

Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!

David A. Wheeler

dwheeler@dwheeler.com

September 26, 2001

Open Source Software / Free Software (OSS/FS) has risen to prominence. If you're not sure what these terms mean, you can get an explanation of these terms and related information from my [list of Open Source Software / Free Software \(OSS/FS\) references at http://www.dwheeler.com/oss_fs_refs.html](http://www.dwheeler.com/oss_fs_refs.html).

However, you shouldn't base a decision to use OSS/FS on a few anecdotes, which is all some sites provide. Instead, this paper emphasizes *quantitative* measures (such as experiments and market studies) on why using OSS/FS products is, in a number of circumstances, a reasonable or even superior approach. I should note that while I find much to like about OSS/FS, I'm not a rabid advocate; I use both proprietary and OSS/FS products myself. Vendors of proprietary products often work hard to find numbers to support their claims; this page provides a useful antidote of hard figures to aid in comparing these proprietary products to OSS/FS.

Note that this paper's goal is *not* to show that all OSS/FS is better than all proprietary software. There are those who believe this is true from ethical, moral, or social grounds, but no numbers could prove such a broad statement. Instead, I'll simply compare commonly-used OSS/FS software with commonly-used proprietary software, to show that, at least in certain situations and by certain measures, OSS/FS is at least as good or better than its proprietary competition.

I'll emphasize the GNU/Linux operating system (which some abbreviate as ``Linux'') and the Apache web server, since these are some of the most visible OSS/FS projects. I'll also primarily compare it to Microsoft Windows, since Windows has significant market share and Microsoft is one of proprietary software's strongest proponents. I'll mention Unix systems in passing as well, though the situation with Unix is more complex; many Unix systems include a number of OSS/FS components or software primarily derived from OSS/FS components. Thus, comparing proprietary Unix systems to OSS/FS systems (when examined as entire systems) is often not as clear-cut. Both Unix and GNU/Linux are ``Unix-like" systems.

Below is data involving market share, reliability, performance, scalability, security, and total cost of ownership. I close with a brief discussion of non-quantitative issues and a list of other sites providing other related information.

Market Share

Many people believe that a product is only a winner if it has significant market share. This is lemming-like, but there's some rationale for this: products with big market shares get applications, trained users, and momentum that reduces future risk. Some writers argue against OSS/FS or GNU/Linux as ``not being mainstream", but if their use is widespread then such statements reflect the past, not the present. There's excellent evidence that OSS/FS has significant market share in several markets:

1. **Apache is the #1 web server on the public Internet, and has been since April 1996.** [Netcraft's](#)

[statistics on webservers](#) have consistently shown Apache (an OSS/FS web server) dominating the public Internet web server market ever since Apache became the #1 web server in April 1996. For example, in August 2000, they polled all the active sites they could (totaling 19,823,296 sites), and found that Apache had 61.66% of the market; the next two largest were Microsoft (19.63%) and iPlanet (aka Netscape's, 7.22%). The June 2001 figures show Apache with 62.42%, Microsoft with 26.14%, and iPlanet with 2.32%. The same result has been determined independently by [E-soft](#) - their report published July 1, 2001 surveyed 3,178,197 web servers and found that Apache was #1 (59.84%), with Microsoft IIS being #2 (28.42%).

2. **GNU/Linux is the #2 web serving operating system on the public Internet (counting by IP address), according to a June 2001 study.** The [Web Server Survey published June 2001](#) by [Netcraft](#) found that GNU/Linux is the #2 operating system for web servers when counting by IP address (and has been consistently gaining market share since February 1999). Every connection to a network accessible to the public is given a unique number called the IP address, and this survey uses IP addresses. Here's a summary of Netcraft's study results:

OS group	Percentage	Composition
Windows	49.2%	Windows 2000, NT4, NT3, Windows 95, Windows 98
Linux	28.5%	Linux
Solaris	7.6%	Solaris 2, Solaris 7, Solaris 8
BSD	6.3%	BSDI BSD/OS, FreeBSD, NetBSD, OpenBSD
Other Unix	2.4%	AIX, Compaq Tru64, HP-UX, IRIX, SCO Unix, SunOS 4 and others
Other non-Unix	2.5%	MacOS, NetWare, proprietary IBM OSs
Unknown	3.6%	not identified by Netcraft operating system detector

Much depends on what you want to measure. Several of the BSDs (FreeBSD, NetBSD, and OpenBSD) are OSS/FS as well; so at least a portion of the 6.3% for BSD should be added to Linux's 28.5% to determine the percentage of OSS/FS operating systems being used as web servers. Thus, it's likely at least 31% of web serving computers use OSS/FS operating systems. There are also regional differences, for example, GNU/Linux leads Windows in Germany, Hungary, the Czech Republic, and Poland.

If you really want to know about the market breakdown of ``Unix vs. Windows," you can find that also in this study. All of the various Windows operating systems are rolled into a single number (even Windows 95/98 and Windows 2000/NT4/NT3 are merged together, although they are fundamentally very different systems). Merging all the Unix-like systems in a similar way produces a total of 44.8% for Unix-like systems (compared to Windows' 49.2%).

Note that these figures would probably be quite different if they were based on web addresses instead of IP addresses; in such a case, the clear majority of web sites are hosted by Unix-like systems. As stated by Netcraft, ``Although Apache running on various Unix systems runs more sites than Windows, Apache is heavily deployed at hosting companies and ISPs who strive to run as many sites as possible on a single computer to save costs."

3. **GNU/Linux is the #1 server operating system on the public Internet (counting by domain name), according to a 1999 survey of primarily European and educational sites.** The first study that I've found that examined GNU/Linux's market penetration is a survey by [Zoebelein in April 1999](#). This survey found that, of the total number of servers deployed on the Internet in 1999 (running at least ftp, news, or http (WWW)) in a database of names they used, the #1 operating system was GNU/Linux (at 28.5%), with others trailing. It's important to note that this survey, which is the first one that I've found to try to answer questions of market share, used existing

databases of servers from the .edu (educational domain) and the RIPE database (which covers Europe, the Middle East, parts of Asia, and parts of Africa), so this isn't really a survey of "the entire Internet" (e.g., it omits ".com" and ".net"). This is a count by domain *name* (e.g., the text name you would type into a web browser for a location) instead of by IP address, so what it's counting is different than the Netcraft June 2001 operating system study. Also, this study counted servers providing ftp and news services (not just web servers).

Here's how the various operating systems fared in the study:

Market Share	Operating System	Composition
GNU/Linux	28.5%	GNU/Linux
Windows	24.4%	All Windows combined (including 95, 98, NT)
Sun	17.7%	Sun Solaris or SunOS
BSD	15.0%	BSD Family (FreeBSD, NetBSD, OpenBSD, BSDI, ...)
IRIX	5.3%	SGI IRIX

A portion of the BSD family is also OSS/FS, so the OSS/FS operating system total is even higher; if over 2/3 of the BSDs are OSS/FS, then the total share of OSS/FS would be about 40%. Advocates of Unix-like systems will notice that the majority (around 66%) were running Unix-like systems, while only around 24% ran a Microsoft Windows variant.

- GNU/Linux is the #2 server operating system sold in 1999 and 2000, and is the fastest-growing.** According to [a June 2000 IDC survey](#) of 1999 licenses, 24% of all servers (counting both Internet and intranet servers) installed in 1999 ran GNU/Linux. Windows NT came in first with 36%; all Unices combined totaled 15%. Again, since some of the Unices are OSS/FS systems (e.g., FreeBSD, OpenBSD, and NetBSD), the number of OSS/FS systems is actually larger than the GNU/Linux figures. Note that it all depends on what you want to count; 39% of all servers installed from this survey were Unix-like (that's 24%+15%), so "Unix-like" servers were actually #1 in installed market share once you count GNU/Linux and Unix together.

IDC released a similar study on January 17, 2001 titled "[Server Operating Environments: 2000 Year in Review](#)". On the server, Windows accounted for 41% of new server operating system sales in 2000, growing by 20% - but GNU/Linux accounted for 27% and grew even faster, by 24%. Other Unices had 13%, and Novell declined (to 17%).

These measures do *not* measure all systems installed that year; some Windows systems are not paid for (they're illegally pirated), and GNU/Linux is often downloaded and installed for multiple systems (since it's legal and free to do so).

- Microsoft sponsored its own research to "prove" that GNU/Linux isn't as widely used, but this research has been shown to be seriously flawed.** Microsoft sponsored a [Gartner Dataquest report](#) claiming only 8.6% of servers shipped in the U.S. during the third quarter of 2000 were Linux-based. However, it's worth noting that Microsoft (as the research sponsor) has every incentive to create low numbers, and these numbers are quite different from IDC's research in the same subject. IDC's Kusnetzky commented that the likely explanation is that Gartner used a very narrow definition of "shipped"; he thought the number was "quite reasonable" if it only surveyed new servers with Linux, "But our research is that this is not how most users get their Linux. We found that just 10 to 15 percent of Linux adoption comes from pre-installed machines... for every paid copy of Linux, there is a free copy that can be replicated 15 times." Note that it's quite difficult to buy a new x86 computer without a Microsoft operating system (Microsoft's contracts with computer makers ensure this), but that doesn't mean that these operating systems are used. Gartner claimed that it used interviews to counter this problem, but its final research results (when

compared to known facts) suggest that Gartner did not really counter this effect. For example, Gartner states that Linux shipments in the supercomputer field were 'zero'. In fact, Linux is widely used on commodity parallel clusters at many scientific sites, including a number of high-profile sites. Many of these systems were assembled in-house, showing that Gartner's method of defining a "shipment" does not appear to correlate to working installations. The Register's article, "[No one's using Linux](#)" (with its companion article "[90% Windows..](#)") discusses this further. In short, Microsoft-sponsored research reported low numbers, but these numbers are quite suspect.

6. **GNU/Linux had 80% as many client shipments in 1999 as Apple's MacOS.** According to [the June 2000 IDC survey](#) of 1999 licenses (5.0% for Mac OS, 4.1% for GNU/Linux), there were almost as many client shipments of GNU/Linux as there were of MacOS - and no one doubts that MacOS is a client system.

Currently, GNU/Linux is a relatively new contender in the client operating system (OS) market, and has relatively little market penetration. This should not be surprising; unlike the wealth of server and developer applications, GNU/Linux has relatively few OSS/FS client applications and many of those client applications are still maturing. There are commercial client applications (e.g. Corel's Word Perfect), but they aren't OSS/FS and are available on other platforms as well. Without strong OSS/FS client applications (in particular an office suite), GNU/Linux has no strong advantage over other client platforms. After all, GNU/Linux combined with a proprietary office suite still lacks the freedoms and low cost of purely OSS/FS systems, and it has to compete with established proprietary systems which have more applications available to them. Microsoft Office is essentially a monopoly controlling the office suite arena, and it isn't available on GNU/Linux (the second-largest player in office suites is Corel; since Corel is partly owned by Microsoft, Corel cannot really be considered a competitor to Microsoft). StarOffice has recently been released as open source (and renamed [OpenOffice](#)), and this has suddenly made it possible to have an OSS/FS office suite. However, its user interface is uncommon (which many dislike), and simply changing its license hasn't suddenly made its interface more like others' user interfaces. Other OSS/FS office application programs, such as AbiWord, Gnumeric, and the KOffice suite, are just now maturing. In addition, the cost of switching all those desktops in the face of compatibility issues with an entrenched monopoly makes it difficult to use anything other than Windows and Office on clients.

Nevertheless, in spite of weaker OSS/FS client applications in several key areas at the time, [an IDC study](#) determined that GNU/Linux captured 4% of the market in 1999. More generally, IDC determined that there were 98.6 million shipments of client operating systems in 1999, of which Windows 3.x, 95 and 98 accounted for 66%, Windows NT had 21%, MacOS had 5%, and GNU/Linux 4%. It will be interesting to see if these numbers begin to change around 2002-2003, when I anticipate OSS/FS client applications (e.g., [AbiWord](#)) will be available that will begin to rival their proprietary competitors in both useability and in the functionality that people need. There's already some evidence that others anticipate this; [Richard Thwaite, director of IT for Ford Europe, stated in 2001 that an open source desktop is their goal, and that they expect the industry to eventually go there](#) (he controls 33,000 desktops, so this would not be a trivial move). It could be argued that this is just a ploy for negotiation with Microsoft - but such ploys only work if they're credible.

7. **Businesses plan to increase their use of GNU/Linux.** A Zona Research study found that more than half of the large enterprise respondents expected increases of up to 25% in the number of GNU/Linux users in their firm, while nearly 20% expected increases of more than 50%. In small companies, more than one third felt that GNU/Linux usage would expand by 50%. The most important factors identified that drove these decisions were reliability, lower price, speed of applications, and scalability. Here are the numbers:

Expected GNU/Linux Use	Small Business	Midsize Business	Large Business	Total
50% increase	21.0%	16%	19.0%	19%
10-25% increase	30.5%	42%	56.5%	44%
No growth	45.5%	42%	24.5%	36%
Reduction	3.0%	0%	0%	1%

You can see more about this study in [``The New Religion: Linux and Open Source''](#) (ZDNet) and in InfoWorld's February 5, 2001 article [``Linux lights up enterprise: But concerns loom about OS vendor profitability.''](#)

8. **The global top 1000 Internet Service Providers expect GNU/Linux use to increase by 154%, according to Idaya's survey conducted January through March 2001.** A [survey](#) conducted by [Idaya](#) of the global top 1000 ISPs found that they expected GNU/Linux to grow a further 154% in 2001. Also, almost two thirds (64%) of ISPs consider the leading open source software meets the standard required for enterprise level applications, comparable with proprietary software. Idaya produces OSS/FS software, so keep that in mind as a potential bias.
9. **IBM found a 30% growth in the number of enterprise-level applications for GNU/Linux in the six month period ending June 2001.** At one time, it was common to claim that [``Not enough applications run under GNU/Linux''](#) for enterprise-level use. However, [IBM found there are more than 2,300 Linux applications \(an increase in 30% over 6 months\)](#) available from IBM and the industry's top independent software vendors (ISVs).
10. **A survey in the second quarter of 2000 found that 95% of all reverse-lookup domain name servers (DNS) used bind, an OSS/FS product.** The Internet is built from many mostly-invisible infrastructure components. This includes domain name servers (DNSs), which turn human-readable machine names (like [``yahoo.com''](#)) and translate them into numeric addresses. Publicly accessible machines also generally support [``reverse lookups''](#), which convert the numbers back to names; for historical reasons, this is implemented using the hidden [``in-addr.arpa''](#) domain. By surveying the in-addr domain, you can gain insight into how the entire Internet is supported. [Bill Manning has surveyed the in-addr domain](#) and found that 95% of all name servers (in 2q2000) performing this important Internet infrastructure task are some version of [``bind''](#). Bind is an OSS/FS program.

Reliability

There are a lot of anecdotal stories that OSS/FS is more reliable, but finally there is quantitative data confirming that mature OSS/FS programs are more reliable:

1. **Equivalent OSS/FS applications are more reliable, according to a 1995 study.** The 1995 [``Fuzz Revisited''](#) paper measured reliability by feeding programs random characters and determining which ones resisted crashing and freeze-ups. Some researchers scoff at this measure, since this approach is unlikely to find subtle failures, but the study authors note that their approach still manages to find many errors in production software and is a useful tool for finding software flaws.

OSS/FS had higher reliability by this measure. It states in section 2.3.1 that:

It is also interesting to compare results of testing the commercial systems to the results from testing "freeware" GNU and Linux. The seven commercial systems in the 1995 study have an average failure rate of 23%, while Linux has a failure rate of 9% and the GNU utilities have a failure rate of only 6%. It is reasonable to ask why a globally scattered group of programmers, with no formal testing support or software

engineering standards can produce code that is more reliable (at least, by our measure) than commercially produced code. Even if you consider only the utilities that were available from GNU or Linux, the failure rates for these two systems are better than the other systems.

There is evidence that Windows NT applications have similar reliability to the proprietary Unix software (e.g., less reliable than the OSS/FS software). A later paper, [`An Empirical Study of the Robustness of Windows NT Applications Using Random Testing`](#), found that with Windows NT GUI applications, they could crash 21% of the applications they tested, hang an additional 24% of the applications, and could crash or hang *all* the tested applications when subjecting them to random Win32 messages. Thus, there's no evidence that proprietary Windows software is more reliable than OSS/FS by this measure.

Now be careful: OSS/FS is not magic pixie dust; beta software of any kind is still buggy! However, the 1995 experiment measured mature OSS/FS to mature proprietary software, and the OSS/FS software was more reliable under this measure.

2. **GNU/Linux is more reliable than Windows NT, according to a 10-month ZDnet experiment.** [ZDnet ran a 10-month test for reliability](#) to compare Caldera Systems OpenLinux, Red Hat Linux, and Microsoft's Windows NT Server 4.0 with Service Pack 3. All three used identical (single-CPU) hardware, and network requests were sent to each server in parallel for standard Internet, file, and print services. The result: NT crashed an average of once every six weeks, each taking about 30 minutes to fix; that's not bad, but neither GNU/Linux server *ever* went down. This ZDnet article also does a good job of identifying GNU/Linux weaknesses (e.g., desktop applications and massive SMP).
3. **GNU/Linux is more reliable than Windows NT, according to a one-year Bloor Research experiment.** [Bloor Research](#) had both operating systems running on relatively old Pentium machines. In the space of one year, GNU/Linux crashed once because of a hardware fault (disk problems), which took 4 hours to fix, giving it a measured availability of 99.95 percent. Windows NT crashed 68 times, caused by hardware problems (disk), memory (26 times), file management (8 times), and a number of odd problems (33 times). All this took 65 hours to fix, giving an availability of 99.26 percent. It's intriguing that the only GNU/Linux problem and a number of the Windows problems were hardware-related; it could be argued that the Windows hardware was worse, or it could be argued that GNU/Linux did a better job of avoiding and containing hardware failures. In any case, file management failure can be blamed on Windows, and the "odd problems" appear to be caused by Windows as well, indicating that GNU/Linux is far more reliable than Windows. Gnet summarized this as saying "the winner here is clearly Linux."
4. **Sites using Microsoft's IIS web serving software have more than double the time offline (on average) than sites using the Apache software, according to a 3-month Swiss evaluation.** These are the results of [Syscontrol AG's analysis of website uptime \(announced February 7, 2000\)](#) They measured over 100 popular Swiss web sites over a three-month period, checking from 4 different locations every 5 minutes (it'd be interesting to see what a larger sample would find!). You can [see their report \(in German\)](#), or a [Babelfish \(machine\) translation of the report](#). Here's their entire set of published data on "average down-time per hour per type of server", plus a 3-month average that I've computed:

Downtime	Apache	Microsoft	Netscape	Other
September	5.21	10.41	3.85	8.72
October	2.66	8.39	2.80	12.05
November	1.83	14.28	3.39	6.85

Average	3.23	11.03	3.35	9.21
----------------	------	-------	------	------

It's hard not to notice that Apache (the OSS web server) had the best results over the three-month average (and with better results over time, too). Indeed, Apache's worst month was better than Microsoft's best month. I believe the difference between Netscape and Apache is statistically insignificant - but this still shows that the freely-available OSS/FS solution (Apache) has a reliability at least as good as the most reliable proprietary solution. The report does note that this might not be solely the fault of the software's quality, since there were several Microsoft IIS sites that had short interruptions at the same time each day (suggesting regular restarts). However, this still begs the question -- why did the IIS sites require so many more regular restarts than the Apache sites? Every outage, even if preplanned, results in a service loss (and for e-commerce sites, a potential loss of sales).

5. **According to a separate uptime study by Netcraft, OSS/FS does very well; as of August 3, 2001, of the 50 sites with the highest uptimes, 92% use Apache and 50% run on OSS/FS operating systems.** Netcraft keeps a track of the 50 often-requested sites with the longest uptimes at <http://uptime.netcraft.com>. Looking at [the August 3, 2001 uptime report](#), I found that 92% (46/50) of the sites use Apache; one site's web server was unknown, and three others were not Apache. Of those three, only one reported to be Microsoft IIS, and that one instance is suspicious because its reported operating system is BSD/OS (this apparant inconsistency can be explained in many ways, e.g., perhaps there is a front-end BSD/OS system that ``masks" the IIS web site, or perhaps the web server is lying about its type to confuse attackers). In this snapshot, 50% (25/50) ran on an open source operating system, and only Unix-like operating systems had these large uptimes (no Windows systems were reported as having the best uptimes).

As with all surveys, this one has weaknesses, as discussed in [Netcraft's Uptime FAQ](#). Their techniques for identifying web server and operating systems can be fooled. Only systems for which Netcraft was sent many requests were included in the survey (so it's not ``every site in the world"). Any site that is requested through the ``what's that site running" query form at Netcraft.com is added to the set of sites that are routinely sampled; Netcraft doesn't routinely monitor all 22 million sites it knows of for performance reasons. Many operating systems don't provide uptime information and thus can't be included; this includes AIX, AS/400, Compaq Tru64, DG/UX, MacOS, NetWare, NT3/Windows 95, NT4/Windows 98, OS/2, OS/390, SCO UNIX, Sony NEWS-OS, SunOS 4, and VM. Thus, this uptime counter can only include systems running on BSD/OS, FreeBSD (but not the default configuration in versions 3 and later), recent versions of HP-UX, IRIX, Linux 2.1 kernel and later (except on Alpha processor based systems), MacOS X, recent versions of NetBSD/OpenBSD, Solaris 2.6 and later, and Windows 2000. Note that Windows NT systems cannot be included in this survey (because their uptimes couldn't be counted), but Windows 2000 systems are included in this survey. Again, no Windows system actually made it into the top 50 for uptimes in this snapshot. Note that HP-UX, (many versions of) Linux, Solaris and recent releases of FreeBSD cycle back to zero after 497 days, exactly as if the machine had been rebooted at that precise point. Thus it is not possible to see an HP-UX, (most) Linux, or Solaris system with an uptime measurement above 497 days, and in fact their uptimes can be misleading (they may be up for a long time, yet not show it). Still, this survey does compare Windows 2000, Linux (up to 497 days usually), FreeBSD, and several other operating systems, and a vast number of web servers, and OSS/FS does quite well.

Of course, there are many anecdotes about Windows reliability vs. Unix. For example, the [Navy's ``Smart Ship" program caused a complete failure of the entire USS Yorktown ship in September 1997](#). Anthony DiGiorgio (a whistle-blower) stated that Windows is ``the source of the Yorktown's computer problems." Ron Redman, deputy technical director of the Fleet Introduction Division of the Aegis Program Executive

Office, said ``there have been numerous software failures associated with [Windows] NT aboard the Yorktown." Redman also said ``Because of politics, some things are being forced on us that without political pressure we might not do, like Windows NT... If it were up to me I probably would not have used Windows NT in this particular application. If we used Unix, we would have a system that has less of a tendency to go down."

One problem with reliability measures is that it takes a long time to gather data on reliability in real-life circumstances. Nevertheless, the available evidence suggests that OSS/FS has a significant edge in reliability.

Performance Data

Comparing GNU/Linux and Windows NT/2000 performance on equivalent hardware has a history of contentious claims and different results based on different assumptions. I think that OSS/FS has at least shown that it's often competitive, and in many circumstances it beats the competition.

Performance benchmarks are very sensitive to the assumptions and environment, so the best benchmark is one you set up yourself to model your intended environment. Failing that, you should use unbiased measures, because it's so easy to create biased measures.

First, here are a few recent studies suggesting that some OSS/FS systems beat their proprietary competition in at least some circumstances:

1. **GNU/Linux with TUX has produced better SPEC values than Windows/IIS in several cases, even when given inferior drive configurations.** One organization that tries to develop unbiased benchmarks is the [SPEC Consortium](#), which develops and maintains a whole series of benchmarks. We can compare Microsoft Windows versus GNU/Linux by comparing SPECweb99 results (which measure web server performance) on identical hardware if both have undergone the same amount of performance optimization effort. Alas, things are not so simple; rarely are the same basic hardware platforms tested with both operating systems, and even when that occurs, as of July 13, 2001 no exactly identical configurations have been tested (they differ in ways such as using a different number of hard drives, or including some faster hard drives). Using all results available by July 13, 2001, there were three hardware configurations, all from Dell, which ran both GNU/Linux (using the TUX web server/accelerator) and Windows (using IIS) on exactly the same underlying hardware. Here are the SPECweb99 results as of July 13, 2001 (larger is better), noting configuration differences:

System	Windows SPEC Result	Linux SPEC Result
Dell PowerEdge 4400/800, 2 800MHz Pentium III Xeon	1060 (IIS 5.0, 1 network controller)	2200 (TUX 1.0, 2 network controllers)
Dell PowerEdge 6400/700, 4 700MHz Pentium III Xeon	1598 (IIS 5.0, 7 9GB 10KRPM drives)	4200 (TUX 1.0, 5 9GB 10KRPM drives)
Dell PowerEdge 8450/700, 8 700MHz Pentium III Xeon	7300/NC (IIS 5.0, 1 9Gb 10KRPM and 8 16Gb 15KRPM drives) then 8001 (IIS 5.0, 7 9Gb 10KRPM and 1 18Gb 15KRPM drive)	7500 (TUX 2.0, 5 9Gb 10KRPM drives)

The first row (the PowerEdge 4400/800) doesn't really prove anything. The IIS system has lower performance, but it only had one network controller and the TUX system has two - so while the TUX system had better performance, that could simply be because it had two network connections

it could use.

The second entry (the PowerEdge 6400/700) certainly suggests that GNU/Linux plus TUX really is much better - the IIS system had two more disk drives available to it (which should increase performance), but the TUX system had more than twice the IIS system's performance.

The last entry for the PowerEdge 8450/700 is even more complex. First, the drives are different - the IIS systems had at least one drive that revolved more quickly than the TUX systems (which should give IIS higher performance overall, since the transfer speed is almost certainly higher). Also, there were more disk drives (which again should give IIS still higher performance). When I originally put this table together showing all data publicly available in April 2001 (covering the third quarter of 1999 through the first quarter of 2001), IIS 5.0 (on an 8-processor Dell PowerEdge 8450/700) had a SPECweb99 value of 7300. Since that time, Microsoft changed the availability of Microsoft SWC 3.0, and by SPECweb99 rules, this means that those test results are "not compliant" (NC). This is subtle; it's not that the test itself was invalid, it's that Microsoft changed what was available and used the SPEC Consortium's own rules to invalidate a test (possibly because the test results were undesirable to Microsoft). A retest then occurred, with yet another disk drive configuration, at which point IIS produced a value of 8001. However, both of these figures are on clearly better hardware - and in one circumstance the better hardware didn't do better.

Thus, in these configurations the GNU/Linux plus TUX system was given inferior hardware yet still sometimes won on performance. Since other factors may be involved, it's hard to judge - there are pathological situations where "better hardware" can have worse performance, or there may be another factor not reported that had a more significant effect. Hopefully in the future there will be many head-to-head tests in a variety of identical configurations.

Note that TUX is intended to be used as a "web accelerator" for many circumstances, where it rapidly handles simple requests and then passes more complex queries to another server (usually Apache). I've quoted the TUX figures because they're the recent performance figures I have available. As of this time I have no SPECweb99 figures or other recent performance measures for Apache on GNU/Linux, or for Apache and TUX together; I also don't have TUX reliability figures. I expect that such measures will appear in the future.

2. **In performance tests by Sys Admin magazine, GNU/Linux beat Solaris (on Intel), Windows 2000, and FreeBSD.** The article ["Which OS is Fastest for High-Performance Network Applications?"](#) in the July 2001 edition of [Sys Admin](#) magazine examined high-performance architectures and found that GNU/Linux beat its competition when compared with Solaris (on Intel), FreeBSD (an OSS/FS system), and Windows 2000. They intentionally ran the systems "out of the box" (untuned), except for increasing the number of simultaneous TCP/IP connections (which is necessary for testing multi-threaded and asynchronous applications). They used the latest versions of operating systems and the exact same machine. They reported (by operating system) the results of two different performance tests.

The FreeBSD developers complained about these tests, noting that FreeBSD by default emphasizes reliability (not speed) and that they expected anyone with a significant performance need would do some tuning first. Thus, [Sys Admin's re-did the tests for FreeBSD after tuning FreeBSD](#). One change they made was switching to "asynchronous" mounting, which makes a system faster (though it increases the risk of data loss in a power failure) - this is the GNU/Linux default and easy to change in FreeBSD, so this was a very small and reasonable modification. However, they also made many other changes, for example, they found and compiled in 17 FreeBSD kernel patches and used various tuning commands. The other operating systems weren't given the chance to "tune" like this, so comparing untuned operating systems to a tuned FreeBSD isn't really fair.

In any case, here are their two performance tests:

1. Their ``real-world" test measured how quickly large quantities of email could be sent using their email delivery server (MailEngine). Up to 100 simultaneous sends there was no difference, but as the number increased the systems began showing significant differences in their hourly email delivery speed. By 500 simultaneous sends GNU/Linux was clearly faster than all except FreeBSD-tuned, and GNU/Linux remained at the top. FreeBSD-tuned had similar performance to GNU/Linux when running 1000 or less simultaneous sends, but FreeBSD-tuned peaked around 1000-1500 simultaneous connections with a steady decline not suffered by GNU/Linux, and FreeBSD-tuned had trouble going beyond 3000 simultaneous connections. By 1500 simultaneous sends, GNU/Linux was sending 1.3 million emails/hour, while Solaris managed approximately 1 million, and Windows 2000 and FreeBSD-untuned were around 0.9 million.
2. Their ``disk I/O test" created, wrote, and read back 10,000 identically-sized files in a single directory, varying the size of the file instances. Here Solaris was the slowest, with FreeBSD-untuned the second-slowest. FreeBSD-tuned, Windows 2000, and GNU/Linux had similar speeds at the smaller file sizes (in some cases FreeBSD-tuned was faster, e.g., 8k and 16k file size), but when the file sizes got to 64k to 128k the operating systems began to show significant performance differences; GNU/Linux was the fastest, then Windows 2000, then FreeBSD. At 128k, FreeBSD was 16% worse than Windows 2000, and 39% worse than GNU/Linux; all were faster than FreeBSD-untuned and Solaris. When totaling these times across file sizes, the results were GNU/Linux: 542 seconds, Windows 2000: 613 seconds, FreeBSD-tuned: 630 seconds, FreeBSD-untuned: 2398 seconds, and Solaris: 3990 seconds.

All operating systems in active development are in a constant battle for performance improvements over their rivals. The history of comparing Windows and GNU/Linux helps put this in perspective:

1. **Ziff-Davis found that GNU/Linux with Apache beat Windows NT 4.0 with IIS by 16%-50% depending on the GNU/Linux distribution.** [Ziff-Davis compared Linux and Windows NT's performance at web serving](#). They found that ``Linux with Apache beats NT 4.0 with IIS, hands down. SuSE, the least effective Linux, is 16% faster than IIS, and Caldera, the leader, is 50% faster."
2. [Mindcraft released a report in April 1999](#) that claimed that **Microsoft Windows NT Server 4.0 is 2.5 times faster than Linux (kernel 2.2) as a File Server and 3.7 times faster as a Web Server when running on a 4-CPU SMP system.** Several people and organizations, such [Linux Weekly News \(LWN\)](#) and [Dan Kegel](#), identified serious problems with this study. An obvious issue was that NT was specially tuned by Microsoft's NT experts, at Microsoft, while GNU/Linux was not tuned at all. Another issue is that the price/performance wasn't considered (nor was total expenditure kept constant - for the same amount of money, the GNU/Linux system could have had better hardware). Mindcraft claimed they asked for help, but they didn't use the documented methods for getting help nor did they purchase a support contract. Many were especially offended that even though this study was funded by Microsoft (one of the contestants) and held at their facility, neither Mindcraft's initial announcement nor its paper made any mention of this conflict-of-interest - and it could be easily claimed that their configuration was designed to put GNU/Linux at a disadvantage. Their configuration was somewhat bizarre - it assumed all web pages were static (typical big sites tend to use many dynamically generated pages) and that there were 100 or so clients connected via 100baseT (in 1999 a more typical situation would be that most clients are using slower 28.8 or 56 Kbps modems).

Careful examination of the benchmark did find some legitimate Linux kernel problems, however. These included a TCP bug, the lack of ``wake one" semantics, and SMP bottlenecks

(see [Dan Kegel's pages](#) for more information). The Linux kernel developers began working on the weaknesses identified by the benchmark.

3. **PC Week confirmed that Windows did indeed do better in this less probable configuration.** In June 30, 1999, Mindcraft released their [Open Benchmark](#) in conjunction with PC Week. While this didn't excuse Mindcraft's biases, it did make a convincing case that there were legitimate problems in the Linux kernel and Apache that made GNU/Linux a poorer-performing product in this somewhat improbable configuration (serving static web pages to clients with high-speed connections). Note that this configuration was considerably different than Ziff-Davis's, so the benchmarks don't necessarily conflict; it's merely that different assumptions can produce different results (as I've already stressed).
4. **Network Computing found that GNU/Linux with Samba ran at essentially the same speed as Windows for file serving.** In their article ["Is it Time for Linux"](#), Network Computing compared Red Hat Linux v5.2 running Samba 2.0.3 against Microsoft Windows NT Server Enterprise Edition on a Pentium II-based HP NetServer LPr. They used Coffee Computing Corp.'s FileMetric 1.0 benchmarking tool (www.coffeecomputing.com), stressing the machine with multiple reads and writes of small, medium and large files over the course of several hours, examining ramp-ups, massive file transfers and saturating the network.

For file serving, they discovered only "negligible performance differences between the two for average workloads... [and] depending on the degree of tuning performed on each installation, either system could be made to surpass the other slightly in terms of file-sharing performance." Linux slightly outperformed NT on file writes, but NT edged out Linux on massive reads. Note that their configuration was primarily network-limited; they stated "At no point were we able to push the CPUs much over 50-percent utilization--the single NIC, full duplex 100BASE-T environment wouldn't allow it."

They also noted that "examining the cost difference between the two licenses brings this testing into an entirely new light... the potential savings on licenses alone is eye-opening. For example, based on the average street price of \$30 for a Windows NT client license, 100 licenses would cost around \$3,000, plus the cost of an NT server license (around \$600). Compare this to the price of a Red Hat Linux CD, or perhaps even a free download, and the savings starts to approach the cost of a low-end workgroup server. Scale that up to a few thousand clients and you begin to see the savings skyrocket."

5. **The German magazine c't found that web sites with NT was better at static content and dual network connections, but GNU/Linux was better for sites with dynamic content and single connections.** Their article [Mixed Double: Linux and NT as Web Server on the Test Bed](#) examined Windows NT with IIS against GNU/Linux (kernel 2.2.9) with Apache on a machine with four Pentium II Xeon CPUs. They found that the performance winner depended on the situation (by now that should not be a surprise). If the web server primarily served static web pages through two high-performance network cards, NT's performance was better. However, they also noted that in sophisticated web sites this result didn't apply, because such sites tend to have primarily dynamic content, and that few sites had this kind of dual-network connection (when only one network board was available, GNU/Linux generally had an edge). See their paper for more figures and background.
6. **The Linux developers' various efforts to improve performance appear to have paid off.** In June 2000, Dell measured the various SPECweb99 values noted above.

There are other benchmarks available, but I've discounted them on various grounds:

1. A more recent set of articles from eWeek on June 2001, shows some eye-popping performance numbers for GNU/Linux with TUX. However, although they compare it to

Microsoft IIS, they don't include Microsoft's SWC (Scaleable Web Cache), Microsoft's response to TUX - and omitting it makes this comparison unfair in my opinion. You can read more at [`Tux: Built for Speed`](#), [`Smart Coding pays off Big`](#), and [Kegel's detailed remarks](#).

2. The ZDNet article [Take that! Linux beats MS in benchmark test](#), loudly trumpeted that GNU/Linux was the May 2001 performance leader in the TPC-H decision support (database) benchmark ("100Gb" category). However, this result should not be taken very seriously; the hardware that Linux ran on was more powerful than that of the runner-up (Windows 2000). Frankly, the more surprising fact than its top score (which can be easily explained by the hardware) is its mere measurement at all with this benchmark - traditionally only Microsoft's numbers were reported for this benchmark at this range. For more information, see [the TPC results](#).

You can get a lot more information on various benchmarks from Kegel's [NT vs. Linux Server Benchmark Comparisons](#), and many more benchmarks from [SPEC](#). You can also see the [dmoz entry on benchmarking](#).

Remember, in benchmarking everything depends on the configuration and assumptions that you make. Many systems are constrained by network bandwidth; in such circumstances buying a faster computer won't help at all. Even when network bandwidth isn't the limitation, neither Windows nor GNU/Linux do well in large-scale symmetric multiprocessing (SMP) configurations; if you want 64-way CPUs with shared memory, neither are appropriate (Sun Solaris, which is not OSS/FS, does much better in this configuration). On the other hand, if you want massive distributed (non-shared) memory, GNU/Linux does quite well, since you can buy more CPUs with a given amount of money. If massive distribution can't help you and you need very high performance, Windows isn't even in the race; today Windows 2000 only runs on Intel x86 compatible chips, while GNU/Linux runs on much higher performance processors as well as the x86.

Scaleability

Which brings us to the topic of scaleability, a simple term with multiple meanings:

1. **GNU/Linux and NetBSD (both OSS/FS) support a wider range of hardware platforms and performance than any other operating system.** Many people mean by "scaleability" to answer the question, "can you use the same software system for both small and large projects?" Often the implied issue is that you'd like to start with a modest system, but have the ability to grow the system as needs demand without expensive modifications. Here OSS/FS is unbeatable; because many people can identify scaleability problems, and because its source code can be optimized for its platform, the scaleability of many OSS/FS products is amazing. Let's specifically look at GNU/Linux. GNU/Linux works on [PDAs](#) (including the [Agenda VR3](#)), [obsolete hardware \(so you needn't throw the hardware away\)](#), common modern PC hardware, over a dozen different chipsets (not just Intel x86s), [mainframes](#), [massive clusters](#), and a [number of supercomputers](#). GNU/Linux can be used for massive parallel processing; a common approach for doing this is the [Beowulf architecture](#). [Sandia's `CPlant`](#) runs on a set of systems running GNU/Linux, and it's the forty-second most powerful computer in the world as of June 2001 (number 42 on the [TOP 500 Supercomputer list, June 2001](#)). There's even a prototype implementation of GNU/Linux on a [wrist watch](#). And GNU/Linux runs on a vast number of different CPU chips, including the [x86](#), [Intel Itanium](#), [ARM](#), [Alpha](#), [IBM AS/400 \(midrange\)](#), [SPARC](#), [MIPS](#), [68k](#), and [Power PC](#).

Another operating system that widely scales to many other hardware platforms is [NetBSD](#).

Thus, you can buy a small GNU/Linux or NetBSD system and grow it as your needs grow; indeed, you can replace small hardware with massively parallel or extremely high-speed processors or very different CPU architectures without switching operating systems.

Windows CE/ME/NT scales down to small platforms, but not to large ones, and it only works on x86 systems. Many Unix systems (such as Solaris) scale well to specific large platforms but not as well to distributed or small platforms. These OSS/FS systems are some of the most scaleable programs around.

2. **OSS/FS development processes can scale to develop large software systems.** At one time it was common to ask if the entire OSS/FS process is "scaleable," that is, if OSS/FS processes could really develop large-scale systems. Bill Gates' 1976 "Open Letter to Hobbyists" asked rhetorically, "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?" He presumed these were unanswerable questions. He was wrong; I developed a quantitative examination of the content of a GNU/Linux distribution. Feel free to see [my reports estimating GNU/Linux's size](#). For Red Hat Linux 6.2, I estimate the size to be over 17 million source lines of code (SLOC). Done traditionally it would have taken 4,500 person-years and over \$600 million to implement. For Red Hat Linux 7.1, I found it to have over 30 million SLOC, representing 8,000 person-years or \$1 billion (a "Gigabuck"). Most developers ascribe to the design principle that components should be divided into smaller components where practical - a practice also applied to GNU/Linux - but some components aren't easily divided, and thus some components are quite large themselves (e.g., over 2 million lines of code for the kernel, mostly in device drivers). Thus, it's no longer reasonable to argue that OSS/FS cannot scale to develop large systems -- because it clearly can.

Security

Quantitatively measuring security is difficult. However, here are a few attempts to do so, and they certainly suggest that OSS/FS has a reasonable approach for security. I'll concentrate in particular on comparing OSS/FS to Windows systems.

1. **J.S. Wurzler Underwriting Managers' "hacker insurance" costs 5-15% more if Windows is used instead of Unix or GNU/Linux for Internet operation.** At least one insurance company has indicated that Windows NT is less secure than Unix or GNU/Linux systems, resulting in higher premiums for Windows-based systems. It's often difficult to find out when a company has been successfully cracked; companies often don't want to divulge such information to the public for a variety of reasons. Indeed, if consumers or business partners lost trust in a company, the resulting loss might be much greater than the original attack. However, insurance companies that insure against cracking can require that they get such information (as a condition of coverage), and can compute future premiums based on that knowledge. According to CNET, Okemos, Mich.-based J.S. Wurzler Underwriting Managers, one of the earliest agencies to offer "hacker insurance" (and thus more likely to have historical data for premium calculation), has begun [charging its clients anywhere from 5 to 15 percent more if they use Microsoft's Windows NT software instead of Unix or GNU/Linux for their Internet operations](#). Walter Kopf, senior vice president of underwriting, said that "We have found out that the possibility for loss is greater using the NT system." He also said the decision is based on findings from hundreds of security assessments the company has done on their small and midsize business clients over the past couple of years.

2. **Most defaced web sites are hosted by Windows, and Windows sites are disproportionately defaced more often than explained by its market share.** Another way to look at security is to look at the operating system used by defaced web sites, and compare them to their market share. A "defaced" web site is a site that has been broken into and has its content changed (usually in a fairly obvious way, since subtle modifications are often not reported). The advantage of this measure is that unlike other kinds of security break-ins (which are often "hushed up"), it's often very difficult for victims to hide the fact that they've been successfully attacked. Historically, this information was maintained by Attrition.org. A summary can be found in [James Middleton's article](#), with the actual data found in [Attrition.org's web site](#). Attrition.org's data showed that 59% of defaced systems ran Windows, 21% Linux, 8% Solaris, 6% BSD, and 6% all others in the period of August 1999 through December 2000. Thus, Windows systems have nearly 3 times as many defacements as GNU/Linux systems. This would make sense if there were 3 times as many Windows systems, but no matter which figures you use, that's simply not true.

Of course, not all sites are broken through their web server and OS - many are broken through exposed passwords, bad web application programming, and so on. But if this is so, why is there such a big difference in the number of defacements based on the operating system? No doubt some other reasons could be put forward (this data only shows a correlation not a cause), but this certainly suggests that OSS/FS can have better security.

[Attrition.org has decided to abandon keeping track of this information due to the difficulty of keeping up with the sheer volume of broken sites](#), and it appeared that tracking this information wouldn't be possible. However, [defaced.alldas.de](#) has decided to perform this valuable service. Their recent reports show that this trend has continued; on July 12, 2001, they report that 66.09% of defaced sites ran Windows, compared to 17.01% for GNU/Linux, out of 20,260 defaced websites.

3. **The Bugtraq vulnerability database suggests that the least vulnerable operating system is OSS/FS, and that all the OSS/FS operating systems in its study were less vulnerable than Windows in 1999-2000.** One approach to examining security is to use a vulnerability database; an analysis of one database is the [Bugtraq Vulnerability Database Statistics](#) page. As of September 17, 2000, here are the total number of vulnerabilities for some leading operating systems:

OS	1997	1998	1999	2000
Debian GNU/Linux	2	2	30	20
OpenBSD	1	2	4	7
Red Hat Linux	5	10	41	40
Solaris	24	31	34	9
Windows NT/2000	4	7	99	85

You shouldn't take these numbers very seriously. Some vulnerabilities are more important than others (some may provide little if exploited or only be vulnerable in unlikely circumstances), and some vulnerabilities are being actively exploited (while others have already been fixed before exploitation). Open source operating systems tend to include many applications that are usually sold separately in proprietary systems (including Windows and Solaris) - for example, Red Hat 7.1 includes two relational database systems, two word processors, two spreadsheet programs, two web servers, and a large number of text editors. In addition, in the open source world, vulnerabilities are discussed publicly, so vulnerabilities

may be identified for software still in development (e.g., ``beta" software). Those with small market shares are likely to have less analysis. The ``small market share" comment won't work with GNU/Linux, of course, since we've already established that GNU/Linux is the #1 or #2 server OS (depending on how you count them). Still, this clearly shows that the three OSS/FS OSs listed (Debian GNU/Linux, OpenBSD, and Red Hat Linux) did much better by this measure than Windows in 1999 and (so far) in 2000. Even if a bizarre GNU/Linux distribution was created explicitly to duplicate all vulnerabilities present in any major GNU/Linux distribution, this intentionally bad GNU/Linux distribution would still do better than Windows (it would have 88 vulnerabilities in 1999, vs. 99 in Windows). The best results were for OpenBSD, an OSS/FS operating system that for years has been specifically focused on security. It could be argued that its smaller number of vulnerabilities is because of its rarer deployment, but the simplest explanation is that OpenBSD has focused strongly on security - and achieved it better than the rest.

This data is partly of interest because [one journalist completely distorted these numbers in an attempt to show that GNU/Linux was worse](#). He took these numbers and then added the GNU/Linux ones so each Linux vulnerability was counted at least twice (once for every distribution it applied to plus one more). By using these nonsensical figures he declared that GNU/Linux was worse than anything. If you read his article, you also need to read [the rebuttal by the manager of the Microsoft Focus Area at SecurityFocus](#) to understand why the journalist's article was so wrong.

Indeed, as noted in Bruce Schneier's [Crypto-gram of September 15, 2000](#), vulnerabilities are affected by other things such as how many attackers exploit the vulnerability, the speed at which a fix is released by a vendor, and the speed at which they're applied by administrators. Nobody's system is invincible.

A more recent [analysis by John McCormick in Tech Republic](#) compared Windows and Linux vulnerabilities using numbers through September 2001. This is an interesting analysis, showing that although Windows NT lead in the number of vulnerabilities in 2000, using the 2001 numbers through September 2001, Windows 2000 had moved to the ``middle of the pack" (with some Linux systems having more, and others having fewer, vulnerabilities). However, it appears that in these numbers, bugs in Linux applications have been counted with Linux, while bugs in Windows applications haven't - and if that's so, this isn't really a fair comparison. This is an issue because typical Linux distributions bundle many applications that are separately purchased from Microsoft (such as web servers, database servers, and so on).

4. **Red Hat (an OSS/FS vendor) responded more rapidly than Microsoft or Sun to advisories; Sun had fewer advisories to respond to yet took the longest to respond.** Another data point is that SecurityPortal has compiled a [list of the time it takes for vendors to respond to vulnerabilities](#). They concluded that:

How did our contestants [fare]? Red Hat had the best score, with 348 recess days on 31 advisories, for an average of 11.23 days from bug to patch. Microsoft had 982 recess days on 61 advisories, averaging 16.10 days from bug to patch. Sun proved itself to be very slow, although having only 8 advisories it accumulated 716 recess days, a whopping three months to fix each bug on average.

Their table of data for 1999 is as shown:

1999 Advisory Analysis			
Vendor	Total Days of Hacker Recess	Total Advisories	Days of Recess per Advisory
Red Hat	348	31	11.23
Microsoft	982	61	16.10
Sun	716	8	89.50

Clearly this table uses a different method for counting security problems than the previous table. Of the three noted here, Sun's Solaris had the fewest vulnerabilities, but it took by far the longest to fix security problems identified. Red Hat was the fastest at fixing security problems, and placed in the middle of these three in number of vulnerabilities. It's worth noting that the OpenBSD operating system (which is OSS/FS) had fewer reported vulnerabilities than all of these. Clearly, having a proprietary operating system doesn't mean you're more secure - Microsoft had the largest number of security advisories, by far, using either counting method.

5. **Apache has a better security record than Microsoft's IIS, as measured by reports of serious vulnerabilities.** Eweek's July 20, 2001 article ["Apache avoids most security woes"](#) examined security advisories dating back to Apache 1.0. They found that Apache's last serious security problem (one where remote attackers could run arbitrary code on the server) was announced in January 1997. A group of less serious problems (including a buffer overflow in the server's logresolve utility) was announced and fixed in January 1998 with Apache 1.2.5. In the three and a half years since then, Apache's only remote security problems have been a handful of denial-of-service and information leakage problems (where attackers can see files or directory listings they shouldn't).

In contrast, in the article ["IT bugs out over IIS security,"](#) eWeek determined that Microsoft has issued [21 security bulletins for IIS from January 2000 through June 2001](#). Determining what this number means is a little difficult, and the article doesn't discuss these complexities, so I've examined Microsoft's bulletins myself to find their true significance. Not all of the bulletins have the same significance, so just stating that there were "21 bulletins" doesn't give the whole picture. However, it's clear that several of these bulletins discuss dangerous vulnerabilities that allow an external user to gain control over the system. I count 5 bulletins on such highly dangerous vulnerabilities for IIS 5.0 (in the period from January 2000 through June 2001), and previous to that time, I count 3 such bulletins for IIS 4.0 (in the period of June 1998 through December 1999). Feel free to examine the bulletins yourself; they are MS01-033, MS01-026, MS01-025, MS01-023, MS00-086, MS99-025, MS99-019, and MS99-003. The [Code Red](#) worm, for example, exploited a vast number of IIS sites through the vulnerabilities identified in the June 2001 security bulletin MS01-033.

In short, by totaling the number of reports of dangerous vulnerabilities (that allow attackers to execute arbitrary code), I find a total of 8 bulletins for IIS from June 1998 through June 2001, while Apache had zero such vulnerabilities for that time period. Apache's last such report was in January 1998, and that one affected the log analyzer not the web server itself. As was noted above, the last such dangerous vulnerability in Apache itself was announced in January 1997.

The article claims there are four reasons for Apache's strong security, and three of these reasons are simply good security practices. Apache installs very few server extensions by default (a "minimalist" approach), all server components run as a non-privileged user (supporting "least privilege"), and all configuration settings are centralized (making it easy

for administrators to know what's going on). However, the article also claims that one of the main reasons Apache is more secure than IIS is that its ``source code for core server files is well-scrutinized," a task that is made much easier by being OSS/FS, and it could be argued that OSS/FS encourages the other good security practices.

Simple counts of vulnerability notices aren't necessarily a good measure, of course. A vendor could intentionally release fewer bulletins - but since Apache's code and its security is publicly discussed, it seems unlikely that Apache is releasing fewer notices. Fewer vulnerability notices could result if the product isn't well scrutinized or is rarely used - but this simply isn't true for Apache. Even the trend line isn't encouraging - using the months of the bulletins (2/99, 6/99, 7/99, 11/00, three in 5/01, and 6/01), I find the time in months between new major IIS vulnerability announcements to be 4, 1, 18, 6, 0, 0, 1, and 3 as of September 2001; this compares to 12 and 44 as of September 2001 for Apache. Given these trends, it looks like IIS's security is slowly improving, but it has little likelihood of meeting Apache's security in the near future. Indeed, these vulnerability counts are corroborated by other measures such as the web site defacement rates.

The issue here isn't whether or not a particular program is invincible (what nonsense!) - the issue here is which is more likely to resist future attacks, based on past performance. It's clear that Apache has much a better security record than IIS, so much so that Gartner Group decided to make an unusual recommendation (described next).

- 6. The Gartner Group is recommending that businesses switch from Microsoft IIS to Apache or iPlanet due to IIS's poor security track record, noting that enterprises had spent \$1.2 billion simply fixing Code Red (IIS-related) vulnerabilities by July 2001.** Microsoft's IIS has such a bad security record that in September 2001, [Gartner Group announced a recommendation](#) that ``businesses hit by both Code Red and Nimda immediately investigate alternatives to IIS, including moving Web applications to Web server software from other vendors such as iPlanet and Apache. Although those Web servers have required some security patches, they have much better security records than IIS and are not under active attack by the vast number of virus and worm writers." Microsoft is sometimes a Gartner Group customer, so this announcement is especially surprising.

In a [background document by Gartner](#), they discuss Code Red's impacts further. By July 2001, Computer Economics (a research firm) estimated that enterprises worldwide had spent \$1.2 billion fixing vulnerabilities in their IT systems that Code Red could exploit (remember, Code Red is designed to only attack IIS systems; systems such as Apache are immune). To be fair, Gartner correctly noted that the problem is not just that IIS has vulnerabilities; part of the problem is that enterprises using IIS are not keeping their IT security up to date, and Gartner openly wondered why this was the case. However, Gartner also asked the question, ``why do Microsoft's software products continue to provide easily exploited openings for such attacks?" This was prescient, since soon after this the ``Nimba" attack surfaced which attacked IIS, Microsoft Outlook, and other Microsoft products.

A brief aside is in order here. Microsoft spokesman Jim Desler tried to counter Gartner's recommendation, trying to label it as ``extreme" and saying that ``serious security vulnerabilities have been found in all Web server products and platforms.. this is an industrywide challenge." While true, this isn't the whole truth. As Gartner points out, ``IIS has a lot more security vulnerabilities than other products and requires more care and feeding." It makes sense to select the product with the best security track record, even if no product has a perfect record.

- 7. The majority of CERT/CC's ``most frequent, high-impact types of security incidents**

and vulnerabilities" only apply to Microsoft's products, and not to OSS/FS products.

Some security vulnerabilities are more important than others, for a variety of reasons. The CERT Coordination Center (CERT/CC) is federally funded to study security vulnerabilities and perform related activities such as publishing security alerts. I sampled their list of ["current activity" of the most frequent, high-impact security incidents and vulnerabilities on September 24, 2001](#), and found yet more evidence that Microsoft's products have poor security compared to others (including OSS/FS). Four of the six most important security vulnerabilities were specific to Microsoft: W32/Nimda, W32/Sircam, cache corruption on Microsoft DNS servers, and "Code Red" related activities. Only one of the six items primarily affected non-Microsoft products (a buffer overflow in telnetd); while this particular vulnerability is important, it's worth noting that many open source systems (such as Red Hat 7.1) normally don't enable this service (telnet) in the first place and thus are less likely to be vulnerable. The sixth item ("scans and probes") is a general note that there is a great deal of scanning and probing on the Internet, and that there are many potential vulnerabilities in all systems. Thus, 4 of 6 issues are high-impact vulnerabilities are specific to Microsoft, 1 of 6 are vulnerabilities primarily affecting Unix-like systems (including OSS/FS operating systems), and 1 of 6 is a general notice about scanning. Again, it's not that OSS/FS products never have security vulnerabilities - but they seem to have fewer of them.

8. **According to a Network Security evaluation, an OSS/FS vulnerability scanner (Nessus) was found to be the best (most effective).** On January 8, 2001, Network Computing's article [Vulnerability Assessment Scanners](#), reported an evaluation of 9 network scanning tools, most of them proprietary. The tools were Axent Technologies' NetRecon,BindView Corp.'s HackerShield, eEye Digital Security's Retina, Internet Security Systems' Internet Scanner, Network Associates' CyberCop Scanner, and two open-source products: Nessus Security Scanner and Security Administrator's Research Assistant (SARA). One product, World Wide Digital Security's System Analyst Integrated Network Tool (SAINT), is open source, with a commercial reporting tool.

In their evaluation, they set up systems with 17 of the most common and critical vulnerabilities, and evaluated how well each tool found the vulnerabilities. Sadly, not one product detected all vulnerabilities; the best scanner was the OSS/FS program Nessus Security Scanner, which found 15 of the 17 (which also received their top total score); the next best was Internet Security Systems' Internet Scanner, which found 13.5 out of 17.

In their words,

Some of us were a bit skeptical of the open-source Nessus project's thoroughness until [Nessus] discovered the greatest number of vulnerabilities. That's a hard fact to argue with, and we are now eating our words ... [Nessus] got the highest overall score simply because it did more things right than the other products.

I agree with the authors that ideally a vulnerability scanner should find every well-known vulnerability, and that "even one hole is too many." Still, perfection is rare in the real world. More importantly, a vulnerability scanner should only be part of the process to secure an organization - it shouldn't be the sole activity. Still, this evaluation suggests that an organization will be *more* secure, not less secure, by using an OSS/FS program. It could be argued that this simply shows that this particular OSS/FS program had more functionality - not more security - but in this case, the product's functionality was to improve security.

Now it should be obvious from these figures that OSS/FS systems are not magically invincible from security flaws. Indeed, some have argued that making the source code available gives attackers an advantage (because they have more information to make an attack). While OSS/FS gives attackers

more information, this thinking ignores an opposing force: having the source code makes it possible for defenders to examine their code and improve it. For a longer description of these issues, see [my discussion on open source and security](#). However, from these figures, it appears that OSS/FS systems are arguably *better* - not just equal - in their resistance to attacks.

Total Cost of Ownership (TCO)

Total cost of ownership (TCO) is an important measure; it doesn't matter if a product starts out cheaply if it costs you more down the line. However, TCO is extremely sensitive to the set of assumptions you make.

Indeed, whatever product you use or support, you can probably find a study to show it has the lowest TCO for some circumstance. Not surprisingly, both Microsoft and [Sun](#) provide studies showing that they have the lowest TCO (but see my comments later about Microsoft's study). [Xephon](#) has a study determining that Mainframes are the cheapest per-user (due to centralized control) at £3450 per user per year; Centralized Unix cost £7350 per user per year, and a decentralized PC environment costs £10850 per user per year. [Xephon](#) appears to be a mainframe-based consultancy, though, and would want the results to come out this way.

In short, what has a smaller TCO depends on your environment and needs. To determine TCO you have to identify all the important cost drivers (the "cost model") and estimate their costs. Don't forget "hidden" costs, such as administration costs, upgrade costs, technical support, end-user operation costs, and so on. However, OSS/FS has a number of strong cost advantages in various categories that, in many cases, will result in its having the smallest TCO.

1. **OSS/FS costs less to initially acquire.** OSS/FS costs much less to get initially. OSS/FS isn't really "free" in the monetary sense to get; the "free" in "free software" refers to freedom, not price (usually summarized as "free speech, not free beer"). You'll still spend money for paper documentation, support, training, system administration, and so on, just as you do with proprietary systems. Many distributions' executable programs can be acquired freely by downloading them ([linux.org provides some pointers on how to get distributions](#)). However, most people (especially beginners) will want to pay a small fee to a distributor for a nicely integrated package with CD-ROMs, paper documentation, and support. Even so, OSS/FS is far less expensive.

For example, look at some of the price differences when trying to configure a server (say a public web server or an intranet file and email server, in which you'd like to use C++ and an RDBMS for some portions of it). This is an example, of course; different missions would involve different components. I used the prices from "Global Computing Supplies" (Suwanee, GA), September 2000, and rounded to the nearest dollar. Here's a quick summary of some costs:

	Microsoft Windows 2000	Red Hat Linux
Operating System	\$1510 (25 client)	\$29 (standard), \$76 deluxe, \$156 professional (all unlimited)
Email Server	\$1300 (10 client)	included (unlimited)
RDBMS Server	\$2100 (10 CALs)	included (unlimited)
C++ Development	\$500	included

Basically, Microsoft Windows 2000 (25 client) costs \$1510; their email server Microsoft

Exchange (10-client access) costs \$1300, their RDBMS server SQL Server 2000 costs \$2100 (with 10 CALs), and their C++ development suite Visual C++ 6.0 costs \$500. Red Hat Linux 6.2 (a widely-used GNU/Linux distribution) costs \$29 for standard (90 days email-based installation support), \$76 for deluxe (above plus 30 days telephone installation support), or \$156 for professional (above plus SSL support for encrypting web traffic); in all cases it includes all of these functionalities (web server, email server, database server, C++, and much more). A public web server with Windows 2000 and an RDBMS might cost \$3610 (\$1510+\$2100) vs. Red Hat Linux's \$156, while an intranet server with Windows 2000 and an email server might cost \$2810 (\$1510+\$1300) vs. Red Hat Linux's \$76.

Both packages have functionality the other doesn't have. The GNU/Linux system always comes with an unlimited number of licenses; the number of clients you'll actually use depends on your requirements. However, this certainly shows that no matter what, Microsoft's server products cost thousands of dollars more per server than the equivalent GNU/Linux system. See [Jimmo's Cost Comparison](#) for another set of data points. Obviously, the price difference depends on exactly what functions you need for a given task.

- 2. Upgrade costs are typically far less.** Long-term upgrade costs are far less for OSS/FS systems. For example, upgrading a Microsoft system will typically cost around half the original purchase. What's worse, you are essentially at their mercy for long-term pricing, because there is only a single supplier (see [Microsoft Turns the Screws](#)). In contrast, the GNU/Linux systems can be downloaded (free), or simply re-purchased (generally for less than \$100), and the single upgrade be used on every system. This doesn't include technical support, but the technical support can be competed (a situation that's not practical for proprietary software). If you don't like your GNU/Linux supplier (e.g., they've become too costly), you can switch.
- 3. OSS/FS can often use older hardware more efficiently than proprietary systems, yielding smaller hardware costs and sometimes eliminating the need for new hardware.** OSS/FS runs faster on faster hardware, of course, but many OSS/FS programs can use older hardware more efficiently than proprietary systems, resulting in lower hardware costs - and in some cases requiring no new costs (because "discarded" systems can suddenly be used again). For example, the [minimum requirements for Microsoft Windows 2000 Server \(according to Microsoft\)](#) are a Pentium-compatible CPU (133 MHz or higher), 128 Mb of RAM minimum (with 256Mb the "recommended minimum"), and a 2 GB hard drive with at least 1.0 Gb free. According to Red Hat, Red Hat Linux 7.1 (a common distribution of GNU/Linux) requires at a minimum an i486 (Pentium-class recommended), 32Mb RAM (64Mb recommended), and 650Mb hard disk space (1.2 Gb recommended).

In Scientific American's August 2001 issue, the article [The Do-It-Yourself Supercomputer](#) discusses how the researchers built a powerful computing platform with a large number of obsolete, discarded computers and GNU/Linux. The result was dubbed the "Stone Soupercomputer"; by May 2001 it contained 133 nodes, with a theoretical peak performance of 1.2 gigaflops.

- 4. As the number of systems and hardware performance increases, this difference in initial and upgrade costs becomes even more substantial.** As the number of servers increases, proprietary solutions become ever more expensive. First, many proprietary systems (including Microsoft) sell per-client licenses; this means that even if your hardware can support more clients, you'll have to pay more to actually use the hardware you've purchased. Secondly, if you want to use more computers, you have to pay for more licenses in proprietary systems. In contrast, for most GNU/Linux distributions, you can install as many copies as you like for no additional fee, and there's no performance limit built into the

software. There may be a fee for additional support, but you can go to competing vendors for this support.

According to [Network World Fusion News](#), Linux is increasingly being used in healthcare, finance, banking, and retail because of its cost advantages when large numbers of identical sites and servers are built. According to their calculations for a 2,000 site deployment, SCO UnixWare would cost \$9 million, Windows would cost \$8 million, and Red Hat Linux costs \$180.

5. **There are many other factors; their effect varies on what you're trying to do.** There are many other factors in TCO, but it's difficult to categorize their effects in general, and it's generally difficult to find justifiable numbers for these other effects. Windows advocates claim that system administrators are cheaper and easier to find than Unix/Linux administrators, while GNU/Linux and Unix advocates argue that fewer such administrators are needed (because administration is easier to automate and the systems are more reliable to start with). Some GNU/Linux advocates have told me that GNU/Linux lends itself to hosting multiple services on a single server in cases where Windows installations must use multiple servers. License compliance administration can be costly for proprietary systems (e.g., time spent by staff to purchase CALS, keep track of licenses, and undergo audits) - a cost that simply isn't relevant to OSS/FS.
6. **For many circumstances, the total cost savings can be substantial; for example, savings exceeding \$250,000 per year were reported by 32% of the CTOs surveyed in a 2001 InfoWorld survey; 60% of these CTOs saved more than \$50,000 annually.** The August 27, 2001 InfoWorld (pages 49-50) reported on a survey of 40 CTOs who were members of the InfoWorld CTO network. In this survey, 32% using OSS reported savings greater than \$250,000; 12% reported savings between 100,001 and \$250,000; and 16% reported saving between \$50,001 and \$100,000. Indeed, only 8% reported annual savings less than \$10,000 (so 92% were saving \$10,000 or more annually). A chief benefit of OSS, according to 93% of the CTOs, was reduced cost of application development or acquisition; 72% said that a chief benefit was reduced development or implementation time (multiple answers were allowed). The CTOs reported using or planning to use OSS for web servers (65%), server operating systems (63%), web-application servers (45%), application development testing (45%), and desktop operating system (38%), among other uses. InfoWorld summarized it this way: ``in early 2000, it seemed as if no one was using open-source software for business-critical tasks... a vast majority of today's corporate IT executives are now using or plan to use OSS operating systems and web servers for their enterprise applications."

Microsoft's TCO study (mentioned earlier) is probably not useful as a starting point for estimating your own TCO. Their study reported the average TCO at sites using Microsoft products compared to the average TCO at sites using Sun systems, but although the Microsoft systems cost 37% less to own, the Solaris systems handled larger databases, more demanding applications connecting to those databases, 63% more concurrent connections, and 243% more hits per day. In other words, the Microsoft systems that did less work were less expensive. This is not a useful starting point if you're using TCO to help determine which system to buy -- to make a valid comparison by TCO, you need to compare the TCOs of systems that both perform the job *you* need to do. A two-part analysis by Thomas Pfau (see [part 1](#) and [part 2](#)) identifies this and many other flaws in the study.

Again, it's TCO that matters, not just certain cost categories. However, given these large differences, in many situations OSS/FS has a smaller TCO than proprietary systems. At one time it was claimed that OSS/FS installation took more time, but nowadays OSS/FS systems can be purchased pre-installed and automatic installers result in equivalent installation labor. Some claim that system administration costs are higher, but studies like Sun's suggest that in many cases the

system administration costs are lower, not higher, for Unix-like systems (at least Sun's). For example, on Unix-like systems it tends to be easier to automate tasks (because you can, but do not need, to use a GUI) - thus over time many manual tasks can be automated (reducing TCO). Retraining costs can be significant - but now that GNU/Linux has modern GUI desktop environments, there's anecdotal evidence that this cost is actually quite small (I've yet to see serious studies quantitatively evaluating this issue). In short, it's often hard to show that a proprietary solution's purported advantages really help offset their demonstrably larger costs in other categories when there's a competing mature OSS/FS product for the given function.

The paper [*Linux as a Replacement for Windows 2000*](#) is an example of an analysis comparing Red Hat Linux 7.1 to Windows 2000; in this customer's case, using Linux instead of Windows 2000 saved \$10,000. The reviewer came from a Windows/DOS background, and after performing an intensive hands-on Linux project lasting several months, determined that ``you will be stunned by the bang for the buck that ... open source software offers."

Does this mean that OSS/FS always have the lowest TCO? No! As I've repeatedly noted, it depends on its use. But the notion that OSS/FS *always* has the larger TCO is simply wrong.

Non-Quantitative Issues

In fairness, I must note that not all issues can be quantitatively measured, and to many they are the most important issues. These issues include freedom, protection from license litigation, and flexibility.

1. **OSS/FS protects its users from the risks and disadvantages of single source solutions.** While ``free software" advocates use the term ``freedom," and some businesses emphasize different terms such as ``multiple sources," the issue is the same: users do not want to be held hostage by any one vendor. Businesses often prefer to buy products in which there is a large set of competing suppliers, because it reduces their risk; they can always switch to another supplier if they're not satisfied or the original supplier goes out of business. This translates into an effect on the products themselves: if customers can easily choose and switch between competing products, the products' prices go down and their quality goes up. Conversely, if there is a near monopoly for a given product, over time the vendor will raise the cost to use the product and limit its uses to those that benefit the monopolist.

Historically, proprietary vendors eventually lose to vendors selling products available from multiple sources, even when their proprietary technology is (at the moment) better. Sony's Betamax format lost to VHS in the videotape market, and IBM's microchannel architecture lost to ISA in the PC architecture market, because customers prefer the reduced risk (and eventually reduced costs) of non-proprietary products. This is sometimes called ``commodification", a term disparaged by proprietary vendors and loved by users. Since users spend the money, users eventually find someone who will provide what they want.

With OSS/FS, users can choose between distributors, and if a supplier abandons them they can switch to another supplier. As a result, I believe OSS/FS tends to lower risks, lower costs, and higher quality products. Users can even band together and maintain the product themselves (this is how the Apache project was founded), making it possible for bands of users to protect themselves from abandonment.

2. **OSS/FS protects its users from licensing management and litigation.** Proprietary vendors make money from the sale of licenses, and are imposing increasingly complex mechanisms on consumers to manage these licenses (such as compulsory registration and an inability to

change hardware). They also litigate against those who don't comply with their licensing management requirements, creating increased legal risks for users. In contrast, OSS/FS users have no fear of litigation from the use and copying of OSS/FS. Licensing issues do come up when OSS/FS software is modified and then redistributed, but to be fair, proprietary software essentially forbids this action (so it's a completely new right). Even in this circumstance, redistributing modified OSS/FS software generally requires following a few simple rules (depending on the license), such as giving credit to others and releasing modifications under the same license.

3. **OSS/FS has greater flexibility.** OSS/FS users can tailor the product as necessary to meet their needs in ways not possible without source code. Users can tailor the product themselves, or hire whoever they believe can solve the problem (including the original developer). Some have claimed that this creates the "danger of forking," that is, of multiple incompatible versions of a product. This is "dangerous" only to those who believe competition is evil - we have multiple versions of cars as well. And in practice, the high cost of maintaining software yourself has resulted in a process in which the change is contributed back to the community. If it's not contributed (e.g., it solves a problem that needed solving but only for a particular situation), then it's still a win for the user - because it solved a user's problem which would not have been solved otherwise.

While I cannot quantitatively measure these issues, these are actually the most important issues to many.

Other Information

Here are some other related information sources:

1. Microsoft has been trying to claim that open source is somehow dangerous, and indeed is its leading critic, yet the Wall Street Journal's Lee Gomes found that "Microsoft Uses Open-Source Code Despite Denying Use of such Software." Here are some interesting quotes from his article:

... But Microsoft's statements Friday suggest the company has itself been taking advantage of the very technology it has insisted would bring dire consequences to others. "I am appalled at the way Microsoft bashes open source on the one hand, while depending on it for its business on the other," said Marshall Kirk McKusick, a leader of the FreeBSD development team.

More recently Microsoft has particularly targeted the GPL license, but "it hasn't previously suggested that there were benign forms of open-source software, and while singling out Linux for special criticism, has tended to criticize all open-source with the same broad brush." The article closes with this statement:

In its campaign against open-source, Microsoft has been unable to come up with examples of companies being harmed by it. One reason, said Eric von Hippel, a Massachusetts Institute of Technology professor who heads up a research effort in the field, is that virtually all the available evidence suggests that open source is "a huge advantage" to companies. "They are able to build on a common standard that is not owned by anyone," he said. "With Windows, Microsoft owns them."

2. Some have commented about this paper that "yes, but with OSS/FS you give up your right to sue if things go wrong." This paper is about numbers, so this is really outside its scope, but the obvious retort is that essentially all proprietary software licenses *also* forbid lawsuits - so

this isn't a difference at all! Also, see [`A Senior Microsoft Attorney Looks at Open-Source Licensing`](#), where Bryan Pfaffenberger argues that ``With open-source software, you don't need warranty protection... because you are, in principle, walking into the deal with your eyes wide open. You know what you're getting, and if you don't, you can find someone who does. Open-source licenses enable the community of users to inspect the code for flaws and to trade knowledge about such flaws, which they most assuredly do. Such licenses allow users to create derivative versions of the code that repair potentially hazardous problems the author couldn't foresee. They let users determine whether the program contains adequate safeguards against safety or security risks. In contrast, the wealthy software firms pushing UCITA are asking us to buy closed-source code that may well contain flaws, and even outright hazards attributable to corporate negligence -- but they won't let us see the code, let alone modify it. You don't know what you're getting."

3. Several studies examine developers (instead of the programs they write), including [`A Quantitative Profile of a Community of Open Source Linux Developers`](#), [Herman, Hertel and Niedner's study \(based on questionnaires\)](#), and the [Who Is Doing It \(WIDI\)](#) study.
4. There are several general information sites about OSS/FS or Unix that might be of interest, such as the [Free Software Foundation \(FSF\)](#), the [Open Source Initiative website](#), the [Linux.org site](#), and the [Unix versus NT site](#),
5. [Recommendations of the Panel on Open Source Software For High End Computing](#); this is the report of a panel created by the (U.S.) President's Information Technology Advisory Committee (PITAC). It recommends that the ``Federal government should encourage the development of open source software as an alternate path for software development for high end computing".
6. Large-scale roll-outs suggest that OSS/FS really is viable for enterprise deployments. Many retailer cash registers are switching to GNU/Linux, according to [Information Week](#); for example, Home Depot plans to roll out 90,000 terminals running Linux by 2003.
7. Several documents were written to counter Microsoft's statements in "[Linux Myths](#)". This included [LWN's response](#) and [Jamin Philip Gray's response](#), and the [FUD-counter site](#). The [shared source](#) page argues that Microsoft's ``shared source" idea is inferior to open source. The letter [Free Software Leaders Stand Together](#) argues against a number of statements by Craig Mundie.
8. [`NT Religious Wars: Why Are DARPA Researchers Afraid of Windows NT?`](#) found that, in spite of strong pressure by paying customers, computer science researchers strongly resisted basing research on Windows. Reasons given were: developers believe Windows is terrible, Windows really is terrible, Microsoft's highly restrictive non-disclosure agreements are at odds with researcher agendas, and there is no clear technology transition path for OS and network research products built on Windows (since only Microsoft can distribute changes to its products).
9. [`How Big Blue Fell For Linux`](#) is an article on how IBM transitioned to becoming a major backer. Now IBM plans to invest \$1 Billion in GNU/Linux, and it's just one company. See the [IBM annual report](#).
10. [`Open Source-onomics: Examining some pseudo-economic arguments about Open Source`](#) by Ganesh Prasad counters ``several myths about the economics of Open Source."
11. For a scientifically unworthy but really funny look at what people who *use* the various operating systems say, take a look at the [Operating System Sucks-Rules-O-Meter](#). It counts how many web pages make statements like ``Linux rocks". It's really just an opinion poll, but

if nothing else it's great for a laugh.

12. A general-purpose site that tries to compare all operating systems (with a bent towards Apple's MacOS) is <http://www.operatingsystems.net>; it has a lot of interesting information, though it tends towards testimonials and such instead of quantitative information.
13. The book *The Cathedral and the Bazaar* by Eric Raymond, available via the [Cathedral-Bazaar web site](#), examines OSS/FS development processes and issues.
14. Microsoft inadvertently advocated OSS/FS in its leaked internal documents, called the ["Halloween" documents](#).
15. Other evaluations include the [Gartner Group's](#) and [GNet's](#) evaluations.
16. For more general information on OSS/FS, see my [list of Open Source Software / Free Software \(OSS/FS\) references at http://www.dwheeler.com/oss_fs_refs.html](http://www.dwheeler.com/oss_fs_refs.html)

Conclusions

OSS/FS has significant market share, is often the most reliable software, and in many cases has the best performance. OSS/FS scales, both in problem size and project size. OSS/FS software generally has far better security, particularly when compared to Windows. Total cost of ownership for OSS/FS is often far less than proprietary software, particularly as the number of platforms increases. These statements are not merely opinions; these effects can be shown *quantitatively*, using a wide variety of measures. This doesn't even consider other issues that are hard to measure, such as freedom from control by a single source, freedom from licensing management (with its accompanying litigation), and increased flexibility. I believe OSS/FS options should be carefully considered any time software or computer hardware is needed.

You can view this page at http://www.dwheeler.com/oss_fs_why.html. Feel free to see my home page at <http://www.dwheeler.com>.

Open Source Software / Free Software (OSS/FS) References

This short paper gives links to important pages related to Open Source Software / Free Software (OSS/FS). This idea / movement / community has risen to prominence, and I've found it very interesting. The following are the links I've found most helpful to understanding it.

Definitions/Names

You can find definitions for the terms [open source software \(OSS, defined in the Open Source Definition\)](#) and [free software \(defined by the Free Software Foundation \(FSF\)\)](#). In practice, nearly all software meeting one definition also meets the other. In summary, OSS/FS programs are programs whose licenses permit users the freedom to run the program for any purpose, to modify the program, and to redistribute the modified program (without payment or restriction on who they can redistribute their program to).

The motives (or at least the emphasis) of the people who use the term ``open source" are sometimes different than those who use the term ``Free Software." The term ``open source software" (a term championed by Eric Raymond) is often used by people who wish to stress aspects such as high reliability and flexibility of the resulting program as the primary motivation for developing such software. In contrast, the term ``Free Software" (used in this way) stresses freedom from control by another (the standard explanation is ``think free speech, not free beer"). The FSF has a page written by its founder, Richard Stallman, on [why the FSF prefers the term ``free software" instead of ``open source software"](#). A [speech by Tony Stanco](#) describes some of the issues of free software; a good quote from it is that "[in cyberspace] software is the functional equivalent to law in real space, because it controls people, just like law does... [it is] much more obedient and therefore dangerous in the wrong hands." In contrast, Eric Raymond's Open Source Initiative declares that the term ``open source" is [``a marketing program for free software"](#) (and recommends using the term ``open source" instead).

In a similar manner, the most widely used OSS/FS operating system is referred to by two names: ``GNU/Linux" and simply ``Linux." ``GNU" is pronounced ``guh-new", and ``Linux" rhymes with ``cynics." Technically, the name ``Linux" is just the name of one system component (the ``kernel"), but often ``Linux" is used to mean the entire system. Richard Stallman has written an article on [why he believes GNU/Linux should be the preferred term when discussing the entire system](#), so those who identify themselves as part of the ``free software movement" tend to use ``GNU/Linux." The advantage of the term ``GNU/Linux" is that it properly gives credit to the organization most responsible for its development - the FSF's GNU project. An advantage of the term ``Linux" is that it's much easier to say. It's also worth noting that many other organizations besides GNU helped develop GNU/Linux. I'll use both terms here; unless noted otherwise, ``GNU/Linux" and ``Linux" both indicate an entire operating system (not just one component).

In fact, different people often have a range of motivations for working on different projects. Some people simply give away their software because they get changes back, resulting in a product better than what they could have produced alone. In a sense, they're getting the product that they want at less

cost than if they tried to develop it themselves. Others develop simply for the pleasure of developing software.

Don't confuse ``open source software" or ``free software" with ``non-commercial" software -- there are many examples of commercial open source / free software, and OSS/FS must be usable for commercial purposes. Antonyms of OSS/FS are ``closed" and ``proprietary" software.

OSS/FS Software Licenses

Essentially all of today's software is licensed; to be OSS/FS, its license has to follow certain rules, and there are a few common licenses that the vast majority of OSS/FS uses. The [Software Release Practice HOWTO](#) discusses briefly why license choices are so important to open source / free software projects:

The license you choose defines the social contract you wish to set up among your co-developers and users ...

Who counts as an author can be very complicated, especially for software that has been worked on by many hands. This is why licenses are important. By setting out the terms under which material can be used, they grant rights to the users that protect them from arbitrary actions by the copyright holders.

In proprietary software, the license terms are designed to protect the copyright. They're a way of granting a few rights to users while reserving as much legal territory is possible for the owner (the copyright holder). The copyright holder is very important, and the license logic so restrictive that the exact technicalities of the license terms are usually unimportant.

In open-source software, the situation is usually the exact opposite; the copyright exists to protect the license. The only rights the copyright holder always keeps are to enforce the license. Otherwise, only a few rights are reserved and most choices pass to the user. In particular, the copyright holder cannot change the terms on a copy you already have. Therefore, in open-source software the copyright holder is almost irrelevant -- but the license terms are very important.

Well-known and widely-used OSS/FS licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD licenses (BSD-old and BSD-new), and the Artistic license. The GPL and LGPL are termed ``copylefting" licenses, that is, these licenses are designed to prevent the code from becoming proprietary. See [Perens' paper](#) for more information comparing these licenses. The LGPL is intended for code libraries; it's quite similar to the GPL, but it permits proprietary programs to link to the library. The MIT and BSD-new licenses let anyone do almost anything with the code except sue the authors.

The most popular OSS/FS license by far is the GPL, and for many OSS/FS projects it's a good license. An OSS/FS program using a license without copyleft protection can be taken by a large company, extended, and made proprietary. Over time the proprietary version may have so many features (or be incompatible) that the original owner has to buy and become dependent on that other company for what was originally their work. A program licensed under the GPL or LGPL, which are copylefting licenses, has a much lower risk of this occurring. Many people writing libraries want proprietary programs to be able to call them, so for them the LGPL is a popular choice. Note that a GPL or LGPL

program can be used for commercial gain - you just can't distribute binaries without distributing their source code. Of course, if it's your desire that the program be used in proprietary programs, then a non-copylefting license is more appropriate (in that case, I'd suggest using the MIT license, which is simpler and clearer than the BSD licenses).

Most OSS/FS programmers shouldn't create their own licenses; creating a good license requires a good lawyer, and the probability of unintentional incompatibility is great. In summary, if you want to develop OSS/FS software, consider the GPL for applications, the LGPL for libraries (if you want proprietary applications to call it), and the MIT license if you want your code incorporated into others' proprietary code. In particular, it's unwise to create an OSS/FS project using a license incompatible with the GPL, because such a license bars code sharing with a vast amount of OSS/FS software. The LGPL, MIT, and BSD-new licenses are compatible with the GPL.

Descriptions/History

Here are some especially useful descriptions of open source/free software, including philosophical approaches, how it's used in practice, history, and so on:

- The [FSF website](#) and the [Open Source Initiative website](#) both contain a wealth of material.
- [The book *Open Sources: Voices from the Open Source Revolution*](#) (1st Edition January 1999), which is completely available on-line as well, contains a number of interesting articles. One especially relevant one is Bruce Perens' ["The Open Source Definition"](#). Other good articles include Eric Raymond's ["A Brief History of Hackerdom"](#).
- The book *The Cathedral and the Bazaar* by Eric Raymond. You can get information on [how to get the book](#), but most of the material is available online via the [Cathedral-Bazaar web site](#) or the [location of "Homesteading the Noosphere"](#).
- Microsoft inadvertently gave open source a boost when some of its internal documents were leaked, exposing that Microsoft was far more concerned than it claimed. This set of documents is termed the ["Halloween" documents](#).
- [The Jargon File](#) is useful because it provides insight into how open source/free software came to be and into the people involved. If nothing else, it clearly shows that this "new" phenomenon has a long history.
- One interesting document is a Master's dissertation on the subject: [Open Source Software as a new business model: The entry of Red Hat Software, Inc. on the operating system market with Linux](#) by Bojidar Mantarov, August 1999. This is a lot less well-known, but it provides useful commentary.
- It's possible to read a number of political approaches into OSS/FS; at one time some tried to claim that open source was essentially communistic. [Ganesh Prasad's *How Does the Capitalist View Open Source?*](#) shows that it also easily fits into a free market / capitalistic viewpoint.
- More recent history is covered by the [Linux Weekly News](#) timelines; here are their [1998](#), [1999](#), and [2000](#) timelines.
- Recently Microsoft has been attacking open source, and particularly the GNU General Public License (GPL) - the most widely-used of such software licenses. This prompted a letter by Bruce Perens and many other OSS/FS leaders titled [Free Software Leaders Stand Together](#).

This letter explains, for example, that ``the business model of Open Source is to reduce the cost of software development and maintenance by distributing it among many collaborators. The success of the Open Source model arises from copyright holders relaxing their control in exchange for more and better collaboration. Developers allow their software to be freely redistributed and modified, asking only for the same privileges in return."

Why use OSS/FS?

There are many good reasons to use OSS/FS, and there's actually quantitative data justifying some of its claims (such as higher reliability). In fact, there's too much data - I needed to separate that out into a separate page. [See my Why OSS/FS page for more information and quantitative evidence for OSS/FS.](#)

Major Projects

Major OSS/FS projects include the [Linux kernel](#), [Apache \(web server\)](#), [Samba \(supports interoperability with Windows clients by acting as a Windows file and print server\)](#), [GNOME \(a desktop environment\)](#), [KDE \(also a desktop environment\)](#), [The GIMP \(bitmapped image editor\)](#), [MySQL \(database emphasizing speed\)](#), [PostgreSQL \(database emphasizing functionality\)](#), [PHP \(hypertext preprocessor used for web development\)](#), [Mailman \(mailing list manager\)](#), [XFree86 \(graphics infrastructure which implements the X window system\)](#), [bind \(domain naming service, a critical Internet infrastructure service\)](#), [GNU Compiler Collection \(GCC, a suite of compilation tools for C, C++, and several other languages\)](#), [Perl \(programming/scripting language\)](#), [Python \(another programming/scripting language\)](#), the open source BSD Operating systems ([FreeBSD \(general purpose\)](#), [OpenBSD \(security-focused\)](#), [NetBSD \(portability-focused\)](#)), and the [Linux Documentation Project \(LDP\)](#).

A number of up and coming projects are at an alpha or beta level. Some projects that have the potential to be very important, have running code, and are working toward more functionality or stability include the following: [Wine \(a program to allow Windows programs to run on Linux/Unix\)](#), [Mozilla \(web browser, sponsored by AOL/Netscape\)](#), [AbiWord \(a word processor\)](#), [Gnumeric \(spreadsheet\)](#), [KOffice \(office suite\)](#), and [GnuCash \(money management\)](#).

Web projects around the world often use ``LAMP", an abbreviation for Linux, Apache, MySQL (sometimes replaced with PostgreSQL), and PHP/Perl/Python. More complex web projects may use major libraries or frameworks, such as PHP-Nuke (based on PHP) and Zope (based on Python).

There isn't really one place to find ``all about GNU/Linux"; you could do worse than looking at the [linux.org information](#).

For information on other packages, you could go to places like [Freshmeat](#) (which lists new software available for GNU/Linux and other systems) and the [FSF list of free software](#).

GNU/Linux Distributions

Few people will want to do all the packaging work for entire operating systems based on GNU/Linux. Thus, "Linux distributors" sprung up, who do that work and sell support, extra services, and so on. Major distributions of GNU/Linux include: [Red Hat](#) (#1 in the U.S. by most measures), [Debian](#) (#1 non-commercial distribution), [SuSE](#) (a major force in Europe), [Caldera](#) (significant for old Novell and SCO users), [TurboLinux](#) (significant in Asia), and [Mandrake](#); there are many others.

It's tricky to figure out the market share of Linux distributions. An IDC study of copies of Linux sold in 1999 ([Red Hat holds huge Linux lead, rivals growing](#)) found that Red Hat shipped 48%, SuSE sold 15%, Caldera Systems and TurboLinux each sold 10%, Mandrakesoft had 4%, Corel 1%, and the others 11%. However, while the market grew 89%, Red Hat's share grew only 69%, suggesting that Red Hat will have more competition ahead.

The fundamental failure of these numbers is that they only count sales. Debian is not usually "sold" in the traditional manner. A copy of Linux can in many cases be downloaded for free (see [LinuxISO.org](#) for one source), and/or installed on as many computers as you wish. Thus, this measure is useful to show that Red Hat is widely used, but it's less useful in showing true market shares.

Community/News

Open source / free software is a community and culture, not just an idea. Thus, you can get news and cultural information from some of the following:

- [Linux Weekly News \(LWN\)](#), in my opinion one of the best news sources; it comes out every Thursday, but you can also view it [daily](#). Other useful news sources include [Linux World](#), [Linux Today](#) (which tries to link to every Linux-related article on other sites), and [Linux News](#).
- The best source for news specific to the Linux kernel is [Kernel Traffic](#), which provides a weekly summary of the linux-kernel mailing list (see the [linux-kernel mailing list FAQ](#) for general information about this). Other useful sites about the Linux kernel are the [Linux Kernel Archives](#) (for Linux kernel source code) and [Kernelnotes](#) (which lists many sites related the Linux kernel).
- Other useful sites include [Linux Headquarters](#).
- [Slashdot](#). It's eclectic, but many scoops get here first.
- For a collection of various rants, look at [OS Opinion](#).
- Commentary on the community and culture is also captured by comic strips, such as [User Friendly](#), [After Y2k](#), [The Widget Box by Harry Martin](#) (especially the classic one on [Penguin Economics](#)), and [TUX: Term Unit X](#). While it's not specific to OSS/FS, a widely-read comic is [Dilbert](#).

Miscellaneous Related Sites

- [SourceForge](#) provides free hosting for OSS projects, and hosts a vast number of them.
- An interesting site is [Advogato](#), which provides free personal sites and uses an experimental "group trust" metric to rank people.
- To sort through the vast number of options for real-time Linux, see [The Real-time Linux Quick Reference Guide](#).
- [The Rise of "Worse is Better"](#) by Richard Gabriel describes an approach often used by open source advocates. Basically, emphasizing simplicity of both design and implementation tends to produce software that's available first (acquiring the market) and is also more flexible (because there's less to change when requirements change).
- [Open Source as ESS](#) by David Rysdam, which applies game theory concepts to software licenses and argues that the GPL fundamentally "wins" over other licenses.
- Evan Leibovitch's opinion piece [License to FUD \(comparing GPL and BSD\)](#) examines the GPL and BSD licensing approaches. He argues that many of the new and most actively developed open-source projects use the GPL license (instead of the BSD or MIT licenses) because of various fears. Programmers fear that their work would be used in a manner they did not support, and companies fear that their work would be used by competitors against them. He argues that the BSD (MIT) approach works best in scenarios such as being reference implementations, but for most other uses, GPL or full proprietary is preferred in the not-so-kind computing environment of today.
- [League for Programming Freedom](#), which opposes the application of patent law to software; there are related sites such as the [Burn All GIFs](#) and [BountyQuest](#) site.
- Various "miscellaneous" vendors include [LinuxTshirts.com](#) and [Linuxmall.com](#).
- There are a number of just plain interesting articles; [here's one contrasting OSS/FS development approaches](#).
- There are many discussions on the legal issues, including [Dan Ravicher's interview](#).
- [Google](#) is a really good web search engine built on Linux (Google has [deployed over 4,000 Linux servers to implement their system](#)). Google provides specialized searches for [Linux](#) and [BSD](#) information.
- [Thom Wyszong](#) has written a nontechnical introduction to Open Source and Free Software.
- If you have problems or questions, there's a right and wrong way to ask them, and only the right way will get a helpful response. To learn the right way, consult [Eric Raymond and Rick Moen's paper on "How to ask smart questions"](#).

You can view this page at http://www.dwheeler.com/oss_fs_refs.html. Feel free to see my home page at <http://www.dwheeler.com>.

David A. Wheeler's Home Page

This is the home page for David A. Wheeler's [open source](#) / [free software](#) projects. Currently, the following projects, papers, and briefings are available here:

Picture of David A. Wheeler

- Security

- [Secure Programming for Linux and Unix HOWTO](#), a set of design and implementation guidelines for developing secure applications on Linux and Unix (GFDL).
- [flawfinder](#), a source code scanner that looks for potential security flaws (GPL).
- [browse](#), a secure version of the BROWSER convention with supporting tools and documentation (the code is MIT/X).
- [Java Security \(a tutorial\)](#) (GPL).

- Open Source / Free Software Information

- [Why Open Source Software / Free Software \(OSS/FS\)? Look at the Numbers!](#), lists information I've found, emphasizing *quantitative evidence* that OSS/FS has value.
- [Open Source Software \(OSS\) / Free Software \(FS\) References](#), which lists important references about OSS/FS.

- Java

- [Java Implementations \(with David K. Friedman\)](#); this paper identifies a number of Java implementations and their properties

- [Java Security \(a tutorial\)](#) (GPL).
- Ada
 - [AdaCGI](#), a library for building web applications using Ada95 and the CGI interface (LGPL).
 - [vim Ada mode](#), a mode for editing Ada code using vim.
- SLOC
 - [Counting Source Lines of Code \(SLOC\)](#), which reports the size of a Linux distribution.
 - [SLOCCount](#), a suite of programs for measuring SLOC (GPL).
- Other Documents I maintain
 - [Software Innovations](#), examining the history of innovation in software.
 - [Linux Program Library HOWTO](#) (GPL). This describes how to create and use program libraries on Linux.
 - [Uri\(7\), documentation on URI/URLs for Linux](#)
- Miscellaneous
 - [ChessClub](#), a set of Perl/sh scripts for managing a Chess Club, permitting web-based submission of chess games with automated annotations (GPL).
 - [mm2frame](#), a program that translates documents using troff/MM macros into Adobe Framemaker (GPL).

- [Miscellaneous \(e.g., docbook to LaTeX translator\)](#)
- [Other Essays](#)

You can also see other works I've developed (at least in part) such as my [Lovelace Ada95 Tutorial](#) (mostly GPL). Smaller works include the [Linux man pages](#) for uri(7) and the rewrite of man(7) and mdoc(7).

For the insatiably curious, [more information about me is also available](#), as well as a interviews of me by [LinuxSecurity.com](#) and [SearchEnterpriseLinux.com](#).

You can contact David A. Wheeler at dwheeler@dwheeler.com.

You are viewing <http://www.dwheeler.com>.

This site is hosted by [Webframe.org](#).

[Secure your Network](#)
[Advertise on Netcraft](#)
[About Netcraft](#)
[Join Netcraft](#)
[Site Map](#)

Netcraft Home

















☐ [What's that site running?](#)
[Web Server Survey](#)
[SSL Server Survey](#)
☐ [Explore web sites](#)
[News](#)



















<input type="checkbox"/> What's that site running ?	Uptime FAQ Top Hosting Locations Longest Uptimes Most Requested Sites

















	Thawte is the Apache Certificate Get our Free Apache Security Guide [Click here!]
---	---

What's that site running?
Example: www.netcraft.com
What's that SSL site running?

Sites with longest running systems by average uptime in the last 90 days

Note: Uptime - the time since last reboot is explained in the FAQ								Generated on 3-Aug-2001
Rank	Site	No. samples	Average	Max	Latest	OS	Server	Netblock Owner
1	www.sre.nl (16 sites)	59 	1244	1272	1272	FreeBSD	Apache/1.3.14 (Unix) ApacheJServ/1.1.2	Internet Access Eindhoven (IAE)
2	www.transactie.org (2 sites)	20 	1130	1142	-	FreeBSD	Apache/1.3.14 (Unix) ApacheJServ/1.1.2	Internet Access Eindhoven (IAE)
3	w11.dion.ne.jp	38 	1080	1118	1118	BSD/OS	Apache/1.1.3 BSDI/3.0	DION (DDI CORPORATION)
4	wwwprod1.telia.com	63 	1005	1048	1049	BSD/OS	Apache/1.3.0 (Unix) PHP/3.0.1	TeliaNet
5	www.fks.bt (2 sites)	120 	1003	1056	-	FreeBSD	Apache/1.3.0 (Unix)	Verio, Inc.
6	www.ees.com	81 	1001	1052	-	FreeBSD	Apache/1.3.20 (Unix) PHP/4.0.4p11	Luce McQuillin Corporation
7	bm98.cup.com (5 sites)	133 	970	1017	1018	FreeBSD	Apache/1.3.0 (Unix)	Hopemoon Internet
8	www.yamagata-cci.or.jp	156 	879	922	923	FreeBSD	Apache/1.3.0 (Unix)	Hopemoon Internet
9	www.bizbase.com	53 	852	902	902	BSD/OS	Microsoft-IIS/4.0	Chase Manhattan Bank
10	www.superior.net (3 sites)	119 	813	859	859	FreeBSD	Apache/1.3.12 (Unix)	BiznessOnline
11	www.daiko-lab.co.jp	47 	812	853	854	FreeBSD	Apache/1.2.4	Daiko Corporation
12	www.aomori-jc.or.jp (3 sites)	102 	791	838	839	FreeBSD	Apache/1.3.3 (Unix)	World Communications
13	www.fat.net (2 sites)	55 	774	819	820	FreeBSD	Apache/1.2.1	Internet America
14	www.imusic.org	109 	772	819	820	BSD/OS	unknown	AboveNet Communications Inc.
15	www.meix-net.or.jp	26 	767	808	809	FreeBSD	Apache/1.3.6 (Unix)	meix corporation
16	prosecutor.co.essex.nj.us	18 	761	768	768	BSD/OS	Apache/1.2.5 FrontPage/3.0.4	Verio, Inc.

17	club21.org	29 	759	797	797	FreeBSD	Apache/1.3.4 (Unix)	Hopemoon Internet
18	osb.sra.co.jp	56 	757	795	795	FreeBSD	Apache/1.3.12 (Unix) ApacheJServ/1.1 PHP/3.0.12-i18n-beta4	Software Research Associates, Inc.
19	shops.ne.jp (2 sites)	55 	757	795	796	FreeBSD	Apache/1.3.4 (Unix)	Hopemoon Internet
20	www.jpix.ad.jp (2 sites)	137 	745	793	794	FreeBSD	Apache/1.3.14 (Unix)	Japan Internet Exchange Co., Ltd.
21	www.crammed.be (2 sites)	67 	745	773	773	IRIX	Apache/1.3.12 (Unix)	ImagiNet - Brussels
22	www.nirvanet.net (2 sites)	118 	744	771	771	IRIX	Apache/1.3.12 (Unix)	ImagiNet - Brussels
23	www.reprocessing.com (4 sites)	64 	743	772	772	IRIX	Apache/1.3.12 (Unix)	ImagiNet - Brussels
24	www.phoenix-united.com (2 sites)	30 	743	764	764	Linux	Apache-AdvancedExtranetServer/1.3.14 (Linux-Mandrake/2mdk) mod_ssl/2.7.1 OpenSSL/0.9.5a PHP/4.0.4pl1	Apache Network is based in France
25	www.francoise-hardy.com	51 	735	754	-	Linux	Apache-AdvancedExtranetServer/1.3.14 (Linux-Mandrake/2mdk) mod_ssl/2.7.1 OpenSSL/0.9.5a PHP/4.0.4pl1	Apache Network is based in France
26	www.etiennedaho.com	90 	733	755	-	Linux	Apache-AdvancedExtranetServer/1.3.14 (Linux-Mandrake/2mdk) mod_ssl/2.7.1 OpenSSL/0.9.5a PHP/4.0.4pl1	Apache Network is based in France
27	www.orangesoft.co.jp	67 	730	776	777	BSD/OS	Apache/1.3.6 (Unix)	JENS Corporation
28	www.crbkenya.com (3 sites)	48 	717	765	766	BSD/OS	Apache/1.2.5 FrontPage/3.0.4	Verio, Inc.
29	01net-mailfriend.com (2 sites)	63 	700	746	747	FreeBSD	Apache/1.3.1 (Unix)	SAKURA Internet Inc.
30	adultshop.co.jp (3 sites)	43 	688	689	690	FreeBSD	Apache/1.3.6 (Unix) PHP/3.0.7	Kabusikikaisya Bestplanning
31	www.port.city.kobe.jp	63 	687	730	731	FreeBSD	Oracle_Web_listener3.0.1/3.0.1.0.0	CITY OF KOBE
32	www.eisai.co.jp	64 	681	727	728	BSD/OS	Apache/1.3.6 (Unix)	Japan Network Information Center
33	www.promise.co.jp	105 	680	729	730	BSD/OS	Apache/1.3.6 (Unix)	Japan Network Information Center
34	www.geysers.net (3 sites)	47 	674	719	720	BSD/OS	Apache/1.2.4 (ntx enhanced server - referer/agent 1.0d6)	Verio, Inc.

35	mail.eofficeplanet.com (9 sites)	67 	672	719	720	FreeBSD	Apache/1.3.6 (Unix) mod_perl/1.20	Critical Path, Inc.: Munich, Germany datacenter
36	www.ampr.org	108 	665	712	-	BSD/OS	Apache/1.2.6	University of California, San Diego
37	www.tfactory.com	6 	663	663	-	NetBSD/OpenBSD	Apache	Mediadeck Projektkoordination GmbH
38	search-cache.ebay.com (3 sites)	150 	652	699	700	BSD/OS	Apache/1.3.6 (Unix)	Exodus Communications
39	search-cache.ebay.compuserve.com	108 	652	700	701	BSD/OS	Apache/1.3.6 (Unix)	Exodus Communications
40	channel.web.aol.com	72 	651	698	699	IRIX	NaviServer/2.0 AOLserver/2.3.3	America Online
41	search-cache.ebay.de	97 	649	697	698	BSD/OS	Apache/1.3.6 (Unix)	Exodus Communications
42	is.yourpimp.com (3 sites)	80 	645	677	678	BSD/OS	Apache/1.3.19 (Unix)	Level 3 Communications, LLC
43	www.min.net (2 sites)	120 	645	692	693	BSD/OS	Apache/1.3.3 (Unix) FrontPage/4.0.4.3	MetroNet Internet Services, LLC
44	www.sapporobank.co.jp (8 sites)	56 	636	673	674	BSD/OS	Apache/1.3.6 (Unix)	Hitachi, Ltd
45	www.mannainternational.com	5 	632	634	-	BSD/OS	Apache/1.2.5 FrontPage/3.0.4	Verio, Inc.
46	www.hadano-cci.or.jp (2 sites)	59 	631	674	675	BSD/OS	Apache/1.3.6 (Unix)	Hitachi, Ltd
47	www.nomura-am.co.jp	13 	626	631	632	BSD/OS	Apache/1.3.6 (Unix)	Japan Network Information Center
48	www.marketingbridge.com	42 	624	667	668	FreeBSD	Apache/1.3.1 (Unix)	Marketing Bridge
49	labyrinth.ne.jp	2 	617	617	-	FreeBSD	Apache/1.3.14 (Unix) PHP/3.0.18-i18n-ja-2	E-Mail. Inc.
50	www.itoyokado.iyg.co.jp	267 	615	660	661	BSD/OS	Apache/1.3.9 (Unix)	Japan Network Information Center

The table shows the top sites by average and peak times since reboot, together with the most recently reported operating system, web server and netblock owner.

For performance reasons, we limit this monitoring process to the most frequently requested sites.

Search Netcraft	What's that site running?	Get your own site noticed, with a direct link to Netcraft
	Example: www.netcraft.com	<div></div> Just cut and paste from here

[What's that site running ?](#)

[Uptime FAQ](#)

[Top Hosting Locations](#)

[Longest Uptimes](#)

[Most Requested Sites](#)

Counting Source Lines of Code (SLOC)

[Click here to get the paper, ``More than a Gigabuck: Estimating GNU/Linux's Size,''](#)
[which presents the latest GNU/Linux size estimates, approach, and analysis.](#)

Picture of David A. Wheeler

My latest size-estimation paper is [More than a Gigabuck: Estimating GNU/Linux's Size](#) (June 2001). Here are a few interesting facts quoting from the paper (which measures Red Hat Linux 7.1):

1. It would cost **over \$1 billion (a Gigabuck) to develop this Linux distribution by conventional proprietary means** in the U.S. (in year 2000 U.S. dollars).
2. It includes **over 30 million physical source lines of code (SLOC)**.
3. It would have required about **8,000 person-years** of development time.
4. Red Hat Linux 7.1 represents over a **60% increase in size, effort, and traditional development costs** over Red Hat Linux 6.2 (which was released about one year earlier).

Many other interesting statistics emerge; here are a few:

- The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), the X window system (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system).
- The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed.
- The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part or as an alternative, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is public domain.

You can get:

1. [``More than a Gigabuck: Estimating GNU/Linux's Size"](#), my latest SLOC analysis paper which analyzes Red Hat Linux 7.1. You can also get some of the supporting information (intended for those who want to do further analysis), such as the [complete summary](#), [summary SLOC](#)

[analysis of the Linux 2.4 kernel](#), [map of build directories to RPM spec files](#), [spec summaries](#), [counts of files](#), and [detailed file-by-file SLOC counts](#). You can also get [version 1.0](#), [version 1.01](#), [version 1.02](#), or [version 1.03](#) of the paper.

2. ["Estimating Linux's Size,"](#) the previous paper which analyzes Red Hat Linux 6.2. Various background files and previous editions are also available. You can see the [ChangeLog](#), along with older versions of the paper ([original paper \(version 1.0\)](#), [version 1.01](#), [version 1.02](#) and [version 1.03](#)). [version 1.04](#)). You can also see some of the summary data: [SLOC sorted by size](#), [filecounts](#), [unsorted SLOC counts](#), [unsorted SLOC counts with long lines](#), and [SLOC counts formatted for computer processing \(tab-separated data\)](#). For license information, you can see [the licenses allocated to each build directory](#). If you want to know what a particular package does, you can find out briefly by looking at the [package \(specification file\) descriptions](#).

When referring to this information, please refer to the URL <http://www.dwheeler.com/sloc>. Some of the other URLs may change, and I may add more measurements later.

If you want to get the tools I used, they're available. I call the set SLOCCount, and you can get SLOCCount at <http://www.dwheeler.com/sloccount>.

You can also view [my home page \(http://www.dwheeler.com\)](http://www.dwheeler.com), or related pages such as my pages on [open source software / free software references](#) and [how to write secure programs](#).

This site is hosted by [Webframe.org](http://www.webframe.org).

More Than a Gigabuck: Estimating GNU/Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

June 30, 2001

Version 1.04

This paper analyzes the amount of source code in GNU/Linux, using Red Hat Linux 7.1 as a representative GNU/Linux distribution, and presents what I believe are interesting results.

In particular, it would cost over \$1 billion (\$1,000 million - a Gigabuck) to develop this GNU/Linux distribution by conventional proprietary means in the U.S. (in year 2000 U.S. dollars). Compare this to the \$600 million estimate for Red Hat Linux version 6.2 (which had been released about one year earlier). Also, Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2. Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2). Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is due to an increased number of mature and maturing open source / free software programs available worldwide.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), the X Window system (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part or as an alternative, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this GNU/Linux distribution. In contrast, only 0.2% of the software is public domain.

This paper is an update of my previous paper on estimating GNU/Linux's size, which measured Red Hat Linux 6.2 [Wheeler 2001]. Since Red Hat Linux 6.2 was released in March 2000, and Red Hat Linux 7.1 was released in April 2001, this paper shows what's changed over approximately one year. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The GNU/Linux operating system has gone from an unknown to a powerful market force. One survey found that more Internet servers use GNU/Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were GNU/Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because GNU/Linux can be obtained at no or low cost. For example, experiments suggest that GNU/Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and GNU/Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a ``typical'' intranet load, using the same load and request set the GNU/Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for GNU/Linux's popularity among many developers and users is that its source code is generally ``open source software'' and/or ``free software''. A program that is ``open source software'' or ``free software'' is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of ``open source software'' is available from the Open Source Initiative [OSI 1999], a more formal definition of ``free software'' (as the term is used in this paper) is available from the Free Software Foundation [FSF 2000], and other general information about these topics is available at Wheeler [2000a].

Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The GNU/Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is ``open source software''/``free software'', and this is also true for all (or nearly all) other components of a typical GNU/Linux distribution. Open source software/free software frees users from being captives of

a particular vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze GNU/Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a GNU/Linux distribution. Microsoft unintentionally published some analysis data in the documents usually called "Halloween I" and "Halloween II" [[Halloween I](#)] [[Halloween II](#)]. Another study focused on the Linux kernel and its growth over time is by [Godfrey \[2000\]](#); this is an interesting study but it focuses solely on the Linux kernel (not the entire operating system). In a previous paper, I examined Red Hat Linux 6.2 and the numbers from the Halloween papers [[Wheeler 2001](#)].

This paper updates my previous paper, showing estimates of the size of one of today's GNU/Linux distributions, and it estimates how much it would cost to rebuild this typical GNU/Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean. I have intentionally written this paper so that you do *not* need to read the previous version of this paper first.

For my purposes, I have selected as my "representative" GNU/Linux distribution Red Hat Linux version 7.1. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [[Shankland 2000b](#)]. Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE (a German distributor) at 15%. Not all GNU/Linux copies are "sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on, or were originally developed from, a version of Red Hat Linux. This doesn't mean the other distributions are less capable, but it suggests that these other distributions are likely to have a similar set of components.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size for the same kind of functionality.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate "all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 7.1 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Note that Red Hat Linux 6.2 was released on March 2000, Red Hat Linux 7 was released on September 2000 (I have not counted its code), and Red Hat Linux 7.1 was released on April 2001. Thus, the differences between Red Hat Linux 7.1 and 6.2 show differences accrued over 13 months (approximately one year).

Clearly there is far more open source / free software available worldwide than is counted in this paper. However, the job of a distributor is to examine these various options and select software that they believe is both sufficiently mature and useful to their target market. Thus, examining a particular distribution results in a selective analysis of such software.

Section 2 briefly describes the approach used to estimate the "size" of this distribution (more details are in Appendix A). Section 3 discusses some of the results. Section 4 presents conclusions, followed by an appendix. GNU/Linux is often called simply "Linux", but technically Linux is only the name of the operating system kernel; to eliminate ambiguity this paper uses the term "GNU/Linux" as the general name for the whole system and "Linux kernel" for just this inner kernel.

2. Approach

My basic approach was to:

1. install the source code files in uncompressed format; this requires carefully selecting the source code to be analyzed.
2. count the number of source lines of code (SLOC); this requires a careful definition of SLOC.
3. use an estimation model to estimate the effort and cost of developing the same system in a proprietary manner; this requires an estimation model.
4. determine the software licenses of each component and develop statistics based on these categories.

More detail on this approach is described in Appendix A. A few summary points are worth mentioning here, however.

2.1 Selecting Source Code

I included all software provided in the Red Hat distribution, but note that Red Hat no longer includes software packages that only apply to other CPU architectures (and thus packages not applying to the x86 family were excluded). I did not include ``old" versions of software, or ``beta" software where non-beta was available. I did include ``beta" software where there was no alternative, because some developers don't remove the ``beta" label even when it's widely used and perceived to be reliable.

I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once (as a tie-breaker, such files are assigned to the first build package it applies to in alphabetic order).

The code in makefiles and Red Hat Package Manager (RPM) specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

Since different languages have different syntaxes, I could only measure the SLOC for the languages that my tool (sloccount) could detect and handle. The languages sloccount could detect and handle are Ada, Assembly, awk, Bourne shell and variants, C, C++, C shell, Expect, Fortran, Java, lex/flex, LISP/Scheme, Makefile, Objective-C, Pascal, Perl, Python, sed, SQL, TCL, and Yacc/bison. Other languages are not counted; these include XUL (used in Mozilla), Javascript (also in Mozilla), PHP, and Objective Caml (an OO dialect of ML). Also code embedded in data is not counted (e.g., code embedded in HTML files). Some systems use their own built-in languages; in general code in these languages is not counted.

2.2 Defining SLOC

The ``physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: ``a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the ``logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the ``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

Note that this required that every file be categorized by language type (so that the correct syntax for comments, strings, and so on could be applied). Also, automatically generated files had to be detected and ignored. Thankfully, my tool ``sloccount" does this automatically.

2.3 Estimation Models

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer ``COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions. Of particular note, basic COCOMO does not include the time to develop translations to other human languages (of documentation, data, and program messages) nor fonts.

There is reason to believe that these models, while imperfect, are still valid for estimating effort in open source / free software projects. Although many open source programs don't need management of human resources, they still require technical management, infrastructure maintenance, and so on. Design documentation is captured less formally in open source projects, but it's often captured by necessity because open source projects tend to have many developers separated geographically. Clearly, the systems must still be programmed. Testing is still done, although as with many of today's proprietary programs, a good deal of testing is done through alpha and beta releases. In addition, quality is enhanced in many open source projects through peer review of submitted code. The estimates may be lower than the actual values because they don't include estimates of human language translations and fonts.

Each software source code package, once uncompressed, produced zero or more "build directories" of source code. Some packages do not actually contain source code (e.g., they only contain configuration information), and some packages are collections of multiple separate pieces (each in different build directories), but in most cases each package uncompresses into a single build directory containing the source code for that package. Each build directory had its effort estimation computed separately; the efforts of each were then totalled. This approach assumes that each build directory was developed essentially separately from the others, which in nearly all cases is quite accurate. This approach slightly underestimates the actual effort in the rare cases where the development of the code in separate build directories are actually highly interrelated; this effect is not expected to invalidate the overall results.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.81 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired. These are the same values as used in my last report.

2.4 Determining Software Licenses

A software license determines how that software can be used and reused, and open source software licensing has been a subject of great debate. The Software Release Practice HOWTO [\[Raymond 2001\]](#) discusses briefly why license choices are so important to open source / free software projects:

The license you choose defines the social contract you wish to set up among your co-developers and users ...

Who counts as an author can be very complicated, especially for software that has been worked on by many hands. This is why licenses are important. By setting out the terms under which material can be used, they grant rights to the users that protect them from arbitrary actions by the copyright holders.

In proprietary software, the license terms are designed to protect the copyright. They're a way of granting a few rights to users while reserving as much legal territory is possible for the owner (the copyright holder). The copyright holder is very important, and the license logic so restrictive that the exact technicalities of the license terms are usually unimportant.

In open-source software, the situation is usually the exact opposite; the copyright exists to protect the license. The only rights the copyright holder always keeps are to enforce the license. Otherwise, only a few rights are reserved and most choices pass to the user. In particular, the copyright holder cannot change the terms on a copy you already have. Therefore, in open-source software the copyright holder is almost irrelevant -- but the license terms are very important.

Well-known open source licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD license, and the Artistic license. The GPL and LGPL are termed "copylefting" licenses, that is, the license is designed to prevent the code from becoming proprietary. See [Perens \[1999\]](#) for more information comparing these licenses. Obvious questions include "what license(s) are developers choosing when they release their software" and "how much code has been released under the various licenses?"

An approximation of the amount of software using various licenses can be found for this particular distribution. Red Hat Linux uses the Red Hat Package Manager (RPM), and RPM supports capturing license data for each package (these are the "Copyright" and "License" fields in the specification file). I used this information to determine how much code was covered by each license. Since this field is simply a string of text, there were some variances in the data that I had to clean up, for example, some entries said "GNU" while most said "GPL". In some cases Red Hat did not include licensing information

with a package. In that case, I wrote a program to attempt to determine the license by looking for certain conventional filenames and contents.

This is an imperfect approach. Some packages contain different pieces of code with difference licenses applying to different pieces. Some packages are ``dual licensed'', that is, they are released under more than one license. Sometimes these other licenses are noted, while at other times they aren't. There are actually two BSD licenses (the ``old'' and ``new'' licenses), but the specification files don't distinguish between them. Also, if the license wasn't one of a small set of common licenses, Red Hat tended to assigned nondescriptive phrases such as ``distributable''. My automated techniques were limited too, in particular, while some licenses (e.g., the GPL and LGPL) are easy to recognize automatically, BSD-like and MIT-like licenses vary the license text and so are more difficult to recognize automatically (and some changes to the license would render them non-open source, non-free software). Thus, when Red Hat did not identify a package's license, a program dual licensed under both the BSD and GPL license might only be labelled as having the GPL using these techniques. Nevertheless, this approach is sufficient to give some insight into the amount of software using various licenses. Future research could examine each license in turn and categorize them; such research might require several lawyers to determine when two licenses in certain circumstances are ``equal.''

One program worth mentioning in this context is Python, which has had several different licenses. Version 1.6 and later (through 2.1) had more complex licenses that the Free Software Foundation (FSF) believes were incompatible with the GPL. Recently this was resolved by another change to the Python license to make Python fully compatible with the GPL. Red Hat Linux 7.1 includes an older version of Python (1.5.2), presumably because of these licensing issues. It can't be because Red Hat is unaware of later versions of Python; Red Hat uses Python in its installation program (which it developed and maintains). Hopefully, the recent resolution of license incompatibilities with the GPL license will enable Red Hat to include the latest versions of Python in the future. In any case, there are several different Python-specific licenses, all of which can legitimately be called the ``Python'' license. Red Hat has labelled Python itself as having a ``Distributable'' license, and package Distutils-1.0.1 is labelled with the ``Python'' license; these labels are kept in this paper.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (sorted by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total counts grouped by their software licenses, section 3.6 presents total SLOC counts, and section 3.7 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 35 largest components (as measured by number of source lines of code), along with their licenses (see section 2.4 for how these license values were determined). In the language section, ``ansic'' means C code, ``asm'' is assembly, ``sh'' is Bourne shell and related shells, and ``cpp'' is C++.

SLOC	Directory	SLOC-by-Language (Sorted)
2437470	kernel-2.4.2	ansic=2285657,asm=144411,sh=3035,perl=2022,yacc=1147,tcl=576,lex=302,awk=248,sed=72 [GPL]
2065224	mozilla	cpp=1279902,ansic=739470,perl=21220,sh=13717,asm=5212,java=3107,yacc=1831,lex=470,csh=271,sed=24 [MPL]
1837608	XFree86-4.0.3	ansic=1750460,asm=35397,cpp=20725,sh=14666,tcl=9182,yacc=3360,perl=1675,lex=1608,awk=393,csh=85,sed=57 [MIT]
984076	gcc-2.96-20000731	ansic=789901,cpp=126738,yacc=19272,sh=17993,asm=14559,lisp=7161,fortran=3814,exp=3705,objc=479,sed=310,perl=144 [GPL]
967263	gdb+dejagnu-20010316	ansic=871288,exp=58422,sh=12054,cpp=8252,yacc=5906,asm=5031,tcl=4477,lisp=1403,sed=248,awk=170,java=7,fortran=5 [GPL]
690983	binutils-2.10.91.0.2	ansic=489993,asm=161236,exp=13234,sh=12835,yacc=5665,cpp=4777,lex=1488,perl=776,sed=561,lisp=394,awk=24 [GPL]

646692	glibc-2.2.2	ansic=548722,asm=88413,sh=6036,perl=2120,awk=1037, yacc=315,sed=49 [LGPL]
627626	emacs-20.7	lisp=453898,ansic=169956,sh=2622,perl=884,asm=253, csh=9,sed=4 [GPL]
474829	LAPACK	fortran=473590,ansic=1239 [Freely distributable]
455980	gimp-1.2.1	ansic=427967,perl=17482,lisp=9648,yacc=502,sh=381 [GPL, LGPL]
402799	mysql-3.23.36	ansic=249350,cpp=84068,perl=25088,tcl=18980,sh=18323, asm=3987,awk=1436,java=1149,sed=418 [LGPL]
395194	tcltk-8.3.1	ansic=291457,tcl=84322,sh=12259,exp=5742,yacc=876, awk=273,perl=265 [BSD]
345949	kdebase-2.1.1	cpp=181210,ansic=158682,sh=4880,perl=1155,python=22 [GPL]
323730	Mesa-3.4	ansic=286437,cpp=18189,asm=10002,sh=7611,objc=1184, python=307 [GPL/MIT]
321123	perl-5.6.0	perl=146755,ansic=118233,sh=49377,lisp=5739,yacc=996, java=23 [Artistic or GPL]
318430	libgcj	ansic=191432,cpp=56843,java=41716,sh=15581,asm=11262, exp=841,perl=731,awk=24 [GPL]
304819	teTeX-1.0	ansic=223491,perl=49789,sh=17634,cpp=9407,pascal=1546, yacc=1507,awk=622,lex=323,sed=314,asm=139,csh=47 [Distributable]
298742	qt-2.3.0	cpp=259310,ansic=34578,yacc=2444,sh=1493,lex=480, perl=422,lisp=15 [GPL]
286113	postgresql-7.0.3	ansic=237184,java=17540,yacc=9740,sh=8975,tcl=7751, lex=1810,perl=1276,python=959,cpp=801,asm=70,csh=5,sed=2 [BSD]
283785	kdelibs-2.1.1	cpp=261334,ansic=17578,sh=1887,java=1538,perl=731, yacc=607,lex=110 [LGPL]
277502	xemacs-21.1.14	ansic=199927,lisp=73366,sh=2948,perl=930,asm=247, csh=62,sed=22 [GPL]
264528	gs5.50	ansic=259471,cpp=2266,asm=968,sh=823,lisp=405,perl=336, yacc=201,lex=58 [GPL]
227354	krb5-1.2.2	ansic=197886,exp=19124,sh=5140,yacc=2474,perl=1529, awk=393,python=348,lex=190,csh=147,sed=123 [MIT]
215473	vnc_unixsrc	ansic=212766,cpp=848,asm=780,perl=648,sh=431 [GPL]
213818	koffice-2.0.1	cpp=197637,sh=7296,yacc=3791,ansic=3213,perl=1801, lex=80 [GPL]
202842	openssl-0.9.6	ansic=131874,cpp=25744,perl=14737,asm=12428,python=10171, yacc=3297,sh=2641,tcl=1583,lisp=224,objc=143 [BSD-like]
200908	Python-1.5.2	python=101017,ansic=96521,lisp=2353,sh=673,perl=342, sed=2 [Distributable]
194799	bind-9.1.0	ansic=173830,sh=12101,yacc=6025,perl=2830,tcl=13

```

[BSD-like]
192394  xpdf-0.92      cpp=167135,ansic=21621,sh=3638
[GPL]
191379  php-4.0.4pl1      ansic=173334,cpp=7033,sh=6591,lex=1867,yacc=1569,
java=437,awk=367,perl=181
[PHP]
190950  pine4.33          ansic=190020,sh=838,csch=62,perl=30
[Freely distributable]
173492  abi               cpp=159595,ansic=12605,perl=725,sh=550,python=17
[GPL]
167663  kdemultimedia-2.1.1 cpp=140731,ansic=23844,tcl=1004,sh=800,asm=598,
lex=578,perl=106,awk=2
[GPL]
163449  4Suite-0.10.1     python=91445,ansic=72004
[Apache-like]
159301  linuxconf-1.24r2  cpp=142970,perl=6738,sh=3821,java=3074,ansic=2613,
python=85
[GPL]

```

Note that the operating system kernel (Linux) is the largest single component, at over 2.4 million lines of code (mostly in C); that compares to 1.5 million lines of code in Red Hat 6.2. See section 3.2 for a more detailed discussion about the Linux kernel.

The next largest component is Mozilla; this is large because it's really a suite of applications including a web browser, email reader, news reader, HTML editor, and so on. Mozilla is the basis for Netscape Navigator 6.0. Mozilla was not included at all in Red Hat Linux 6.2.

The next largest component is the X Window system, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to gain functionality and size), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, the symbolic debugger, a set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). Emacs is next largest, which should not be a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system.

Note that language implementations tend to be written in themselves, particularly for their libraries. Perl's implementation is written mostly in Perl, and Python is written mostly in Python. Intriguingly, this is not true for Tcl.

In many senses, what is the "largest" component is an artifact of packaging. GNOME and KDE are actually huge, but both are packaged as a set of components instead of being delivered as a single large component. The amount of C code in "kdebase" seemed suspiciously high to one KDE developer, but it turns out that Red Hat includes "lesstiflite" (a Motif clone) in kdebase to support Netscape plug-ins. My thanks to Waldo Bastian (of KDE) for pointing out this unusual situation in kdebase and determining its cause, and to Bernhard Rosenkraenzer (of Red Hat) for confirming the reason for this kdebase addition.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the Linux kernel (at over 2.4 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 1,400,000 lines (57% of the Linux kernel) was in the "drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of peripherals. No other subdirectory comes close to this size - the second largest is the "arch" directory (at over 446,000 SLOC, 18% of the kernel), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is over 168,000 SLOC.

Richard Stallman and others have argued that the resulting system often called "Linux" should instead be called "GNU/Linux" [Stallman 2000]. In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to "free software" (free as in

the freedom to use, modify, and redistribute software for any purpose). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called ``Linux". Using the term ``Linux" is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux" or ``GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc, gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux" and not just ``Linux," and using the term GNU/Linux both credits its contributions and eliminates some ambiguity. Thus, I've decided to switch to the ``GNU/Linux" terminology here.

For more information on the sizes of the Linux kernel components, see

http://www.dwheeler.com/sloc/redhat71-v1/kernel_sloc.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code (using the naming conventions of sloccount, the program used to count SLOC):

Language	SLOC (%)
C	21461450 (71.18%)
C++	4575907 (15.18%)
Shell (Bourne-like)	793238 (2.63%)
Lisp	722430 (2.40%)
Assembly	565536 (1.88%)
Perl	562900 (1.87%)
Fortran	493297 (1.64%)
Python	285050 (0.95%)
Tcl	213014 (0.71%)
Java	147285 (0.49%)
yacc/bison	122325 (0.41%)
Expect	103701 (0.34%)
lex/flex	41967 (0.14%)
awk/gawk	17431 (0.06%)
Objective-C	14645 (0.05%)
Ada	13200 (0.04%)
C shell	10753 (0.04%)
Pascal	4045 (0.01%)
sed	3940 (0.01%)

Here you can see that C is pre-eminent (with over 71% of the code), followed by C++, shell, LISP, assembly, Perl, Fortran, and Python. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has about 4.5 million lines of code, a very respectable showing, but is far less than C (over 21 million SLOC). Still, there's increasing use of C++ code; in the last survey, C had 80.55% and C++ had 7.51%. There is slightly less C code in the total percentage of code, most of which is being taken by C++. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++

became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

LISP continues to place very highly, far more than Perl, Python, Fortran, or Java. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 87% (627626/722430) of the LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages. Perl includes 5739 lines of LISP, and Python includes another 2353 of LISP that is directly used to support elaborate Emacs modes for program editing. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a "control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

Some may be surprised at the number of different languages, but I believe this should be considered not a weakness but a strength. This GNU/Linux distribution supports a wide number of languages, enabling developers to choose the "best tool for the job."

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 94 different lex/flex files, and 138 yacc/bison files. Some build directories use lex/flex or yacc/bison more than once.

Other insights can be gained from the file counts. There were 352,549 files, of which 130,488 were counted source code files (ignoring duplicate files and automatically generated ones). Not included in this count were 10,807 files which contained duplicate contents, and 1,587 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 21,461,450 SLOC contained in 78,676 files, resulting in an "average" C file containing 273 (14218806/52088) physical source lines of code. Intriguingly enough, Red Hat Linux 6.2 had essentially the same average number of physical lines of C code.

3.5 Total Counts by License

Here are the various license types, sorted by the SLOC in the packages with those licenses (see section 2.4 for how these license values were determined):

```
15185987 (50.36%) GPL
2498084 (8.28%) MIT
2305001 (7.64%) LGPL
2065224 (6.85%) MPL
1826601 (6.06%) Distributable
1315348 (4.36%) BSD
907867 (3.01%) BSD-like
766859 (2.54%) Freely distributable
692561 (2.30%) Free
455980 (1.51%) GPL, LGPL
323730 (1.07%) GPL/MIT
321123 (1.07%) Artistic or GPL
191379 (0.63%) PHP
173161 (0.57%) Apache-like
161451 (0.54%) OpenLDAP
146647 (0.49%) LGPL/GPL
103439 (0.34%) GPL (programs), relaxed LGPL (libraries),
and public domain (docs)
```


103291	(0.34%)	Apache
73650	(0.24%)	W3C
73356	(0.24%)	IBM Public License
66554	(0.22%)	University of Washington's Free-Fork License
59354	(0.20%)	Public domain
39828	(0.13%)	GPL and Artistic
31019	(0.10%)	GPL or BSD
25944	(0.09%)	GPL/BSD
20740	(0.07%)	Not listed
20722	(0.07%)	MIT-like
18353	(0.06%)	GPL/LGPL
12987	(0.04%)	Distributable - most of it GPL
8031	(0.03%)	Python
6234	(0.02%)	GPL/distributable
4894	(0.02%)	Freely redistributable
1977	(0.01%)	Artistic
1941	(0.01%)	GPL (not Firmware)
606	(0.00%)	Proprietary

These can be grouped by totalling up SLOC for licenses containing certain key phrases:

16673212	(55.30%)	GPL
3029420	(10.05%)	LGPL
2842536	(9.43%)	MIT
2612681	(8.67%)	distributable
2280178	(7.56%)	BSD
2065224	(6.85%)	MPL
162793	(0.54%)	public domain

From these numbers, you can determine that:

1. The GPL is far and away the most common license (by lines of code) of any single license. In fact, the category ``GPL" (packages with only this one license) all by itself accounts for 50.36% of the packages. By totalling the SLOC for all packages that include "GPL" in the license text, the total rises to 55%. No matter how you look at it, the GPL is the dominant single license in this distribution.
2. The next most common licenses were the LGPL, MIT, BSD, and MPL licenses (in order). This is in line with expectations: the most well-known and well-used open source licenses are the GPL, LGPL, MIT, and BSD licenses. Although the MPL does well in terms of SLOC, there is only one program in this distribution that uses it - Mozilla. There is some use of the ``Artistic" license, but its use is far less; note that papers such as [Perens \[1999\]](#) specifically recommend against using the the Artistic license due to its legal ambiguities.
3. Very little software is released as public domain software (``no copyright"). In this distribution, only 0.2% of the software is in packages labelled as ``public domain" (note that the 0.54% figure above includes the ``sane" package which has documentation in the public domain). There may be several factors that account for this. First, if a developer wishes to get credit for their work, this is a poor ``license;" by law anyone can claim ownership of ``public domain" software. Second, there may be a fear of litigation; both the MIT and BSD licenses permit essentially arbitrary use but forbid lawsuits. While licenses such as MIT's and BSD's are not proof against a lawsuit, they at least provide some legal protection, while releasing software to the public domain provides absolutely no protection. Finally, any software released into the public domain can be modified and re-licensed under any other license, so there's nothing that keeps updated public domain software in the public domain.
4. There is a tiny amount of proprietary code, which is entirely in one component - Netscape Communicator / Navigator. This component uses the Motif toolkit (which is not open source) and has proprietary code mixed into it. As a result, almost none of the code for this package is included on the CD-ROM - only a small amount of ``placeholder" code is there. In the future it is expected that this component will be replaced by Mozilla.
5. The packages which are clearly MIT-like/BSD-like licenses (totalling the MIT, BSD, MIT-like, BSD-like, and none/public domain entries) total 4,742,021 SLOC (15.92%). It's worth noting that 1,837,608 of these lines (39%) is accounted for by the XFree86 X server, an infrastructure component used for GNU/Linux's graphical user interface (GUI).
6. If the license types "distributable", "freely distributable", "MPL", "Free", "Artistic", "Apache", "Apache-like", and "IBM Public license" software was also considered MIT-like/BSD-like, the total SLOC would be 7,954,474 (26%, down from 36%). Unfortunately, the information to determine which of these other packages are simply

BSD-like/MIT-like licenses is not included in the specification files.

7. The packages which include copylefting licenses (GPL or LGPL) total 63%. Limiting to only those that are GPL, LGPL, or both yields 60%, the same percentage as in Red Hat Linux 6.2 and a clear majority.

It is quite clear that in this distribution the GPL is the dominant license and that copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses. This is a simple quantitative explanation why several visible projects (Mozilla, Troll Tech's Qt, and Python) have changed their licenses so that they're compatible with the GPL. When there is so much GPL software, GPL compatibility is critically important to the survival of many open source projects. See the Free Software Foundation's information on [Various Licenses and Comments about Them \[FSF 2001a\]](#) for information on GPL compatibility, and the [GPL FAQ \[FSF 2001b\]](#) for more information on the GPL in general.

The most common open source licenses in this distribution (by SLOC) are the GPL, MIT, LGPL, and BSD licenses (as well as the MPL, but note that it's only used by one project). Note that this is consistent with [Perens \[1999\]](#), who pleads that developers use an existing license instead of developing a new license where possible.

As of this writing, the GPL has received the most attention of these licenses, because Microsoft has specifically been attacking the GPL license. The GPL license permits commercial use of the program, but requires that distributors of modified versions of the program must also release the source code to their changes under the same terms. Therefore, software released under the GPL resists Microsoft's usual ``embrace and extend'' approach to destroy competitors - Microsoft can use and change GPL'ed code, but it cannot make its changes to that code proprietary. As a counter-example, Kerberos (a security component released using an MIT license instead of the GPL) was recently incorporated by Microsoft into their products, and then extended in an incompatible way to prevent users from fully interoperating between products [\[Schneier 2000\]](#). Had Kerberos been released under a GPL or LGPL license, this would have been much more difficult. The presence of so many GPL and LGPL components should make GNU/Linux distributions more resistant to being ``embraced, extended, and extinguished."

3.6 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 30,152,114 physical source lines of code (SLOC); I will simplify this to ``over 30 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (as of 1998)	20 million
Red Hat Linux 6.2 (2000)	17 million

These numbers come from Bruce Schneier's *Crypto-Gram* [\[Schneier 2000\]](#), except for the Space Shuttle numbers which come from a National Academy of Sciences study [\[NAS 1996\]](#) and the Red Hat Linux 6.2 numbers which come from [\[Wheeler 2001\]](#). Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of most of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Note that a deployed ``minimal system'' would have less code; see the paper analyzing Red Hat Linux 6.2 for more discussion about this [\[Wheeler 2001\]](#).

Note that the Red Hat Linux 7.1 system includes a number of applications - in many cases a choice for each category. There are two major desktop environments (GNOME and KDE), plus various lightweight options. There are two word processors (Abiword and KWord), two spreadsheets (Gnumeric and KSpread), two relational database systems (MySQL and Postgres), and two web servers (Apache and TUX). In short, Red Hat Linux 7.1 includes a large number of applications, many of which are not included in its Microsoft or Sun equivalents.

At first blush, this bundling of applications with the operating system might appear similar to Microsoft's policy of combining applications with operating systems (which got Microsoft into legal trouble). However, it's worth noting some

differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take GNU/Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as "secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with GNU/Linux. Third, this distribution (and many others) include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while GNU/Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with smaller SLOC counts can sometimes provide greater functionality than programs with larger SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between GNU/Linux and either Solaris or Windows NT would necessarily cause GNU/Linux to take less effort to develop for a similar size. To see this, let's pretend that GNU/Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that GNU/Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, GNU/Linux's reliability suggests that developing GNU/Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized proprietary systems.

3.7 Effort and Cost Estimates

Finally, given all the assumptions shown previously, the effort values are:

Total Physical Source Lines of Code (SLOC)	= 30152114
Estimated Development Effort in Person-Years (Person-Months)	= 7955.75 (95469)
(Basic COCOMO model, Person-Months = $2.4 * (KSLOC^{1.05})$)	
Estimated Schedule in Years (Months)	= 6.53 (78.31)
(Basic COCOMO model, Months = $2.5 * (person-months^{0.38})$)	
Total Estimated Cost to Develop	= \$ 1074713481
(average salary = \$56286/year, overhead = 2.4).	

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2 (which had been released about one year earlier). Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2).

Had this GNU/Linux distribution been developed by conventional proprietary means, it would have cost over \$1.08 billion (1,000 million) to develop in the U.S. (in year 2000 dollars). Compare this to the \$600 million estimate for version 6.2. Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is quite extraordinary, since this represents approximately one year.

This does not mean that all of the code added to the distribution in this thirteen month time period was actually written in that time period. In many cases, it represents the addition of whole new packages that have been in development for years, but have only now become sufficiently mature to include in the distribution. Also, many projects are developed over time and then released once testing is complete, and the time periods between releases can be more than a year. Still, from the user's point of view, this is a valid viewpoint - within one year of time much more functionality became available within their distribution.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), the X Window system (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary

tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). Note that Mozilla and LAPACK were not included in Red Hat 6.2 at all.

The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed. C is still the predominant language but less so, with C++ primarily taking the difference. This appears in part to reflect many programmers' choice of C++ over C for GUI development. The increased amount of Fortran is primarily due to the inclusion of LAPACK.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this GNU/Linux distribution. In contrast, only 0.2% of the software is in the public domain.

Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches. Back in 1976, Bill Gates published his "Open Letter to Hobbyists", claiming that if software was freely shared it would prevent the writing of good software. He asked rhetorically, "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?" He presumed these were unanswerable questions, and both he and others based an industry on this assumption [Moody 2001]. Now, however, there are thousands of developers who are writing their own excellent code, and then giving it away. Gates was fundamentally wrong: sharing source code, and allowing others to extend it, is indeed a practical approach to developing large-scale systems - and its products can be more reliable.

It would be interesting to re-run these values on other GNU/Linux distributions (such as SuSE and Debian), and other open source systems (such as FreeBSD). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost.

As was noted in the previous paper, some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like "Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not "reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ".m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

More information is available at <http://www.dwheeler.com/sloc>.

Appendix A. Details of Approach

This appendix discusses some of the issues I had to deal with when performing the analysis, hopefully in enough detail that someone could repeat the effort.

In particular, installing the source code required two steps:

1. install the source code files (converting source RPM packages into "spec" files and compressed source files),
2. unpack the source code files (which generates uncompressed source code, and the licensing information),

I then ran sloccount version 1.9 to analyze the source code files, and examined the warning messages and fixed any serious problems that arose. This was not as easy as it sounds; the previous paper (analyzing Red Hat Linux 6.2) discusses this in more detail [Wheeler 2001]. I've since released the tools I used to count code as the program sloccount, available at

<http://www.dwheeler.com/sloccount>.

One complication should be mentioned here. Although the Red Hat Linux 7.1 package comes with a CD-ROM labelled "Source Code", the "Source Code" CD-ROM doesn't contain all the source code. Skipping the source code on the "binary" CD-ROM would produce an invalid count, because 224 source code packages are placed there (including important packages like ssl, perl, python, and samba). It's likely this was done because of CD-ROM space needs - there is so much source code that, even when compressed, it doesn't fit on one CD-ROM.

I then searched for "old" versions of programs that were also included on the CD (so that the same program wouldn't be counted twice), or those required for non-Intel x86 operation (since these would not be fully counted anyway). I did this by examining any specification (in /usr/src/redhat/SPECS) with "compat" or "10" or "11" in its title (it turned out all of them were old and needed removing). I also examined anything ending in a digit or "x" followed by ".spec", which located qt1x.spec. Through this process I removed:

```
compat-egcs.spec  compat-glibc.spec  compat-libs.spec  kdel-compat.spec
gtk+10.spec  libxml10.spec  x86-compat-libs.spec  qt1x.spec
```

I also removed any ``beta" software which had a non-beta version available (beta software was identified by searching for ``beta" in the package or specification file name). This removed:

```
glib-gtkbeta.spec  gtk+-gtkbeta.spec  pango-gtkbeta.spec
```

I also removed "mysqlclient9.spec". This specification contained the older MySQL client library version 3.23.22, as shipped with Red Hat Linux 7, for use with applications linked against it. I did include "mysql.spec", which had the code for the newer version 3.23.36 of MySQL (a relational database package).

Note that unlike Red Hat Linux 6.2, Red Hat Linux 7.1 didn't have two versions of bash or ncurses, so I didn't have to remove old versions of them. I left db1, db2, and db3 in, because it can be argued that none of these three necessarily replaces the other two.

One complication was in handling the graphical subsystem "XFree86". Version 4 of XFree86 was used for all client-side applications, but Red Hat uses both version 3 and version 4 to implement various X servers. I looked at the XFree86 source package for version 4, and it turned out that server code was included in the package. Rather than have XFree86 counted essentially twice (once as version 3, and another as version 4), I only counted the code in version 4 of XFree86. This could be argued both ways; I understand that version 4 is a massive rewrite of much of the version 3 server, so counting it twice is actually not irrational. And unintentionally, I ended up counting a small amount of version 3 code through reuse by another program. It turns out that vnc_unixsrc includes (through reuse) portions of the X Window system version 3 code; in their words, ``a cut-down version of the standard XFree86 distribution (``server only" distribution) without many of the later X extensions or hardware-specific code." VNC won't work without that code, and clearly there was effort to build version 3 and to rebuild version 4, so I let these counts stand.

I then unpacked the source code by running code that in essence did this:

```
cd /usr/src/redhat/SPECS
rpm -bp *.spec
```

This uncompresses the source code and applies all patches used by the actual system. Since I wanted to count the amount of code actually included in the system, it was important to include the patches. The actual code to unpack the source code was more complex, because it also marked every unpacked directory (in the BUILD directory) to identify the spec file it came from and the license of the program. The license was determined by (1) looking at the "Copyright" and "License" fields of the spec file, and if that didn't work, (2) looking at various files in the build directory, such as "LICENSE", "COPYING*", and "Artistic". Unfortunately, MIT-like and BSD-like licenses can be harder to detect (because their text can be varied), but many licenses (such as the GPL and LGPL) can be detected with great confidence. I used the "spec" file as the primary source, because this was placed by a human (who could better understand legal technicalities than a machine).

I actually had to repeat the unpacking more than once; the RPM system would notice a missing dependency for building the software and protest. This required installation of the missing component (in some cases I didn't have to install the program and could have forced installation, but I did not want to risk corrupting the results by failing to install a package).

A surprising development was that the packages "imap" and "samba" reported errors in unpacking. For imap, patch #5 (imap-4.7c2-flock.patch) and for samba, patch #21 (samba-ia64.patch of source/passdb/pass_check.c) would cause unpacking to halt. I unpacked the software and simply counted what was there; this appears to be what the original developers did.

I examined the reported license values, in particular for all code more than 100,000 source lines of code (as the largest components, wrong values for these components would be more likely to cause significant error). I found that Perl had been assigned "GPL" in its spec file, but this isn't the whole story; as documented in its README file, Perl can be used under either the GPL or Artistic license, so its license entry was changed to "GPL or Artistic". Mozilla's licensing situation is more complex; some portions of it are actually under a separate dual licensing scheme (licensed under both the GPL and Netscape Public License, i.e., NPL). However, labelling it as "MPL, NPL, and GPL" would probably overstate the amount of code licensed under the GPL, so I left its entry as the MPL license.

Note that the unpacked source files (including source code, fonts, documentation, and so on) totalled more than 4.4 Gigabytes.

I ran the analysis code as a normal user, so I first had to set the permissions for users to read the code. I then reverted to normal user account, and used sloccount version 1.9 to measure the source code, using the following bash command:

```
sloccount --multiproject /usr/src/redhat/BUILD > sloc-actions 2>&1 &
```

Note that I did not use the "--follow" option of sloccount. Some programs, notably pine, include a symbolic link to other directories such as /usr/lib. Thus, using --follow would have included files outside of the intended directory in the analysis.

I looked over various error reports and determined that none would fundamentally invalidate the results. For example, there were several errors in the XFree86 source code involving improperly formatted strings. It appears that these are syntax errors in the code that are preprocessed away (and thus not noticed by the compiler). I intend to report these problems to the XFree86 project. One program was a bash shell script that began with "#! /usr/bin/env bash", which sloccount's heuristics could not handle at the time. I then modified sloccount to correctly determine its type (it's a bash shell script).

Note that sloccount creates a large number of small files. This isn't fundamentally a problem, but because of the large scale of the system I found that I ran out of inodes if I tried to store multiple copies of results. Those who try to duplicate this activity may want to specially format their filesystems to include more inodes.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/develpro.html>.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the ``Defense Acquisition Deskbook." <http://web2.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[FSF 2001a] Free Software Foundation (FSF). 2001. *Various Licenses and Comments about Them*. <http://www.gnu.org/philosophy/license-list.html>.

[FSF 2001b] Free Software Foundation (FSF). 2001. *General Public License (GPL) Frequently Asked Questions (FAQ)* <http://www.gnu.org/copyleft/gpl-faq.html>.

[Godfrey 2000] Godfrey, Michael W., and Qiang Tu. Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo. ``Evolution in Open Source Software: A Case Study." *2000 Intl Conference on Software Maintenance*. <http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf>

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

[Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>

[Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990.

http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf

[Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter.

<http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>

[Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.

[Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*.

<http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.

[Moody 2001] Moody, Glyn. 2001. *Rebel Code*. ISBN 0713995203.

[NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*.

<http://www.nap.edu/html/statsoft/chap2.html>

[OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.

[Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>

[Perens 1999] Perens, Bruce. January 1999. *Open Sources: Voices from the Open Source Revolution*. "The Open Source Definition". ISBN 1-56592-582-3. <http://www.oreilly.com/catalog/opensources/book/perens.html>

[Raymond 1999] Raymond, Eric S. January 1999. "A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.

[Raymond 2001] Raymond, Eric S. February 22, 2001. *Software Release Practice HOWTO*.

<http://www.linuxdoc.org/HOWTO/Software-Release-Practice-HOWTO/index.html>

[Schneier 2000] Schneier, Bruce. March 15, 2000. "Software Complexity and Security". *Crypto-Gram*.

<http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name..."

<http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet.

<http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*.

http://www.dwheeler.com/oss_fs_refs.html.

[Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.

[Wheeler 2001]. Wheeler, David A. May 9, 2001 (minor update from November 6, 2000). *Estimating Linux's Size*. Version 1.04. <http://www.dwheeler.com/sloc>.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

Trademark owners own any trademarks mentioned.

This paper is (C) Copyright 2001 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. Talk to me to request republication rights. When referring to the paper, please refer to it as "More than a Gigabuck: Estimating GNU/Linux's Size" by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

SLOC	Directory	SLOC-by-Language (Sorted)
2437470	kernel-2.4.2	ansic=2285657,asm=144411,sh=3035,perl=2022,yacc=1147, tcl=576,lex=302,awk=248,sed=72 [GPL]
2065224	mozilla	cpp=1279902,ansic=739470,perl=21220,sh=13717,asm=5212, java=3107,yacc=1831,lex=470,csh=271,sed=24 [MPL]
1837608	XFree86-4.0.3	ansic=1750460,asm=35397,cpp=20725,sh=14666,tcl=9182, yacc=3360,perl=1675,lex=1608,awk=393,csh=85,sed=57 [MIT]
984076	gcc-2.96-20000731	ansic=789901,cpp=126738,yacc=19272,sh=17993,asm=14559, lisp=7161,fortran=3814,exp=3705,objc=479,sed=310,perl=144 [GPL]
967263	gdb+dejagnum-20010316	ansic=871288,exp=58422,sh=12054,cpp=8252,yacc=5906, asm=5031,tcl=4477,lisp=1403,sed=248,awk=170,java=7,fortran=5 [GPL]
690983	binutils-2.10.91.0.2	ansic=489993,asm=161236,exp=13234,sh=12835, yacc=5665,cpp=4777,lex=1488,perl=776,sed=561,lisp=394,awk=24 [GPL]
646692	glibc-2.2.2	ansic=548722,asm=88413,sh=6036,perl=2120,awk=1037, yacc=315,sed=49 [LGPL]
627626	emacs-20.7	lisp=453898,ansic=169956,sh=2622,perl=884,asm=253, csh=9,sed=4 [GPL]
474829	LAPACK	fortran=473590,ansic=1239 [Freely distributable]
455980	gimp-1.2.1	ansic=427967,perl=17482,lisp=9648,yacc=502,sh=381 [GPL, LGPL]
402799	mysql-3.23.36	ansic=249350,cpp=84068,perl=25088,tcl=18980,sh=18323, asm=3987,awk=1436,java=1149,sed=418 [LGPL]
395194	tcltk-8.3.1	ansic=291457,tcl=84322,sh=12259,exp=5742,yacc=876, awk=273,perl=265 [BSD]
345949	kdebase-2.1.1	cpp=181210,ansic=158682,sh=4880,perl=1155,python=22 [GPL]
323730	Mesa-3.4	ansic=286437,cpp=18189,asm=10002,sh=7611,objc=1184, python=307 [GPL/MIT]
321123	perl-5.6.0	perl=146755,ansic=118233,sh=49377,lisp=5739,yacc=996, java=23 [Artistic or GPL]
318430	libgcj	ansic=191432,cpp=56843,java=41716,sh=15581,asm=11262, exp=841,perl=731,awk=24 [GPL]
304819	teTeX-1.0	ansic=223491,perl=49789,sh=17634,cpp=9407,pascal=1546, yacc=1507,awk=622,lex=323,sed=314,asm=139,csh=47 [Distributable]
298742	qt-2.3.0	cpp=259310,ansic=34578,yacc=2444,sh=1493,lex=480, perl=422,lisp=15 [GPL]
286113	postgresql-7.0.3	ansic=237184,java=17540,yacc=9740,sh=8975,tcl=7751, lex=1810,perl=1276,python=959,cpp=801,asm=70,csh=5,sed=2 [BSD]
283785	kdelibs-2.1.1	cpp=261334,ansic=17578,sh=1887,java=1538,perl=731, yacc=607,lex=110 [LGPL]
277502	xemacs-21.1.14	ansic=199927,lisp=73366,sh=2948,perl=930,asm=247, csh=62,sed=22 [GPL]
264528	gs5.50	ansic=259471,cpp=2266,asm=968,sh=823,lisp=405,perl=336, yacc=201,lex=58 [GPL]
227354	krb5-1.2.2	ansic=197886,exp=19124,sh=5140,yacc=2474,perl=1529,

		awk=393,python=348,lex=190,csch=147,sed=123 [MIT]
215473	vnc_unixsrc	ansic=212766,cpp=848,asm=780,perl=648,sh=431 [GPL]
213818	koffice-2.0.1	cpp=197637,sh=7296,yacc=3791,ansic=3213,perl=1801, lex=80 [GPL]
202842	openssl-0.9.6	ansic=131874,cpp=25744,perl=14737,asm=12428,python=10171, yacc=3297,sh=2641,tcl=1583,lisp=224,objc=143 [BSD-like]
200908	Python-1.5.2	python=101017,ansic=96521,lisp=2353,sh=673,perl=342, sed=2 [Distributable]
194799	bind-9.1.0	ansic=173830,sh=12101,yacc=6025,perl=2830,tcl=13 [BSD-like]
192394	xpdf-0.92	cpp=167135,ansic=21621,sh=3638 [GPL]
191379	php-4.0.4pl1	ansic=173334,cpp=7033,sh=6591,lex=1867,yacc=1569, java=437,awk=367,perl=181 [PHP]
190950	pine4.33	ansic=190020,sh=838,csch=62,perl=30 [Freely distributable]
173492	abi	cpp=159595,ansic=12605,perl=725,sh=550,python=17 [GPL]
167663	kdemultimedia-2.1.1	cpp=140731,ansic=23844,tcl=1004,sh=800,asm=598, lex=578,perl=106,awk=2 [GPL]
163449	4Suite-0.10.1	python=91445,ansic=72004 [Apache-like]
159301	linuxconf-1.24r2	cpp=142970,perl=6738,sh=3821,java=3074,ansic=2613, python=85 [GPL]
156168	vim60z	ansic=154611,awk=688,sh=498,perl=365,csch=6 [Free]
154738	anaconda-7.1	ansic=125372,python=20874,sh=5447,yacc=1175,lex=966, perl=904 [GPL]
152764	kaffe-1.0.6	java=70408,ansic=70048,sh=6619,cpp=4611,perl=982, asm=84,awk=12 [GPL]
149535	doxygen-1.2.6	cpp=135809,lex=12119,perl=1028,sh=346,yacc=233 [GPL]
139421	octave-2.1.33	cpp=82407,ansic=28240,fortran=15646,sh=4506,yacc=2687, lex=2526,perl=1676,lisp=1610,exp=123 [GPL]
138945	gtk+-1.2.9	ansic=138103,perl=328,awk=274,sh=233,lisp=7 [LGPL]
138667	ImageMagick-5.2.7	ansic=115458,cpp=13092,sh=7331,perl=2328,tcl=458 [Free]
130133	gnome-libs-1.2.8	ansic=128282,sh=730,perl=667,awk=277,lisp=177 [LGPL]
123368	gsl-0.7	ansic=123216,sh=152 [GPL]
122179	openjade-1.3	cpp=112755,ansic=6507,sh=2423,perl=489,sed=5
121390	tripwire-2.3.0-50	cpp=84011,ansic=34686,perl=1167,sh=623,yacc=497, lex=354,sed=40,awk=12 [GPL]
119385	kdepim-2.1.1	cpp=72755,ansic=39618,yacc=5204,perl=1214,lex=466, awk=100,sh=28 [GPL]
118802	lam-6.5.1	ansic=100176,sh=9619,cpp=8789,fortran=218 [BSD-like]
117227	lynx2-8-4	ansic=116438,perl=583,sh=206 [GPL]
117159	mc-4.5.51	ansic=115411,sh=1198,perl=346,awk=148,csch=56

		[GPL]
114822	netpbm-9.9	ansic=113181, csh=804, perl=476, sh=361
		[Free]
113302	xlispstat-3-52-18	ansic=91514, lisp=21769, sh=18, csh=1
		[Distributable]
112075	gnumeric-0.61	ansic=110711, yacc=612, perl=501, lisp=193, python=58
		[GPL]
110682	lclint-2.5q	ansic=105377, lex=2821, yacc=2484
		[GPL]
110140	isd4k-utils	ansic=96344, sh=6546, perl=4520, cpp=2708, tcl=22
		[GPL]
109387	kdenetwork-2.1.1	cpp=101022, perl=6268, ansic=1972, sh=76, tcl=49
		[GPL]
108502	TiMidity+-2.10.3a2	ansic=106958, tcl=1045, lisp=499
		[GPL]
106216	samba-2.0.7	ansic=97694, sh=4173, perl=2698, asm=1278, csh=162, awk=161, sed=50
		[GPL]
104234	blt2.4u	ansic=89860, tcl=14222, sh=152
		[MIT]
104080	xlockmore-4.17.2	ansic=97496, cpp=3204, tcl=1318, sh=1200, java=458, perl=404
		[MIT]
103479	db-3.1.17	ansic=61029, tcl=17318, perl=9890, sh=8158, cpp=3085, java=2188, awk=1400, sed=411
		[BSD-like]
103439	sane-1.0.3	ansic=95179, sh=6382, java=1421, lisp=457
		[GPL (programs), relaxed LGPL (libraries), and public domain (docs)]
103291	apache_1.3.19	ansic=91118, sh=10202, perl=1602, lex=190, yacc=97, cpp=55, lisp=27
		[Apache]
103035	gated-public-3_6	ansic=101595, lex=941, sh=499
		[Distributable]
102559	xscreensaver-3.29	ansic=100974, perl=1105, sh=480
		[BSD]
101601	openldap-2.0.7	ansic=89504, sh=10060, cpp=1107, perl=890, tcl=40
		[OpenLDAP]
100588	krb4-1.0.5	ansic=84876, asm=5163, cpp=3775, perl=2508, sh=2487, yacc=1510, lex=236, awk=33
		[Freely distributable]
98142	WindowMaker-0.64.0	ansic=95772, sh=1588, perl=563, lisp=219
		[GPL]
97920	ucd-snmp-4.2	ansic=84445, perl=7261, sh=6214
		[BSD-like]
95528	FreeWnn-1.1.1-a017	ansic=95391, cpp=88, sh=49
		[Distributable]
90367	cvs-1.11	ansic=69953, sh=18478, perl=910, yacc=834, csh=185, lisp=7
		[GPL]
89173	unixODBC-1.8.13	ansic=64065, cpp=22618, sh=2220, lex=190, yacc=80
		[LGPL]
88835	inn-2.3.1	ansic=67615, sh=9722, perl=9580, yacc=1566, lex=249, python=100, tcl=3
		[GPL]
88654	kdevelop-1.4.1	cpp=52239, ansic=26963, sh=4988, perl=2931, asm=1239, lex=174, java=77, awk=34, csh=9
		[GPL]
87834	htdig-3.2.0b3	ansic=42344, cpp=40977, sh=2325, perl=2188
		[GPL]
87306	freetype-2.0.1	ansic=76064, sh=7499, python=1863, cpp=1662, lex=218
		[BSD-like]
84629	xpilot-4.3.0	ansic=77919, tcl=3479, cpp=1896, sh=1267, perl=68
		[GPL]
84244	kdegraphics-2.1.1	ansic=42828, cpp=41388, sh=28
		[GPL]

82088	groff-1.16.1	cpp=69739,ansic=6459,yacc=2969,asm=1865,perl=522,sh=403,sed=131 [GPL]
80433	kdegames-2.1.1	cpp=71840,ansic=8289,python=209,sh=66,perl=29 [GPL]
80259	Canna35b2	ansic=79073,yacc=403,lex=379,cpp=369,sh=28,awk=7 [Distributable]
79153	enlightenment-0.16.4	ansic=78364,sh=642,perl=147 [GPL]
78444	extace-1.4.4	ansic=69839,sh=5711,perl=2894 [GPL]
77231	ntp-4.0.99k	ansic=72069,perl=4106,sh=595,awk=417,asm=37,sed=7 [Distributable]
77182	freeciv-1.11.4	ansic=77019,sh=163 [GPL]
77077	gmp-3.1.1	ansic=43307,asm=25229,sh=8027,lisp=186,yacc=150,lex=78,perl=76,fortran=24 [LGPL]
76834	bash-2.04	ansic=62346,sh=9676,yacc=3034,perl=1730,asm=48 [GPL]
75303	dia-0.86	ansic=75303 [GPL]
74295	gnome-core-1.2.4	ansic=73984,perl=241,sh=70 [LGPL/GPL]
73885	VFLib2-2.25.1	perl=49664,ansic=20684,sh=3537 [GPL]
73660	cdrecord-1.9	ansic=71178,sh=2255,perl=223,sed=4 [GPL]
73650	w3c-libwww-5.2.8	ansic=64587,sh=4678,cpp=3181,perl=1204 [W3C]
73356	jikes-1.13	cpp=70781,sh=2575 [IBM Public License]
69489	squid-2.3.STABLE4	ansic=66882,sh=1570,perl=1037 [GPL]
69303	fvwm-2.2.4	ansic=63534,cpp=2463,perl=1835,sh=723,yacc=596,lex=152 [GPL]
69247	linux-86	ansic=63329,asm=5276,sh=642 [GPL]
67944	ncurses-5.2	ansic=48144,ada=12937,cpp=3726,sh=2323,awk=552,sed=136,perl=126 [Distributable]
66554	imap-2000	ansic=66494,sh=60 [University of Washington's Free-Fork License]
65302	sendmail-8.11.2	ansic=61141,perl=3096,sh=1065 [BSD]
64627	gaim-0.11.0pre4	ansic=57897,sh=6602,perl=128 [GPL]
64418	SDL-1.1.7	ansic=53659,sh=7583,cpp=1621,asm=1506,perl=49 [LGPL]
63792	sgml-tools-1.0.9	cpp=39470,ansic=19560,perl=3219,lex=560,sh=532,lisp=309,awk=142 [Free]
63603	kdeutils-2.1.1	cpp=63384,sh=219 [GPL]
62774	wv	ansic=59459,sh=2368,perl=646,awk=273,cs=28 [GPL]
62007	amanda-2.4.2p2	ansic=50259,sh=9404,perl=1619,awk=333,lex=167,tcl=118,yacc=107 [BSD]
61316	kdeadmin-2.1.1	cpp=49153,sh=9715,perl=1778,ansic=412,python=258 [GPL]
59850	openldap-1.2.11	ansic=58673,sh=950,python=201,tcl=26 [OpenLDAP]
59681	gnupg-1.0.4	ansic=54373,asm=4524,sh=784 [GPL]

58359	lsf_4.51	ansic=52317,sh=4970,perl=856,awk=214,asm=2 [Free]
58264	xfig.3.2.3c	ansic=58264 [Free]
56943	a2ps-4.13	ansic=41775,sh=10215,lex=2286,perl=1156,yacc=757, ada=263,java=151,lisp=122,sed=107,objc=87,python=24 [GPL]
55667	uucp-1.06.1	ansic=52078,sh=3400,perl=189 [GPL]
55484	nvi-1.79	ansic=53850,tcl=1124,perl=365,sh=98,awk=34,csch=13 [GPL]
54081	glade-0.5.9	ansic=50848,sh=3233 [GPL]
53485	guile-1.3.4	ansic=43663,lisp=7781,asm=1514,sh=315,awk=162,csch=50 [GPL]
52693	nmh-1.0.4	ansic=50810,sh=1792,awk=74,sed=17 [Free]
52356	mutt-1.2.5	ansic=51866,sh=490 [GPL]
51765	balsa-1.1.1	ansic=51472,sh=233,awk=60 [GPL]
50105	unzip-5.41	ansic=44728,cpp=3788,asm=1542,sh=47 [BSD]
49908	imlib-1.9.8.1	ansic=48982,sh=926 [LGPL]
49736	mgetty-1.1.25	ansic=37210,perl=6516,sh=4971,tcl=756,lisp=283 [GPL]
48512	licq-1.0.2	cpp=41123,sh=3481,perl=2396,ansic=1464,csch=48 [GPL]
48217	rpm-4.0.2	ansic=44845,sh=2017,perl=1355 [GPL]
47721	tcsh-6.10.00	ansic=43878,sh=2378,csch=796,lisp=669 [Distributable]
47368	ncpfs-2.2.0.18	ansic=47161,sh=181,tcl=26 [GPL]
47081	slang-1.4.2	ansic=46234,sh=847 [GPL]
45527	kinput2-v3	ansic=45527 [Distributable]
45378	gnuplot-3.7.1	ansic=43276,lisp=661,asm=539,objc=387,csch=297,perl=138, sh=80 [Distributable]
44505	ppp-2.4.0	ansic=44102,sh=321,exp=82 [Distributable]
44060	xmms-1.2.4	ansic=42160,asm=1714,sh=186 [GPL]
43930	am-utils-6.0.5	ansic=34144,sh=6533,perl=2424,lex=454,yacc=375 [BSD]
43432	libiconv	ansic=41249,sh=2183 [GPL]
42745	elm2.5.3	ansic=32930,sh=9774,awk=41 [Distributable]
42561	gnome-games-1.2.0	ansic=32242,lisp=7132,cpp=3187 [LGPL]
42007	gnome-pim-1.2.0	ansic=40204,yacc=1803 [GPL]
41786	textutils-2.0.11	sh=20403,ansic=19827,perl=1496,sed=60 [GPL]
41388	p2c-1.22	ansic=38788,pascal=2499,perl=101 [Distributable]
41045	mount-2.10r	ansic=40399,sh=464,perl=65,csch=62,sed=55 [GPL]
40760	ORBit-0.5.7	ansic=38314,yacc=1751,lex=364,sh=331 [LGPL/GPL]
40143	fileutils-4.0.36	ansic=32835,sh=5772,yacc=877,perl=659

		[GPL]
39874	libmng-1.0.0	ansic=39751,sh=123
		[BSD-like]
39828	LPRng-3.7.4	ansic=34921,sh=2812,perl=2095
		[GPL and Artistic]
39770	x3270-3.2	ansic=38822,sh=772,exp=176
		[MIT]
39739	ircii-4.4Z	ansic=39205,sh=508,lex=26
		[Distributable]
39231	strace	ansic=36728,sh=2084,perl=375,lisp=44
		[Distributable]
38805	openssh-2.5.2p2	ansic=36224,sh=2229,perl=352
		[BSD]
38672	zsh-3.0.8	ansic=36424,sh=1763,perl=331,awk=148,sed=6
		[GPL]
38516	librep-0.13.3	ansic=27430,lisp=10788,sh=298
		[GPL]
38458	pilot-link	ansic=31145,java=2167,cpp=1714,perl=1431,sh=1058, yacc=661,python=268,tcl=14
		[GPL]
37742	gnome-applets-1.2.4	ansic=37395,sh=203,perl=144
		[GPL]
37599	Xaw3d-1.5	ansic=37226,yacc=247,lex=126
		[MIT]
36836	xpuzzles-5.5.2	ansic=36836
		[MIT]
36819	exmh-2.2	tcl=36325,perl=411,sh=49,exp=34
		[Free]
36806	gal-0.4.1	ansic=36466,perl=340
		[GPL]
35853	pan-0.9.5	ansic=35853
		[GPL]
35814	xloadimage-4.1	ansic=35707,sh=107
		[MIT]
35750	skkinput-2.03	ansic=35750
		[GPL]
35369	jed-B0.99-12	ansic=35369
		[GPL]
34851	gnome-utils-1.2.1	ansic=33450,yacc=824,lisp=577
		[GPL]
34755	gphoto-0.4.3	ansic=34715,sh=40
		[GPL]
34472	gnome-print-0.25	ansic=34472
		[LGPL]
34393	sawfish-0.36	lisp=17573,ansic=14732,sh=2069,perl=19
		[GPL]
34384	Inti-0.6preview	cpp=29578,python=4798,ansic=7,sh=1
		[LGPL]
33420	xchat-1.6.3	ansic=33200,perl=121,python=76,sh=23
		[GPL]
33082	e2fsprogs-1.19	ansic=29994,sh=2538,awk=407,sed=121,perl=22
		[GPL]
32314	ncftp-3.0.2	ansic=31368,cpp=595,sh=351
		[Distributable]
31989	libpng-1.0.9	ansic=31916,sh=63,cpp=10
		[Distributable]
31224	kdesdk-2.1.1	cpp=27358,perl=1036,lisp=857,ansic=835,sh=785,python=353
		[GPL]
30950	nasm-0.98	ansic=28284,perl=1982,asm=617,sh=67
		[GPL]
30618	lm_sensors-2.5.5	ansic=26472,perl=3507,yacc=275,lex=265,sh=99
		[GPL]
30518	screen-3.9.8	ansic=28931,sh=1587
		[GPL]
30414	scheme-3.2	lisp=19775,ansic=10515,sh=124

		[GPL]
30379	tcpdump-3.4	ansic=29516,yacc=236,sh=222,lex=206,awk=191,csch=8
		[BSD]
29224	slrn-0.9.6.4	ansic=29213,sh=11
		[GPL]
29069	texinfo-4.0	ansic=26960,sh=1168,awk=451,perl=256,lisp=213,sed=21
		[GPL]
27823	tiff-v3.5.5	ansic=23632,sh=4191
		[Distributable]
27665	mikmod-3.1.6	ansic=27600,sh=55,awk=10
		[LGPL]
26946	xpaint	ansic=26923,sh=23
		[MIT]
26672	lout-3.17	ansic=26672
		[GPL]
26629	ddskk20010225	lisp=25951,ansic=500,awk=178
		[GPL]
26430	libxml-1.8.10	ansic=26353,sh=77
		[LGPL]
26205	db2	ansic=25710,cpp=454,asm=41
		[GPL]
26152	mars_nwe	ansic=25923,sh=229
		[GPL]
26022	gv-3.5.8	ansic=25928,sh=94
		[GPL]
25995	xrn-9.02	ansic=24687,yacc=888,sh=249,lex=92,perl=35,awk=31,csch=13
		[Distributable]
25656	shadow-20000826	ansic=23431,sh=1336,yacc=856,perl=33
		[BSD]
25602	kdesupport-2.1	cpp=14659,ansic=10393,perl=322,sh=228
		[LGPL/GPL]
25126	gawk-3.0.6	ansic=20213,awk=2628,yacc=2064,sh=221
		[GPL]
25083	gd-1.8.3	ansic=24941,perl=140,sh=2
		[BSD-like]
24989	links-0.95	ansic=23084,sh=1804,awk=67,perl=34
		[GPL]
24814	findutils-4.1.6	ansic=13732,sh=10791,exp=291
		[GPL]
24769	pdksh-5.2.14	ansic=23595,perl=945,sh=189,sed=40
		[Public domain]
24704	enscript-1.6.1	ansic=23403,lex=429,perl=308,sh=291,yacc=164,lisp=109
		[GPL]
24349	gnome-python-1.0.53	python=14348,ansic=10001
		[LGPL]
23896	sox-12.17.1	ansic=22292,sh=1471,perl=133
		[Distributable]
23867	kterm-6.2.0	ansic=23867
		[Distributable]
23726	make-3.79.1	ansic=22697,perl=966,sh=63
		[GPL]
23470	xboard-4.1.0	ansic=22465,lex=922,sh=66,sed=12,csch=5
		[GPL]
21941	wu-ftpd	ansic=19120,yacc=1983,sh=634,perl=204
		[BSD]
21827	xsane-0.62	ansic=20958,sh=864,sed=5
		[GPL]
21468	pam-0.74	ansic=19864,yacc=693,sh=527,perl=324,lex=60
		[GPL or BSD]
21171	netkit-telnet-0.17-pre20000412	ansic=14962,cpp=6209
		[BSD]
21017	kdbg-1.2.0	cpp=15454,sh=5563
		[GPL]
20740	src_contrib	ansic=20740

20722	libungif-4.1.0b1	ansic=20458,sh=210,perl=54 [MIT-like]
20709	cyrus-sasl-1.5.24	ansic=18812,java=1536,cpp=361 [Distributable]
20708	fetchmail-5.7.4	ansic=15655,sh=2087,python=1595,perl=491,yacc=422, lex=250,awk=124,lisp=84 [GPL]
20619	xinetd-2.1.8.9pre14	ansic=18916,sh=1547,perl=156 [BSD-like]
20323	xcdroast-0.98alpha8	ansic=20290,sh=33 [GPL]
20207	pcmcia-cs-3.1.24	ansic=18841,sh=1366 [GPL]
20133	ical-2.2	cpp=12657,tcl=6765,sh=624,perl=60,ansic=27 [Distributable]
19932	alchemist-0.16	ansic=12011,sh=7240,python=681 [GPL]
19829	parted-1.4.7	ansic=18919,sh=874,asm=36 [GPL]
19396	mttools-3.9.7	ansic=16600,sh=2788,sed=8 [GPL]
19197	gtk-engines-0.10	ansic=14143,sh=5054 [GPL]
19174	SDL_mixer-1.1.0	ansic=16967,sh=2207 [LGPL]
19067	joe	ansic=18843,asm=224 [GPL]
19061	glib-1.2.9	ansic=18931,sh=130 [LGPL]
18482	transfig.3.2.3c	ansic=18419,sh=38,cs=25 [Distributable]
18353	control-center-1.2.2	ansic=18227,sh=126 [GPL/LGPL]
18333	util-linux-2.10s	ansic=17969,sh=364 [Distributable]
18227	libvorbis-1.0beta4	ansic=17853,perl=216,sh=158 [GPL/BSD]
18182	sh-utils-2.0	ansic=13335,sh=3027,perl=949,yacc=871 [GPL]
18018	xboing	ansic=18004,sh=14 [MIT]
17915	smpeg-0.4.2	cpp=15006,ansic=2289,asm=542,sh=78 [LGPL]
17830	less-358	ansic=17823,awk=7 [GPL]
17802	gettext-0.10.35	ansic=13466,lisp=2030,sh=1983,yacc=261,perl=53,sed=9 [GPL]
17428	expat	ansic=11916,sh=5173,cpp=339 [GPL]
17392	rxvt-2.7.5	ansic=15057,sh=2335 [Distributable]
17293	reiserfsprogs-3.x.0f	ansic=17270,sh=23 [GPL]
16750	mpg123-0.59r	ansic=15831,asm=919 [Distributable]
16395	dhcp-2.0p15	ansic=15446,sh=949 [Distributable]
15993	automake-1.4	perl=10622,sh=4967,ansic=404 [GPL]
15852	taper-6.9b	ansic=15852 [GPL]
15777	tamago-4.0.6	lisp=15777 [GPL]
15718	iptables-1.2.1a	ansic=15399,sh=319 [GPL]

15672	mod_perl-1.24_01	perl=9739,ansic=5679,sh=254 [GPL]
15616	man-1.5h1	sh=7575,ansic=7419,perl=317,awk=305 [GPL]
15526	rsync-2.4.6	ansic=14264,sh=1024,perl=179,awk=59 [GPL]
15465	aspell-.32.6	sh=8414,cpp=6569,perl=409,ansic=73 [LGPL]
15403	mm2.7	ansic=8046,csch=7199,sh=158 [Distributable]
15380	modutils-2.4.2	ansic=13508,sh=1013,lex=484,yacc=375 [GPL]
15265	wget-1.6	ansic=15191,perl=51,sh=23 [GPL]
15066	nfs-utils-0.3.1	ansic=14704,sh=335,perl=27 [GPL]
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,csch=29 [GPL]
14763	flex-2.5.4	ansic=13000,lex=1045,yacc=605,awk=72,sh=29,sed=12 [GPL]
14721	mawk-1.3.3	ansic=13008,yacc=994,awk=629,sh=90 [GPL]
14718	wmakerconf-2.6.1	ansic=14408,perl=171,sh=139 [GPL]
14587	multimedia	ansic=14577,sh=10 [GPL]
14314	libgtop-1.0.10	ansic=13566,perl=653,sh=64,asm=31 [LGPL]
14297	rcc-5.7	ansic=12237,sh=2060 [GPL]
14239	console-tools-0.3.3	ansic=12052,yacc=986,sh=800,lex=291,perl=110 [GPL]
13780	ash-0.3.7.orig	ansic=13356,sh=190,yacc=139,lex=95 [BSD]
13716	iproute2	ansic=12637,sh=978,perl=101 [GPL]
13626	tar-1.13.19	ansic=12930,sh=585,perl=111 [GPL]
13623	Maelstrom-3.0.1	cpp=10838,ansic=2781,sh=4 [LGPL]
13431	libunicode-0.4	ansic=13317,sh=105,cpp=9 [LGPL]
13144	gftp-2.0.7b	ansic=13121,sh=23 [GPL]
12987	xpat2-1.06	ansic=10716,cpp=2243,sh=28 [Distributable - most of it GPL]
12822	rp3-1.1.10	cpp=9233,ansic=3456,sh=133 [GPL]
12781	net-tools-1.57	ansic=12679,sh=102 [GPL]
12717	esound-0.2.22	ansic=7771,sh=4860,csch=86 [GPL]
12675	minicom-1.83.1	ansic=12559,sh=116 [GPL]
12657	macutils	ansic=12657 [Distributable]
12608	pmake-1.45	ansic=12549,sh=59 [BSD]
12589	gnome-objc-1.0.2	objc=12365,sh=212,ansic=12 [LGPL]
12313	jpeg-6a	ansic=12313 [Distributable]
12259	lrzsz-0.12.20	ansic=9544,sh=1700,exp=1015 [GPL]
12036	libodbc++-0.2.2pre4	cpp=10347,sh=1689

		[LGPL]
11764	bc-1.06	ansic=10135,yacc=971,lex=374,sh=284
		[GPL]
11630	ypserv-1.3.11	ansic=11140,sh=418,perl=42,sed=30
		[GPL]
11529	xdelta-1.1.1	ansic=9840,lisp=1525,sh=164
		[GPL]
11509	nss_ldap-149	ansic=11222,perl=241,sh=46
		[LGPL]
11438	db.1.85	ansic=10632,sh=665,csch=141
		[BSD]
11344	xmorph-2000apr28	ansic=10828,tcl=516
		[GPL]
11113	dump-0.4b21	ansic=10196,sh=912,sed=5
		[BSD]
11090	ctags-4.0.3	ansic=11067,sh=23
		[GPL]
10976	gpm-1.19.3	ansic=9132,yacc=1120,sh=422,lisp=222,awk=74,sed=6
		[GPL]
10914	gqview-0.8.1	ansic=10914
		[GPL]
10914	diffutils-2.7	ansic=10914
		[GPL]
10901	dip-3.3.7o	ansic=8214,asm=2591,sh=96
		[GPL]
10860	lilo-21.4.4	asm=5975,ansic=3701,sh=748,perl=433,cpp=3
		[MIT]
10775	dmalloc-4.8.1	ansic=10575,perl=145,cpp=34,sh=21
		[Public domain]
10680	procps-2.0.7	ansic=10678,sh=2
		[GPL]
10568	nut-0.44.1	ansic=10286,sh=282
		[GPL]
10554	m4-1.4.1	ansic=10156,lisp=243,sh=155
		[GPL]
10554	gnorpm-0.96	ansic=10554
		[GPL]
10480	sudo-1.6.3p6	ansic=8890,sh=1590
		[BSD]
10436	gdk-pixbuf-0.8.0	ansic=9968,asm=417,sh=51
		[LGPL]
10288	kdebindings-2.1.1	cpp=5737,ansic=3010,python=881,perl=320,java=272,sh=68
		[GPL]
10194	lv4494	ansic=8505,sh=1200,perl=485,csch=4
		[Distributable]
10077	Mysql-Mysql-modules-1.2215	perl=7859,ansic=2218
		[Distributable]
9928	procmail-3.14	ansic=8091,sh=1837
		[Distributable]
9926	gnome-media-1.2.0	ansic=9915,sed=11
		[LGPL]
9712	mod_dav-1.0.2-1.3.6	ansic=9077,python=635
		[Apache-like]
9688	grep-2.4.2	ansic=9527,sh=103,awk=49,sed=9
		[GPL]
9686	kudzu-0.98.10	ansic=9299,python=239,perl=97,sh=51
		[GPL]
9625	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=279
		[Public domain]
9551	pwdb-0.61.1	ansic=9488,sh=63
		[GPL or BSD]
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
		[BSD]
9432	rpm2html-1.5	ansic=9310,perl=122

		[BSD-like]
9336	switchdesk-3.9.5	sh=7035,perl=1328,ansic=411,cpp=304,python=258
		[GPL]
9114	gtop-1.0.11	ansic=8774,cpp=340
		[LGPL]
9077	njamd-0.8.0	ansic=8794,sh=231,perl=52
		[GPL]
9050	im-140	perl=8342,sh=708
		[Distributable]
9039	pax-1.5	ansic=9039
		[BSD]
8816	dialog-0.9a-20001217	ansic=5321,sh=3145,perl=350
		[GPL]
8779	pxe-linux	cpp=4469,ansic=3629,asm=681
		[BSD]
8713	pspell-.11.2	sh=3784,cpp=2918,ansic=1961,perl=50
		[LGPL]
8682	DBI-1.14	perl=8211,ansic=471
		[Distributable]
8644	pnm2ppa-1.04	ansic=8179,sh=465
		[GPL]
8572	psgml-1.2.1	lisp=8572
		[GPL]
8513	pinfo-0.6.0	ansic=8405,sh=108
		[GPL]
8418	gedit-0.9.4	ansic=8273,sh=145
		[GPL]
8287	gq-0.4.0	ansic=8264,sh=23
		[GPL]
8031	Distutils-1.0.1	python=7050,ansic=980,sh=1
		[Python]
7948	gzip-1.3	ansic=5765,sh=1993,asm=179,perl=11
		[GPL]
7869	bison-1.28	ansic=7820,sh=49
		[GPL]
7786	newt-0.50.22	ansic=7245,python=541
		[LGPL]
7782	sharutils-4.2.1	ansic=5723,perl=1741,sh=318
		[GPL]
7748	pkgconfig-0.5.0	ansic=7748
		[GPL]
7740	sed-3.02	ansic=7301,sh=359,sh=80
		[GPL]
7724	psiconv	ansic=7701,sh=23
		[GPL]
7717	xosview-1.7.3	cpp=7330,ansic=367,awk=20
		[GPL/BSD]
7711	rpmfind-1.6	ansic=7711
		[BSD-like]
7686	gdbm-1.8.0	sh=4371,ansic=3290,cpp=25
		[GPL]
7622	cpio-2.4.2	ansic=7603,sh=19
		[GPL]
7430	cdparanoia-III-alpha9.7	ansic=6209,sh=1221
		[GPL]
7427	ed-0.2	ansic=7263,sh=164
		[GPL]
7376	xbl-1.0j	ansic=7376
		[GPL]
7267	apel-10.2	lisp=7267
7175	ee-0.3.12	ansic=6993,sh=182
		[GPL]
7162	yacc	ansic=6095,yacc=1030,sh=37
		[Public domain]
7155	expat-1.95.1	ansic=7132,sh=23

		[MIT]
7096	xgammon-0.98	ansic=6507,lex=589
		[GPL]
7095	iputils	ansic=7095
		[BSD]
7079	gnome-linuxconf-0.64	ansic=7069,sh=10
		[GPL]
7055	kdoc-2.1.1	perl=6964,cpp=54,sh=37
		[GPL]
7046	mailx-8.1.1	ansic=7041,sh=5
		[BSD]
6980	audiofile-0.1.11	ansic=6921,sh=59
		[LGPL]
6970	patch-2.5.4	ansic=6916,sed=54
		[GPL]
6915	syslinux-1.52	asm=6382,ansic=335,perl=198
		[BSD]
6850	ftpcopy-0.3.4	ansic=6637,sh=213
		[Free]
6629	libglade-0.14	ansic=6352,python=154,sh=123
		[LGPL]
6524	pidentd-3.0.12	ansic=6503,sh=21
		[Public domain]
6396	clime-1.13.6	lisp=6396
6348	bug-buddy-1.2	ansic=4161,sh=2187
		[GPL]
6344	internet-config-0.40	cpp=5465,sh=571,perl=308
		[GPL]
6234	awesfx-0.4.3a	ansic=6234
		[GPL/distributable]
6230	gsm-1.0-pl10	ansic=6230
		[BSD]
6180	sash-3.4	ansic=6180
		[GPL]
6122	at-3.1.8	sh=2777,ansic=1510,perl=1220,yacc=524,lex=91
		[GPL]
6116	lslk	ansic=5325,sh=791
		[Free]
6093	indent-2.2.6	ansic=6070,sh=23
		[GPL]
6090	joystick-1.2.15	ansic=6086,sh=4
		[GPL]
6053	sysvinit-2.78	ansic=5276,sh=777
		[GPL]
6035	ytalk-3.1.1	ansic=6035
		[BSD]
5995	isapnptools-1.22	ansic=4430,yacc=1382,perl=123,sh=60
		[GPL]
5990	gdm-2.0beta2	ansic=5878,sh=112
		[LGPL/GPL]
5852	irda-utils-0.9.13	ansic=5535,sh=234,perl=83
		[GPL]
5753	perl-NKF-1.71	sh=3692,ansic=1508,perl=553
		[BSD]
5577	acct-6.3.2	ansic=5016,cpp=287,sh=274
		[GPL]
5567	kdetoys-2.1.1	cpp=5296,ansic=194,perl=64,sh=13
		[GPL]
5526	efax-0.9	ansic=4570,sh=956
		[GPL]
5431	firewall-config-0.95	cpp=2808,perl=1295,ansic=699,sh=629
		[GPL]
5383	bzip2-1.0.1	ansic=5383
		[BSD]
5283	semi-1.13.7	lisp=5283

5213	Xconfigurator-4.9.27	ansic=5161,perl=52 [GPL]
5200	initscripts-5.83	sh=3099,ansic=1998,python=65,csch=38 [GPL]
5134	python-xmlrpc-1.4	python=2733,perl=1451,ansic=950 [GPL]
5112	netkit-ftp-0.17	ansic=5112 [BSD]
5096	libtool-1.3.5	sh=3404,ansic=1692 [GPL]
5064	up2date-2.5.2	python=4746,ansic=264,sh=54 [GPL]
4921	rep-gtk-0.15	ansic=3577,lisp=1102,sh=242 [GPL]
4894	xcpustate-2.5	ansic=4894 [Freely redistributable]
4854	kon2-0.3.9b	ansic=4763,sh=91 [Distributable]
4792	libelf-0.6.4	ansic=3310,sh=1482 [Distributable]
4731	asp2php-0.75.11	ansic=4731 [GPL]
4638	printconf-0.2.12	ansic=2098,python=2071,perl=354,sh=115 [GPL]
4596	sysstat-3.3.5	ansic=3945,tcl=517,sh=134 [GPL]
4562	rp-pppoe-2.6	ansic=3383,sh=1179 [GPL]
4503	cipe-1.4.5	ansic=4155,asm=289,perl=45,sh=14 [GPL]
4436	libole2-0.1.7	ansic=4222,sh=214 [GPL]
4430	gperf-2.7	cpp=2947,exp=745,ansic=695,sh=43 [GPL]
4400	libPropList-0.10.1	ansic=4089,lex=172,yacc=139 [LGPL]
4395	plugger-3.2	ansic=3803,cpp=576,java=16 [GPL]
4287	kakasi-2.3.2	ansic=4287 [GPL]
4281	pciutils-2.1.8	ansic=4206,sh=75 [GPL]
4245	SDL_image-1.1.0	sh=2216,ansic=2029 [LGPL]
4185	memprof-0.4.1	ansic=4185 [GPL]
4172	fnlib-0.5	ansic=2509,sh=1663 [LGPL]
4043	gtk-doc-0.4b1	perl=3917,sh=126 [LGPL]
3916	syslogd-1.4rh	ansic=3854,sh=62 [GPL]
3896	ltrace-0.3.10	ansic=2986,sh=854,awk=56 [GPL]
3783	xdaliclock-2.18	ansic=3783 [MIT]
3780	ipchains-1.3.10	ansic=2781,sh=999 [GPL]
3765	quota-3.00	ansic=3765 [BSD]
3716	tcp_wrappers_7.6	ansic=3716 [Distributable]
3706	ksymoops-2.4.0	ansic=3706 [GPL]
3680	dosfstools-2.2	ansic=3680

		[GPL]
3641	jpeg-6b	sh=2967,ansic=674
		[Distributable]
3603	netkit-rsh-0.17-pre20000412	ansic=3603
		[BSD]
3591	autofs-3.1.7	ansic=3155,sh=436
		[GPL]
3442	yp-tools-2.4	ansic=3419,sh=23
		[GPL]
3415	ext2ed-0.1	ansic=3415
		[GPL]
3401	netkit-tftp-0.17	ansic=3401
		[BSD]
3334	mtx-1.2.10	ansic=3068,python=141,perl=117,sh=8
		[GPL]
3219	playmidi-2.4	ansic=3217,sed=2
		[GPL]
3207	dhcpcd-1.3.18-pl8	ansic=2927,sh=280
		[GPL]
3107	file-3.33	ansic=3038,perl=46,sh=23
		[Distributable]
3101	magicdev-0.3.5	ansic=3101
		[GPL]
3073	mpage-2.5.1	ansic=3004,sh=69
		[BSD]
3012	apmd	ansic=2617,sh=395
		[GPL]
2925	ypbind-mt-1.7	ansic=2875,sh=50
		[GPL]
2915	autorun-2.65	sh=2137,cpp=778
		[GPL]
2888	vixie-cron-3.0.1	ansic=2875,sh=13
		[Distributable]
2842	rhn_register-1.3.1	python=2835,sh=7
		[GPL]
2835	gkermit-1.0	ansic=2835
		[GPL]
2791	xjewel-1.6	ansic=2791
		[MIT]
2790	usermode-1.42	ansic=2772,sh=18
		[GPL]
2758	autoconf-2.13	sh=2226,perl=283,exp=167,ansic=82
		[GPL]
2727	libghttp-1.0.8	ansic=2727
		[LGPL]
2720	sysctlconfig-0.13	ansic=2697,sh=23
		[GPL]
2717	mtr-0.42	ansic=2717
		[GPL]
2680	authconfig-4.1.6	ansic=2680
		[GPL]
2676	psutils	ansic=1741,perl=509,sh=426
		[Distributable]
2661	cdp-0.33	ansic=2661
		[GPL]
2653	pythonlib-1.28	python=2653
		[GPL]
2636	raidtools	ansic=2630,sh=6
		[GPL]
2627	aumix-2.7	ansic=2448,sh=179
		[GPL]
2569	stunnel-3.13	ansic=2519,perl=47,sh=3
		[GPL]
2542	gnome-kerberos-0.2.2	ansic=2542
		[GPL]

2475	DBD-Pg-0.95	ansic=1415,perl=1060 [Distributable]
2464	sliplogin-2.1.1	ansic=2273,sh=143,perl=48 [BSD]
2458	apacheconf-0.7	python=2451,ansic=5,sh=2 [GPL]
2449	wireless_tools.20	ansic=2449 [GPL]
2443	netkit-routed-0.17	ansic=2443 [BSD]
2409	nc	ansic=1672,sh=737 [GPL]
2342	docbook-utils-0.6	perl=1420,sh=922 [GPL]
2317	kde-i18n-2.1.1	cpp=1320,perl=841,sh=156 [GPL]
2233	bindconf-1.4	python=2233 [GPL]
2155	unarj-2.43	ansic=2155 [Distributable]
2113	wmconfig-0.9.10	ansic=2077,sh=36 [GPL]
2103	slocate-2.5	ansic=1992,sh=111 [GPL]
2100	pump-0.8.11	ansic=2100 [MIT]
2099	Perl-RPM-0.291	perl=1985,ansic=114 [Distributable]
2098	rpmlint-0.28	python=2095,sh=3 [GPL]
2097	pam_krb5-1.31-1	ansic=1938,yacc=144,lex=15 [LGPL]
2065	units-1.55	ansic=1963,perl=102 [GPL]
2064	netkit-ntalk-0.17-pre20000412	ansic=2064 [BSD]
2030	zip-2.3	ansic=2030 [Distributable]
1990	sndconfig-0.64.8	ansic=1990 [GPL]
1987	SGMLSpM	perl=1725,lisp=262 [GPL]
1984	cleanfeed-0.95.7b	perl=1984 [Distributable]
1977	cracklib,2.7	ansic=1923,perl=46,sh=8 [Artistic]
1941	isicom	ansic=1898,sh=43 [GPL (not Firmware)]
1917	urlview-0.9	ansic=1680,sh=237 [GPL]
1899	netkit-rusers-0.17	ansic=1899 [BSD]
1899	gnome-lokkit-0.42	ansic=1532,sh=247,perl=120 [GPL]
1841	cdecl-2.5	ansic=1001,yacc=765,lex=75 [Distributable]
1835	which-2.12	ansic=1812,sh=23 [GPL]
1826	adjtimex-1.11	ansic=1674,sh=152 [Distributable]
1771	readline-4.1	ansic=1771 [GPL]
1765	fbset-2.1	ansic=1401,yacc=130,lex=121,perl=113 [GPL]
1737	logrotate-3.5.4	ansic=1570,sh=167

		[GPL]
1727	setserial-2.17	sh=912,ansic=815
		[GPL]
1673	traceroute-1.4a5	ansic=1636,awk=37
		[BSD]
1670	libogg-1.0beta4	ansic=1613,sh=57
		[BSD]
1648	netcfg-2.36	python=1646,sh=2
		[GPL]
1630	psmisc	ansic=1624,sh=6
		[Distributable]
1604	fortune-mod-9708	ansic=1604
		[BSD]
1475	ksconfig-1.2	python=1473,sh=2
		[GPL]
1445	hdparm-3.9	ansic=1362,sh=83
		[Distributable]
1435	ncompress-4.2.4	ansic=1435
		[Distributable]
1395	time-1.7	ansic=1395
		[GPL]
1361	mt-st-0.5b	ansic=1361
		[BSD]
1358	tux-2.0.26	ansic=1008,sh=350
		[GPL]
1336	pl_PL	perl=1291,sh=45
		[Distributable]
1329	nss_db-2.2	ansic=1329
		[GPL]
1313	usbview-1.0	ansic=1313
		[GPL]
1305	bsd-finger-0.17-pre20000412	ansic=1305
		[BSD]
1290	cxhextris	ansic=1290
		[Distributable]
1271	procinfo-17	ansic=1145,perl=126
		[GPL]
1209	auth_ldap-1.4.7	ansic=1209
		[GPL]
1194	passwd-0.64.1	ansic=1194
		[BSD]
1159	anacron-2.3	ansic=1159
		[GPL]
1140	xbill-2.0	c++=1140
		[MIT]
1110	smpeg-xmms-0.3.3	ansic=1087,sh=23
		[GPL]
1055	kpppload-1.04	c++=1050,sh=5
		[GPL]
1031	netkit-rwho-0.17	ansic=1031
		[BSD]
1013	trojka	ansic=1013
		[Distributable]
987	xmailbox-2.5	ansic=987
		[MIT]
978	kcc	ansic=978
		[GPL]
957	chkconfig-1.2.22	ansic=957
		[GPL]
953	ElectricFence-2.2.2	ansic=944,sh=9
		[GPL]
936	portmap_4	ansic=936
		[BSD]
928	mouseconfig-4.21	ansic=928
		[Distributable]

920	glms-1.03	ansic=761,sh=159 [GPL]
919	mkinitrd-3.0.10	ansic=600,sh=319 [GPL]
906	File-MMagic-1.06	perl=906 [Distributable]
896	XFree86-jpfonts-2.0	awk=802,perl=87,sh=7 [Distributable]
817	emh-1.10.1	lisp=817
797	termcap-2.0.8	ansic=797 [LGPL]
787	xsysinfo-1.7	ansic=787 [MIT]
778	lockdev-1.0.0	ansic=653,perl=102,sh=23 [LGPL]
770	giftrans-1.12.2	ansic=770 [BSD]
745	MAKEDEV-3.1.0	ansic=718,sh=27 [GPL]
728	tree-1.2	ansic=728 [GPL]
696	rarpd	ansic=615,sh=81 [GPL]
659	man-pages-ja-0.4	sh=659 [Distributable]
659	eject-2.0.2	ansic=659 [GPL]
649	br.ispell-2.4	awk=649 [GPL]
626	freecdb-0.62	ansic=614,sh=12 [Distributable]
616	diffstat-1.27	ansic=616 [Distributable]
615	hotplug-2001_02_14	sh=615 [GPL]
606	netscape-4.76	sh=606 [Proprietary]
549	genromfs-0.3	ansic=549 [GPL]
548	tksysv-1.3	tcl=526,sh=22 [GPL]
511	docbook-style-dsssl-1.59	perl=511 [Distributable]
506	netkit-bootparamd-0.17-pre20000412	ansic=506 [BSD]
499	xinitrc-3.6	sh=499 [Public domain]
497	locale_config-0.2	ansic=497 [GPL]
497	tmpwatch-2.7.1	ansic=345,sh=152 [GPL]
492	dos2unix-3.1	ansic=492 [Freely distributable]
459	control-panel-3.18	ansic=374,tcl=85 [GPL]
444	unix2dos-2.2	ansic=444 [Distributable]
401	sgml-common-0.5	sh=401 [GPL]
369	kbdconfig-1.9.12	ansic=369 [GPL]
368	vlock-1.3	ansic=368 [GPL]
367	timetool-2.8	tcl=367 [GPL]

355	Text-Kakasi-1.04	perl=355 [GPL]
354	chkfontpath-1.9.5	ansic=354 [GPL]
350	timeconfig-3.2	ansic=319,sh=31 [GPL]
348	mkbootdisk-1.4.2	sh=348 [GPL]
347	mingetty-0.9.4	ansic=347 [GPL]
344	zlib-1.1.3	ansic=174,sh=170 [BSD]
302	symlinks-1.2	ansic=302 [Distributable]
301	xsri-1.0	ansic=301 [GPL]
294	netkit-rwall-0.17	ansic=294 [BSD]
291	sysreport-1.2	sh=291 [GPL]
290	biff+comsat-0.17-pre20000412	ansic=290 [BSD]
276	stat-2.2	ansic=276 [GPL]
262	bdflush-1.5	ansic=203,asm=59 [Distributable]
250	whois-1.0.6	ansic=249,sh=1 [LGPL]
249	rmail-mime-1.13.0	lisp=249
244	man-pages-cs-0.14	sh=244 [Distributable]
240	open-1.4	ansic=240 [GPL]
238	dbskkd-cdb-1.01	ansic=227,sh=11 [GPL]
236	xtoolwait-1.2	ansic=236 [GPL]
229	mkkickstart-2.3	sh=229 [GPL]
222	utempter-0.5.2	ansic=222 [MIT]
221	hellas	sh=179,perl=42 [Distributable]
213	rhmask	ansic=213 [GPL]
152	gimp-data-extras-1.2.0	sh=152 [GPL]
134	rdate-1.0	ansic=134 [GPL]
133	unixfonts	sh=133 [GPL]
131	statserial-1.1	ansic=121,sh=10 [BSD]
94	mktemp-1.5	ansic=94 [BSD]
93	koi8-ub	perl=93 [Distributable]
87	modemtool-1.22	python=73,sh=14 [GPL]
75	setuptool-1.7	ansic=75 [GPL]
56	shaper	ansic=56 [GPL]
40	aspell-da-1.4.9	perl=40 [GPL]

11	words-2	sh=11
		[Free]
10	xtt-fonts-0.19990222	sh=10
		[Distributable]
7	trXFree86-2.1.2	tcl=7
		[Distributable]
6	aspell-swedish-0.2	sh=6
		[GPL]
1	man-pages-it-0.3.0	sed=1
		[Distributable]
0	ghostscript-fonts-5.50	(none)
		[GPL]
0	users-guide-1.2	(none)
		[GPL]
0	skkdic-20010122	(none)
		[GPL]
0	urw-fonts-2.0	(none)
		[GPL,]
0	Kappa20-0.3	(none)
		[Public domain]
0	oo	(none)
		[GPL]
0	jisksp16-1990	(none)
		[Public domain]
0	docbook-dtd41-xml-1.0	(none)
		[Distributable]
0	docbook-dtd41-sgml-1.0	(none)
		[Distributable]
0	docbook-dtd40-sgml-1.0	(none)
		[Distributable]
0	docbook-dtd31-sgml-1.0	(none)
		[Distributable]
0	docbook-dtd30-sgml-1.0	(none)
		[Distributable]
0	specspo-7.1	(none)
		[GPL]
0	desktop-backgrounds-1.1	(none)
		[LGPL]
0	manpages-ru-0.6	(none)
		[Distributable]
0	jisksp14	(none)
		[Public domain]
0	src_top_dir	(none)
0	manpages-es-0.6a	(none)
		[Distributable]
0	man-pages-de-0.2	(none)
		[Distributable]
0	manpages-da-0.1.1	(none)
		[Distributable]
0	jadetex-3.3	(none)
		[Distributable]
0	man-pages-1.35	(none)
		[Distributable]
0	indexhtml-7.1	(none)
		[Distributable]
0	man-fr-0.9	(none)
		[Distributable]
0	gnome-audio-1.0.0	(none)
		[LGPL]
0	rootfiles	(none)
		[Public domain]
0	aspell-ca-0.1	(none)
		[GPL]
0	pvm	(none)
		[Freely distributable]

0	knm_new-1.1	(none)
		[GPL]
0	caching-nameserver-7.1	(none)
		[Public domain]
0	redhat-logos	(none)
0	aspell-de-0.1.1	(none)
		[GPL]
0	aspell-es-0.2	(none)
		[GPL]
0	aspell-it-0.1	(none)
		[GPL]
0	aspell-nl-0.1	(none)
		[GPL]
0	wvdial-1.41	(none)
		[LGPL]
0	lost+found	(none)
0	aspell-no-0.2	(none)
		[GPL]
0	francais	(none)
		[GPL]
0	setup-2.4.7	(none)
		[Public domain]
0	mailcap-2.1.4	(none)
		[Public domain]
0	nkf-1.92	(none)
		[BSD-like]
0	watanabe-vf	(none)
		[Distributable]
0	abidistfiles	(none)
		[GPL]

ansic:	21461450	(71.18%)
cpp:	4575907	(15.18%)
sh:	793238	(2.63%)
lisp:	722430	(2.40%)
asm:	565536	(1.88%)
perl:	562900	(1.87%)
fortran:	493297	(1.64%)
python:	285050	(0.95%)
tcl:	213014	(0.71%)
java:	147285	(0.49%)
yacc:	122325	(0.41%)
exp:	103701	(0.34%)
lex:	41967	(0.14%)
awk:	17431	(0.06%)
objc:	14645	(0.05%)
ada:	13200	(0.04%)
csh:	10753	(0.04%)
pascal:	4045	(0.01%)
sed:	3940	(0.01%)

Licenses:

15185987	(50.36%)	GPL
2498084	(8.28%)	MIT
2305001	(7.64%)	LGPL
2065224	(6.85%)	MPL
1826601	(6.06%)	Distributable
1315348	(4.36%)	BSD
907867	(3.01%)	BSD-like
766859	(2.54%)	Freely distributable
692561	(2.30%)	Free
455980	(1.51%)	GPL, LGPL
323730	(1.07%)	GPL/MIT

321123	(1.07%)	Artistic or GPL
191379	(0.63%)	PHP
173161	(0.57%)	Apache-like
161451	(0.54%)	OpenLDAP
146647	(0.49%)	LGPL/GPL
103439	(0.34%)	GPL (programs), relaxed LGPL (libraries), and public domain (docs)
103291	(0.34%)	Apache
73650	(0.24%)	W3C
73356	(0.24%)	IBM Public License
66554	(0.22%)	University of Washington's Free-Fork License
59354	(0.20%)	Public domain
39828	(0.13%)	GPL and Artistic
31019	(0.10%)	GPL or BSD
25944	(0.09%)	GPL/BSD
20740	(0.07%)	Not listed
20722	(0.07%)	MIT-like
18353	(0.06%)	GPL/LGPL
12987	(0.04%)	Distributable - most of it GPL
8031	(0.03%)	Python
6234	(0.02%)	GPL/distributable
4894	(0.02%)	Freely redistributable
1977	(0.01%)	Artistic
1941	(0.01%)	GPL (not Firmware)
606	(0.00%)	Proprietary

Percentage of Licenses containing selected key phrases:

16673212	(55.30%)	GPL
3029420	(10.05%)	LGPL
2842536	(9.43%)	MIT
2612681	(8.67%)	distributable
2280178	(7.56%)	BSD
2065224	(6.85%)	MPL
162793	(0.54%)	public domain

Total Physical Source Lines of Code (SLOC)	=	30152114
Estimated Development Effort in Person-Years (Person-Months)	=	7955.75 (95468.99)
(Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))		
Estimated Schedule in Years (Months)	=	6.53 (78.31)
(Basic COCOMO model, Months = 2.5 * (person-months**0.38))		
Total Estimated Cost to Develop	=	\$ 1074713481
(average salary = \$56286/year, overhead = 2.4).		

Please credit this data as "generated using 'SLOCCount' by David A. Wheeler."

```
Have a non-directory at the top, so creating directory top_dir
Adding /usr/src/redhat/BUILD/kernel-2.4.2/linux/COPYING to top_dir
Adding /usr/src/redhat/BUILD/kernel-2.4.2/linux/CREDITS to top_dir
Creating filelist for Documentation
Adding /usr/src/redhat/BUILD/kernel-2.4.2/linux/MAINTAINERS to top_dir
Adding /usr/src/redhat/BUILD/kernel-2.4.2/linux/Makefile to top_dir
Adding /usr/src/redhat/BUILD/kernel-2.4.2/linux/README to top_dir
Adding /usr/src/redhat/BUILD/kernel-2.4.2/linux/REPORTING-BUGS to top_dir
Adding /usr/src/redhat/BUILD/kernel-2.4.2/linux/Rules.make to top_dir
Creating filelist for arch
Creating filelist for configs
Creating filelist for devfsd
Creating filelist for drivers
Creating filelist for fs
Creating filelist for include
Creating filelist for init
Creating filelist for ipc
Creating filelist for kernel
Creating filelist for lib
Creating filelist for mm
Creating filelist for net
Creating filelist for scripts
Categorizing files.
Warning: in include, number of duplicates=101
Counting files.
Counting SLOC.
```

SLOC	Directory	SLOC-by-Language (Sorted)
1400422	drivers	ansic=1396471,asm=1541,yacc=1147,perl=816,lex=302,sh=145
446030	arch	ansic=301963,asm=143011,sh=492,perl=449,awk=115
240761	include	ansic=240030,cpp=731
168277	fs	ansic=168277
150627	net	ansic=150422,awk=133,sed=72
9999	mm	ansic=9999
9293	kernel	ansic=9293
8098	scripts	ansic=5225,sh=1540,perl=757,tcl=576
2459	ipc	ansic=2459
1940	devfsd	ansic=1940
1453	lib	ansic=1453
858	Documentation	sh=858
702	init	ansic=702
0	configs	(none)
0	top_dir	(none)

ansic:	2288234	(93.74%)
asm:	144552	(5.92%)
sh:	3035	(0.12%)
perl:	2022	(0.08%)
yacc:	1147	(0.05%)
cpp:	731	(0.03%)
tcl:	576	(0.02%)
lex:	302	(0.01%)
awk:	248	(0.01%)
sed:	72	(0.00%)

Total Physical Source Lines of Code (SLOC)	=	2440919
Estimated Development Effort in Person-Years (Person-Months)	=	721.04 (8652.51)
(Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))		
Estimated Schedule in Years (Months)	=	6.53 (78.35)

(Basic COCOMO model, Months = $2.5 * (\text{person-months}^{0.38})$)

Estimated Average Number of Developers (Effort/Schedule) = 110.43

Total Estimated Cost to Develop = \$ 97403073

(average salary = \$56286/year, overhead = 2.4).

Please credit this data as "generated using 'SLOCCount' by David A. Wheeler."

/usr/src/redhat/BUILD/4Suite-0.10.1:4Suite.spec
/usr/src/redhat/BUILD/Canna35b2:Canna.spec
/usr/src/redhat/BUILD/DBD-Pg-0.95:perl-DBD-Pg.spec
/usr/src/redhat/BUILD/DBI-1.14:perl-DBI.spec
/usr/src/redhat/BUILD/Distutils-1.0.1:Distutils.spec
/usr/src/redhat/BUILD/ElectricFence-2.2.2:ElectricFence.spec
/usr/src/redhat/BUILD/File-MMagic-1.06:perl-File-MMagic.spec
/usr/src/redhat/BUILD/FreeWnn-1.1.1-a017:FreeWnn.spec
/usr/src/redhat/BUILD/ImageMagick-5.2.7:ImageMagick.spec
/usr/src/redhat/BUILD/Inti-0.6preview:inti.spec
/usr/src/redhat/BUILD/Kappa20-0.3:kappa20.spec
/usr/src/redhat/BUILD/LAPACK:lapack.spec
/usr/src/redhat/BUILD/LPRng-3.7.4:LPRng.spec
/usr/src/redhat/BUILD/MAKEDEV-3.1.0:MAKEDEV.spec
/usr/src/redhat/BUILD/Maelstrom-3.0.1:Maelstrom.spec
/usr/src/redhat/BUILD/Mesa-3.4:Mesa.spec
/usr/src/redhat/BUILD/Msql-Mysql-modules-1.2215:perl-DBD-Mysql.spec
/usr/src/redhat/BUILD/ORBit-0.5.7:ORBit.spec
/usr/src/redhat/BUILD/Perl-RPM-0.291:Perl-RPM.spec
/usr/src/redhat/BUILD/Python-1.5.2:python.spec
/usr/src/redhat/BUILD/SDL-1.1.7:SDL.spec
/usr/src/redhat/BUILD/SDL_image-1.1.0:SDL_image.spec
/usr/src/redhat/BUILD/SDL_mixer-1.1.0:SDL_mixer.spec
/usr/src/redhat/BUILD/SGMLSpM:perl-SGMLSpM.spec
/usr/src/redhat/BUILD/Text-Kakasi-1.04:perl-Text-Kakasi.spec
/usr/src/redhat/BUILD/TiMidity+-2.10.3a2:timidity.spec
/usr/src/redhat/BUILD/VFLib2-2.25.1:VFLib2.spec
/usr/src/redhat/BUILD/WindowMaker-0.64.0:WindowMaker.spec
/usr/src/redhat/BUILD/XFree86-4.0.3:XFree86.spec
/usr/src/redhat/BUILD/XFree86-jpfonts-2.0:XFree86-jpfonts.spec
/usr/src/redhat/BUILD/Xaw3d-1.5:Xaw3d.spec
/usr/src/redhat/BUILD/Xconfigurator-4.9.27:Xconfigurator.spec
/usr/src/redhat/BUILD/a2ps-4.13:a2ps.spec
/usr/src/redhat/BUILD/abi:abiword.spec
/usr/src/redhat/BUILD/abidistfiles:abiword.spec
/usr/src/redhat/BUILD/acct-6.3.2:psacct-6.3.spec
/usr/src/redhat/BUILD/adjtimex-1.11:adjtimex.spec
/usr/src/redhat/BUILD/alchemist-0.16:alchemist.spec
/usr/src/redhat/BUILD/am-utils-6.0.5:am-utils.spec
/usr/src/redhat/BUILD/amanda-2.4.2p2:amanda.spec
/usr/src/redhat/BUILD/anaconda-7.1:anaconda.spec
/usr/src/redhat/BUILD/anacron-2.3:anacron.spec
/usr/src/redhat/BUILD/apache_1.3.19:apache.spec
/usr/src/redhat/BUILD/apacheconf-0.7:apacheconf.spec
/usr/src/redhat/BUILD/apel-10.2:semi-emacs.spec
/usr/src/redhat/BUILD/apmd:apmd.spec
/usr/src/redhat/BUILD/ash-0.3.7.orig:ash.spec
/usr/src/redhat/BUILD/asp2php-0.75.11:asp2php.spec
/usr/src/redhat/BUILD/aspell-.32.6:aspell.spec
/usr/src/redhat/BUILD/aspell-ca-0.1:aspell-ca.spec
/usr/src/redhat/BUILD/aspell-da-1.4.9:aspell-da.spec
/usr/src/redhat/BUILD/aspell-de-0.1.1:aspell-de.spec
/usr/src/redhat/BUILD/aspell-es-0.2:aspell-es.spec
/usr/src/redhat/BUILD/aspell-it-0.1:aspell-it.spec
/usr/src/redhat/BUILD/aspell-nl-0.1:aspell-nl.spec
/usr/src/redhat/BUILD/aspell-no-0.2:aspell-no.spec
/usr/src/redhat/BUILD/aspell-swedish-0.2:aspell-sv.spec
/usr/src/redhat/BUILD/at-3.1.8:at.spec
/usr/src/redhat/BUILD/audiofile-0.1.11:audiofile.spec
/usr/src/redhat/BUILD/aumix-2.7:aumix.spec
/usr/src/redhat/BUILD/auth_ldap-1.4.7:auth_ldap.spec
/usr/src/redhat/BUILD/authconfig-4.1.6:authconfig.spec
/usr/src/redhat/BUILD/autoconf-2.13:autoconf.spec
/usr/src/redhat/BUILD/autofs-3.1.7:autofs.spec
/usr/src/redhat/BUILD/automake-1.4:automake.spec

/usr/src/redhat/BUILD/autorun-2.65:autorun.spec
/usr/src/redhat/BUILD/awesfx-0.4.3a:awesfx.spec
/usr/src/redhat/BUILD/balsa-1.1.1:balsa.spec
/usr/src/redhat/BUILD/bash-2.04:bash.spec
/usr/src/redhat/BUILD/bc-1.06:bc.spec
/usr/src/redhat/BUILD/bdflush-1.5:bdflush-1.5.spec
/usr/src/redhat/BUILD/biff+comsat-0.17-pre20000412:comsat.spec
/usr/src/redhat/BUILD/bind-9.1.0:bind.spec
/usr/src/redhat/BUILD/bindconf-1.4:bindconf.spec
/usr/src/redhat/BUILD/binutils-2.10.91.0.2:binutils.spec
/usr/src/redhat/BUILD/bison-1.28:bison.spec
/usr/src/redhat/BUILD/blt2.4u:blt.spec
/usr/src/redhat/BUILD/br.ispell-2.4:aspell-pt_BR.spec
/usr/src/redhat/BUILD/bsd-finger-0.17-pre20000412:finger.spec
/usr/src/redhat/BUILD/bug-buddy-1.2:bug-buddy.spec
/usr/src/redhat/BUILD/bzip2-1.0.1:bzip2.spec
/usr/src/redhat/BUILD/caching-nameserver-7.1:caching-nameserver.spec
/usr/src/redhat/BUILD/cdecl-2.5:cdecl-2.5.spec
/usr/src/redhat/BUILD/cdp-0.33:cdp.spec
/usr/src/redhat/BUILD/cdparanoia-III-alpha9.7:cdparanoia.spec
/usr/src/redhat/BUILD/cdrecord-1.9:cdrecord.spec
/usr/src/redhat/BUILD/chkconfig-1.2.22:chkconfig.spec
/usr/src/redhat/BUILD/chkfontpath-1.9.5:chkfontpath.spec
/usr/src/redhat/BUILD/cipe-1.4.5:cipe.spec
/usr/src/redhat/BUILD/cleanfeed-0.95.7b:cleanfeed.spec
/usr/src/redhat/BUILD/clime-1.13.6:semi-emacs.spec
/usr/src/redhat/BUILD/console-tools-0.3.3:console-tools.spec
/usr/src/redhat/BUILD/control-center-1.2.2:control-center.spec
/usr/src/redhat/BUILD/control-panel-3.18:control-panel.spec
/usr/src/redhat/BUILD/cpio-2.4.2:cpio.spec
/usr/src/redhat/BUILD/cproto-4.6:cproto.spec
/usr/src/redhat/BUILD/cracklib,2.7:cracklib.spec
/usr/src/redhat/BUILD/ctags-4.0.3:ctags.spec
/usr/src/redhat/BUILD/cvs-1.11:cvs.spec
/usr/src/redhat/BUILD/cxhextris:cxhextris.spec
/usr/src/redhat/BUILD/cyrus-sasl-1.5.24:cyrus-sasl.spec
/usr/src/redhat/BUILD/db-3.1.17:db3.spec
/usr/src/redhat/BUILD/db.1.85:db1.spec
/usr/src/redhat/BUILD/db2:db2.spec
/usr/src/redhat/BUILD/dbskkd-cdb-1.01:dbskkd-cdb.spec
/usr/src/redhat/BUILD/ddskk20010225:ddskk.spec
/usr/src/redhat/BUILD/desktop-backgrounds-1.1:desktop-backgrounds.spec
/usr/src/redhat/BUILD/dhcp-2.0pl5:dhcp.spec
/usr/src/redhat/BUILD/dhcpd-1.3.18-pl8:dhcpd.spec
/usr/src/redhat/BUILD/dia-0.86:dia.spec
/usr/src/redhat/BUILD/dialog-0.9a-20001217:dialog.spec
/usr/src/redhat/BUILD/diffstat-1.27:diffstat.spec
/usr/src/redhat/BUILD/diffutils-2.7:diffutils.spec
/usr/src/redhat/BUILD/dip-3.3.7o:dip.spec
/usr/src/redhat/BUILD/dmalloc-4.8.1:dmalloc.spec
/usr/src/redhat/BUILD/docbook-dtd30-sgml-1.0:docbook-dtd30-sgml.spec
/usr/src/redhat/BUILD/docbook-dtd31-sgml-1.0:docbook-dtd31-sgml.spec
/usr/src/redhat/BUILD/docbook-dtd40-sgml-1.0:docbook-dtd40-sgml.spec
/usr/src/redhat/BUILD/docbook-dtd41-sgml-1.0:docbook-dtd41-sgml.spec
/usr/src/redhat/BUILD/docbook-dtd41-xml-1.0:docbook-dtd41-xml.spec
/usr/src/redhat/BUILD/docbook-style-dsssl-1.59:docbook-style-dsssl.spec
/usr/src/redhat/BUILD/docbook-utils-0.6:docbook-utils.spec
/usr/src/redhat/BUILD/dos2unix-3.1:dos2unix.spec
/usr/src/redhat/BUILD/dosfstools-2.2:dosfstools.spec
/usr/src/redhat/BUILD/doxygen-1.2.6:doxygen.spec
/usr/src/redhat/BUILD/dump-0.4b21:dump.spec
/usr/src/redhat/BUILD/e2fsprogs-1.19:e2fsprogs.spec
/usr/src/redhat/BUILD/ed-0.2:ed.spec
/usr/src/redhat/BUILD/ee-0.3.12:ee.spec
/usr/src/redhat/BUILD/efax-0.9:efax.spec

/usr/src/redhat/BUILD/eject-2.0.2:eject.spec
/usr/src/redhat/BUILD/elm2.5.3:elm.spec
/usr/src/redhat/BUILD/emacs-20.7:emacs.spec
/usr/src/redhat/BUILD/emh-1.10.1:semi-emacs.spec
/usr/src/redhat/BUILD/enlightenment-0.16.4:e.spec
/usr/src/redhat/BUILD/enscript-1.6.1:enscript.spec
/usr/src/redhat/BUILD/esound-0.2.22:esound.spec
/usr/src/redhat/BUILD/exmh-2.2:exmh.spec
/usr/src/redhat/BUILD/expat-1.95.1:expat.spec
/usr/src/redhat/BUILD/expat:abiword.spec
/usr/src/redhat/BUILD/ext2ed-0.1:ext2ed.spec
/usr/src/redhat/BUILD/extace-1.4.4:extace.spec
/usr/src/redhat/BUILD/fbset-2.1:fbset.spec
/usr/src/redhat/BUILD/fetchmail-5.7.4:fetchmail.spec
/usr/src/redhat/BUILD/file-3.33:file.spec
/usr/src/redhat/BUILD/fileutils-4.0.36:fileutils.spec
/usr/src/redhat/BUILD/findutils-4.1.6:findutils.spec
/usr/src/redhat/BUILD/firewall-config-0.95:firewall-config.spec
/usr/src/redhat/BUILD/flex-2.5.4:flex.spec
/usr/src/redhat/BUILD/fnlib-0.5:fnlib.spec
/usr/src/redhat/BUILD/fortune-mod-9708:fortune-mod.spec
/usr/src/redhat/BUILD/francais:aspell-fr.spec
/usr/src/redhat/BUILD/freecdb-0.62:freecdb.spec
/usr/src/redhat/BUILD/freeciv-1.11.4:freeciv.spec
/usr/src/redhat/BUILD/freetype-2.0.1:freetype.spec
/usr/src/redhat/BUILD/ftpcopy-0.3.4:ftpcopy.spec
/usr/src/redhat/BUILD/fvwm-2.2.4:fvwm2.spec
/usr/src/redhat/BUILD/gaim-0.11.0pre4:gaim.spec
/usr/src/redhat/BUILD/gal-0.4.1:gal.spec
/usr/src/redhat/BUILD/gated-public-3_6:gated.spec
/usr/src/redhat/BUILD/gawk-3.0.6:gawk.spec
/usr/src/redhat/BUILD/gcc-2.96-20000731:gcc.spec
/usr/src/redhat/BUILD/gd-1.8.3:gd.spec
/usr/src/redhat/BUILD/gdb+dejaguu-20010316:gdb.spec
/usr/src/redhat/BUILD/gdbm-1.8.0:gdbm.spec
/usr/src/redhat/BUILD/gdk-pixbuf-0.8.0:gdk-pixbuf.spec
/usr/src/redhat/BUILD/gdm-2.0beta2:gdm.spec
/usr/src/redhat/BUILD/gedit-0.9.4:gedit.spec
/usr/src/redhat/BUILD/genromfs-0.3:genromfs.spec
/usr/src/redhat/BUILD/gettext-0.10.35:gettext.spec
/usr/src/redhat/BUILD/gftp-2.0.7b:gftp.spec
/usr/src/redhat/BUILD/ghostscript-fonts-5.50:ghostscript-fonts.spec
/usr/src/redhat/BUILD/giftrans-1.12.2:giftrans.spec
/usr/src/redhat/BUILD/gimp-1.2.1:gimp.spec
/usr/src/redhat/BUILD/gimp-data-extras-1.2.0:gimp-data-extras.spec
/usr/src/redhat/BUILD/gkernit-1.0:gkernit.spec
/usr/src/redhat/BUILD/glade-0.5.9:glade.spec
/usr/src/redhat/BUILD/glib-1.2.9:glib.spec
/usr/src/redhat/BUILD/glibc-2.2.2:glibc.spec
/usr/src/redhat/BUILD/glms-1.03:glms.spec
/usr/src/redhat/BUILD/gmp-3.1.1:gmp.spec
/usr/src/redhat/BUILD/gnome-applets-1.2.4:gnome-applets.spec
/usr/src/redhat/BUILD/gnome-audio-1.0.0:gnome-audio.spec
/usr/src/redhat/BUILD/gnome-core-1.2.4:gnome-core.spec
/usr/src/redhat/BUILD/gnome-games-1.2.0:gnome-games.spec
/usr/src/redhat/BUILD/gnome-kerberos-0.2.2:gnome-kerberos.spec
/usr/src/redhat/BUILD/gnome-libs-1.2.8:gnome-libs.spec
/usr/src/redhat/BUILD/gnome-linuxconf-0.64:gnome-linuxconf.spec
/usr/src/redhat/BUILD/gnome-lokkit-0.42:gnome-lokkit.spec
/usr/src/redhat/BUILD/gnome-media-1.2.0:gnome-media.spec
/usr/src/redhat/BUILD/gnome-objc-1.0.2:gnome-objc.spec
/usr/src/redhat/BUILD/gnome-pim-1.2.0:gnome-pim.spec
/usr/src/redhat/BUILD/gnome-print-0.25:gnome-print.spec
/usr/src/redhat/BUILD/gnome-python-1.0.53:gnome-python.spec
/usr/src/redhat/BUILD/gnome-utils-1.2.1:gnome-utils.spec

/usr/src/redhat/BUILD/gnorp-0.96:gnorp.spec
/usr/src/redhat/BUILD/gnuchess-4.0.pl80:gnuchess.spec
/usr/src/redhat/BUILD/gnumeric-0.61:gnumeric.spec
/usr/src/redhat/BUILD/gnupg-1.0.4:gnupg.spec
/usr/src/redhat/BUILD/gnuplot-3.7.1:gnuplot.spec
/usr/src/redhat/BUILD/gperf-2.7:gperf.spec
/usr/src/redhat/BUILD/gphoto-0.4.3:gphoto.spec
/usr/src/redhat/BUILD/gpm-1.19.3:gpm.spec
/usr/src/redhat/BUILD/gq-0.4.0:gq.spec
/usr/src/redhat/BUILD/gqview-0.8.1:gqview.spec
/usr/src/redhat/BUILD/grep-2.4.2:grep.spec
/usr/src/redhat/BUILD/groff-1.16.1:groff.spec
/usr/src/redhat/BUILD/gs5.50:ghostscript.spec
/usr/src/redhat/BUILD/gsl-0.7:gsl.spec
/usr/src/redhat/BUILD/gsm-1.0-pl10:gsm.spec
/usr/src/redhat/BUILD/gtk+-1.2.9:gtk+.spec
/usr/src/redhat/BUILD/gtk-doc-0.4b1:gtk-doc.spec
/usr/src/redhat/BUILD/gtk-engines-0.10:gtk-engines.spec
/usr/src/redhat/BUILD/gtop-1.0.11:gtop.spec
/usr/src/redhat/BUILD/guile-1.3.4:guile.spec
/usr/src/redhat/BUILD/gv-3.5.8:gv.spec
/usr/src/redhat/BUILD/gzip-1.3:gzip.spec
/usr/src/redhat/BUILD/hdparm-3.9:hdparm.spec
/usr/src/redhat/BUILD/hellas:XFree86-ISO8859-7.spec
/usr/src/redhat/BUILD/hotplug-2001_02_14:hotplug.spec
/usr/src/redhat/BUILD/htdig-3.2.0b3:htdig.spec
/usr/src/redhat/BUILD/ical-2.2:ical.spec
/usr/src/redhat/BUILD/im-140:im.spec
/usr/src/redhat/BUILD/imap-2000:imap.spec
/usr/src/redhat/BUILD/imlib-1.9.8.1:imlib.spec
/usr/src/redhat/BUILD/indent-2.2.6:indent.spec
/usr/src/redhat/BUILD/indexhtml-7.1:indexhtml.spec
/usr/src/redhat/BUILD/initscripts-5.83:initscripts.spec
/usr/src/redhat/BUILD/inn-2.3.1:inn.spec
/usr/src/redhat/BUILD/internet-config-0.40:internet-config.spec
/usr/src/redhat/BUILD/ipchains-1.3.10:ipchains.spec
/usr/src/redhat/BUILD/iproute2:iproute.spec
/usr/src/redhat/BUILD/iptables-1.2.1a:iptables.spec
/usr/src/redhat/BUILD/iptables:iptables.spec
/usr/src/redhat/BUILD/ircii-4.4Z:ircii.spec
/usr/src/redhat/BUILD/irda-utils-0.9.13:irda-utils.spec
/usr/src/redhat/BUILD/isapnptools-1.22:isapnptools.spec
/usr/src/redhat/BUILD/isdn4k-utils:isdn4k-utils.spec
/usr/src/redhat/BUILD/isicom:isicom.spec
/usr/src/redhat/BUILD/jadetex-3.3:jadetex.spec
/usr/src/redhat/BUILD/jed-B0.99-12:jed.spec
/usr/src/redhat/BUILD/jikes-1.13:jikes.spec
/usr/src/redhat/BUILD/jisksp14:jisksp14.spec
/usr/src/redhat/BUILD/jisksp16-1990:jisksp16-1990.spec
/usr/src/redhat/BUILD/joe:joe.spec
/usr/src/redhat/BUILD/joystick-1.2.15:joystick.spec
/usr/src/redhat/BUILD/jpeg-6a:libjpeg-6a.spec
/usr/src/redhat/BUILD/jpeg-6b:libjpeg.spec
/usr/src/redhat/BUILD/kaffe-1.0.6:kaffe.spec
/usr/src/redhat/BUILD/kakasi-2.3.2:kakasi.spec
/usr/src/redhat/BUILD/kbdconfig-1.9.12:kbdconfig.spec
/usr/src/redhat/BUILD/kcc:kcc.spec
/usr/src/redhat/BUILD/kdbg-1.2.0:kdbg.spec
/usr/src/redhat/BUILD/kde-il8n-2.1.1:kde-il8n-2.1.spec
/usr/src/redhat/BUILD/kdeadmin-2.1.1:kdeadmin-2.1.spec
/usr/src/redhat/BUILD/kdebase-2.1.1:kdebase-2.1.spec
/usr/src/redhat/BUILD/kdebindings-2.1.1:kdebindings-2.1.spec
/usr/src/redhat/BUILD/kdegames-2.1.1:kdegames-2.1.spec
/usr/src/redhat/BUILD/kdegraphics-2.1.1:kdegraphics-2.1.spec
/usr/src/redhat/BUILD/kdelibs-2.1.1:kdelibs-2.1.spec

/usr/src/redhat/BUILD/kdemultimedia-2.1.1:kdemultimedia-2.1.spec
/usr/src/redhat/BUILD/kdenetwork-2.1.1:kdenetwork-2.1.spec
/usr/src/redhat/BUILD/kdepim-2.1.1:kdepim-2.1.spec
/usr/src/redhat/BUILD/kdesdk-2.1.1:kdesdk-2.1.spec
/usr/src/redhat/BUILD/kdesupport-2.1:kdesupport-2.1.spec
/usr/src/redhat/BUILD/kdetoys-2.1.1:kdetoys-2.1.spec
/usr/src/redhat/BUILD/kdeutils-2.1.1:kdeutils-2.1.spec
/usr/src/redhat/BUILD/kdevelop-1.4.1:kdevelop-1.4.spec
/usr/src/redhat/BUILD/kdoc-2.1.1:kdoc-2.1.spec
/usr/src/redhat/BUILD/kernel-2.4.2:kernel-2.4.spec
/usr/src/redhat/BUILD/kinput2-v3:kinput2.spec
/usr/src/redhat/BUILD/knm_new-1.1:knm_new.spec
/usr/src/redhat/BUILD/koffice-2.0.1:koffice-2.0.spec
/usr/src/redhat/BUILD/koi8-ub:XFree86-KOI8-R.spec
/usr/src/redhat/BUILD/kon2-0.3.9b:kon2.spec
/usr/src/redhat/BUILD/kpppload-1.04:kpppload.spec
/usr/src/redhat/BUILD/krb4-1.0.5:krbafs.spec
/usr/src/redhat/BUILD/krb5-1.2.2:krb5.spec
/usr/src/redhat/BUILD/ksconfig-1.2:ksconfig.spec
/usr/src/redhat/BUILD/ksymoops-2.4.0:ksymoops.spec
/usr/src/redhat/BUILD/kterm-6.2.0:kterm.spec
/usr/src/redhat/BUILD/kudzu-0.98.10:kudzu.spec
/usr/src/redhat/BUILD/lam-6.5.1:lam.spec
/usr/src/redhat/BUILD/lclint-2.5q:lclint.spec
/usr/src/redhat/BUILD/less-358:less.spec
/usr/src/redhat/BUILD/libPropList-0.10.1:libPropList.spec
/usr/src/redhat/BUILD/libelf-0.6.4:libelf-0.6.4.spec
/usr/src/redhat/BUILD/libgcj:libgcj.spec
/usr/src/redhat/BUILD/libghttp-1.0.8:libghttp.spec
/usr/src/redhat/BUILD/libglade-0.14:libglade.spec
/usr/src/redhat/BUILD/libgtop-1.0.10:libgtop.spec
/usr/src/redhat/BUILD/libiconv:abiword.spec
/usr/src/redhat/BUILD/libmng-1.0.0:libmng.spec
/usr/src/redhat/BUILD/libodbc++-0.2.2pre4:libodbc++.spec
/usr/src/redhat/BUILD/libogg-1.0beta4:libogg.spec
/usr/src/redhat/BUILD/libole2-0.1.7:libole2.spec
/usr/src/redhat/BUILD/libpng-1.0.9:libpng.spec
/usr/src/redhat/BUILD/librep-0.13.3:librep.spec
/usr/src/redhat/BUILD/libtool-1.3.5:libtool.spec
/usr/src/redhat/BUILD/libungif-4.1.0b1:libungif.spec
/usr/src/redhat/BUILD/libunicode-0.4:libunicode.spec
/usr/src/redhat/BUILD/libvorbis-1.0beta4:vorbis.spec
/usr/src/redhat/BUILD/libxml-1.8.10:libxml.spec
/usr/src/redhat/BUILD/licq-1.0.2:licq.spec
/usr/src/redhat/BUILD/lilo-21.4.4:lilo.spec
/usr/src/redhat/BUILD/links-0.95:links.spec
/usr/src/redhat/BUILD/linux-86:dev86.spec
/usr/src/redhat/BUILD/linuxconf-1.24r2:linuxconf.spec
/usr/src/redhat/BUILD/lm_sensors-2.5.5:lm_sensors.spec
/usr/src/redhat/BUILD/locale_config-0.2:locale_config.spec
/usr/src/redhat/BUILD/lockdev-1.0.0:lockdev.spec
/usr/src/redhat/BUILD/logrotate-3.5.4:logrotate.spec
/usr/src/redhat/BUILD/lout-3.17:lout.spec
/usr/src/redhat/BUILD/lrzs-0.12.20:lrzs.spec
/usr/src/redhat/BUILD/lslk:lslk.spec
/usr/src/redhat/BUILD/lsof_4.51:lsof.spec
/usr/src/redhat/BUILD/ltrace-0.3.10:ltrace.spec
/usr/src/redhat/BUILD/lv4494:lv.spec
/usr/src/redhat/BUILD/lynx2-8-4:lynx.spec
/usr/src/redhat/BUILD/m4-1.4.1:m4.spec
/usr/src/redhat/BUILD/macutils:macutils.spec
/usr/src/redhat/BUILD/magicdev-0.3.5:magicdev.spec
/usr/src/redhat/BUILD/mailcap-2.1.4:mailcap.spec
/usr/src/redhat/BUILD/mailx-8.1.1:mailx.spec
/usr/src/redhat/BUILD/make-3.79.1:make.spec

/usr/src/redhat/BUILD/man-1.5h1:man.spec
/usr/src/redhat/BUILD/man-fr-0.9:man-pages-fr.spec
/usr/src/redhat/BUILD/man-pages-1.35:man-pages.spec
/usr/src/redhat/BUILD/man-pages-cs-0.14:man-pages-cs.spec
/usr/src/redhat/BUILD/man-pages-de-0.2:man-pages-de.spec
/usr/src/redhat/BUILD/man-pages-it-0.3.0:man-pages-it.spec
/usr/src/redhat/BUILD/man-pages-ja-0.4:man-pages-ja.spec
/usr/src/redhat/BUILD/manpages-da-0.1.1:man-pages-da.spec
/usr/src/redhat/BUILD/manpages-es-0.6a:man-pages-es.spec
/usr/src/redhat/BUILD/manpages-ru-0.6:man-pages-ru.spec
/usr/src/redhat/BUILD/mars_nwe:mars-nwe.spec
/usr/src/redhat/BUILD/mawk-1.3.3:mawk.spec
/usr/src/redhat/BUILD/mc-4.5.51:mc.spec
/usr/src/redhat/BUILD/memprof-0.4.1:memprof.spec
/usr/src/redhat/BUILD/mgetty-1.1.25:mgetty.spec
/usr/src/redhat/BUILD/mikmod-3.1.6:mikmod.spec
/usr/src/redhat/BUILD/mingetty-0.9.4:mingetty.spec
/usr/src/redhat/BUILD/minicom-1.83.1:minicom.spec
/usr/src/redhat/BUILD/mkbootdisk-1.4.2:mkbootdisk.spec
/usr/src/redhat/BUILD/mkinitrd-3.0.10:mkinitrd.spec
/usr/src/redhat/BUILD/mkkickstart-2.3:mkkickstart.spec
/usr/src/redhat/BUILD/mktemp-1.5:mktemp.spec
/usr/src/redhat/BUILD/mm2.7:metamail.spec
/usr/src/redhat/BUILD/mod_dav-1.0.2-1.3.6:mod_dav.spec
/usr/src/redhat/BUILD/mod_perl-1.24_01:mod_perl.spec
/usr/src/redhat/BUILD/modemtool-1.22:modemtool-1.22.spec
/usr/src/redhat/BUILD/modutils-2.4.2:modutils.spec
/usr/src/redhat/BUILD/mount-2.10r:mount.spec
/usr/src/redhat/BUILD/mouseconfig-4.21:mouseconfig.spec
/usr/src/redhat/BUILD/mozilla:mozilla-0.7.spec
/usr/src/redhat/BUILD/mpage-2.5.1:mpage.spec
/usr/src/redhat/BUILD/mpg123-0.59r:mpg123.spec
/usr/src/redhat/BUILD/mt-st-0.5b:mt-st.spec
/usr/src/redhat/BUILD/mtools-3.9.7:mtools.spec
/usr/src/redhat/BUILD/mtr-0.42:mtr.spec
/usr/src/redhat/BUILD/mtx-1.2.10:mtx.spec
/usr/src/redhat/BUILD/multimedia:multimedia.spec
/usr/src/redhat/BUILD/mutt-1.2.5:mutt.spec
/usr/src/redhat/BUILD/mysql-3.23.36:mysql.spec
/usr/src/redhat/BUILD/nasm-0.98:nasm.spec
/usr/src/redhat/BUILD/nc:nc.spec
/usr/src/redhat/BUILD/ncftp-3.0.2:ncftp.spec
/usr/src/redhat/BUILD/ncompress-4.2.4:ncompress.spec
/usr/src/redhat/BUILD/ncpfs-2.2.0.18:ncpfs.spec
/usr/src/redhat/BUILD/ncurses-5.2:ncurses.spec
/usr/src/redhat/BUILD/net-tools-1.57:net-tools.spec
/usr/src/redhat/BUILD/netcfg-2.36:netcfg.spec
/usr/src/redhat/BUILD/netkit-bootparamd-0.17-pre20000412:bootparamd.spec
/usr/src/redhat/BUILD/netkit-ftp-0.17:ftp.spec
/usr/src/redhat/BUILD/netkit-ntalk-0.17-pre20000412:talk.spec
/usr/src/redhat/BUILD/netkit-routed-0.17:routed.spec
/usr/src/redhat/BUILD/netkit-rsh-0.17-pre20000412:rsh.spec
/usr/src/redhat/BUILD/netkit-rusers-0.17:rusers.spec
/usr/src/redhat/BUILD/netkit-rwall-0.17:rwall.spec
/usr/src/redhat/BUILD/netkit-rwho-0.17:rwho.spec
/usr/src/redhat/BUILD/netkit-telnet-0.17-pre20000412:telnet.spec
/usr/src/redhat/BUILD/netkit-tftp-0.17:tftp.spec
/usr/src/redhat/BUILD/netpbm-9.9:netpbm.spec
/usr/src/redhat/BUILD/netscape-4.76:netscape.spec
/usr/src/redhat/BUILD/newt-0.50.22:newt.spec
/usr/src/redhat/BUILD/nfs-utils-0.3.1:nfs-utils.spec
/usr/src/redhat/BUILD/njamd-0.8.0:njamd.spec
/usr/src/redhat/BUILD/nkf-1.92:nkf.spec
/usr/src/redhat/BUILD/nmh-1.0.4:nmh.spec
/usr/src/redhat/BUILD/nss_db-2.2:nss_db.spec

/usr/src/redhat/BUILD/nss_ldap-149:nss_ldap.spec
/usr/src/redhat/BUILD/ntp-4.0.99k:ntp.spec
/usr/src/redhat/BUILD/nut-0.44.1:nut.spec
/usr/src/redhat/BUILD/nvi-1.79:nvi-ml7n.spec
/usr/src/redhat/BUILD/octave-2.1.33:octave.spec
/usr/src/redhat/BUILD/oo:ttfonts.spec
/usr/src/redhat/BUILD/open-1.4:open-1.4.spec
/usr/src/redhat/BUILD/openjade-1.3:openjade.spec
/usr/src/redhat/BUILD/openldap-1.2.11:openldap12.spec
/usr/src/redhat/BUILD/openldap-2.0.7:openldap.spec
/usr/src/redhat/BUILD/openssh-2.5.2p2:openssh.spec
/usr/src/redhat/BUILD/openssl-0.9.6:openssl.spec
/usr/src/redhat/BUILD/p2c-1.22:p2c.spec
/usr/src/redhat/BUILD/pam-0.74:pam.spec
/usr/src/redhat/BUILD/pam_krb5-1.31-1:pam_krb5.spec
/usr/src/redhat/BUILD/pan-0.9.5:pan.spec
/usr/src/redhat/BUILD/parted-1.4.7:parted.spec
/usr/src/redhat/BUILD/passwd-0.64.1:passwd.spec
/usr/src/redhat/BUILD/patch-2.5.4:patch.spec
/usr/src/redhat/BUILD/pax-1.5:pax.spec
/usr/src/redhat/BUILD/pciutils-2.1.8:pciutils.spec
/usr/src/redhat/BUILD/pcmcia-cs-3.1.24:kernel-pcmcia-cs.spec
/usr/src/redhat/BUILD/pdksh-5.2.14:pdksh.spec
/usr/src/redhat/BUILD/perl-5.6.0:perl.spec
/usr/src/redhat/BUILD/perl-NKF-1.71:perl-NKF.spec
/usr/src/redhat/BUILD/php-4.0.4pl1:php.spec
/usr/src/redhat/BUILD/pidentd-3.0.12:pidentd.spec
/usr/src/redhat/BUILD/pilot-link:pilot-link.spec
/usr/src/redhat/BUILD/pine4.33:pine.spec
/usr/src/redhat/BUILD/pinfo-0.6.0:pinfo.spec
/usr/src/redhat/BUILD/pkgconfig-0.5.0:pkgconfig.spec
/usr/src/redhat/BUILD/pl_PL:man-pages-pl.spec
/usr/src/redhat/BUILD/playmidi-2.4:playmidi.spec
/usr/src/redhat/BUILD/plugger-3.2:plugger.spec
/usr/src/redhat/BUILD/pmake-1.45:pmake.spec
/usr/src/redhat/BUILD/pnm2ppa-1.04:pnm2ppa-1.0.spec
/usr/src/redhat/BUILD/portmap_4:portmap.spec
/usr/src/redhat/BUILD/postgresql-7.0.3:postgresql.spec
/usr/src/redhat/BUILD/ppp-2.4.0:ppp.spec
/usr/src/redhat/BUILD/printconf-0.2.12:printconf.spec
/usr/src/redhat/BUILD/procinfo-17:procinfo.spec
/usr/src/redhat/BUILD/procmail-3.14:procmail.spec
/usr/src/redhat/BUILD/procps-2.0.7:procps.spec
/usr/src/redhat/BUILD/psgml-1.2.1:psgml.spec
/usr/src/redhat/BUILD/psiconv:abiword.spec
/usr/src/redhat/BUILD/psmisc:psmisc.spec
/usr/src/redhat/BUILD/pspell-.11.2:pspell.spec
/usr/src/redhat/BUILD/psutils:psutils.spec
/usr/src/redhat/BUILD/pump-0.8.11:pump.spec
/usr/src/redhat/BUILD/pvm:pvm.spec
/usr/src/redhat/BUILD/pwdb-0.61.1:pwdb.spec
/usr/src/redhat/BUILD/pxe-linux:pxe.spec
/usr/src/redhat/BUILD/python-xmlrpc-1.4:xmlrpc.spec
/usr/src/redhat/BUILD/pythonlib-1.28:pythonlib.spec
/usr/src/redhat/BUILD/qt-2.3.0:qt.spec
/usr/src/redhat/BUILD/quota-3.00:quota.spec
/usr/src/redhat/BUILD/raidtools:raidtools.spec
/usr/src/redhat/BUILD/rarpd:rarpd.spec
/usr/src/redhat/BUILD/rcs-5.7:rcs.spec
/usr/src/redhat/BUILD/rdate-1.0:rdate.spec
/usr/src/redhat/BUILD/rdist-6.1.5:rdist.spec
/usr/src/redhat/BUILD/readline-4.1:readline.spec
/usr/src/redhat/BUILD/redhat-logos:redhat-logos.spec
/usr/src/redhat/BUILD/reiserfsprogs-3.x.0f:reiserfs-utils.spec
/usr/src/redhat/BUILD/rep-gtk-0.15:rep-gtk.spec

/usr/src/redhat/BUILD/rhmask:rhmask.spec
/usr/src/redhat/BUILD/rhn_register-1.3.1:rhn_register.spec
/usr/src/redhat/BUILD/rmail-mime-1.13.0:semi-emacs.spec
/usr/src/redhat/BUILD/rootfiles:rootfiles.spec
/usr/src/redhat/BUILD/rp-pppoe-2.6:rp-pppoe.spec
/usr/src/redhat/BUILD/rp3-1.1.10:rp3.spec
/usr/src/redhat/BUILD/rpm-4.0.2:rpm.spec
/usr/src/redhat/BUILD/rpm2html-1.5:rpm2html.spec
/usr/src/redhat/BUILD/rpmfind-1.6:rpmfind.spec
/usr/src/redhat/BUILD/rpmlint-0.28:rpmlint.spec
/usr/src/redhat/BUILD/rsync-2.4.6:rsync.spec
/usr/src/redhat/BUILD/rxvt-2.7.5:rxvt.spec
/usr/src/redhat/BUILD/samba-2.0.7:samba.spec
/usr/src/redhat/BUILD/sane-1.0.3:sane.spec
/usr/src/redhat/BUILD/sash-3.4:sash.spec
/usr/src/redhat/BUILD/sawfish-0.36:sawfish.spec
/usr/src/redhat/BUILD/scheme-3.2:umb-scheme-3.2.spec
/usr/src/redhat/BUILD/screen-3.9.8:screen.spec
/usr/src/redhat/BUILD/sed-3.02:sed.spec
/usr/src/redhat/BUILD/semi-1.13.7:semi-emacs.spec
/usr/src/redhat/BUILD/sendmail-8.11.2:sendmail.spec
/usr/src/redhat/BUILD/setserial-2.17:setserial.spec
/usr/src/redhat/BUILD/setup-2.4.7:setup.spec
/usr/src/redhat/BUILD/setuptools-1.7:setuptools.spec
/usr/src/redhat/BUILD/sgml-common-0.5:sgml-common.spec
/usr/src/redhat/BUILD/sgml-tools-1.0.9:sgml-tools.spec
/usr/src/redhat/BUILD/sh-utils-2.0:sh-utils.spec
/usr/src/redhat/BUILD/shadow-20000826:shadow-utils.spec
/usr/src/redhat/BUILD/shaper:shapecfg.spec
/usr/src/redhat/BUILD/sharutils-4.2.1:sharutils.spec
/usr/src/redhat/BUILD/skksdic-20010122:skksdic.spec
/usr/src/redhat/BUILD/skksinput-2.03:skksinput.spec
/usr/src/redhat/BUILD/slang-1.4.2:slang.spec
/usr/src/redhat/BUILD/sliplogin-2.1.1:sliplogin.spec
/usr/src/redhat/BUILD/slocate-2.5:slocate.spec
/usr/src/redhat/BUILD/slrn-0.9.6.4:slrn.spec
/usr/src/redhat/BUILD/smpeg-0.4.2:smpeg.spec
/usr/src/redhat/BUILD/smpeg-xmms-0.3.3:smpeg-xmms.spec
/usr/src/redhat/BUILD/sndconfig-0.64.8:sndconfig.spec
/usr/src/redhat/BUILD/sox-12.17.1:sox.spec
/usr/src/redhat/BUILD/specspo-7.1:specspo.spec
/usr/src/redhat/BUILD/squid-2.3.STABLE4:squid.spec
/usr/src/redhat/BUILD/src:Wnn6-SDK.spec
/usr/src/redhat/BUILD/stat-2.2:stat.spec
/usr/src/redhat/BUILD/statserial-1.1:statserial.spec
/usr/src/redhat/BUILD/strace:strace.spec
/usr/src/redhat/BUILD/stunnel-3.13:stunnel.spec
/usr/src/redhat/BUILD/sudo-1.6.3p6:sudo.spec
/usr/src/redhat/BUILD/switchdesk-3.9.5:switchdesk.spec
/usr/src/redhat/BUILD/symlinks-1.2:symlinks-1.2.spec
/usr/src/redhat/BUILD/sysctlconfig-0.13:sysctlconfig.spec
/usr/src/redhat/BUILD/sysklogd-1.4rh:sysklogd.spec
/usr/src/redhat/BUILD/syslinux-1.52:syslinux.spec
/usr/src/redhat/BUILD/sysreport-1.2:sysreport.spec
/usr/src/redhat/BUILD/sysstat-3.3.5:sysstat.spec
/usr/src/redhat/BUILD/sysvinit-2.78:SysVinit.spec
/usr/src/redhat/BUILD/tamago-4.0.6:tamago.spec
/usr/src/redhat/BUILD/taper-6.9b:taper.spec
/usr/src/redhat/BUILD/tar-1.13.19:tar.spec
/usr/src/redhat/BUILD/tcltk-8.3.1:tcltk.spec
/usr/src/redhat/BUILD/tcp_wrappers_7.6:tcp_wrappers.spec
/usr/src/redhat/BUILD/tcpdump-3.4:tcpdump.spec
/usr/src/redhat/BUILD/tcsh-6.10.00:tcsh.spec
/usr/src/redhat/BUILD/teTeX-1.0:tetex.spec
/usr/src/redhat/BUILD/termcap-2.0.8:libtermcap.spec

/usr/src/redhat/BUILD/texinfo-4.0:texinfo.spec
/usr/src/redhat/BUILD/textutils-2.0.11:textutils.spec
/usr/src/redhat/BUILD/tiff-v3.5.5:libtiff.spec
/usr/src/redhat/BUILD/time-1.7:time.spec
/usr/src/redhat/BUILD/timeconfig-3.2:timeconfig.spec
/usr/src/redhat/BUILD/timetool-2.8:timetool.spec
/usr/src/redhat/BUILD/tksysv-1.3:tksysv.spec
/usr/src/redhat/BUILD/tmpwatch-2.7.1:tmpwatch.spec
/usr/src/redhat/BUILD/trXFree86-2.1.2:XFree86-IS08859-9.spec
/usr/src/redhat/BUILD/traceroute-1.4a5:traceroute.spec
/usr/src/redhat/BUILD/transfig.3.2.3c:transfig.spec
/usr/src/redhat/BUILD/tree-1.2:tree.spec
/usr/src/redhat/BUILD/tripwire-2.3.0-50:tripwire.spec
/usr/src/redhat/BUILD/trojka:trojka.spec
/usr/src/redhat/BUILD/tux-2.0.26:tux.spec
/usr/src/redhat/BUILD/ucd-snmp-4.2:ucd-snmp.spec
/usr/src/redhat/BUILD/unarj-2.43:unarj.spec
/usr/src/redhat/BUILD/units-1.55:units.spec
/usr/src/redhat/BUILD/unix2dos-2.2:unix2dos.spec
/usr/src/redhat/BUILD/unixODBC-1.8.13:unixODBC.spec
/usr/src/redhat/BUILD/unixfonts:abiword.spec
/usr/src/redhat/BUILD/unzip-5.41:unzip.spec
/usr/src/redhat/BUILD/up2date-2.5.2:up2date.spec
/usr/src/redhat/BUILD/urlview-0.9:urlview.spec
/usr/src/redhat/BUILD/urw-fonts-2.0:urw-fonts.spec
/usr/src/redhat/BUILD/usbview-1.0:usbview.spec
/usr/src/redhat/BUILD/usermode-1.42:usermode.spec
/usr/src/redhat/BUILD/users-guide-1.2:users-guide.spec
/usr/src/redhat/BUILD/utempter-0.5.2:utempter.spec
/usr/src/redhat/BUILD/util-linux-2.10s:util-linux.spec
/usr/src/redhat/BUILD/uucp-1.06.1:uucp.spec
/usr/src/redhat/BUILD/vim60z:vim.spec
/usr/src/redhat/BUILD/vixie-cron-3.0.1:vixie-cron.spec
/usr/src/redhat/BUILD/vlock-1.3:vlock.spec
/usr/src/redhat/BUILD/vnc_unixsrc:vnc.spec
/usr/src/redhat/BUILD/w3c-libwww-5.2.8:w3c-libwww.spec
/usr/src/redhat/BUILD/watanabe-vf:watanabe.spec
/usr/src/redhat/BUILD/wget-1.6:wget.spec
/usr/src/redhat/BUILD/which-2.12:which-2.spec
/usr/src/redhat/BUILD/whois-1.0.6:whois.spec
/usr/src/redhat/BUILD/wireless_tools.20:wireless-tools.spec
/usr/src/redhat/BUILD/wmakerconf-2.6.1:wmakerconf.spec
/usr/src/redhat/BUILD/wmconfig-0.9.10:wmconfig.spec
/usr/src/redhat/BUILD/words-2:words-2.spec
/usr/src/redhat/BUILD/wu-ftpd:wu-ftpd.spec
/usr/src/redhat/BUILD/wv:abiword.spec
/usr/src/redhat/BUILD/wvdial-1.41:wvdial.spec
/usr/src/redhat/BUILD/x3270-3.2:x3270.spec
/usr/src/redhat/BUILD/xbill-2.0:xbill.spec
/usr/src/redhat/BUILD/xbl-1.0j:xbl.spec
/usr/src/redhat/BUILD/xboard-4.1.0:xboard.spec
/usr/src/redhat/BUILD/xboing:xboing.spec
/usr/src/redhat/BUILD/xcdroast-0.98alpha8:xcdroast.spec
/usr/src/redhat/BUILD/xchat-1.6.3:xchat.spec
/usr/src/redhat/BUILD/xcpustate-2.5:xcpustate.spec
/usr/src/redhat/BUILD/xdaliclock-2.18:xdaliclock.spec
/usr/src/redhat/BUILD/xdelta-1.1.1:xdelta.spec
/usr/src/redhat/BUILD/xemacs-21.1.14:xemacs.spec
/usr/src/redhat/BUILD/xfig.3.2.3c:xfig.spec
/usr/src/redhat/BUILD/xgammon-0.98:xgammon.spec
/usr/src/redhat/BUILD/xinetd-2.1.8.9pre14:xinetd.spec
/usr/src/redhat/BUILD/xinitrc-3.6:xinitrc.spec
/usr/src/redhat/BUILD/xjewel-1.6:xjewel.spec
/usr/src/redhat/BUILD/xlispstat-3-52-18:xlispstat.spec
/usr/src/redhat/BUILD/xloadimage-4.1:xloadimage-4.1.spec

/usr/src/redhat/BUILD/xlockmore-4.17.2:xlockmore.spec
/usr/src/redhat/BUILD/xmailbox-2.5:xmailbox-2.5.spec
/usr/src/redhat/BUILD/xmms-1.2.4:xmms.spec
/usr/src/redhat/BUILD/xmorph-2000apr28:xmorph.spec
/usr/src/redhat/BUILD/xosview-1.7.3:xosview.spec
/usr/src/redhat/BUILD/xpaint:xpaint.spec
/usr/src/redhat/BUILD/xpat2-1.06:xpat2.spec
/usr/src/redhat/BUILD/xpdf-0.92:xpdf.spec
/usr/src/redhat/BUILD/xpilot-4.3.0:xpilot.spec
/usr/src/redhat/BUILD/xpuzzles-5.5.2:xpuzzles.spec
/usr/src/redhat/BUILD/xrn-9.02:xrn.spec
/usr/src/redhat/BUILD/xsane-0.62:xsane.spec
/usr/src/redhat/BUILD/xscreensaver-3.29:xscreensaver.spec
/usr/src/redhat/BUILD/xsri-1.0:xsri.spec
/usr/src/redhat/BUILD/xsysinfo-1.7:xsysinfo.spec
/usr/src/redhat/BUILD/xtoolwait-1.2:xtoolwait-1.2.spec
/usr/src/redhat/BUILD/xtt-fonts-0.19990222:xtt-fonts.spec
/usr/src/redhat/BUILD/yacc:byacc-1.9.spec
/usr/src/redhat/BUILD/yp-tools-2.4:yp-tools.spec
/usr/src/redhat/BUILD/ypbind-mt-1.7:ypbind.spec
/usr/src/redhat/BUILD/ypserv-1.3.11:ypserv.spec
/usr/src/redhat/BUILD/ytalk-3.1.1:ytalk.spec
/usr/src/redhat/BUILD/zip-2.3:zip.spec
/usr/src/redhat/BUILD/zlib-1.1.3:zlib.spec
/usr/src/redhat/BUILD/zsh-3.0.8:zsh.spec

4Suite.spec: A collections of XML-related technologies for python
Canna.spec: Japanese input system
Canna.spec: Development library and header file for Canna
Canna.spec: Runtime library for Canna
Distutils.spec: Python Distribution Utilities
ElectricFence.spec: A debugger which detects memory allocation violations.
FreeWnn.spec: FreeWnn Japanese Input System
FreeWnn.spec: development library and header file for FreeWnn
FreeWnn.spec: common File for Wnn
FreeWnn.spec: runtime library for FreeWnn
FreeWnn.spec: cWnn Chinese Input System
FreeWnn.spec: cWnn/tWnn Chinese Input System common file
FreeWnn.spec: development library and header file for cWnn,tWnn.
FreeWnn.spec: tWnn Chinese Input System
FreeWnn.spec: kWnn Korean Input System
FreeWnn.spec: development library and header file for kWnn
ImageMagick.spec: An X application for displaying and manipulating images.
ImageMagick.spec: Static libraries and header files for ImageMagick app development.
ImageMagick.spec: ImageMagick Magick++ library
ImageMagick.spec: C++ bindings for the ImageMagick library
LPRng.spec: LPRng Print Spooler
MAKEDEV.spec: A program used for creating the device files in /dev.
MAKEDEV.spec: The most commonly-used entries in the /dev directory.
Maelstrom.spec: Maelstrom
Mesa.spec: A 3-D graphics library similar to OpenGL.
Mesa.spec: Development files for the Mesa 3-D graphics library.
Mesa.spec: Demos for the Mesa 3D graphics library.
ORBit.spec: A high-performance CORBA Object Request Broker.
ORBit.spec: Development libraries, header files and utilities for ORBit.
Perl-RPM.spec: Perl-RPM module for perl
SDL.spec: Simple DirectMedia Layer
SDL.spec: Libraries, includes and more to develop SDL applications.
SDL_image.spec: Simple DirectMedia Layer - Sample Image Loading Library
SDL_image.spec: Libraries, includes and more to develop SDL applications.
SDL_mixer.spec: Simple DirectMedia Layer - Sample Mixer Library
SDL_mixer.spec: Libraries, includes and more to develop SDL applications using the SDL mixer.
SysVinit.spec: Programs which control basic system processes.
VFlib2.spec: The vector font library (VFlib2)
VFlib2.spec: header files and static library for VFlib v2.24.0.
VFlib2.spec: Other useful files for using VFlib2
WindowMaker.spec: A window manager for the X Window System.
WindowMaker.spec: Libraries bundled with WindowMaker
Wnn6-SDK.spec: Wnn6 Client Library
Wnn6-SDK.spec: development library and header file for Wnn6
XFree86-ISO8859-7.spec: Greek language fonts for the X Window System.
XFree86-ISO8859-7.spec: ISO 8859-7 fonts in 75 dpi resolution for the X Window System.
XFree86-ISO8859-7.spec: ISO 8859-7 fonts in 100 dpi resolution for the X Window System.
XFree86-ISO8859-7.spec: Type 1 scalable Greek (ISO 8859-7) fonts
XFree86-ISO8859-9.spec: Turkish language fonts and modmaps for X.
XFree86-ISO8859-9.spec: 75 dpi Turkish (ISO8859-9) fonts for X.
XFree86-ISO8859-9.spec: 100 dpi Turkish (ISO8859-9) fonts for X.
XFree86-KOI8-R.spec: Russian and Ukrainian language fonts for the X Window System.
XFree86-KOI8-R.spec: A set of 75 dpi Russian and Ukrainian language fonts for X.
XFree86-KOI8-R.spec: KOI8-R fonts in 100 dpi resolution for the X Window System.
XFree86-jpfonts.spec: Japanese fixed fonts for X11
XFree86.spec: The basic fonts, programs and docs for an X workstation.
XFree86.spec: A font server for the X Window System.
XFree86.spec: A simple window manager
XFree86.spec: X11R6 static libraries, headers and programming man pages.
XFree86.spec: X Display Manager
XFree86.spec: Shared libraries needed by the X Window System version 11 release 6.4
XFree86.spec: X Window System 100dpi fonts.
XFree86.spec: A set of 75 dpi resolution fonts for the X Window System.
XFree86.spec: Cyrillic fonts for X.

XFree86.spec: A set of 100 dpi Central European language fonts for X.
XFree86.spec: A set of 75 dpi Central European language fonts for X.
XFree86.spec: A set of Type1 Central European language fonts for X.
XFree86.spec: Documentation on various X11 programming interfaces.
XFree86.spec: A nested XFree86 server.
XFree86.spec: A virtual framebuffer X Windows System server for XFree86.
XFree86.spec: Video for Linux (V4L) support for XFree86
XFree86.spec: Various tools for XFree86
XFree86.spec: XFree86 configurator
Xaw3d.spec: A version of the MIT Athena widget set for X.
Xaw3d.spec: Header files and static libraries for development using Xaw3d.
Xconfigurator.spec: The Red Hat Linux configuration tool for the X Window System.
a2ps.spec: Converts text and other types of files to PostScript(TM).
abiword.spec: The AbiWord word processor
adjtimex.spec: A utility for adjusting kernel time variables.
alchemist.spec: A multi-sourced configuration back-end.
am-utils.spec: Automount utilities including an updated version of Amd.
amanda.spec: A network-capable tape backup solution.
amanda.spec: The client component of the AMANDA tape backup system.
amanda.spec: The server side of the AMANDA tape backup system.
amanda.spec: Libraries and documentation of the AMANDA tape backup system.
anaconda.spec: The Red Hat Linux installation program.
anaconda.spec: Red Hat Linux installer portions needed only for fresh installs.
anacron.spec: A cron-like program that can run jobs lost during downtime.
anonftp.spec: A fast, read-only, anonymous FTP server.
apache.spec: The most widely used web server on the Internet.
apache.spec: Cryptography support for the Apache web server.
apache.spec: Development tools for the Apache web server.
apache.spec: Documentation for the Apache web server.
apacheconf.spec: Apache configuration tool
apmd.spec: Advanced Power Management (APM) BIOS utilities for laptops.
ash.spec: A smaller version of the Bourne shell (sh).
asp2php.spec: asp2php converts WWW Active Server Pages to PHP pages
asp2php.spec: gtk+ frontend for asp2php
aspell-ca.spec: Catalan files for aspell
aspell-da.spec: Danish files for aspell
aspell-de.spec: German files for aspell
aspell-es.spec: Spanish files for aspell
aspell-fr.spec: French files for aspell
aspell-it.spec: Italian files for aspell
aspell-nl.spec: Dutch files for aspell
aspell-no.spec: Norwegian files for aspell
aspell-pt_BR.spec: Aspell dictionary for Brazilian Portuguese
aspell-sv.spec: Swedish files for aspell
aspell.spec: A spelling checker.
aspell.spec: The static libraries and header files needed for Aspell development.
aspell.spec: British dictionary for aspell
aspell.spec: Canadian dictionary
at.spec: Job spooling tools.
audiofile.spec: A library for accessing various audio file formats.
audiofile.spec: Libraries, includes and other files to develop audiofile applications.
aumix.spec: An ncurses-based audio mixer.
aumix.spec: An X11-based audio mixer
auth_ldap.spec: This is an LDAP authentication module for Apache.
authconfig.spec: Text-mode tool for setting up NIS and shadow passwords.
autoconf.spec: A GNU tool for automatically configuring source code.
autofs.spec: A tool for automatically mounting and unmounting filesystems.
automake.spec: A GNU tool for automatically creating Makefiles.
autorun.spec: A CD-ROM mounting utility.
awesfx.spec: Utility programs for the AWE32 sound driver.
balsa.spec: Balsa Mail Client
basesystem.spec: The skeleton package which defines a simple Red Hat Linux system.
bash.spec: The GNU Bourne Again shell (bash) version %{version}.
bash.spec: Documentation for the GNU Bourne Again shell (bash) version %{version}.
bc.spec: GNU's bc (a numeric processing language) and dc (a calculator).

bdflush-1.5.spec: The daemon which starts the flushing of dirty buffers back to disk.
bind.spec: A DNS (Domain Name System) server.
bind.spec: A DNS (Domain Name System) server.
bind.spec: A DNS (Domain Name System) server.
bindconf.spec: Red Hat DNS configuration tool
binutils.spec: A GNU collection of binary utilities.
bison.spec: A GNU general-purpose parser generator.
blt.spec: A Tk toolkit extension.
bootparamd.spec: A server process which provides boot information to diskless clients.
bug-buddy.spec: Utility to ease the reporting of bugs within the GNOME Desktop Environment.
byacc-1.9.spec: A public domain Yacc parser generator.
bzip2.spec: A file compression utility.
bzip2.spec: Header files and libraries for developing apps which will use bzip2.
caching-nameserver.spec: The configuration files for setting up a caching name server.
cdecl-2.5.spec: Programs for encoding and decoding C and C++ function declarations.
cdp.spec: An interactive text-mode program for playing audio CD-ROMs.
cdparanoia.spec: A Compact Disc Digital Audio (CDDA) extraction tool (or ripper).
cdparanoia.spec: Development tools for libcddda_paranoia (Paranoia III).
cdrecord.spec: A command line CD/DVD recording program.
cdrecord.spec: The libschily SCSI user level transport library.
cdrecord.spec: Creates an image of an ISO9660 filesystem.
cdrecord.spec: A utility for sampling/copying .wav files from digital audio CDs.
chkconfig.spec: A system tool for maintaining the /etc/rc*.d hierarchy.
chkconfig.spec: A tool to set the stop/start of system services in a runlevel.
chkfontpath.spec: Simple interface for editing the font path for the X font server.
cipe.spec: Kernel Module and Daemon for VPN
cleanfeed.spec: A spam filter for Usenet news servers.
comsat.spec: A mail checker client and the comsat mail checking server.
console-tools.spec: Tools for configuring the console.
control-center.spec: The GNOME Control Center.
control-center.spec: The GNOME Control Center development environment.
control-panel.spec: A Red Hat sysadmin utility program launcher for X.
cpio.spec: A GNU archiving program.
cproto.spec: Generates function prototypes and variable declarations from C code.
cracklib.spec: A password-checking library.
cracklib.spec: The standard CrackLib dictionaries.
crontabs.spec: Root crontab files used to schedule the execution of programs.
ctags.spec: A C programming language indexing and/or cross-reference tool.
cvs.spec: A version control system.
cxhextris.spec: An X Window System color version of xhextris.
cyrus-sasl.spec: The Cyrus SASL library.
cyrus-sasl.spec: Files needed for developing applications with Cyrus SASL.
cyrus-sasl.spec: GSSAPI support for Cyrus SASL.
db1.spec: The BSD database library for C (version 1).
db1.spec: Development libs/header files for Berkeley DB (version 1) library.
db2.spec: The BSD database library for C (version 2).
db2.spec: Development libs/header files for Berkeley DB (version 2) library.
db3.spec: The Berkeley DB database library for C.
db3.spec: Command line tools for managing Berkeley DB databases.
db3.spec: Development libraries/header files for the Berkeley DB library.
dbskkd-cdb.spec: A dictionary server for the SKK Japanese input method system
ddskk.spec: Daredevil SKK - Simple Kana to Kanji conversion program
ddskk.spec: Emacs Lisp source file of skk
desktop-backgrounds.spec: Desktop background images.
dev86.spec: A real mode 80x86 assembler and linker.
dhcp.spec: A DHCP (Dynamic Host Configuration Protocol) server and relay agent.
dhcpcd.spec: A DHCP (Dynamic Host Configuration Protocol) client.
dia.spec: A diagram drawing program.
dialog.spec: A utility for creating TTY dialog boxes.
diffstat.spec: A utility which provides statistics based on the output of diff.
diffutils.spec: A GNU collection of diff utilities.
dip.spec: Handles the connections needed for dialup IP links.
dmalloc.spec: Debug Malloc (Dmalloc)
docbook-dtd30-sgml.spec: SGML document type definition for DocBook.

docbook-dtd31-sgml.spec: SGML document type definition for DocBook 3.1.
docbook-dtd40-sgml.spec: SGML document type definition for DocBook 4.0.
docbook-dtd41-sgml.spec: SGML document type definition for DocBook 4.1.
docbook-dtd41-xml.spec: XML document type definition for DocBook 4.1.
docbook-style-dsssl.spec: Norman Walsh's modular stylesheets for DocBook.
docbook-utils.spec: Shell scripts to manage DocBook documents.
docbook-utils.spec: A script to convert DocBook documents to PDF.
dos2unix.spec: Text file format converter
dosfstools.spec: Utilities for making and checking MS-DOS FAT filesystems on Linux.
doxygen.spec: A documentation system for C and C++.
dump.spec: Programs for backing up and restoring filesystems.
dump.spec: Provides certain programs with access to remote tape devices.
dump.spec: Statically linked versions of dump and restore.
e.spec: The Enlightenment window manager.
e2fsprogs.spec: Utilities for managing the second extended (ext2) filesystem.
e2fsprogs.spec: Ext2 filesystem-specific static libraries and headers.
ed.spec: The GNU line editor.
ee.spec: The Electric Eyes image viewer application.
efax.spec: A program for faxing using a Class 1, 2 or 2.0 fax modem.
eject.spec: A program that ejects removable media using software control.
elm.spec: The elm mail user agent.
emacs.spec: The libraries needed to run the GNU Emacs text editor.
emacs.spec: The sources for elisp programs included with Emacs.
emacs.spec: Emacs Lisp code for input methods for international characters.
emacs.spec: The Emacs text editor without support for the X Window System.
emacs.spec: The Emacs text editor for the X Window System.
enscript.spec: A plain ASCII to PostScript converter.
esound.spec: Allows several audio streams to play on a single audio device.
esound.spec: Development files for Esound applications.
exmh.spec: The exmh mail handling system.
expat.spec: A library for parsing XML.
expat.spec: Libraries and include files to develop XML applications with expat.
ext2ed.spec: An ext2 filesystem editor.
extace.spec: A GNOME sound displayer.
fbset.spec: Tools for managing a frame buffer's video mode properties.
fetchmail.spec: A remote mail retrieval and forwarding utility.
fetchmail.spec: A GUI utility for configuring your fetchmail preferences.
file.spec: A utility for determining file types.
filesystem.spec: The basic directory layout for a Linux system.
fileutils.spec: The GNU versions of common file management utilities.
findutils.spec: The GNU versions of find utilities (find and xargs).
finger.spec: The finger client.
finger.spec: The finger daemon.
firewall-config.spec: A configuration tool for IP firewalls and masquerading.
flex.spec: A tool for creating scanners (text pattern recognizers).
fnlib.spec: Color font rendering library for X11R6.
fnlib.spec: Headers, static libraries and documentation for Fnlib.
fortune-mod.spec: A program which will display a fortune.
freecdb.spec: A fast lookup database library and utilities
freeciv.spec: Civilization clone (game)
freetype.spec: A free and portable TrueType font rendering engine.
freetype.spec: A free and portable TrueType font rendering engine.
freetype.spec: A free and portable TrueType font rendering engine.
ftp.spec: The standard UNIX FTP (File Transfer Protocol) client.
ftpcopy.spec: A mirroring tool.
fvwm2.spec: An improved version of the FVWM window manager for X.
fvwm2.spec: Graphics used by the FVWM and FVWM2 window managers.
gaim.spec: A GTK+ clone of the AOL Instant Messenger client.
gal.spec: The GNOME Application Library
gal.spec: Development files for the GNOME Applications library
gal.spec: The GNOME Application Library
gated.spec: The public release version of the Gated routing daemon.
gawk.spec: The GNU version of the awk text processing utility.
gcc.spec: Various compilers (C, C++, Objective-C, Chill, ...)
gcc.spec: C++ support for gcc

gcc.spec: GNU c++ library
gcc.spec: Header files and libraries for C++ development
gcc.spec: Older GNU c++ library
gcc.spec: Header files and libraries for C++ development (libg++)
gcc.spec: Objective C support for gcc
gcc.spec: Fortran 77 support for gcc
gcc.spec: CHILL support for gcc
gcc.spec: Java support for gcc
gcc.spec: The C compiler optimized for generating SPARC 32bit code
gcc.spec: The C++ compiler optimized for generating SPARC 32bit code
gcc.spec: The C Preprocessor.
gd.spec: A graphics library for drawing image files in various formats.
gd.spec: Utility programs that use libgd.
gd.spec: The development libraries and header files for gd.
gdb.spec: A GNU source-level debugger for C, C++ and Fortran.
gdbm.spec: A GNU set of database routines which use extensible hashing.
gdbm.spec: Development libraries and header files for the gdbm library.
gdk-pixbuf.spec: Image loading library used with GNOME
gdk-pixbuf.spec: Libraries and headers for application development with gdk-pixbuf
gdm.spec: The GNOME Display Manager.
gedit.spec: gEdit is a small but powerful text editor for GNOME.
gedit.spec: The files needed for developing plug-ins for the gEdit editor.
genromfs.spec: Utility for creating romfs filesystems.
gettext.spec: GNU libraries and utilities for producing multi-lingual messages.
gftp.spec: A multi-threaded FTP client for the X Window System.
ghostscript-fonts.spec: Fonts for the Ghostscript PostScript(TM) interpreter.
ghostscript.spec: A PostScript(TM) interpreter and renderer.
giftrans.spec: A program for making transparent GIFs from non-transparent GIFs.
gimp-data-extras.spec: The GNU Image Manipulation Program
gimp.spec: The GNU Image Manipulation Program
gimp.spec: GIMP plugin and extension development kit
gimp.spec: GIMP perl extensions and plugins.
gkermit.spec: A utility for transferring files using the Kermit protocol.
glade.spec: A GTK+ GUI builder.
glib.spec: A library of handy utility functions.
glib.spec: The GIMP ToolKit (GTK+) and GIMP Drawing Kit (GDK) support library.
glibc.spec: The GNU libc libraries.
glibc.spec: Header and object files for development using standard C libraries.
glibc.spec: The GNU libc libraries, including support for gprof profiling.
glibc.spec: Common binaries and locale data for glibc
glibc.spec: A Name Service Caching Daemon (nscd).
glms.spec: A GNOME hardware monitoring applet.
gmp.spec: A GNU arbitrary precision library.
gmp.spec: Development tools for the GNU MP arbitrary precision library.
gnome-applets.spec: Small applications which embed themselves in the GNOME panel
gnome-audio.spec: Sounds for GNOME events.
gnome-audio.spec: Files needed for customizing GNOME event sounds.
gnome-core.spec: The core programs for the GNOME GUI desktop environment.
gnome-core.spec: GNOME core libraries, headers and more.
gnome-games.spec: GNOME games.
gnome-games.spec: GNOME games development libraries.
gnome-kerberos.spec: Kerberos 5 tools for GNOME.
gnome-libs.spec: Main GNOME libraries
gnome-libs.spec: Libraries and headers for GNOME application development
gnome-linuxconf.spec: The GNOME front-end for linuxconf.
gnome-lokkit.spec: Firewall configuration application for an average end user.
gnome-lokkit.spec: Firewall configuration application for an average end user.
gnome-media.spec: GNOME media programs.
gnome-objc.spec: Objective C libraries for the GNOME desktop environment.
gnome-objc.spec: Files needed to develop Objective C GNOME applications.
gnome-pim.spec: The GNOME Personal Information Manager.
gnome-pim.spec: GNOME PIM development files
gnome-print.spec: Printing libraries for GNOME
gnome-print.spec: Printing libraries for GNOME

gnome-print.spec: Libraries and include files for developing GNOME applications.
gnome-python.spec: The sources for the PyGTK and PyGNOME Python extension modules.
gnome-python.spec: Python bindings for the GTK+ widget set.
gnome-python.spec: A wrapper for the libglade library for use with PyGTK
gnome-python.spec: GNOME support for the libglade python wrapper
gnome-python.spec: Python bindings for the GNOME libraries.
gnome-python.spec: Python bindings for GNOME Panel applets.
gnome-python.spec: Python bindings for GNOME Panel applets.
gnome-utils.spec: GNOME utility programs.
gnorpm.spec: A graphical front-end to RPM for GNOME.
gnuchess.spec: The GNU chess program.
gnumeric.spec: A full-featured spreadsheet for GNOME.
gnumeric.spec: Files necessary to develop gnumeric-based applications.
gnupg.spec: A GNU utility for secure communication and data storage.
gnuplot.spec: A program for plotting mathematical expressions and data.
gperf.spec: A perfect hash function generator.
gphoto.spec: Allows you to retrieve and manipulate images from digital cameras
gpm.spec: A mouse server for the Linux console.
gpm.spec: A mouse server for the Linux console.
gq.spec: A graphical LDAP directory browser and editor.
gqview.spec: An image viewer.
grep.spec: The GNU versions of grep pattern matching utilities.
groff.spec: A document formatting system.
groff.spec: Parts of the groff formatting system that require Perl.
groff.spec: An X previewer for groff text processor output.
gsl.spec: The GNU Scientific Library for numerical analysis.
gsm.spec: A GSM sound format compressor/decompressor.
gsm.spec: A development library and headers for GSM use
gtk+.spec: The GIMP ToolKit (GTK+), a library for creating GUIs for X.
gtk+.spec: Development tools for GTK+ (GIMP ToolKit) applications.
gtk-doc.spec: API documentation generation tool for GTK+ and GNOME
gtk-engines.spec: Theme engines for GTK+.
gtop.spec: The GNOME system monitor.
guile.spec: A GNU implementation of Scheme for application extensibility.
guile.spec: Libraries and header files for the GUILE extensibility library.
gv.spec: A X front-end for the Ghostscript PostScript(TM) interpreter.
gzip.spec: The GNU data compression program.
hdparm.spec: A utility for displaying and/or setting hard disk parameters.
hotplug.spec: A helper application for loading modules for USB devices
htdig.spec: ht://Dig - Web search engine
htdig.spec: Scripts and HTML code needed for using ht://Dig as a web search engine
htmlview.spec: Script that calls up whatever HTML viewer is installed/preferred
ical.spec: An X Window System-based calendar program.
im.spec: Internet Message
imap.spec: Server daemons for IMAP and POP network mail protocols.
imap.spec: Development tools for programs which will use the IMAP library.
imlib.spec: An image loading and rendering library for X11R6.
imlib.spec: Development tools for Imlib applications.
imlib.spec: A configuration editor for the Imlib library.
indent.spec: A GNU program for formatting C code.
indexhtml.spec: The Web page you'll see after installing Red Hat Linux.
initscripts.spec: The inittab file and the /etc/init.d scripts.
inn.spec: The InterNetNews (INN) system, an Usenet news server.
inn.spec: The INN (InterNetNews) library.
inn.spec: Sends Usenet articles to a local news server for distribution.
internet-config.spec: Configuration tool for internet connections
inti.spec: Inti Libraries
inti.spec: Inti foundation libraries
ipchains.spec: Tools for managing Linux kernel packet filtering capabilities.
iproute.spec: Enhanced IP routing and network devices configuration tools
iptables.spec: Tools for managing Linux kernel packet filtering capabilities.
iptables.spec: IPv6 support for iptables
iputils.spec: The ping program for checking to see if network hosts are alive.
ircii.spec: An Internet Relay Chat (IRC) client.
irda-utils.spec: Utilities for infrared communication between devices.

isapnptools.spec: Utilities for configuring ISA Plug-and-Play (PnP) devices.
isd4k-utl.spec: Utilities for configuring an ISDN subsystem.
isd4k-utl.spec: An ISDN connection load average display for the X Window System.
isicom.spec: Multitech IntelligentSerialInternal (ISI) Support Tools
jadetex.spec: TeX macros used by Jade TeX output.
jcode.pl.spec: Perl library for Japanese character code conversion
jed.spec: A fast, compact editor based on the S-Lang screen library.
jed.spec: Files needed by any Jed text editor.
jed.spec: The X Window System version of the Jed text editor.
jed.spec: A grep utility which can recursively descend through directories.
jikes.spec: A Java source file to bytecode compiler.
jisksp14.spec: 14 dots jis auxiliary kanji font
jisksp16-1990.spec: 16 dots jis auxiliary kanji font
joe.spec: An easy to use, modeless text editor.
joystick.spec: Utilities for configuring most popular joysticks.
kaffe.spec: A free virtual machine for running Java(TM) code.
kakasi.spec: KAKASI - kanji kana simple inverter
kakasi.spec: header file and libraries of KAKASI
kakasi.spec: The base dictionary of KAKASI
kappa20.spec: Kappa 20dot Font
kbdconfig.spec: A text-based interface for setting and loading a keyboard map.
kcc.spec: Kanji Code Converter
kdbg.spec: A GUI for gdb, the GNU debugger, and KDE.
kde-i18n-2.1.spec: Internationalization support for KDE
kde-i18n-2.1.spec: Afrikaans language support for KDE
kde-i18n-2.1.spec: Bulgarian language support for KDE
kde-i18n-2.1.spec: Breton language support for KDE
kde-i18n-2.1.spec: Catalan language support for KDE
kde-i18n-2.1.spec: Czech language support for KDE
kde-i18n-2.1.spec: Cymraeg language support for KDE
kde-i18n-2.1.spec: Danish language support for KDE
kde-i18n-2.1.spec: German language support for KDE
kde-i18n-2.1.spec: Greek language support for KDE
kde-i18n-2.1.spec: British English support for KDE
kde-i18n-2.1.spec: Esperanto support for KDE
kde-i18n-2.1.spec: Spanish language support for KDE
kde-i18n-2.1.spec: Estonian language support for KDE
kde-i18n-2.1.spec: Basque language support for KDE
kde-i18n-2.1.spec: Finnish language support for KDE
kde-i18n-2.1.spec: French language support for KDE
kde-i18n-2.1.spec: Irish language support for KDE
kde-i18n-2.1.spec: Galician language support for KDE
kde-i18n-2.1.spec: Hebrew language support for KDE
kde-i18n-2.1.spec: Croatian language support for KDE
kde-i18n-2.1.spec: Hungarian language support for KDE
kde-i18n-2.1.spec: Icelandic language support for KDE
kde-i18n-2.1.spec: Italian language support for KDE
kde-i18n-2.1.spec: Japanese language support for KDE
kde-i18n-2.1.spec: Korean language support for KDE
kde-i18n-2.1.spec: Lithuanian language support for KDE
kde-i18n-2.1.spec: Maori language support for KDE
kde-i18n-2.1.spec: Macedonian language support for KDE
kde-i18n-2.1.spec: Dutch language support for KDE
kde-i18n-2.1.spec: Norwegian (Bokmaal) language support for KDE
kde-i18n-2.1.spec: Norwegian (Nynorsk) language support for KDE
kde-i18n-2.1.spec: Polish language support for KDE
kde-i18n-2.1.spec: Portuguese language support for KDE
kde-i18n-2.1.spec: Brazil Portuguese language support for KDE
kde-i18n-2.1.spec: Romanian language support for KDE
kde-i18n-2.1.spec: Russian language support for KDE
kde-i18n-2.1.spec: Slovak language support for KDE
kde-i18n-2.1.spec: Slovenian language support for KDE
kde-i18n-2.1.spec: Serbian language support for KDE
kde-i18n-2.1.spec: Swedish language support for KDE
kde-i18n-2.1.spec: Tamil language support for KDE

kde-118n-2.1.spec: Thai language support for KDE
kde-118n-2.1.spec: Turkish language support for KDE
kde-118n-2.1.spec: Ukrainian language support for KDE
kde-118n-2.1.spec: Walloon language support for KDE
kde-118n-2.1.spec: Chinese (Simplified Chinese) language support for KDE
kde-118n-2.1.spec: Chinese (Big5) language support for KDE
kdeadmin-2.1.spec: K Desktop Environment - Admin tools
kdebase-2.1.spec: K Desktop Environment - core files
kdebindings-2.1.spec: KDE bindings to non-C++ languages
kdebindings-2.1.spec: Development files for %{name}
kdebindings-2.1.spec: KDE bindings to mozilla
kdegames-2.1.spec: K Desktop Environment - Games
kdegraphics-2.1.spec: K Desktop Environment - Graphics Applications
kdelibs-2.1.spec: K Desktop Environment - Libraries
kdelibs-2.1.spec: Header files and documentation for compiling KDE applications.
kdelibs-2.1.spec: K Desktop Environment - Libraries for sound support
kdelibs-2.1.spec: arts (analog real-time synthesizer) - the KDE 2.x sound system
kdelibs-2.1.spec: Header files and documentation for compiling KDE applications with sound
kdemultimedia-2.1.spec: Multimedia applications for the K Desktop Environment (KDE).
kdenetwork-2.1.spec: K Desktop Environment - Network Applications
kdenetwork-2.1.spec: K Desktop Environment - PPP Network Applications
kdepim-2.1.spec: PIM (Personal Information Manager) for KDE
kdesdk-2.1.spec: KDE SDK
kdesdk-2.1.spec: Development files for %{name}
kdesupport-2.1.spec: K Desktop Environment - Support Libraries
kdesupport-2.1.spec: Header files and documentation for KDE Support Libraries
kdetoys-2.1.spec: K Desktop Environment - Toys and Amusements
kdeutils-2.1.spec: K Desktop Environment - Utilities
kdevelop-1.4.spec: Integrated Development Environment for C++/C
kdoc-2.1.spec: Documentation for the K Desktop Environment (KDE) 2.0.
kernel-2.4.spec: The Linux kernel (the core of the Linux operating system)
kernel-2.4.spec: The source code for the Linux kernel.
kernel-2.4.spec: Header files for the Linux kernel.
kernel-2.4.spec: Various documentation bits found in the kernel source.
kernel-2.4.spec: Device filesystem management daemon
kernel-2.4.spec: The Linux kernel compiled for SMP machines.
kernel-2.4.spec: The Linux Kernel compiled with options for Enterprise server usage.
kernel-2.4.spec: The version of the Linux kernel used on installation boot disks.
kernel-2.4.spec: The Linux kernel used on installation boot disks for SMP machines.
kernel-2.4.spec: The Linux Kernel compiled for the Alpha Jensen platform.
kernel-pcmcia-cs.spec: The daemon and device drivers for using PCMCIA adapters.
kinput2.spec: kinput2 - kanji input server for X11
kinput2.spec: kinput2 for Canna
kinput2.spec: kinput2 for Wnn4
kinput2.spec: kinput2 for both Canna and Wnn4
kinput2.spec: kinput2 for Wnn6
kinput2.spec: kinput2 for both Canna and Wnn6
knm_new.spec: Kaname-cho font, revised version
koffice-2.0.spec: Set of office applications for KDE
kon2.spec: KON - Kanji ON Linux console
kon2.spec: Fonts for KON
kpppload.spec: A PPP connection load monitor for KDE.
krb5.spec: The Kerberos network authentication system.
krb5.spec: Development files needed for compiling Kerberos 5 programs.
krb5.spec: The shared libraries used by Kerberos 5.
krb5.spec: The server programs for Kerberos 5.
krb5.spec: Kerberos 5 programs for use on workstations.
krbafs.spec: A Kerberos to AFS bridging library, built against Kerberos 5.
krbafs.spec: Kerberos/AFS utility binaries.
ksconfig.spec: A graphical interface for making kickstart files.
ksymoops.spec: Kernel oops and error message decoder
kterm.spec: A Kanji (Japanese character set) terminal emulator for X.
kudzu.spec: The Red Hat Linux hardware probing tool.
kudzu.spec: Development files needed for hardware probing using kudzu.

lam.spec: LAM (Local Area Multicomputer) programming environment
lapack.spec: The LAPACK libraries for numerical linear algebra.
lapack.spec: The BLAS (Basic Linear Algebra Subprograms) library for Linux.
lapack.spec: Man pages for BLAS (Basic Linear Algebra Subprograms) routines.
lapack.spec: Documentation for the LAPACK numerical linear algebra libraries.
lclint.spec: An implementation of the lint program
less.spec: A text file browser similar to more, but better.
libPropList.spec: Ensures program compatibility with GNUstep/OPENSTEP.
libelf-0.6.4.spec: An ELF object file access library.
libgcj.spec: Java runtime library for gcc
libgcj.spec: Libraries for Java development using gcc
libghttp.spec: GNOME http client library.
libghttp.spec: GNOME http client development
libglade.spec: The libglade library for loading user interfaces
libglade.spec: The files needed for libglade application development.
libgtop.spec: The LibGTop library
libgtop.spec: Libraries, includes and other files to develop LibGTop applications.
libgtop.spec: These are examples for LibGTop, a library which retrieves information about your system, such as CPU and memory usage.
libjpeg-6a.spec: A library for manipulating JPEG image format files.
libjpeg.spec: A library for manipulating JPEG image format files.
libjpeg.spec: Development tools for programs which will use the libjpeg library.
libmng.spec: Library for supporting MNG (Animated PNG) graphics
libmng.spec: Development files for the MNG (Animated PNG) library
libmng.spec: MNG (Animated PNG) library for static linking
libodbc++.spec: An ODBC class library that emulates the JDBC interface
libodbc++.spec: Development files for programs which use the odbc++ library.
libodbc++.spec: qt libodbc++ libraries
libogg.spec: Ogg Bitstream Library
libogg.spec: Ogg Bitstream Library Development
libole2.spec: Structured Storage OLE2 library
libole2.spec: Libraries, includes, etc to develop libole2 applications
libpng.spec: A library of functions for manipulating PNG image format files.
libpng.spec: Development tools for programs to manipulate PNG image format files.
librep.spec: An embeddable LISP environment.
librep.spec: Include files and link libraries for librep development.
libtermcap.spec: A basic system library for accessing the termcap database.
libtermcap.spec: Development tools for programs which will access the termcap database.
libtiff.spec: A library of functions for manipulating TIFF format image files.
libtiff.spec: Development tools for programs which will use the libtiff library.
libtool.spec: The GNU libtool, which simplifies the use of shared libraries.
libtool.spec: Runtime libraries for GNU libtool.
libungif.spec: A library for manipulating GIF format image files.
libungif.spec: Development tools for programs which will use the libungif library.
libungif.spec: Programs for manipulating GIF format image files.
libunicode.spec: A unicode manipulation library
libunicode.spec: A unicode manipulation library
libxml.spec: An XML library.
libxml.spec: Libraries, includes and other files to develop libxml applications.
licq.spec: An ICQ clone for online messaging.
lilo.spec: The boot loader for Linux and other operating systems.
links.spec: text mode www browser with support for frames
linuxconf.spec: A system configuration tool.
linuxconf.spec: The tools needed for developing linuxconf modules.
lm_sensors.spec: Hardware monitoring tools.
lm_sensors.spec: Hardware monitoring tools.
lm_sensors.spec: Development files for programs which will use lm_sensors.
locale_config.spec: Locale configuration
lockdev.spec: A library for locking devices
lockdev.spec: headers and a static library for lockdev
logrotate.spec: Rotates, compresses, removes and mails system log files.
lout.spec: The Lout document formatting language.
lout.spec: The documentation for the Lout document formatting language.
lrzsz.spec: The lrz and lsz modem communications programs.
lslk.spec: A lock file lister.

lsf.spec: A utility which lists open files on a Linux/UNIX system.
ltrace.spec: Tracks runtime library calls from dynamically linked executables.
lv.spec: A Powerful Multilingual File Viewer
lynx.spec: A text-based Web browser.
m4.spec: The GNU macro processor.
macutils.spec: Utilities for manipulating Macintosh file formats.
magicdev.spec: A GNOME daemon for automatically mounting/playing CDs.
mailcap.spec: Associates helper applications with particular file types.
mailx.spec: The /bin/mail program for sending e-mail messages.
make.spec: A GNU tool which simplifies the build process for users.
man-pages-cs.spec: Czech man (manual) pages from the Linux Documentation Project
man-pages-da.spec: Danish man (manual) pages from the Linux Documentation Project
man-pages-de.spec: German man (manual) pages from the Linux Documentation Project
man-pages-es.spec: Spanish man (manual) pages from the Linux Documentation Project
man-pages-fr.spec: French man (manual) pages from the Linux Documentation Project
man-pages-it.spec: Italian man (manual) pages from the Linux Documentation Project
man-pages-ja.spec: Japanese man (manual) pages from the Linux Documentation Project
man-pages-pl.spec: Polish man (manual) pages from the Linux Documentation Project
man-pages-ru.spec: Russian man (manual) pages from the Linux Documentation Project
man-pages.spec: Man (manual) pages from the Linux Documentation Project.
man.spec: A set of documentation tools: man, apropos and whatis.
mars-nwe.spec: NetWare file and print servers which run on Linux systems.
mawk.spec: An interpreter for the awk programming language.
mc.spec: A user-friendly file manager and visual shell.
mc.spec: The GNOME version of the Midnight Commander file manager.
mc.spec: Server for the Midnight Commander network file management system.
memprof.spec: Tool for memory profiling and leak detection
metamail.spec: A program for handling multimedia mail using the mailcap file.
mgetty.spec: A getty replacement for use with data and fax modems.
mgetty.spec: Provides support for sending faxes over a modem.
mgetty.spec: A program for using your modem and mgetty as an answering machine.
mgetty.spec: An X Window System fax viewer.
mikmod.spec: A MOD music file player.
mingetty.spec: A compact getty program for virtual consoles only.
minicom.spec: A text-based modem control and terminal emulation program.
mkbootdisk.spec: Creates an initial ramdisk image for preloading modules.
mkinitrd.spec: Creates an initial ramdisk image for preloading modules.
mkkickstart.spec: Writes a kickstart description of the current machine.
mktemp.spec: A small utility for safely making /tmp files.
mkxauth.spec: A utility for managing .Xauthority files.
mod_dav.spec: A DAV module for Apache.
mod_perl.spec: An embedded Perl interpreter for the Apache Web server.
modemtool-1.22.spec: A tool for selecting the serial port your modem is connected to.
modutils.spec: The kernel daemon (kernelld) and kernel module utilities.
mount.spec: Programs for mounting and unmounting filesystems.
mount.spec: Programs for setting up and configuring loopback devices.
mouseconfig.spec: The Red Hat Linux mouse configuration tool.
mozilla-0.7.spec: Web browser and mail reader
mozilla-0.7.spec: Development files for Mozilla
mozilla-0.7.spec: Mozilla-based mail system
mozilla-0.7.spec: SSL support for Mozilla.
mpage.spec: A tool for printing multiple pages of text on each printed page.
mpg123.spec: An MPEG audio player.
mt-st.spec: Install mt-st if you need a tool to control tape drives.
mtools.spec: Programs for accessing MS-DOS disks without mounting the disks.
mtr.spec: Ping/Traceroute network diagnostic tool
mtr.spec: Ping/Traceroute network diagnostic tool - GTK Interface
mtx.spec: Controls the robotic mechanism in DDS Tape drive autoloaders.
multimedia.spec: Several X utilities mainly for use with multimedia files.
mutt.spec: A text mode mail user agent.
mysql.spec: MySQL client program and shared library
mysql.spec: MySQL server
mysql.spec: MySQL devel
nasm.spec: The Netwide Assembler, a portable x86 assembler with Intel-like syntax
nasm.spec: Extensive documentation for NASM

nasm.spec: Tools for the RDOFF binary format, sometimes used with NASM.
nc.spec: Reads and writes data across network connections using TCP or UDP.
ncftp.spec: An improved FTP client.
ncompress.spec: Fast compression and decompression utilities.
ncpfs.spec: Utilities for the ncpfs filesystem, a NetWare client for Linux.
ncpfs.spec: Tools for configuring and debugging IPX interfaces and networks.
ncurses.spec: A CRT screen handling and optimization package.
ncurses.spec: The development files for applications which use ncurses.
net-tools.spec: The basic tools for setting up networking.
netcfg.spec: A network configuration tool.
netpbm.spec: A library for handling different graphics file formats.
netpbm.spec: Development tools for programs which will use the netpbm libraries.
netpbm.spec: Tools for manipulating graphics files in netpbm supported formats.
netscape.spec: The Netscape Communicator suite of tools.
netscape.spec: Files shared by Netscape Navigator and Communicator.
netscape.spec: The Netscape Communicator suite of tools.
netscape.spec: The Netscape Navigator Web browser.
newt.spec: A development library for text mode user interfaces.
newt.spec: Newt windowing toolkit development files.
newt.spec: Snack for python2
nfs-utils.spec: NFS utilities and supporting daemons for the kernel NFS server.
njamd.spec: An advanced debugger which detects memory allocation violations.
nkf.spec: Network Kanji code conversion Filter
nmh.spec: A capable mail handling system with a command line interface.
nss_db.spec: NSS library for DB
nss_db.spec: NSS compatibility library for DB
nss_ldap.spec: NSS library and PAM module for LDAP.
ntp.spec: Synchronizes system time using the Network Time Protocol (NTP).
nut.spec: Network UPS Tools
nut.spec: Network UPS Tools client monitoring utilities
nut.spec: CGI utilities for the Network UPS Tools
nvi-m17n.spec: Multilingualized nex/nvi text editors
nvi-m17n.spec: Multilingualized nex/nvi text editors (canna version)
nvi-m17n.spec: Multilingualized nex/nvi text editors (non canna version)
octave.spec: A high-level language for numerical computations.
open-1.4.spec: A tool which will start a program on a virtual console.
openjade.spec: A DSSSL implementation.
openldap.spec: The configuration files, libraries and documentation for OpenLDAP.
openldap.spec: OpenLDAP development libraries and header files.
openldap.spec: OpenLDAP servers and related files.
openldap.spec: Client programs for OpenLDAP.
openldap12.spec: LDAP libraries.
openssh.spec: OpenSSH free Secure Shell (SSH) implementation
openssh.spec: OpenSSH Secure Shell protocol clients
openssh.spec: OpenSSH Secure Shell protocol server (sshd)
openssh.spec: OpenSSH X11 passphrase dialog
openssh.spec: OpenSSH GNOME passphrase dialog
openssl.spec: Secure Sockets Layer Toolkit
openssl.spec: OpenSSL libraries and development headers.
openssl.spec: OpenSSL scripts which require Perl.
openssl.spec: Support for using OpenSSL in python scripts.
p2c.spec: A Pascal to C translator.
p2c.spec: Files for p2c Pascal to C translator development.
pam.spec: A security tool which provides authentication for applications.
pam.spec: Files needed for developing PAM-aware applications and modules for PAM.
pam_krb5.spec: A Pluggable Authentication Module for Kerberos 5.
pan.spec: A GNOME/GTK+ news reader for X.
parted.spec: The GNU disk partition manipulation program.
parted.spec: The GNU disk partition manipulation program development files.
passwd.spec: The passwd utility for setting/changing passwords using PAM.
patch.spec: The GNU patch command, for modifying/upgrading files.
pax.spec: POSIX File System Archiver
pciutils.spec: Linux PCI utilities.
pciutils.spec: Linux PCI development library.
pdksh.spec: A public domain clone of the Korn shell (ksh).

perl-DBD-Mysql.spec: A MySQL interface for perl
perl-DBD-Pg.spec: A PostgreSQL interface for perl
perl-DBI.spec: A database access API for perl
perl-File-MMAGIC.spec: file command like perl5 module
perl-NKF.spec: Perl extension for Network Kanji Filter
perl-Text-Kakasi.spec: kakasi library module for perl
perl.spec: The Perl programming language.
php.spec: The PHP scripting language.
php.spec: Files needed for building PHP extensions.
php.spec: A module for PHP applications that use IMAP.
php.spec: A module for PHP applications that use LDAP.
php.spec: The PHP manual, in HTML format.
php.spec: A module for PHP applications that use MySQL databases.
php.spec: A module for PHP applications that use PostgreSQL databases.
pidentd.spec: An implementation of the RFC1413 identification server.
pilot-link.spec: File transfer utilities between Linux and PalmPilots.
pilot-link.spec: PalmPilot development header files.
pine.spec: A commonly used, MIME compliant mail and news reader.
pinfo.spec: An info file viewer.
pkgconfig.spec: A tool for memory profiling and leak detection.
playmidi.spec: A MIDI sound file player.
playmidi.spec: An X Window System based MIDI sound file player.
plugger.spec: A generic netscape plug-in
pmake.spec: The BSD 4.4 version of make.
pnm2ppa-1.0.spec: Drivers for printing to HP PPA printers.
portmap.spec: A program which manages RPC connections.
postgresql.spec: PostgreSQL client programs and libraries.
postgresql.spec: The programs needed to create and run a PostgreSQL server.
postgresql.spec: PostgreSQL development header files and libraries.
postgresql.spec: A Tcl client library, and the PL/Tcl procedural language for PostgreSQL.
postgresql.spec: Tk shell and tk-based GUI for PostgreSQL.
postgresql.spec: The ODBC driver needed for accessing a PostgreSQL DB using ODBC.
postgresql.spec: Development module needed for Perl code to access a PostgreSQL DB.
postgresql.spec: Development module for Python code to access a PostgreSQL DB.
postgresql.spec: Files needed for Java programs to access a PostgreSQL database.
postgresql.spec: The test suite distributed with PostgreSQL.
ppp.spec: The PPP (Point-to-Point Protocol) daemon.
printconf.spec: A printer configuration backend/frontend combo.
printconf.spec: printconf-backend summary
procinfo.spec: A tool for gathering and displaying system information.
procmail.spec: The procmail mail processing program.
procps.spec: Utilities for monitoring your system and processes on your system.
procps.spec: An X based system message monitoring utility.
psacct-6.3.spec: Utilities for monitoring process activities.
psgml.spec: A GNU Emacs major mode for editing SGML documents.
psmisc.spec: Utilities for managing processes on your system.
pspell.spec: Portable Spell Checker Interface Library.
pspell.spec: Static libraries and header files for pspell
psutils.spec: PostScript Utilities
pump.spec: A Bootp and DHCP client for automatic IP configuration.
pump.spec: Development tools for sending dhcp requests
pvm.spec: Libraries for distributed computing.
pvm.spec: TCL/TK graphical frontend to monitor and manage a PVM cluster.
pwdb.spec: The password database library.
pxe.spec: A Linux PXE (Preboot eXecution Environment) server.
python.spec: An interpreted, interactive object-oriented programming language.
python.spec: The libraries and header files needed for Python development.
python.spec: A collection of development tools included with Python.
python.spec: Documentation for the Python programming language.
python.spec: A graphical user interface for the Python scripting language.
pythonlib.spec: A library of Python code used by various Red Hat Linux programs.
qt.spec: The shared library for the Qt GUI toolkit.
qt.spec: The shared library for the Qt GUI toolkit for framebuffer devices.
qt.spec: Development files and documentation for the Qt GUI toolkit.

qt.spec: Development files and documentation for the Qt GUI toolkit for framebuffer devices.

qt.spec: An Xt (X Toolkit) compatibility add-on for the Qt GUI toolkit.

qt.spec: Version of the Qt GUI toolkit for static linking

qt.spec: Version of the Qt GUI toolkit for framebuffer devices for static linking

qt.spec: Interface designer (IDE) for the Qt toolkit

qt.spec: Interface designer (IDE) for the Qt toolkit for framebuffer devices

quota.spec: System administration tools for monitoring users' disk usage.

raidtools.spec: Tools for creating and maintaining software RAID devices.

rarpd.spec: The RARP daemon.

rcs.spec: Revision Control System (RCS) file version management tools.

rdate.spec: Tool for getting the date/time from another machine on your network.

rdist.spec: Maintains identical copies of files on multiple machines.

readline.spec: A library for editing typed in command lines.

readline.spec: Files needed to develop programs which use the readline library.

redhat-logos.spec: Red Hat-related icons and pictures.

reiserfs-utils.spec: Tools for creating, repairing and debugging ReiserFS filesystems

rep-gtk.spec: GTK+ binding for librep Lisp environment

rep-gtk.spec: librep binding for the libglade library for loading user interfaces.

rep-gtk.spec: GNOME binding for librep

rhmask.spec: Generates and restores mask files based on original and update files.

rhn_register.spec: Red Hat Network Services registration program.

rhn_register.spec: GUI client for the RHN registration program.

rootfiles.spec: The basic required files for the root user's directory.

routed.spec: The routing daemon which maintains routing tables.

rp-pppoe.spec: PPP Over Ethernet (xDSL support)

rp3.spec: The Red Hat graphical PPP management tool.

rpm.spec: The Red Hat package management system.

rpm.spec: Development files for applications which will manipulate RPM packages.

rpm.spec: Scripts and executable programs used to build packages.

rpm.spec: Python bindings for apps which will manipulate RPM packages.

rpm.spec: A C library for parsing command line parameters.

rpm2html.spec: Translates an RPM database and dependency information into HTML.

rpmfind.spec: Finds and transfers RPM files for a specified program.

rpmlint.spec: A development tool for checking the correctness of RPM packages.

rsh.spec: Clients for remote access commands (rsh, rlogin, rcp).

rsh.spec: Servers for remote access commands (rsh, rlogin, rcp).

rsync.spec: A program for synchronizing files over a network.

rusers.spec: Displays the users logged into machines on the local network.

rusers.spec: Server for the rusers protocol.

rwall.spec: Client for sending messages to a host's logged in users.

rwall.spec: Server for sending messages to a host's logged in users.

rwho.spec: Displays who is logged in to local network machines.

rxvt.spec: A color VT102 terminal emulator for the X Window System.

samba.spec: Samba SMB server.

samba.spec: Samba (SMB) client programs.

samba.spec: Files used by both Samba servers and clients.

samba.spec: The Samba SMB server configuration program."

sane.spec: Scanner access software.

sane.spec: The SANE (a universal scanner interface) development toolkit.

sash.spec: A statically linked shell, including some built-in basic commands.

sawfish.spec: An extensible window manager for the X Window System.

sawfish.spec: A GUI for creating sawfish window manager themes.

screen.spec: A screen manager that supports multiple logins on one terminal.

sed.spec: A GNU stream text editor.

semi-emacs.spec: Library to provide MIME feature for Emacs 20

sendmail.spec: A widely used Mail Transport Agent (MTA).

sendmail.spec: Documentation about the Sendmail Mail Transport Agent program.

sendmail.spec: The files needed to reconfigure Sendmail.

setserial.spec: A utility for configuring serial ports.

setup.spec: A set of system configuration and setup files.

setuptool.spec: A text mode system configuration tool.

sgml-common.spec: Common SGML catalog and DTD files.

sgml-tools.spec: A text formatting package based on SGML.

sh-utils.spec: A set of GNU utilities commonly used in shell scripts.

shadow-utils.spec: Utilities for managing shadow password files and user/group accounts.

shapecfg.spec: A configuration tool for setting traffic bandwidth parameters.

sharutils.spec: The GNU shar utilities for packaging and unpacking shell archives.

skkdic.spec: Dictionary for SKK (Simple Kana-Kanji conversion program)

skkinput.spec: SKK like Japanese-input application

slang.spec: The shared library for the S-Lang extension language.

slang.spec: The static library and header files for development using S-Lang.

sliplogin.spec: A login program for SLIP connections.

slocate.spec: Finds files on a system via a central database.

slrn.spec: A threaded Internet news reader.

slrn.spec: Offline news reading support for the SLRN news reader.

smpeg-xmms.spec: MPEG video plugin for XMMS.

smpeg.spec: SDL MPEG Library

smpeg.spec: Libraries, includes and more to develop SMPEG applications.

sndconfig.spec: The Red Hat Linux sound configuration tool.

sox.spec: A general purpose sound file conversion tool.

sox.spec: The SoX sound file format converter libraries.

specspo.spec: Red Hat package descriptions, summaries, and groups.

squid.spec: The Squid proxy caching server.

stat.spec: A tool for finding out information about a specified file.

statserial.spec: A tool which displays the status of serial port modem lines.

strace.spec: Tracks and displays system calls associated with a running process.

stunnel.spec: SSL-encrypting socket wrapper.

sudo.spec: Allows restricted root access for specified users.

switchdesk.spec: A desktop environment switcher for GNOME, KDE and AnotherLevel.

switchdesk.spec: A KDE interface for the Desktop Switcher.

switchdesk.spec: A GNOME interface for the Desktop Switcher.

symlinks-1.2.spec: A utility which maintains a system's symbolic links.

sysctlconfig.spec: A configuration tool for operating system tunable parameters

sysklogd.spec: System logging and kernel message trapping daemons.

syslinux.spec: Simple kernel loader which boots from a FAT filesystem

sysreport.spec: Gathers system hardware and configuration information.

sysstat.spec: Includes the sar and iostat system monitoring commands.

talk.spec: Talk client for one-on-one Internet chatting.

talk.spec: The talk server for one-on-one Internet chatting.

tamago.spec: Tamago Version 4

taper.spec: A menu-driven file backup system.

tar.spec: A GNU file archiving program.

tcltk.spec: A Tcl/Tk development environment: tcl, tk, tix, tclX, expect, and itcl.

tcltk.spec: An embeddable scripting language.

tcltk.spec: The Tk GUI toolkit for Tcl, with shared libraries.

tcltk.spec: A tcl extension for simplifying program-script interaction.

tcltk.spec: Extensions for Tcl.

tcltk.spec: A set of capable widgets for Tk.

tcltk.spec: Object-oriented mega-widgets for Tcl.

tcltk.spec: A library of utility modules for Tcl.

tcp_wrappers.spec: A security tool which acts as a wrapper for TCP daemons.

tcpdump.spec: A network traffic monitoring tool.

tcpdump.spec: A system-independent interface for user-level packet capture.

tcpdump.spec: Network monitoring tools for tracking IP addresses on a network.

tcsh.spec: An enhanced version of csh, the C shell.

telnet.spec: The client program for the telnet remote login protocol.

telnet.spec: The server program for the telnet remote login protocol.

termcap.spec: The terminal feature database used by certain applications.

tetex.spec: The TeX text formatting system.

tetex.spec: The LaTeX front end for the TeX text formatting system.

tetex.spec: An X viewer for DVI files.

tetex.spec: A DVI to PostScript converter for the TeX text formatting system.

tetex.spec: A DVI to HP PCL (Printer Control Language) converter.

tetex.spec: A converter for PostScript(TM) font metric files, for use with TeX.

tetex.spec: The font files for the TeX text formatting system.

tetex.spec: The documentation files for the TeX text formatting system.

texinfo.spec: Tools needed to create Texinfo format documentation files.

texinfo.spec: A stand-alone TTY-based reader for GNU texinfo documentation.

textutils.spec: A set of GNU text file modifying utilities.
tftp.spec: The client for the Trivial File Transfer Protocol (TFTP).
tftp.spec: The server for the Trivial File Transfer Protocol (TFTP).
time.spec: A GNU utility for monitoring a program's use of system resources.
timeconfig.spec: Text mode tools for setting system time parameters.
timetool.spec: A utility for setting the system's date and time.
timidity.spec: A software wavetable MIDI synthesizer.
tksysv.spec: An X editor for editing runlevel services.
tmpwatch.spec: A utility for removing files based on when they were last accessed.
traceroute.spec: Traces the route taken by packets over a TCP/IP network.
transfig.spec: A utility for converting FIG files (made by xfig) to other formats.
tree.spec: A utility which displays a tree view of the contents of directories.
tripwire.spec: A system integrity assessment tool.
trojka.spec: A non-X game of falling blocks.
ttfonts.spec: Some TrueType fonts
tux.spec: User-space component of TUX kernel-based threaded HTTP server
ucd-snmp.spec: A collection of SNMP protocol tools from UC-Davis.
ucd-snmp.spec: Network management utilities using SNMP, from the UCD-SNMP project.
ucd-snmp.spec: The development environment for the UCD-SNMP project.
umb-scheme-3.2.spec: An implementation of the Scheme programming language.
unarj.spec: An uncompressor for .arj format archive files.
units.spec: A utility for converting amounts from one unit to another.
unix2dos.spec: unix2dos - UNIX to DOS text file format converter
unixODBC.spec: A complete ODBC Driver Manager for Linux
unixODBC.spec: Development files for programs which will use the unixODBC library.
unixODBC.spec: KDE DriverManager components for ODBC
unzip.spec: A utility for unpacking zip files.
up2date.spec: Automatically update RPMs for a Red Hat Linux System
up2date.spec: GUI client for Update Agent.
urlview.spec: An URL extractor/viewer for use with Mutt.
urw-fonts.spec: Free versions of the 35 standard PostScript fonts.
usbview.spec: USB topology and device viewer
usermode.spec: Graphical tools for certain user account management tasks.
users-guide.spec: GNOME Users Guide
utempter.spec: A privileged helper for utmp/wtmp updates.
util-linux.spec: A collection of basic system utilities.
uucp.spec: The uucp utility for copying files between systems.
vim.spec: The VIM editor.
vim.spec: The common files needed by any version of the VIM editor.
vim.spec: A minimal version of the VIM editor.
vim.spec: A version of the VIM editor which includes recent enhancements.
vim.spec: The VIM version of the vi editor for the X Window System.
vixie-cron.spec: The Vixie cron daemon for executing specified programs at set times.
vlock.spec: A program which locks one or more virtual consoles.
vnc.spec: A remote display system.
vnc.spec: A VNC server.
vnc.spec: Complete documentation for VNC.
vorbis.spec: The Vorbis General Audio Compression Codec libraries and tools
vorbis.spec: Development tools for Vorbis applications
w3c-libwww.spec: HTTP library of common code
w3c-libwww.spec: Libraries and header files for programs that use libwww.
w3c-libwww.spec: Applications built using Libwww web library: e.g. Robot, command line tool, etc.
watanabe.spec: Watanabe font in SYOTAI CLUB format
wget.spec: A utility for retrieving files using the HTTP or FTP protocols.
which-2.spec: Displays where a particular program in your path is located.
whois.spec: Internet whois/nicname client.
wireless-tools.spec: Wireless ethernet configuration tools
wmakerconf.spec: A configuration tool for the Window Maker window manager for X.
wmconfig.spec: A helper application for configuring X window managers.
words-2.spec: A dictionary of English words for the /usr/dict directory.
wu-ftpd.spec: An FTP daemon provided by Washington University.
wvdial.spec: A heuristic autodialer for PPP connections.
x3270.spec: An X Window System based IBM 3278/3279 terminal emulator.
x3270.spec: IBM 3278/3279 terminal emulator for text mode.

xbill.spec: Stop Bill from loading his OS into all the computers.
xbl.spec: 3d geometry game
xboard.spec: An X Window System graphical chessboard.
xboing.spec: A Breakout style X Window System based game.
xcdroast.spec: An X Window System based tool for creating CDs.
xchat.spec: A GTK+ IRC (chat) client.
xcpustate.spec: An X Window System based CPU state monitor.
xdaliclock.spec: A clock for the X Window System.
xdelta.spec: A binary file delta generator and an RCS replacement library.
xdelta.spec: Static library and header files for Xdelta development.
xemacs.spec: An X Window System based version of GNU Emacs.
xemacs.spec: The .el source files for XEmacs.
xemacs.spec: Information files for XEmacs.
xfig.spec: An X Window System tool for drawing basic vector graphics.
xgammon.spec: An X Window System based backgammon game.
xinetd.spec: A secure replacement for inetd.
xinitrc.spec: The default startup script for the X Window System.
xjewel.spec: An X Window System game of falling jewel blocks.
xlispstat.spec: An implementation of the Lisp language with statistics extensions.
xloadimage-4.1.spec: An X Window System based image viewer.
xlockmore.spec: An X terminal locking program.
xmailbox-2.5.spec: An X Window System utility which notifies you of new mail.
xmlrpc.spec: A set of Python modules for XML-RPC support
xmms.spec: An MP3 player for X which resembles Winamp.
xmms.spec: Static libraries and header files for Xmms plug-in development.
xmms.spec: A GNOME panel applet for the Xmms multimedia player.
xmorph.spec: An X Window System tool for creating morphed images.
xosview.spec: An X Window System utility for monitoring system resources.
xpaint.spec: An X Window System image editing or paint program.
xpat2.spec: A set of Solitaire type games for the X Window System.
xpdf.spec: A PDF file viewer for the X Window System.
xpilot.spec: An X Window System based multiplayer aerial combat game.
xpuzzles.spec: Geometric puzzles and toys for the X Window System.
xrn.spec: An X Window System based news reader.
xsane.spec: An X Window System front-end for the SANE scanner interface.
xsane.spec: A GIMP plugin which provides a scanner interface.
xscreensaver.spec: A set of X Window System screensavers.
xsri.spec: A program for displaying images on the background for X.
xsysinfo.spec: An X Window System kernel parameter monitoring tool.
xtoolwait-1.2.spec: A utility which aims to decrease X session startup time.
xtt-fonts.spec: Free Japanese TrueType fonts (mincho & gothic)
yp-tools.spec: NIS (or YP) client programs.
ypbind.spec: The NIS daemon which binds NIS clients to an NIS domain.
ypserv.spec: The NIS (Network Information Service) server.
ytalk.spec: A chat program for multiple users.
zip.spec: A file compression and packaging utility compatible with PKZIP.
zlib.spec: The zlib compression and decompression library.
zlib.spec: Header files and libraries for developing apps which will use zlib.
zsh.spec: A shell similar to ksh, but with improvements.

#Files	Directory	#Files-by-Language (Sorted)
30776	kde-118n-2.1.1	not=26714, unknown=3996, auto=34, dup=17, perl=6, sh=6, cpp=2, makefile=1 [GPL]
28588	mozilla	not=11080, unknown=7016, cpp=6376, ansic=2426, makefile=878, dup=436, perl=164, sh=82, java=47, asm=45, auto=15, zero=12, csh=6, lex=2, yacc=2, sed=1 [MPL]
15498	XFree86-4.0.3	ansic=5502, not=4888, unknown=3966, dup=696, cpp=129, sh=81, asm=78, tcl=59, makefile=26, auto=23, perl=21, awk=8, yacc=8, lex=6, csh=4, sed=3 [MIT]
11962	teTeX-1.0	not=7204, unknown=2909, ansic=883, dup=552, perl=207, sh=97, cpp=40, auto=33, makefile=12, awk=8, sed=5, csh=3, lex=2, zero=2, pascal=2, yacc=2, asm=1 [Distributable]
10263	gcc-2.96-20000731	ansic=3833, cpp=3450, unknown=1472, not=995, dup=136, fortran=91, auto=80, sh=64, asm=56, exp=56, objc=8, lisp=8, yacc=6, sed=5, perl=2, zero=1 [GPL]
9060	linuxconf-1.24r2	not=5886, unknown=1944, cpp=824, sh=110, makefile=101, perl=79, dup=57, ansic=33, java=21, zero=4, python=1 [GPL]
8880	kernel-2.4.2	ansic=7150, asm=421, not=377, unknown=361, makefile=324, dup=160, auto=36, sh=28, perl=11, awk=5, zero=2, tcl=2, lex=1, sed=1, yacc=1 [GPL]
8555	glibc-2.2.2	ansic=5776, unknown=1079, asm=904, dup=414, not=147, makefile=139, sh=49, awk=23, auto=10, perl=8, sed=5, yacc=1 [LGPL]
6884	kdebase-2.1.1	not=4360, cpp=1133, unknown=911, ansic=361, dup=66, sh=41, perl=6, makefile=3, auto=2, python=1 [GPL]
4956	kdelibs-2.1.1	not=2586, cpp=1349, unknown=875, ansic=78, dup=28, sh=14, java=9, perl=7, auto=5, makefile=2, yacc=2, lex=1 [LGPL]
4883	gdb+dejagnu-20010316	ansic=2261, unknown=1062, not=591, exp=357, dup=302, asm=105, auto=85, cpp=41, tcl=30, sh=26, sed=9, yacc=6, awk=3, makefile=2, fortran=1, java=1, lisp=1 [GPL]
4352	LAPACK	fortran=2739, unknown=1414, not=149, dup=36, makefile=13, ansic=1 [Freely distributable]
4196	mysql-3.23.36	ansic=1297, not=1109, unknown=1030, dup=227, tcl=142, cpp=140, auto=63, perl=59, asm=56, sh=51, java=14, awk=6, sed=1, zero=1 [LGPL]
4132	kdemultimedia-2.1.1	not=1942, cpp=1136, unknown=902, ansic=79, dup=27, auto=23, sh=8, tcl=4, makefile=4, perl=3, asm=2, lex=1, awk=1 [GPL]
4001	qt-2.3.0	not=1917, unknown=1108, cpp=839, ansic=85, dup=36, sh=5, auto=4, perl=3, lisp=1, lex=1, makefile=1, yacc=1 [GPL]
3760	kdegames-2.1.1	not=2360, unknown=775, cpp=533, dup=44, ansic=39, makefile=3, auto=2, sh=2, python=1, perl=1 [GPL]
3501	tcltk-8.3.1	unknown=1148, not=980, ansic=608, tcl=500, sh=129, dup=61, exp=37, auto=27, zero=4, makefile=3, awk=2, perl=1, yacc=1 [BSD]
3499	octave-2.1.33	unknown=2059, cpp=543, dup=220, ansic=218, fortran=211, not=204, sh=30, auto=8, lisp=2, lex=1, exp=1, perl=1, yacc=1 [GPL]
3228	php-4.0.4pl1	not=1949, ansic=629, unknown=555, dup=30, sh=28, cpp=11, perl=6, auto=6, awk=4, java=3, yacc=3, zero=2, lex=2 [PHP]
3193	koffice-2.0.1	not=1712, cpp=1103, unknown=316, dup=23, sh=18, ansic=11, perl=4, makefile=3, lex=1, auto=1, yacc=1 [GPL]
3141	gimp-1.2.1	not=1111, ansic=1078, unknown=705, perl=106, lisp=94, dup=32, auto=8, sh=4, yacc=3 [GPL, LGPL]
2960	binutils-2.10.91.0.2	ansic=1046, unknown=807, asm=611, sh=179, not=155,

```
exp=82,dup=17,sed=16,cpp=11,auto=10,yacc=9,perl=8,lex=6,lisp=1,awk=1,zero=1
[GPL]
2854    openssl-0.9.6    unknown=1038,ansic=751,not=261,python=159,perl=148,
dup=138,makefile=116,cpp=93,sh=50,tcl=49,auto=17,zero=16,asm=10,objc=4,lisp=3,yacc=1
[BSD-like]
2766    libgcj          java=730,not=645,ansic=589,unknown=434,asm=141,dup=101,
cpp=82,sh=15,auto=12,makefile=10,exp=4,perl=2,awk=1
[GPL]
2434    kdenetwork-2.1.1 not=1100,cpp=645,unknown=575,perl=37,auto=25,ansic=23,
dup=20,sh=5,makefile=2,zero=1,tcl=1
[GPL]
2428    kaffe-1.0.6      java=939,not=740,ansic=450,unknown=220,sh=37,cpp=23,
dup=10,auto=4,perl=3,asm=1,awk=1
[GPL]
2387    kdevelop-1.4.1   not=1350,ansic=577,cpp=232,unknown=168,dup=22,asm=14,
sh=8,perl=8,auto=3,lex=1,makefile=1,csh=1,awk=1,java=1
[GPL]
2351    postgresql-7.0.3 ansic=926,unknown=549,not=393,sql=123,makefile=112,
java=91,sh=62,dup=20,cpp=20,tcl=19,perl=10,python=9,lex=4,yacc=4,auto=3,asm=3,sed=1,zero=1,csh=1
[BSD]
2237    anaconda-7.1     not=1436,ansic=427,unknown=166,python=96,makefile=44,
sh=40,auto=10,perl=9,lex=3,yacc=3,dup=2,zero=1
[GPL]
2197    krb5-1.2.2       ansic=1183,not=413,unknown=409,exp=59,sh=50,auto=31,
dup=21,perl=11,makefile=5,yacc=5,sed=4,awk=3,csh=1,lex=1,python=1
[MIT]
2158    lam-6.5.1        ansic=1135,not=830,cpp=120,unknown=45,sh=16,dup=6,
auto=3,fortran=3
[BSD-like]
2140    emacs-20.7       not=674,lisp=615,ansic=467,unknown=345,dup=15,sh=10,
zero=5,asm=2,makefile=2,auto=2,perl=1,csh=1,sed=1
[GPL]
2125    abi             cpp=1244,not=462,unknown=210,makefile=128,ansic=45,
dup=15,sh=12,perl=7,python=1,zero=1
[GPL]
2090    gnome-applets-1.2.4 not=1577,unknown=323,ansic=158,dup=26,sh=3,auto=2,
perl=1
[GPL]
2065    tripwire-2.3.0-50 cpp=882,unknown=403,not=252,dup=226,ansic=215,
auto=54,perl=16,sh=5,sed=5,zero=2,makefile=2,lex=1,awk=1,yacc=1
[GPL]
1895    Mesa-3.4         ansic=972,unknown=332,dup=265,not=173,cpp=53,sh=39,
asm=32,objc=11,makefile=8,auto=4,zero=3,python=3
[GPL/MIT]
1869    kdepim-2.1.1     not=801,cpp=559,unknown=231,ansic=169,dup=75,perl=9,
yacc=8,sh=4,auto=3,zero=3,lex=3,awk=3,makefile=1
[GPL]
1700    bind-9.1.0       ansic=776,unknown=504,not=327,sh=66,perl=14,dup=6,
auto=5,tcl=1,yacc=1
[BSD-like]
1668    perl-5.6.0       perl=849,unknown=451,ansic=165,sh=93,not=67,dup=31,
makefile=6,java=2,yacc=2,auto=1,lisp=1
[Artistic or GPL]
1651    Python-1.5.2     python=784,not=280,ansic=247,unknown=193,dup=109,
sh=22,makefile=9,lisp=3,auto=2,sed=1,perl=1
[Distributable]
1576    xemacs-21.1.14   ansic=603,not=351,unknown=341,lisp=218,dup=27,sh=21,
makefile=7,perl=2,auto=2,csh=1,asm=1,sed=1,zero=1
[GPL]
1573    samba-2.0.7      unknown=629,not=543,ansic=246,sh=72,dup=48,perl=21,
awk=4,makefile=3,csh=3,auto=2,sed=1,asm=1
[GPL]
1566    gs5.50           ansic=953,unknown=308,not=241,sh=29,dup=13,cpp=10,
auto=4,asm=3,perl=2,lex=1,lisp=1,yacc=1
[GPL]
1539    4Suite-0.10.1    python=889,unknown=205,dup=158,not=108,ansic=86,
auto=82,zero=10,makefile=1
[Apache-like]
```

1517	textutils-2.0.11	unknown=1068,not=188,dup=187,ansic=35,sh=23,perl=12, sed=2,auto=2 [GPL]
1439	gnome-libs-1.2.8	not=826,ansic=418,unknown=142,dup=34,auto=7,sh=5, perl=3,lisp=2,awk=2 [LGPL]
1419	gnome-core-1.2.4	not=951,ansic=226,unknown=204,dup=33,auto=2,perl=2, sh=1 [LGPL/GPL]
1419	gnumeric-0.61	not=792,ansic=401,unknown=186,dup=27,perl=6,lisp=2, auto=2,python=2,yacc=1 [GPL]
1349	db-3.1.17	not=689,ansic=239,tcl=150,unknown=100,auto=52,perl=33, java=30,sh=23,cpp=17,awk=9,dup=3,sed=3,makefile=1 [BSD-like]
1308	kdeutils-2.1.1	not=721,cpp=347,unknown=207,dup=27,makefile=2,sh=2, auto=2 [GPL]
1299	kdeadmin-2.1.1	not=617,cpp=509,unknown=134,sh=14,dup=8,ansic=6, perl=4,auto=4,makefile=2,python=1 [GPL]
1291	man-fr-0.9	not=1229,unknown=62 [Distributable]
1280	kdegraphics-2.1.1	not=647,cpp=302,unknown=193,ansic=109,dup=19,auto=4, sh=4,makefile=2 [GPL]
1167	unixODBC-1.8.13	ansic=624,not=282,dup=129,cpp=87,unknown=28,auto=10, sh=4,sql=1,lex=1,yacc=1 [LGPL]
1160	apache_1.3.19	not=421,unknown=342,ansic=280,dup=49,sh=39,perl=15, auto=7,makefile=3,lisp=1,lex=1,cpp=1,yacc=1 [Apache]
1153	FreeWnn-1.1.1-a017	unknown=575,not=318,ansic=256,dup=1,sh=1,cpp=1, auto=1 [Distributable]
1139	gsl-0.7	ansic=832,not=190,unknown=98,dup=12,sh=5,zero=1, auto=1 [GPL]
1133	gnome-utils-1.2.1	not=780,ansic=167,unknown=135,dup=45,auto=4,lisp=1, yacc=1 [GPL]
1124	netpbm-9.9	ansic=509,not=301,unknown=278,makefile=16,sh=8,csh=6, perl=3,dup=2,auto=1 [Free]
1116	lclint-2.5q	ansic=627,unknown=347,not=117,auto=8,makefile=7, dup=6,yacc=3,lex=1 [GPL]
1108	w3c-libwww-5.2.8	not=537,ansic=364,unknown=127,cpp=44,dup=20,perl=9, sh=5,auto=1,zero=1 [W3C]
1086	squid-2.3.STABLE4	unknown=746,ansic=222,not=84,perl=19,sh=9,makefile=3, auto=2,dup=1 [GPL]
1068	man-pages-ja-0.4	not=1043,unknown=17,sh=6,makefile=2 [Distributable]
1061	pine4.33	unknown=398,dup=357,ansic=223,not=56,sh=14,makefile=9, csh=2,auto=1,perl=1 [Freely distributable]
1057	isd4k-utils	ansic=345,not=310,unknown=222,dup=43,sh=35,auto=31, perl=27,makefile=20,cpp=20,tcl=2,sql=2 [GPL]
1032	vnc_unixsrc	ansic=603,unknown=262,not=129,dup=13,sh=10,cpp=5, asm=4,perl=2,auto=2,makefile=2 [GPL]
1030	xfig.3.2.3c	not=832,ansic=175,unknown=22,makefile=1 [Free]
1007	krb4-1.0.5	ansic=533,not=223,unknown=176,cpp=33,perl=18,sh=12, yacc=3,makefile=2,asm=2,lex=2,awk=1,dup=1,auto=1 [Freely distributable]
985	WindowMaker-0.64.0	not=422,unknown=272,ansic=259,sh=15,dup=10,perl=3, auto=2,makefile=1,lisp=1 [GPL]

984	man-pages-1.35	not=979,unknown=4,makefile=1 [Distributable]
952	a2ps-4.13	not=372,unknown=286,ansic=210,sh=52,auto=6,perl=6, lex=6,sed=3,lisp=2,sql=2,yacc=2,python=1,ada=1,objc=1,java=1,dup=1 [GPL]
935	wv	not=566,ansic=248,unknown=67,sh=24,dup=16,awk=6, perl=3,csh=2,auto=1,zero=1,makefile=1 [GPL]
931	ncurses-5.2	ansic=292,not=274,unknown=170,ada=141,sh=21,cpp=16, awk=6,auto=3,dup=3,perl=2,sed=2,makefile=1 [Distributable]
917	gnome-games-1.2.0	not=512,ansic=152,unknown=149,cpp=41,lisp=34,dup=27, auto=2 [LGPL]
915	openldap-2.0.7	ansic=392,not=209,unknown=193,sh=59,perl=19,sql=18, cpp=16,dup=5,auto=2,makefile=1,tcl=1 [OpenLDAP]
864	manpages-es-0.6a	not=856,unknown=7,makefile=1 [Distributable]
858	SDL-1.1.7	not=477,ansic=317,unknown=19,cpp=15,sh=10,asm=6, dup=6,makefile=3,perl=3,auto=2 [LGPL]
852	console-tools-0.3.3	unknown=598,not=125,ansic=73,dup=33,sh=14,perl=4, auto=3,lex=1,yacc=1 [GPL]
850	groff-1.16.1	unknown=529,cpp=137,not=101,ansic=48,sh=12,sed=6, makefile=5,perl=3,auto=3,yacc=3,dup=2,asm=1 [GPL]
819	sgml-tools-1.0.9	unknown=247,dup=243,not=126,ansic=82,cpp=81,perl=23, makefile=4,sh=4,auto=4,lisp=2,lex=2,awk=1 [Free]
816	kdetoys-2.1.1	not=639,unknown=91,cpp=56,dup=21,ansic=5,makefile=1, sh=1,auto=1,perl=1 [GPL]
804	ucd-snmp-4.2	ansic=360,not=198,unknown=135,sh=61,perl=37,dup=11, auto=2 [BSD-like]
803	gmp-3.1.1	ansic=436,asm=252,not=78,unknown=22,sh=5,dup=4,perl=1, lex=1,auto=1,fortran=1,lisp=1,yacc=1 [LGPL]
802	mc-4.5.51	ansic=339,not=293,unknown=111,dup=24,sh=22,perl=5, auto=3,csh=2,zero=2,awk=1 [GPL]
801	dia-0.86	not=321,ansic=275,unknown=176,dup=27,auto=2 [GPL]
801	xboard-4.1.0	not=448,unknown=309,ansic=28,sh=5,dup=5,sed=3,auto=1, csh=1,lex=1 [GPL]
792	htdig-3.2.0b3	cpp=281,not=193,ansic=179,unknown=64,auto=28,sh=22, perl=15,dup=8,makefile=2 [GPL]
785	vim60z	unknown=462,not=205,ansic=89,sh=11,makefile=6,perl=4, awk=3,auto=2,dup=1,zero=1,csh=1 [Free]
768	bash-2.04	unknown=349,ansic=237,sh=99,not=62,dup=8,auto=5, makefile=4,asm=2,perl=1,yacc=1 [GPL]
741	kdesdk-2.1.1	not=393,unknown=126,cpp=121,dup=38,auto=27,sh=15, perl=12,ansic=5,python=2,makefile=1,lisp=1 [GPL]
740	ircii-4.4Z	unknown=624,ansic=97,not=14,sh=3,auto=1,lex=1 [Distributable]
715	linux-86	ansic=420,unknown=153,not=48,makefile=36,asm=31, dup=13,sh=12,zero=2 [GPL]
702	openldap-1.2.11	ansic=315,unknown=190,not=171,sh=14,dup=6,makefile=2, auto=2,tcl=1,python=1 [OpenLDAP]
692	licq-1.0.2	not=268,unknown=162,cpp=139,dup=70,ansic=21,sh=17, perl=8,auto=6,csh=1

		[GPL]
673	pl_PL	not=648,unknown=22,auto=1,perl=1,sh=1
		[Distributable]
666	gtk+-1.2.9	ansic=313,not=196,unknown=100,makefile=35,dup=9, sh=4,perl=3,awk=3,auto=2,lisp=1
		[LGPL]
655	cdrecord-1.9	unknown=292,ansic=265,not=57,sh=18,makefile=17,auto=2, perl=2,dup=1,sh=1
		[GPL]
644	mgetty-1.1.25	unknown=224,ansic=209,not=98,sh=49,perl=33,makefile=24, lisp=3,tcl=3,dup=1
		[GPL]
631	freetype-2.0.1	ansic=296,unknown=156,not=129,dup=15,cpp=14,sh=8, makefile=5,auto=4,python=3,lex=1
		[BSD-like]
629	freeciv-1.11.4	ansic=339,unknown=150,not=105,dup=21,sh=10,auto=4
		[GPL]
609	fvwm-2.2.4	not=321,ansic=191,unknown=57,cpp=15,perl=10,sh=9, dup=3,lex=1,auto=1,yacc=1
		[GPL]
607	openjade-1.3	cpp=371,unknown=123,not=69,ansic=27,perl=5,dup=5, makefile=3,sh=3,sh=1
607	ntp-4.0.99k	ansic=260,not=197,unknown=78,sh=22,dup=17,perl=15, awk=9,asm=5,auto=3,sh=1
		[Distributable]
593	e2fsprogs-1.19	not=217,ansic=190,unknown=132,dup=21,sh=13,zero=8, auto=5,awk=4,sh=2,perl=1
		[GPL]
588	fileutils-4.0.36	ansic=178,not=177,unknown=93,sh=92,dup=42,perl=3, auto=2,yacc=1
		[GPL]
583	doxygen-1.2.6	cpp=259,unknown=222,not=72,lex=9,makefile=8,auto=8, perl=3,sh=1,yacc=1
		[GPL]
567	ImageMagick-5.2.7	not=201,ansic=166,unknown=91,perl=44,cpp=42,sh=9, dup=9,auto=2,makefile=2,tcl=1
		[Free]
562	libxml-1.8.10	unknown=454,not=60,ansic=39,dup=6,sh=2,auto=1
		[LGPL]
551	glade-0.5.9	not=301,ansic=176,unknown=39,dup=26,auto=6,sh=3
		[GPL]
543	imap-2000	ansic=307,dup=110,unknown=82,not=32,makefile=8,sh=4
		[University of Washington's Free-Fork License]
537	inn-2.3.1	ansic=216,not=120,unknown=81,perl=45,sh=42,makefile=24, python=3,yacc=2,lex=1,tcl=1,dup=1,auto=1
		[GPL]
534	xlockmore-4.17.2	not=235,ansic=186,unknown=67,cpp=19,sh=8,perl=6, tcl=4,java=3,dup=2,makefile=2,auto=2
		[MIT]
531	libgtop-1.0.10	ansic=266,not=125,dup=78,unknown=48,auto=6,perl=4, sh=3,asm=1
		[LGPL]
528	sh-utils-2.0	unknown=221,not=147,dup=83,ansic=59,sh=10,perl=5, auto=2,yacc=1
		[GPL]
525	sawfish-0.36	not=287,lisp=133,unknown=52,ansic=29,sh=11,perl=8, dup=4,auto=1
		[GPL]
518	man-pages-de-0.2	not=514,unknown=3,makefile=1
		[Distributable]
513	VFLib2-2.25.1	unknown=167,not=151,perl=128,ansic=44,sh=10,dup=6, makefile=4,auto=3
		[GPL]
512	xscreensaver-3.29	ansic=233,not=207,unknown=49,dup=14,sh=5,auto=3, perl=1
		[BSD]
506	Inti-0.6preview	cpp=263,unknown=144,not=48,dup=27,python=20,sh=1, auto=1,zero=1,ansic=1
		[LGPL]
506	rpm-4.0.2	ansic=146,not=143,unknown=79,dup=66,sh=59,perl=9, auto=4
		[GPL]

502	xlispstat-3-52-18	ansic=231,lisp=137,unknown=61,not=54,makefile=7, auto=5,dup=4,sh=2,csch=1 [Distributable]
501	xmms-1.2.4	ansic=205,not=148,unknown=101,dup=34,asm=6,auto=4, sh=3 [GPL]
492	sendmail-8.11.2	not=189,unknown=166,ansic=90,sh=21,makefile=14,perl=11, dup=1 [BSD]
482	lout-3.17	unknown=411,ansic=50,not=20,makefile=1 [GPL]
481	xpilot-4.3.0	ansic=220,unknown=149,not=76,cpp=28,makefile=4,sh=2, tcl=1,perl=1 [GPL]
479	balsa-1.1.1	ansic=202,not=184,unknown=59,dup=27,sh=3,auto=3, awk=1 [GPL]
473	control-center-1.2.2	not=301,ansic=76,unknown=67,dup=26,auto=2,sh=1 [GPL/LGPL]
473	libvorbis-1.0beta4	unknown=157,not=138,ansic=106,dup=36,auto=16, sh=14,makefile=3,perl=3 [GPL/BSD]
466	pam-0.74	not=150,ansic=120,unknown=119,makefile=50,sh=16, auto=3,perl=3,yacc=3,lex=1,dup=1 [GPL or BSD]
462	gnupg-1.0.4	ansic=178,unknown=83,not=70,asm=69,sh=34,dup=23, auto=4,makefile=1 [GPL]
458	tcsh-6.10.00	unknown=361,ansic=75,not=12,sh=5,dup=1,makefile=1, csch=1,auto=1,lisp=1 [Distributable]
449	aspell-.32.6	unknown=238,not=99,cpp=46,sh=43,perl=14,dup=3,ansic=3, makefile=2,auto=1 [LGPL]
448	gawk-3.0.6	unknown=162,awk=127,not=75,ansic=68,sh=8,dup=5,auto=1, makefile=1,yacc=1 [GPL]
442	xboing	not=225,unknown=144,ansic=72,sh=1 [MIT]
431	shadow-20000826	ansic=153,not=123,unknown=107,dup=27,sh=16,auto=2, makefile=1,perl=1,yacc=1 [BSD]
428	lynx2-8-4	ansic=205,unknown=123,not=72,dup=19,sh=5,perl=2, makefile=1,auto=1 [GPL]
427	gaim-0.11.0pre4	not=192,ansic=156,unknown=38,dup=31,sh=5,perl=3, auto=2 [GPL]
425	kdesupport-2.1	not=148,unknown=74,cpp=72,dup=61,ansic=55,sh=7,perl=3, auto=3,makefile=2 [LGPL/GPL]
423	Canna35b2	unknown=196,ansic=130,not=88,cpp=4,sh=2,lex=1,awk=1, yacc=1 [Distributable]
422	cvs-1.11	ansic=219,unknown=105,not=67,sh=13,perl=8,dup=3, auto=3,csch=1,lisp=1,makefile=1,yacc=1 [GPL]
415	trXFree86-2.1.2	not=404,unknown=9,makefile=1,tcl=1 [Distributable]
412	am-utils-6.0.5	ansic=210,not=134,unknown=30,sh=20,perl=8,auto=3, dup=3,lex=2,yacc=2 [BSD]
410	nvi-1.79	ansic=262,unknown=79,not=29,dup=15,makefile=7,perl=5, tcl=5,awk=4,sh=2,csch=1,auto=1 [GPL]
410	x3270-3.2	dup=139,ansic=134,not=89,unknown=27,sh=12,auto=6, exp=2,makefile=1 [MIT]
405	wmconfig-0.9.10	unknown=345,not=27,ansic=23,auto=4,dup=4,sh=2 [GPL]
405	gphoto-0.4.3	ansic=227,not=141,unknown=27,dup=7,auto=2,sh=1

```

[GPL]
399      util-linux-2.10s dup=152,not=112,unknown=84,ansic=42,makefile=7,
      auto=1,sh=1
[Distributable]
396      sane-1.0.3      ansic=167,unknown=109,not=96,java=10,sh=7,dup=4,
      auto=2,lisp=1
[GPL (programs), relaxed LGPL (libraries), and public domain
(docs)]
394      extace-1.4.4      ansic=210,not=77,unknown=72,dup=11,sh=10,perl=7,
      makefile=4,auto=3
[GPL]
391      gal-0.4.1      ansic=174,not=155,unknown=32,dup=26,auto=2,perl=2
[GPL]
387      gnuplot-3.7.1      unknown=241,ansic=94,not=27,objc=7,sh=4,asm=3,dup=2,
      makefile=2,csh=2,lisp=2,perl=2,auto=1
[Distributable]
384      xpdf-0.92      cpp=123,not=91,ansic=66,unknown=54,dup=44,sh=4,auto=2
[GPL]
382      TiMidity+-2.10.3a2 ansic=234,not=96,unknown=39,dup=6,tcl=4,auto=2,
      lisp=1
[GPL]
375      mount-2.10r      ansic=165,not=110,unknown=63,makefile=15,sh=10,dup=6,
      csh=3,perl=1,sh=1,auto=1
[GPL]
374      guile-1.3.4      ansic=210,not=53,unknown=32,lisp=31,asm=19,makefile=9,
      dup=8,awk=6,sh=3,auto=2,csh=1
[GPL]
360      enlightenment-0.16.4 not=164,ansic=112,unknown=60,dup=10,sh=7,perl=3,
      auto=2,makefile=1,zero=1
[GPL]
358      pilot-link      not=111,ansic=100,unknown=59,java=58,cpp=10,perl=7,
      dup=5,sh=2,python=2,auto=1,zero=1,tcl=1,yacc=1
[GPL]
358      jed-B0.99-12      unknown=213,ansic=96,not=44,dup=2,makefile=2,auto=1
[GPL]
357      users-guide-1.2 not=346,unknown=7,dup=3,auto=1
[GPL]
354      tiff-v3.5.5      not=138,unknown=88,ansic=72,dup=39,sh=9,makefile=8
[Distributable]
352      nmh-1.0.4      ansic=210,not=86,unknown=45,sh=7,sh=1,awk=1,auto=1,
      dup=1
[Free]
351      koi8-ub      not=325,unknown=23,perl=2,makefile=1
[Distributable]
349      unzip-5.41      unknown=149,ansic=130,not=32,makefile=10,asm=9,cpp=9,
      dup=5,zero=4,sh=1
[BSD]
345      kdebindings-2.1.1 not=155,unknown=65,cpp=43,ansic=21,auto=21,dup=17,
      python=8,sh=6,java=5,perl=3,makefile=1
[GPL]
344      ORBit-0.5.7      ansic=148,not=106,unknown=48,dup=29,sh=7,auto=4,
      lex=1,yacc=1
[LGPL/GPL]
343      gated-public-3_6 ansic=164,not=147,unknown=21,sh=7,auto=2,makefile=1,
      lex=1
[Distributable]
342      gedit-0.9.4      not=197,unknown=61,ansic=55,dup=26,auto=2,sh=1
[GPL]
337      LPRng-3.7.4      ansic=108,unknown=91,not=69,sh=42,perl=15,dup=6,
      makefile=4,auto=2
[GPL and Artistic]
336      man-pages-it-0.3.0 not=279,unknown=55,sh=1,makefile=1
[Distributable]
335      tcpdump-3.4      ansic=155,unknown=112,dup=25,not=21,awk=8,sh=6,auto=5,
      lex=1,csh=1,yacc=1
[BSD]
335      gnome-objc-1.0.2 objc=245,not=57,unknown=20,dup=7,auto=3,sh=2,ansic=1
[LGPL]
333      gnome-print-0.25 ansic=122,not=107,unknown=75,dup=27,auto=2
[LGPL]
332      blt2.4u      not=136,ansic=87,tcl=76,unknown=28,dup=3,auto=1,sh=1
[MIT]
```

```
324      openssh-2.5.2p2  ansic=217,unknown=58,not=29,sh=13,perl=3,auto=2,
                        dup=2
                        [BSD]
322      gtk-engines-0.10 not=233,unknown=32,dup=23,ansic=19,sh=9,auto=6
                        [GPL]
320      ncpfs-2.2.0.18  ansic=166,not=82,unknown=33,dup=26,sh=9,auto=2,tcl=1,
                        makefile=1
                        [GPL]
318      uucp-1.06.1     ansic=245,unknown=38,not=21,sh=10,perl=2,auto=1,
                        dup=1
                        [GPL]
315      docbook-style-dsssl-1.59 unknown=190,not=122,perl=2,makefile=1
                        [Distributable]
315      pan-0.9.5       ansic=144,not=126,unknown=35,auto=6,dup=4
                        [GPL]
314      wu-ftpd         unknown=163,not=70,ansic=62,auto=6,sh=6,dup=5,perl=1,
                        yacc=1
                        [BSD]
309      gnome-pim-1.2.0 not=131,ansic=114,unknown=52,dup=8,auto=2,yacc=2
                        [GPL]
306      amanda-2.4.2p2  ansic=154,not=70,unknown=44,sh=22,perl=6,dup=2,makefile=2,
                        lex=1,awk=1,auto=1,tcl=1,sql=1,yacc=1
                        [BSD]
302      zip-2.3         dup=164,unknown=109,not=21,ansic=8
                        [Distributable]
301      rp3-1.1.10     not=93,cpp=81,unknown=55,dup=39,ansic=19,sh=6,makefile=5,
                        auto=3
                        [GPL]
300      fnlib-0.5       not=180,unknown=105,ansic=7,dup=5,sh=2,auto=1
                        [LGPL]
299      xpat2-1.06     unknown=155,ansic=62,not=54,cpp=16,makefile=7,sh=3,
                        dup=2
                        [Distributable - most of it GPL]
298      automake-1.4   sh=200,not=75,unknown=14,dup=4,perl=3,auto=1,ansic=1
                        [GPL]
295      enscript-1.6.1 not=115,ansic=64,unknown=48,sh=48,dup=11,perl=4,
                        auto=2,lex=1,lisp=1,yacc=1
                        [GPL]
290      mutt-1.2.5     ansic=137,not=70,unknown=61,dup=13,sh=7,auto=2
                        [GPL]
286      findutils-4.1.6 not=99,unknown=75,dup=59,ansic=44,sh=4,exp=3,auto=2
                        [GPL]
285      cyrus-sasl-1.5.24 not=124,ansic=48,unknown=42,cpp=39,java=24,dup=7,
                        auto=1
                        [Distributable]
283      elm2.5.3       ansic=169,unknown=79,not=19,sh=8,makefile=7,awk=1
                        [Distributable]
280      hellas         not=253,unknown=19,perl=4,dup=2,sh=2
                        [Distributable]
280      xpuzzles-5.5.2  ansic=88,not=77,dup=52,unknown=51,auto=11,makefile=1
                        [MIT]
278      lsof_4.51      ansic=194,unknown=27,sh=20,makefile=15,perl=10,not=6,
                        dup=3,awk=2,asm=1
                        [Free]
272      pcmcia-cs-3.1.24 dup=129,unknown=55,not=50,ansic=18,sh=16,makefile=4
                        [GPL]
269      gnome-media-1.2.0 not=125,unknown=65,ansic=49,dup=27,auto=2,sed=1
                        [LGPL]
266      libungif-4.1.0b1 not=122,ansic=85,unknown=34,dup=13,sh=6,auto=5,
                        perl=1
                        [MIT-like]
259      tar-1.13.19    dup=80,not=74,unknown=45,ansic=37,sh=20,auto=2,perl=1
                        [GPL]
256      Xaw3d-1.5      ansic=135,dup=105,unknown=13,lex=1,makefile=1,yacc=1
                        [MIT]
254      ed-0.2         unknown=217,ansic=18,not=12,sh=4,dup=1,auto=1,makefile=1
                        [GPL]
251      iptables-1.2.1a unknown=92,ansic=83,not=36,makefile=29,sh=11
                        [GPL]
249      texinfo-4.0    ansic=78,not=60,dup=43,unknown=41,sh=18,awk=3,auto=3,
                        lisp=1,perl=1,sed=1
```

		[GPL]
247	libiconv	ansic=118,unknown=100,not=17,sh=6,dup=2,auto=2,makefile=2
		[GPL]
245	kdbg-1.2.0	not=137,cpp=69,unknown=31,dup=4,sh=3,auto=1
		[GPL]
244	nfs-utils-0.3.1	ansic=102,not=60,unknown=38,makefile=25,sh=15,perl=2, dup=1,auto=1
		[GPL]
243	exmh-2.2	tcl=124,unknown=54,not=52,perl=6,sh=4,exp=2,makefile=1
		[Free]
238	xchat-1.6.3	ansic=74,not=70,unknown=62,dup=21,perl=3,python=3, auto=3,makefile=1,sh=1
		[GPL]
236	mod_perl-1.24_01	unknown=116,perl=87,not=18,ansic=11,makefile=2,sh=2
		[GPL]
236	librep-0.13.3	lisp=78,ansic=55,not=45,dup=26,unknown=22,sh=9,auto=1
		[GPL]
234	gnorpm-0.96	not=99,unknown=55,ansic=55,dup=23,auto=2
		[GPL]
230	xloadimage.4.1	ansic=161,unknown=49,not=12,sh=4,dup=2,makefile=2
		[MIT]
227	slang-1.4.2	ansic=99,unknown=89,not=26,makefile=6,dup=4,auto=2, sh=1
		[GPL]
225	gettext-0.10.35	ansic=59,not=54,unknown=47,dup=31,sh=27,auto=2,yacc=2, lisp=1,sed=1,perl=1
		[GPL]
225	make-3.79.1	unknown=128,ansic=53,not=29,dup=9,sh=3,perl=2,auto=1
		[GPL]
224	mtools-3.9.7	ansic=87,unknown=83,not=35,sh=12,dup=3,auto=2,sed=2
		[GPL]
223	gnome-python-1.0.53	python=79,not=77,unknown=37,ansic=16,dup=11, auto=3
		[LGPL]
222	multimedia	ansic=111,not=89,unknown=13,makefile=7,sh=1,zero=1
		[GPL]
215	man-1.5h1	not=99,unknown=56,ansic=34,sh=20,awk=2,perl=2,makefile=1, dup=1
		[GPL]
213	ical-2.2	cpp=77,tcl=66,not=38,unknown=17,sh=8,auto=2,ansic=2, dup=1,makefile=1,perl=1
		[Distributable]
212	ncftp-3.0.2	ansic=117,unknown=40,not=36,cpp=9,sh=7,dup=2,auto=1
		[Distributable]
208	xsane-0.62	not=93,unknown=46,ansic=37,dup=25,auto=3,sh=2,zero=1, sed=1
		[GPL]
206	printconf-0.2.12	not=112,unknown=52,auto=15,dup=7,sh=6,ansic=5,perl=4, python=4,zero=1
		[GPL]
202	libodbc++-0.2.2pre4	not=130,cpp=47,unknown=17,dup=4,sh=3,auto=1
		[LGPL]
202	gdm-2.0beta2	not=77,unknown=70,dup=27,ansic=19,sh=5,auto=4
		[LGPL/GPL]
201	kinput2-v3	ansic=134,unknown=62,not=5
		[Distributable]
200	bug-buddy-1.2	not=99,unknown=53,ansic=26,dup=19,auto=2,sh=1
		[GPL]
199	gv-3.5.8	ansic=125,unknown=36,not=30,sh=6,dup=2
		[GPL]
199	mikmod-3.1.6	ansic=89,not=52,unknown=41,dup=13,auto=2,awk=1,sh=1
		[LGPL]
195	xinetd-2.1.8.9pre14	ansic=122,not=38,unknown=17,dup=8,sh=5,perl=2, makefile=1,zero=1,auto=1
		[BSD-like]
193	parted-1.4.7	ansic=76,not=55,dup=26,unknown=24,sh=9,auto=2,asm=1
		[GPL]
191	grep-2.4.2	not=74,unknown=45,dup=34,ansic=22,sh=10,awk=3,auto=2, sed=1
		[GPL]
190	man-pages-cs-0.14	not=182,unknown=6,makefile=1,sh=1
		[Distributable]

189	ee-0.3.12	not=92,unknown=38,ansic=30,dup=25,auto=2,sh=2 [GPL]
189	ftpcopy-0.3.4	ansic=159,unknown=10,not=9,sh=9,dup=1,makefile=1 [Free]
188	gtop-1.0.11	not=74,unknown=43,ansic=41,dup=26,cpp=2,auto=2 [LGPL]
187	apacheconf-0.7	not=102,unknown=39,dup=26,auto=13,python=4,ansic=2, sh=1 [GPL]
186	lm_sensors-2.5.5	unknown=86,ansic=64,not=13,perl=8,sh=7,dup=4,makefile=2, lex=1,yacc=1 [GPL]
185	ppp-2.4.0	ansic=90,unknown=68,sh=14,not=9,dup=2,makefile=1, exp=1 [Distributable]
183	scheme-3.2	lisp=117,unknown=30,ansic=29,not=4,makefile=2,sh=1 [GPL]
180	manpages-ru-0.6	not=170,unknown=9,makefile=1 [Distributable]
179	libpng-1.0.9	unknown=97,ansic=41,not=35,sh=3,cpp=2,makefile=1 [Distributable]
177	fetchmail-5.7.4	ansic=65,unknown=41,not=25,dup=20,sh=12,perl=6,auto=2, awk=2,lisp=1,lex=1,python=1,yacc=1 [GPL]
177	gimp-data-extras-1.2.0	unknown=156,not=17,dup=2,auto=1,sh=1 [GPL]
175	net-tools-1.57	ansic=78,not=60,unknown=32,makefile=4,sh=1 [GPL]
174	strace	ansic=76,unknown=40,not=29,sh=17,dup=4,perl=3,makefile=2, auto=2,lisp=1 [Distributable]
174	glms-1.03	unknown=93,not=41,dup=26,auto=8,sh=3,ansic=3 [GPL]
172	file-3.33	unknown=143,ansic=15,not=10,dup=1,sh=1,auto=1,perl=1 [Distributable]
171	nasm-0.98	ansic=82,unknown=38,asm=20,not=18,dup=5,perl=3,auto=2, makefile=2,sh=1 [GPL]
169	plugger-3.2	not=112,unknown=24,makefile=15,ansic=11,auto=4,cpp=2, java=1 [GPL]
169	pspell-.11.2	not=77,unknown=37,cpp=32,dup=8,ansic=6,auto=3,sh=3, perl=2,zero=1 [LGPL]
169	wmakerconf-2.6.1	not=64,unknown=35,ansic=34,dup=26,perl=6,auto=3, sh=1 [GPL]
166	rxvt-2.7.5	unknown=90,ansic=37,not=28,dup=5,sh=4,auto=2 [Distributable]
164	nss_ldap-149	ansic=64,unknown=62,not=21,dup=10,perl=3,auto=2,sh=2 [LGPL]
163	xpaint	ansic=83,not=64,unknown=15,sh=1 [MIT]
162	m4-1.4.1	unknown=82,not=43,ansic=25,dup=6,sh=3,auto=2,lisp=1 [GPL]
162	db2	ansic=113,auto=21,unknown=20,cpp=3,asm=2,not=2,makefile=1 [GPL]
153	xosview-1.7.3	cpp=112,not=19,unknown=13,dup=4,ansic=2,makefile=1, auto=1,awk=1 [GPL/BSD]
152	jpeg-6b	dup=90,unknown=37,not=20,sh=2,ansic=2,auto=1 [Distributable]
152	pkgconfig-0.5.0	dup=68,not=43,unknown=19,ansic=19,auto=2,zero=1 [GPL]
151	gftp-2.0.7b	not=69,unknown=34,ansic=23,dup=21,auto=2,makefile=1, sh=1 [GPL]
150	wget-1.6	unknown=51,not=47,ansic=43,dup=6,sh=1,auto=1,perl=1 [GPL]
150	slrn-0.9.6.4	ansic=66,unknown=62,not=15,dup=3,auto=2,makefile=1, sh=1

		[GPL]
150	rdist-6.1.5	ansic=70,unknown=59,dup=5,sh=5,not=5,makefile=3, perl=2,yacc=1
		[BSD]
148	ypserv-1.3.11	ansic=57,not=52,unknown=21,sh=9,auto=3,dup=3,perl=1, sed=1,makefile=1
		[GPL]
147	lrzsz-0.12.20	ansic=39,not=31,dup=26,unknown=22,exp=19,sh=7,auto=2, makefile=1
		[GPL]
147	transfig.3.2.3c	not=63,ansic=61,unknown=15,makefile=4,csch=2,sh=2
		[Distributable]
147	modutils-2.4.2	ansic=63,not=34,dup=25,unknown=21,sh=1,auto=1,lex=1, yacc=1
		[GPL]
146	sed-3.02	unknown=66,sed=26,not=25,ansic=16,dup=8,sh=4,auto=1
		[GPL]
145	sysctlconfig-0.13	dup=47,ansic=43,not=39,unknown=14,auto=1,sh=1
		[GPL]
145	Maelstrom-3.0.1	c++=56,unknown=40,not=34,ansic=6,dup=5,auto=3,sh=1
		[LGPL]
144	lv4494	ansic=86,unknown=31,perl=13,not=9,sh=2,csch=1,auto=1, dup=1
		[Distributable]
143	mawk-1.3.3	ansic=67,unknown=37,awk=25,not=9,sh=2,makefile=1, auto=1,yacc=1
		[GPL]
142	libglade-0.14	not=74,dup=28,unknown=21,ansic=14,sh=2,auto=2,python=1
		[LGPL]
137	ddskk20010225	lisp=64,unknown=42,not=16,awk=6,sql=5,makefile=3, ansic=1
		[GPL]
137	internet-config-0.40	not=85,c++=34,unknown=14,sh=2,auto=1,perl=1
		[GPL]
136	ash-0.3.7.orig	unknown=69,ansic=56,sh=5,not=2,makefile=2,lex=1, yacc=1
		[BSD]
134	jpeg-6a	ansic=48,dup=36,unknown=29,not=20,auto=1
		[Distributable]
134	switchdesk-3.9.5	not=69,dup=26,unknown=15,sh=13,ansic=4,c++=3,perl=2, auto=1,python=1
		[GPL]
133	less-358	unknown=73,ansic=48,not=7,auto=2,dup=2,awk=1
		[GPL]
133	sox-12.17.1	ansic=90,unknown=17,not=13,sh=7,perl=2,dup=2,makefile=1, auto=1
		[Distributable]
133	mars_nwe	ansic=66,unknown=57,not=6,sh=3,makefile=1
		[GPL]
132	gnome-linuxconf-0.64	ansic=74,not=39,unknown=9,dup=5,auto=4,sh=1
		[GPL]
131	wvdial-1.41	dup=88,unknown=32,not=10,auto=1
		[LGPL]
130	libogg-1.0beta4	not=86,unknown=22,dup=10,sh=5,ansic=5,auto=1,makefile=1
		[BSD]
130	db.1.85	ansic=74,unknown=17,not=17,dup=10,makefile=9,sh=2, csch=1
		[BSD]
129	macutils	ansic=96,unknown=14,not=9,makefile=9,dup=1
		[Distributable]
128	gdk-pixbuf-0.8.0	not=69,ansic=35,unknown=11,dup=7,asm=4,auto=1,sh=1
		[LGPL]
128	njamd-0.8.0	not=57,ansic=44,unknown=15,dup=7,sh=3,auto=1,perl=1
		[GPL]
128	bindconf-1.4	not=102,unknown=16,python=4,auto=3,dup=3
		[GPL]
127	iproute2	ansic=74,unknown=20,sh=15,not=9,makefile=5,perl=3, dup=1
		[GPL]
126	audiofile-0.1.11	ansic=71,unknown=24,not=21,dup=7,sh=2,auto=1
		[LGPL]
126	libtool-1.3.5	not=42,sh=35,unknown=19,ansic=14,dup=11,auto=5

		[GPL]
126	xrn-9.02	ansic=89,unknown=20,not=6,csch=3,sh=2,yacc=2,awk=1, perl=1,lex=1,makefile=1
		[Distributable]
125	links-0.95	unknown=50,ansic=42,not=15,sh=11,awk=3,dup=2,auto=1, perl=1
		[GPL]
125	pdcksh-5.2.14	ansic=53,unknown=51,not=9,sh=6,dup=2,makefile=1, auto=1,perl=1,sed=1
		[Public domain]
124	firewall-config-0.95	not=46,cpp=21,ansic=21,dup=15,unknown=14,sh=4, perl=2,auto=1
		[GPL]
123	mailx-8.1.1	unknown=87,ansic=31,makefile=3,not=1,sh=1
		[BSD]
123	pnm2ppa-1.04	ansic=42,unknown=40,not=28,sh=10,makefile=3
		[GPL]
123	initscripts-5.83	not=42,sh=40,unknown=19,ansic=15,makefile=3,python=2, dup=1,csch=1
		[GPL]
121	kon2-0.3.9b	unknown=52,dup=29,ansic=24,not=10,makefile=5,sh=1
		[Distributable]
120	screen-3.9.8	ansic=46,unknown=45,not=17,sh=9,dup=1,auto=1,makefile=1
		[GPL]
120	sharutils-4.2.1	not=35,unknown=33,ansic=31,dup=11,sh=5,perl=3,auto=2
		[GPL]
120	netkit-telnet-0.17-pre20000412	ansic=47,cpp=26,not=24,unknown=15, makefile=6,dup=1,auto=1
		[BSD]
120	glib-1.2.9	ansic=56,not=36,unknown=18,dup=7,sh=2,auto=1
		[LGPL]
120	mm2.7	csch=44,not=24,unknown=24,ansic=18,sh=6,makefile=4
		[Distributable]
119	esound-0.2.22	not=59,ansic=37,unknown=12,dup=5,sh=4,csch=1,auto=1
		[GPL]
118	gnuchess-4.0.pl80	unknown=48,ansic=39,not=25,sh=2,makefile=1,csch=1, auto=1,dup=1
		[GPL]
117	gnome-lokkit-0.42	not=69,unknown=34,ansic=9,sh=4,perl=1
		[GPL]
117	memprof-0.4.1	not=40,unknown=30,dup=27,ansic=18,auto=2
		[GPL]
117	pinfo-0.6.0	ansic=44,not=35,unknown=19,dup=16,auto=2,sh=1
		[GPL]
116	minicom-1.83.1	unknown=55,ansic=37,not=20,makefile=2,sh=2
		[GPL]
116	sysvinit-2.78	unknown=41,not=25,ansic=25,sh=23,makefile=2
		[GPL]
115	bison-1.28	ansic=32,not=28,unknown=28,dup=24,auto=2,sh=1
		[GPL]
113	gperf-2.7	unknown=40,not=31,cpp=26,auto=5,exp=5,ansic=4,sh=1, dup=1
		[GPL]
113	zsh-3.0.8	ansic=43,not=25,unknown=23,sh=14,dup=2,awk=2,perl=2, sed=1,auto=1
		[GPL]
112	urw-fonts-2.0	not=70,unknown=42
		[GPL,]
112	readline-4.1	dup=64,unknown=27,not=18,ansic=2,auto=1
		[GPL]
111	mpg123-0.59r	ansic=68,unknown=26,not=11,asm=3,makefile=2,dup=1
		[Distributable]
111	magicdev-0.3.5	not=40,unknown=36,dup=22,ansic=11,auto=2
		[GPL]
110	gqview-0.8.1	not=43,ansic=24,dup=22,unknown=19,auto=2
		[GPL]
110	libmng-1.0.0	ansic=44,not=29,unknown=22,dup=10,auto=2,sh=2,makefile=1
		[BSD-like]
110	pwdb-0.61.1	ansic=68,not=13,dup=10,unknown=10,makefile=7,sh=2
		[GPL or BSD]
110	gzip-1.3	not=37,ansic=33,unknown=28,dup=6,sh=3,asm=1,perl=1,

		auto=1
		[GPL]
110	joe	ansic=85,unknown=19,not=4,asm=1,makefile=1
		[GPL]
109	dhcp-2.0p15	ansic=66,unknown=22,not=14,sh=7
		[Distributable]
108	skkinput-2.03	ansic=87,unknown=16,not=3,makefile=2
		[GPL]
107	nut-0.44.1	ansic=48,not=34,unknown=16,sh=6,dup=1,auto=1,makefile=1
		[GPL]
106	zlib-1.1.3	dup=47,unknown=47,not=8,makefile=1,sh=1,auto=1,ansic=1
		[BSD]
106	xbill-2.0	not=65,cpp=18,dup=15,unknown=8
		[MIT]
105	aumix-2.7	not=39,unknown=27,dup=21,ansic=14,sh=2,auto=1,zero=1
		[GPL]
105	kudzu-0.98.10	not=38,ansic=36,dup=13,unknown=12,python=2,makefile=2,perl=1,sh=1
		[GPL]
105	jikes-1.13	cpp=75,not=17,unknown=7,sh=4,auto=1,dup=1
		[IBM Public License]
104	up2date-2.5.2	unknown=70,python=14,not=13,makefile=3,dup=2,sh=1,ansic=1
		[GPL]
103	nss_db-2.2	dup=45,unknown=21,ansic=21,not=14,auto=2
		[GPL]
100	docbook-utils-0.6	not=54,sh=22,unknown=18,perl=3,dup=2,auto=1
		[GPL]
99	imlib-1.9.8.1	ansic=35,not=34,unknown=15,dup=12,sh=2,auto=1
		[LGPL]
99	sudo-1.6.3p6	ansic=55,unknown=25,not=13,sh=3,auto=2,dup=1
		[BSD]
98	yp-tools-2.4	not=39,ansic=21,dup=20,unknown=15,auto=2,sh=1
		[GPL]
98	lilo-21.4.4	unknown=31,ansic=26,not=18,asm=14,sh=3,perl=2,makefile=2,dup=1,cpp=1
		[MIT]
97	dialog-0.9a-20001217	sh=33,unknown=22,ansic=20,not=17,perl=3,auto=1,dup=1
		[GPL]
97	p2c-1.22	ansic=39,unknown=38,not=11,pascal=5,makefile=3,perl=1
		[Distributable]
96	bc-1.06	unknown=36,ansic=32,not=20,sh=4,yacc=2,lex=1,auto=1
		[GPL]
96	im-140	unknown=38,not=27,perl=26,auto=2,dup=2,sh=1
		[Distributable]
95	SDL_mixer-1.1.0	ansic=62,not=15,unknown=9,dup=6,sh=2,auto=1
		[LGPL]
93	kdoc-2.1.1	not=43,perl=24,unknown=18,sh=3,cpp=2,makefile=1,auto=1,dup=1
		[GPL]
93	rsync-2.4.6	ansic=55,not=13,dup=10,unknown=8,auto=2,sh=2,awk=1,perl=1,makefile=1
		[GPL]
92	dip-3.3.7o	unknown=48,ansic=24,not=13,sh=3,asm=2,makefile=2
		[GPL]
92	procps-2.0.7	ansic=47,not=25,unknown=16,makefile=3,sh=1
		[GPL]
92	flex-2.5.4	unknown=37,ansic=30,not=11,lex=4,dup=2,makefile=2,sh=2,sed=1,auto=1,awk=1,yacc=1
		[GPL]
91	procmail-3.14	ansic=50,unknown=19,not=11,sh=8,makefile=3
		[Distributable]
91	unixfonts	unknown=49,not=41,sh=1
		[GPL]
91	gpm-1.19.3	unknown=35,ansic=25,not=19,sh=4,awk=2,lisp=2,auto=1,dup=1,sed=1,yacc=1
		[GPL]
89	pmake-1.45	ansic=55,unknown=26,makefile=5,not=2,sh=1
		[BSD]
89	xdaliclock-2.18	not=58,ansic=16,dup=6,unknown=6,auto=2,makefile=1
		[MIT]

89	kakasi-2.3.2	ansic=25,dup=22,unknown=21,not=20,auto=1 [GPL]
89	xbl-1.0j	ansic=49,unknown=28,not=11,auto=1 [GPL]
88	smpeg-0.4.2	cpp=43,not=18,unknown=8,ansic=8,dup=6,sh=2,asm=2, auto=1 [LGPL]
88	pxe-linux	ansic=41,cpp=19,unknown=17,makefile=9,asm=2 [BSD]
88	Distutils-1.0.1	python=62,not=10,unknown=10,ansic=4,sh=1,zero=1 [Python]
86	isapnptools-1.22	unknown=45,ansic=26,not=10,perl=2,makefile=1,sh=1, yacc=1 [GPL]
85	fortune-mod-9708	unknown=66,not=7,makefile=6,ansic=6 [BSD]
84	cproto-4.6	unknown=35,ansic=23,not=15,sh=6,makefile=2,auto=1, lex=1,yacc=1 [Public domain]
84	ypbind-mt-1.7	not=27,dup=23,unknown=15,ansic=15,auto=2,sh=2 [GPL]
84	pidentd-3.0.12	ansic=57,unknown=12,not=10,dup=3,auto=1,sh=1 [Public domain]
83	taper-6.9b	ansic=50,unknown=15,not=14,makefile=3,dup=1 [GPL]
82	XFree86-jpfonts-2.0	not=39,unknown=20,dup=13,awk=5,auto=2,makefile=1, sh=1,perl=1 [Distributable]
82	irda-utils-0.9.13	ansic=31,not=19,unknown=13,makefile=8,sh=5,dup=4, perl=1,auto=1 [GPL]
81	at-3.1.8	unknown=40,ansic=14,not=10,sh=7,auto=5,perl=2,lex=1, dup=1,yacc=1 [GPL]
80	reiserfsprogs-3.x.0f	ansic=44,not=24,unknown=8,dup=2,auto=1,sh=1 [GPL]
79	libunicode-0.4	ansic=42,not=19,unknown=8,dup=6,sh=2,cpp=1,auto=1 [LGPL]
77	lsk	ansic=50,sh=9,unknown=8,makefile=7,not=3 [Free]
76	pam_krb5-1.31-1	unknown=55,not=11,ansic=6,lex=1,dup=1,auto=1,yacc=1 [LGPL]
75	cpio-2.4.2	ansic=44,unknown=18,not=9,makefile=1,sh=1,auto=1, dup=1 [GPL]
74	vixie-cron-3.0.1	unknown=49,ansic=17,not=6,makefile=1,sh=1 [Distributable]
74	gdbm-1.8.0	ansic=47,unknown=10,not=8,sh=4,dup=3,cpp=1,auto=1 [GPL]
74	patch-2.5.4	ansic=26,dup=19,not=17,unknown=9,sed=2,auto=1 [GPL]
74	ltrace-0.3.10	ansic=39,unknown=11,not=10,dup=4,makefile=4,sh=3, awk=2,auto=1 [GPL]
73	xdelta-1.1.1	ansic=26,unknown=17,not=16,dup=9,sh=3,auto=1,lisp=1 [GPL]
73	apmd	not=25,unknown=23,sh=11,ansic=11,makefile=3 [GPL]
72	rcc-5.7	ansic=29,not=23,unknown=14,sh=3,dup=2,auto=1 [GPL]
72	dmalloc-4.8.1	ansic=36,unknown=19,not=11,perl=2,dup=1,sh=1,cpp=1, auto=1 [Public domain]
72	xmorph-2000apr28	ansic=41,unknown=16,tcl=10,not=4,makefile=1 [GPL]
72	autorun-2.65	unknown=31,not=26,dup=5,cpp=4,sh=4,auto=1,makefile=1 [GPL]
72	ctags-4.0.3	ansic=44,unknown=20,not=6,auto=1,sh=1 [GPL]
71	libole2-0.1.7	not=44,ansic=9,unknown=8,dup=7,sh=2,auto=1 [GPL]

```
70      netkit-rsh-0.17-pre20000412 unknown=24,not=20,ansic=17,makefile=8,  
      auto=1  
      [BSD]  
70      awesfx-0.4.3a      ansic=43,unknown=23,not=3,makefile=1  
      [GPL/distributable]  
69      Msql-Mysql-modules-1.2215 perl=33,unknown=19,not=12,ansic=5  
      [Distributable]  
69      xcdroast-0.98alpha8 not=37,ansic=19,unknown=11,makefile=1,sh=1  
      [GPL]  
69      specspro-7.1      not=58,unknown=8,makefile=3  
      [GPL]  
69      libelf-0.6.4      ansic=48,not=9,unknown=7,dup=2,sh=2,auto=1  
      [Distributable]  
68      SGMLSpM          not=46,unknown=10,perl=9,makefile=2,lisp=1  
      [GPL]  
68      gnome-kerberos-0.2.2 dup=23,not=20,ansic=12,unknown=11,auto=2  
      [GPL]  
68      tcp_wrappers_7.6 ansic=41,unknown=17,not=8,makefile=2  
      [Distributable]  
68      apel-10.2        lisp=58,unknown=8,not=1,makefile=1  
68      gq-0.4.0         ansic=36,not=21,unknown=7,dup=2,auto=1,sh=1  
      [GPL]  
67      raidtools       unknown=27,not=19,ansic=18,auto=2,sh=1  
      [GPL]  
67      autofs-3.1.7     ansic=23,unknown=20,not=16,makefile=5,sh=2,auto=1  
      [GPL]  
66      psiconv         ansic=34,not=14,dup=9,unknown=7,auto=1,sh=1  
      [GPL]  
66      tamago-4.0.6     lisp=33,unknown=25,not=6,auto=1,dup=1  
      [GPL]  
65      quota-3.00      unknown=29,ansic=23,not=12,makefile=1  
      [BSD]  
64      dump-0.4b21     ansic=30,not=17,unknown=12,sh=2,dup=1,auto=1,sed=1  
      [BSD]  
64      psutils         not=23,ansic=14,perl=13,unknown=12,sh=2  
      [Distributable]  
64      gsm-1.0-pl10    ansic=41,unknown=14,not=8,makefile=1  
      [BSD]  
63      ghostscript-fonts-5.50 unknown=55,not=8  
      [GPL]  
62      indent-2.2.6    unknown=23,ansic=21,not=13,dup=3,auto=1,sh=1  
      [GPL]  
62      mtr-0.42        ansic=21,unknown=14,not=13,dup=10,auto=4  
      [GPL]  
61      usermode-1.42   not=38,ansic=14,unknown=6,makefile=2,sh=1  
      [GPL]  
61      acct-6.3.2      ansic=28,not=17,unknown=10,sh=3,auto=1,cpp=1,dup=1  
      [GPL]  
61      Xconfigurator-4.9.27 not=38,ansic=10,unknown=8,makefile=3,perl=2  
      [GPL]  
61      expat           ansic=32,not=11,unknown=7,dup=6,sh=3,cpp=1,auto=1  
      [GPL]  
61      gd-1.8.3        ansic=44,not=9,unknown=5,perl=1,makefile=1,sh=1  
      [BSD-like]  
60      expat-1.95.1    dup=37,not=10,unknown=7,ansic=4,auto=1,sh=1  
      [MIT]  
59      sliplogin-2.1.1 unknown=35,ansic=13,not=6,makefile=2,sh=2,perl=1  
      [BSD]  
59      src_contrib     ansic=48,unknown=10,not=1  
59      cracklib,2.7    unknown=41,ansic=11,makefile=3,not=2,perl=1,sh=1  
      [Artistic]  
59      bzip2-1.0.1     not=20,unknown=20,ansic=11,dup=7,auto=1  
      [BSD]  
58      xjewel-1.6      not=30,ansic=21,unknown=7  
      [MIT]  
56      rpm2html-1.5    ansic=22,unknown=18,not=11,dup=3,auto=1,perl=1  
      [BSD-like]  
56      dhcpcd-1.3.18-pl8 ansic=16,unknown=15,not=11,sh=7,dup=5,auto=1,makefile=1  
      [GPL]  
55      kterm-6.2.0     ansic=30,unknown=23,not=2  
      [Distributable]  
55      traceroute-1.4a5 unknown=23,dup=10,ansic=10,not=6,auto=4,awk=2
```

		[BSD]
54	rep-gtk-0.15	unknown=17,not=11,ansic=11,sh=8,dup=4,lisp=2,auto=1
		[GPL]
53	which-2.12	not=20,unknown=11,ansic=11,dup=8,auto=2,sh=1
		[GPL]
53	kpppload-1.04	not=23,cpp=11,unknown=11,dup=6,auto=1,sh=1
		[GPL]
52	diffutils-2.7	ansic=29,unknown=12,not=7,dup=3,auto=1
		[GPL]
52	playmidi-2.4	unknown=29,ansic=16,not=3,sed=1,makefile=1,auto=1,
		dup=1
		[GPL]
52	jadetex-3.3	unknown=25,not=24,makefile=3
		[Distributable]
51	sysstat-3.3.5	not=17,unknown=16,ansic=10,sh=6,makefile=1,tcl=1
		[GPL]
51	gtk-doc-0.4b1	not=26,unknown=11,perl=6,dup=3,sh=3,auto=2
		[LGPL]
51	rp-pppoe-2.6	not=14,sh=12,ansic=12,unknown=11,auto=1,dup=1
		[GPL]
50	mpage-2.5.1	unknown=25,ansic=12,not=10,sh=2,makefile=1
		[BSD]
50	autoconf-2.13	unknown=20,not=14,sh=8,exp=4,perl=1,auto=1,dup=1,
		ansic=1
		[GPL]
49	mod_dav-1.0.2-1.3.6	ansic=18,unknown=12,not=9,dup=6,python=2,auto=1,
		makefile=1
		[Apache-like]
49	pax-1.5	ansic=32,unknown=9,not=6,makefile=2
		[BSD]
49	pciutils-2.1.8	unknown=22,ansic=18,not=6,makefile=2,sh=1
		[GPL]
49	tux-2.0.26	not=16,unknown=14,ansic=10,sh=7,makefile=2
		[GPL]
48	netkit-tftp-0.17	ansic=16,not=15,makefile=7,unknown=7,auto=2,dup=1
		[BSD]
48	cdparanoia-III-alpha9.7	ansic=32,not=8,unknown=4,sh=2,auto=1,zero=1
		[GPL]
48	cipe-1.4.5	ansic=20,unknown=12,not=10,asm=2,sh=2,perl=1,auto=1
		[GPL]
45	mouseconfig-4.21	not=34,makefile=4,unknown=4,ansic=3
		[Distributable]
45	stunnel-3.13	unknown=17,not=13,ansic=8,dup=4,sh=1,auto=1,perl=1
		[GPL]
45	authconfig-4.1.6	not=32,ansic=6,unknown=4,makefile=3
		[GPL]
45	sndconfig-0.64.8	not=33,unknown=7,ansic=3,makefile=2
		[GPL]
44	libghttp-1.0.8	ansic=18,not=11,dup=7,unknown=7,auto=1
		[LGPL]
44	timeconfig-3.2	not=35,unknown=4,makefile=2,dup=1,sh=1,ansic=1
		[GPL]
43	dosfstools-2.2	ansic=18,not=12,unknown=10,makefile=3
		[GPL]
43	chkconfig-1.2.22	not=33,unknown=4,ansic=4,makefile=2
		[GPL]
43	ksconfig-1.2	not=29,python=6,unknown=4,makefile=2,dup=1,sh=1
		[GPL]
43	mtx-1.2.10	ansic=16,unknown=13,not=9,perl=2,sh=1,python=1,makefile=1
		[GPL]
43	kbdconfig-1.9.12	not=35,unknown=4,makefile=3,ansic=1
		[GPL]
42	newt-0.50.22	ansic=27,not=7,python=3,unknown=3,auto=1,dup=1
		[LGPL]
42	slocate-2.5	not=15,unknown=11,ansic=8,dup=4,sh=3,auto=1
		[GPL]
42	control-panel-3.18	not=33,unknown=4,makefile=2,tcl=2,ansic=1
		[GPL]
42	alchemist-0.16	ansic=13,not=11,unknown=6,python=4,sh=4,dup=3,auto=1
		[GPL]
42	netkit-rusers-0.17	not=18,ansic=11,unknown=7,makefile=5,auto=1

```

    [BSD]
41    rhn_register-1.3.1 not=13,unknown=12,python=11,makefile=3,sh=2
    [GPL]
41    yacc          ansic=13,unknown=11,not=9,auto=4,yacc=2,sh=1,makefile=1
    [Public domain]
40    netkit-ntalk-0.17-pre20000412 ansic=21,not=9,unknown=6,makefile=3,
    auto=1
    [BSD]
39    xcpustate-2.5  ansic=28,unknown=9,not=2
    [Freely redistributable]
39    joystick-1.2.15 ansic=24,not=9,unknown=4,makefile=1,sh=1
    [GPL]
39    libPropList-0.10.1 ansic=15,not=8,dup=7,unknown=6,lex=1,auto=1,yacc=1
    [LGPL]
39    sysklogd-1.4rh unknown=16,ansic=14,not=7,makefile=1,sh=1
    [GPL]
39    DBI-1.14       perl=21,unknown=12,ansic=4,not=2
    [Distributable]
39    rpmfind-1.6    ansic=22,not=8,unknown=5,dup=3,auto=1
    [BSD-like]
38    sgml-common-0.5 unknown=26,not=7,sh=4,makefile=1
    [GPL]
37    clime-1.13.6   lisp=21,unknown=12,not=3,makefile=1
36    iputils        ansic=18,unknown=11,not=5,dup=1,makefile=1
    [BSD]
36    nc             sh=12,unknown=12,not=5,ansic=5,makefile=2
    [GPL]
36    netkit-routed-0.17 ansic=18,not=9,unknown=4,makefile=3,auto=2
    [BSD]
35    SDL_image-1.1.0 ansic=13,not=9,dup=5,unknown=4,sh=3,auto=1
    [LGPL]
35    docbook-dtd41-xml-1.0 unknown=30,not=4,makefile=1
    [Distributable]
35    rpmlint-0.28    python=21,unknown=9,not=3,makefile=1,sh=1
    [GPL]
35    ytalk-3.1.1     ansic=16,not=8,unknown=8,dup=2,auto=1
    [BSD]
34    ext2ed-0.1      ansic=14,unknown=11,not=8,makefile=1
    [GPL]
34    cdp-0.33        ansic=20,unknown=8,not=4,makefile=1,zero=1
    [GPL]
34    syslinux-1.52   unknown=15,not=7,asm=5,perl=5,makefile=1,ansic=1
    [BSD]
33    usbview-1.0     not=11,ansic=10,unknown=7,dup=4,auto=1
    [GPL]
33    netscape-4.76   unknown=31,sh=2
    [Proprietary]
33    netkit-rwho-0.17 not=11,unknown=10,ansic=6,makefile=4,auto=1,dup=1
    [BSD]
33    MAKEDEV-3.1.0   unknown=25,sh=3,not=2,ansic=2,makefile=1
    [GPL]
32    gnome-audio-1.0.0 unknown=28,not=2,makefile=2
    [LGPL]
32    redhat-logos    not=28,unknown=4
31    bsd-finger-0.17-pre20000412 ansic=10,not=9,unknown=8,makefile=3,
    auto=1
    [BSD]
31    time-1.7        ansic=11,unknown=9,not=7,dup=3,auto=1
    [GPL]
31    python-xmlrpc-1.4 unknown=9,python=8,perl=7,not=4,makefile=2,ansic=1
    [GPL]
30    freedb-0.62     ansic=13,unknown=12,makefile=2,sh=2,not=1
    [Distributable]
30    semi-1.13.7     lisp=14,unknown=11,not=4,makefile=1
30    urlview-0.9     not=9,unknown=7,ansic=7,sh=4,dup=2,auto=1
    [GPL]
29    xgammon-0.98    ansic=16,unknown=8,not=4,lex=1
    [GPL]
28    psgml-1.2.1     lisp=12,unknown=9,not=3,auto=2,dup=2
    [GPL]
28    units-1.55      not=9,unknown=9,dup=4,ansic=4,auto=1,perl=1
    [GPL]
```


28	ipchains-1.3.10	not=12,unknown=5,sh=4,ansic=4,makefile=3 [GPL]
28	netkit-ftp-0.17	ansic=12,not=8,unknown=5,makefile=2,auto=1 [BSD]
28	asp2php-0.75.11	ansic=19,not=4,unknown=4,makefile=1 [GPL]
27	Perl-RPM-0.291	perl=13,unknown=11,not=2,ansic=1 [Distributable]
27	sash-3.4	ansic=14,unknown=10,not=2,makefile=1 [GPL]
27	xmailbox-2.5	not=13,unknown=9,ansic=5 [MIT]
26	termcap-2.0.8	unknown=18,ansic=4,not=3,makefile=1 [LGPL]
26	ElectricFence-2.2.2	unknown=12,ansic=6,not=4,makefile=3,sh=1 [GPL]
26	lockdev-1.0.0	unknown=13,ansic=4,sh=3,not=2,perl=2,makefile=2 [LGPL]
26	smpeg-xmms-0.3.3	not=10,dup=6,unknown=5,ansic=3,auto=1,sh=1 [GPL]
25	efax-0.9	ansic=10,unknown=7,not=5,sh=2,makefile=1 [GPL]
24	logrotate-3.5.4	not=9,ansic=7,unknown=4,sh=3,makefile=1 [GPL]
24	netkit-rwall-0.17	not=10,unknown=5,ansic=4,makefile=3,auto=1,dup=1 [BSD]
24	procinfo-17	unknown=11,not=6,perl=3,ansic=3,makefile=1 [GPL]
24	oo	unknown=24 [GPL]
22	setup-2.4.7	unknown=21,makefile=1 [Public domain]
22	indexhtml-7.1	not=18,unknown=3,makefile=1 [Distributable]
22	psmisc	not=7,unknown=7,ansic=6,makefile=1,sh=1 [Distributable]
22	knm_new-1.1	not=15,unknown=7 [GPL]
22	fbset-2.1	unknown=8,not=5,ansic=4,makefile=2,lex=1,perl=1, yacc=1 [GPL]
20	ksymoops-2.4.0	ansic=10,unknown=5,not=4,makefile=1 [GPL]
20	pythonlib-1.28	python=14,unknown=4,not=1,makefile=1 [GPL]
20	setserial-2.17	unknown=7,not=6,ansic=3,sh=2,auto=1,dup=1 [GPL]
19	skkdic-20010122	unknown=14,not=4,makefile=1 [GPL]
19	anacron-2.3	ansic=10,not=6,unknown=2,makefile=1 [GPL]
19	unarj-2.43	unknown=8,not=6,ansic=4,makefile=1 [Distributable]
18	docbook-dtd41-sgml-1.0	unknown=12,not=5,makefile=1 [Distributable]
18	netkit-bootparamd-0.17-pre20000412	not=7,unknown=4,ansic=4,makefile=2, auto=1 [BSD]
18	ncompress-4.2.4	unknown=10,not=5,ansic=2,makefile=1 [Distributable]
18	xtt-fonts-0.19990222	unknown=16,makefile=1,sh=1 [Distributable]
18	adjtimex-1.11	not=7,unknown=6,ansic=3,auto=1,sh=1 [Distributable]
18	biff+comsat-0.17-pre20000412	not=8,makefile=3,unknown=3,ansic=3, auto=1 [BSD]
18	docbook-dtd31-sgml-1.0	unknown=12,not=5,makefile=1 [Distributable]
17	vlock-1.3	ansic=7,unknown=6,not=3,makefile=1 [GPL]

17	auth_ldap-1.4.7	unknown=8,not=4,ansic=4,makefile=1 [GPL]
17	mt-st-0.5b	unknown=10,not=4,ansic=2,makefile=1 [BSD]
17	portmap_4	ansic=7,unknown=6,not=2,dup=1,makefile=1 [BSD]
17	perl-NKF-1.71	unknown=7,not=4,perl=3,makefile=1,sh=1,ansic=1 [BSD]
17	passwd-0.64.1	ansic=8,not=4,unknown=4,makefile=1 [BSD]
16	docbook-dtd40-sgml-1.0	unknown=12,not=3,makefile=1 [Distributable]
16	wireless_tools.20	ansic=7,not=4,unknown=4,makefile=1 [GPL]
16	hotplug-2001_02_14	sh=8,unknown=6,not=2 [GPL]
16	xsyinfo-1.7	unknown=7,ansic=6,not=2,makefile=1 [MIT]
16	cleanfeed-0.95.7b	unknown=10,not=4,perl=2 [Distributable]
16	open-1.4	unknown=7,not=4,ansic=3,dup=1,makefile=1 [GPL]
15	cxhextris	unknown=7,ansic=4,not=3,makefile=1 [Distributable]
15	pump-0.8.11	ansic=5,unknown=4,not=3,dup=2,makefile=1 [MIT]
15	tksysv-1.3	not=7,unknown=5,sh=2,tcl=1 [GPL]
15	mingetty-0.9.4	unknown=10,not=3,makefile=1,ansic=1 [GPL]
15	trojka	ansic=9,unknown=3,not=2,makefile=1 [Distributable]
15	eject-2.0.2	unknown=8,not=5,makefile=1,ansic=1 [GPL]
15	gkermi-1.0	unknown=6,ansic=5,not=2,auto=1,makefile=1 [GPL]
15	xinitrc-3.6	sh=8,unknown=6,makefile=1 [Public domain]
15	DBD-Pg-0.95	unknown=7,perl=3,ansic=3,not=2 [Distributable]
14	bdf flush-1.5	unknown=6,not=3,asm=2,makefile=2,ansic=1 [Distributable]
14	docbook-dtd30-sgml-1.0	unknown=10,not=3,makefile=1 [Distributable]
14	diffstat-1.27	not=6,unknown=3,dup=2,ansic=2,auto=1 [Distributable]
14	setuptool-1.7	not=7,unknown=4,makefile=2,ansic=1 [GPL]
13	hdparm-3.9	unknown=6,not=3,sh=2,makefile=1,ansic=1 [Distributable]
13	whois-1.0.6	not=5,unknown=4,auto=1,dup=1,sh=1,ansic=1 [LGPL]
13	Kappa20-0.3	not=9,unknown=3,makefile=1 [Public domain]
13	mkinitrd-3.0.10	unknown=4,not=3,ansic=3,makefile=2,sh=1 [GPL]
13	manpages-da-0.1.1	not=8,unknown=4,makefile=1 [Distributable]
13	cdecl-2.5	unknown=6,not=3,lex=1,makefile=1,ansic=1,yacc=1 [Distributable]
12	isicom	unknown=7,ansic=3,makefile=1,sh=1 [GPL (not Firmware)]
12	emh-1.10.1	unknown=6,lisp=4,not=1,makefile=1
11	dbskkd-cdb-1.01	unknown=5,ansic=3,sh=2,makefile=1 [GPL]
11	Text-Kakasi-1.04	unknown=6,not=3,perl=2 [GPL]
11	aspell-da-1.4.9	unknown=7,not=2,perl=1,makefile=1 [GPL]
11	stat-2.2	unknown=5,not=3,ansic=2,makefile=1 [GPL]
11	dos2unix-3.1	unknown=7,ansic=2,not=1,makefile=1

		[Freely distributable]
11	netcfg-2.36	unknown=6,not=2,python=1,makefile=1,sh=1 [GPL]
11	statserial-1.1	not=4,unknown=4,makefile=1,sh=1,ansic=1 [BSD]
10	File-MMagic-1.06	unknown=7,not=2,perl=1 [Distributable]
10	br.ispell-2.4	unknown=5,not=3,awk=1,makefile=1 [GPL]
10	genromfs-0.3	not=4,unknown=4,makefile=1,ansic=1 [GPL]
10	français	unknown=7,not=2,makefile=1 [GPL]
9	sysreport-1.2	not=3,sh=3,unknown=3 [GPL]
9	utempter-0.5.2	ansic=4,unknown=3,not=1,makefile=1 [MIT]
9	modemtool-1.22	unknown=4,not=2,python=1,makefile=1,sh=1 [GPL]
9	mktemp-1.5	unknown=6,not=1,makefile=1,ansic=1 [BSD]
9	tree-1.2	unknown=4,not=3,makefile=1,ansic=1 [GPL]
9	tmpwatch-2.7.1	unknown=4,not=2,makefile=1,sh=1,ansic=1 [GPL]
9	xtoolwait-1.2	unknown=5,not=3,ansic=1 [GPL]
9	symlinks-1.2	unknown=6,not=1,makefile=1,ansic=1 [Distributable]
9	rmail-mime-1.13.0	unknown=6,not=1,lisp=1,makefile=1
8	mkbootdisk-1.4.2	unknown=3,not=2,sh=2,makefile=1 [GPL]
8	rootfiles	unknown=8 [Public domain]
8	rarpd	not=2,sh=2,unknown=2,makefile=1,ansic=1 [GPL]
8	abidistfiles	unknown=5,not=3 [GPL]
8	chkfontpath-1.9.5	unknown=3,not=2,makefile=2,ansic=1 [GPL]
7	words-2	unknown=5,sh=2 [Free]
7	aspell-swedish-0.2	unknown=5,not=1,sh=1 [GPL]
7	mailcap-2.1.4	unknown=5,not=1,makefile=1 [Public domain]
7	kcc	unknown=3,not=2,makefile=1,ansic=1 [GPL]
7	timetool-2.8	unknown=4,not=1,makefile=1,tcl=1 [GPL]
7	locale_config-0.2	unknown=3,ansic=3,makefile=1 [GPL]
6	xsri-1.0	unknown=3,not=1,makefile=1,ansic=1 [GPL]
6	watanabe-vf	unknown=5,not=1 [Distributable]
6	mkkickstart-2.3	unknown=3,not=1,makefile=1,sh=1 [GPL]
6	rhmask	unknown=3,not=1,makefile=1,ansic=1 [GPL]
6	unix2dos-2.2	unknown=3,ansic=2,not=1 [Distributable]
6	rdate-1.0	not=2,unknown=2,makefile=1,ansic=1 [GPL]
6	aspell-no-0.2	unknown=5,not=1 [GPL]
5	aspell-de-0.1.1	unknown=4,not=1 [GPL]
5	shaper	unknown=3,makefile=1,ansic=1 [GPL]
5	aspell-es-0.2	unknown=4,not=1

		[GPL]
5	desktop-backgrounds-1.1	unknown=4,not=1
		[LGPL]
5	aspell-ca-0.1	unknown=5
		[GPL]
4	aspell-it-0.1	unknown=4
		[GPL]
4	aspell-nl-0.1	unknown=4
		[GPL]
4	giftrans-1.12.2	unknown=2,not=1,ansic=1
		[BSD]
3	pvm	unknown=3
		[Freely distributable]
3	jisksp16-1990	unknown=2,not=1
		[Public domain]
3	jisksp14	unknown=2,not=1
		[Public domain]
2	caching-nameserver-7.1	unknown=2
		[Public domain]
2	nkf-1.92	unknown=2
		[BSD-like]
2	src_top_dir	unknown=2
1	lost+found	unknown=1

not:	134339	(38.11%)
ansic:	78676	(22.32%)
unknown:	75222	(21.34%)
cpp:	26045	(7.39%)
dup:	10807	(3.07%)
sh:	4577	(1.30%)
fortran:	3046	(0.86%)
perl:	3012	(0.85%)
asm:	2959	(0.84%)
makefile:	2954	(0.84%)
python:	2237	(0.63%)
java:	1990	(0.56%)
lisp:	1720	(0.49%)
auto:	1587	(0.45%)
tcl:	1265	(0.36%)
exp:	632	(0.18%)
awk:	336	(0.10%)
objc:	276	(0.08%)
sql:	152	(0.04%)
ada:	142	(0.04%)
yacc:	138	(0.04%)
sed:	129	(0.04%)
zero:	106	(0.03%)
csh:	101	(0.03%)
lex:	94	(0.03%)
pascal:	7	(0.00%)

ansic:	78676	(60.29%)
cpp:	26045	(19.96%)
sh:	4577	(3.51%)
fortran:	3046	(2.33%)
perl:	3012	(2.31%)
asm:	2959	(2.27%)
python:	2237	(1.71%)
java:	1990	(1.53%)
lisp:	1720	(1.32%)
tcl:	1265	(0.97%)
exp:	632	(0.48%)
awk:	336	(0.26%)
objc:	276	(0.21%)
ada:	142	(0.11%)
yacc:	138	(0.11%)
sed:	129	(0.10%)
csh:	101	(0.08%)
lex:	94	(0.07%)
pascal:	7	(0.01%)

Licenses:

184442	(52.32%)	GPL
32508	(9.22%)	Distributable
29529	(8.38%)	LGPL
28588	(8.11%)	MPL
20820	(5.91%)	MIT
11633	(3.30%)	BSD
9959	(2.82%)	BSD-like
6434	(1.82%)	Freely distributable
5471	(1.55%)	Free
3228	(0.92%)	PHP
3141	(0.89%)	GPL, LGPL
2390	(0.68%)	LGPL/GPL
1895	(0.54%)	GPL/MIT
1668	(0.47%)	Artistic or GPL
1617	(0.46%)	OpenLDAP
1588	(0.45%)	Apache-like
1160	(0.33%)	Apache
1108	(0.31%)	W3C
626	(0.18%)	GPL/BSD
576	(0.16%)	GPL or BSD
543	(0.15%)	University of Washington's Free-Fork License
479	(0.14%)	Public domain
473	(0.13%)	GPL/LGPL
396	(0.11%)	GPL (programs), relaxed LGPL (libraries), and public domain (docs)
337	(0.10%)	GPL and Artistic
299	(0.08%)	Distributable - most of it GPL
266	(0.08%)	MIT-like
112	(0.03%)	GPL,
105	(0.03%)	IBM Public License
88	(0.02%)	Python
70	(0.02%)	GPL/distributable
61	(0.02%)	Not listed
59	(0.02%)	Artistic
39	(0.01%)	Freely redistributable
33	(0.01%)	Proprietary
12	(0.00%)	GPL (not Firmware)

Percentage of Licenses containing selected key phrases:

196437	(55.72%)	GPL
39311	(11.15%)	distributable
35929	(10.19%)	LGPL
28588	(8.11%)	MPL
22981	(6.52%)	MIT
22794	(6.47%)	BSD
875	(0.25%)	public domain

Total Number of Files = 352549

Total Number of Source Code Files = 130488

Please credit this data as "generated using 'SLOCCount' by David A. Wheeler."

More Than a Gigabuck: Estimating GNU/Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

June 20, 2001

Version 1.0

This paper analyzes the amount of source code in GNU/Linux, using Red Hat Linux 7.1 as a representative GNU/Linux distribution, and presents what I believe are interesting results.

In particular, it would cost over \$1 billion (\$1,000 million - a Gigabuck) to develop this Linux distribution by conventional proprietary means in the U.S. (in year 2000 U.S. dollars). Compare this to the \$600 million estimate for Red Hat Linux version 6.2 (which had been released about one year earlier). Also, Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2. Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2). Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is due to an increased number of mature and maturing open source / free software programs available worldwide.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), X-windows (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part or as an alternative, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is public domain.

This paper is an update of my previous paper on estimating GNU/Linux's size, which measured Red Hat Linux 6.2 [Wheeler 2001]. Since Red Hat Linux 6.2 was released in March 2000, and Red Hat Linux 7.1 was released in April 2001, this paper shows what's changed over approximately one year.

More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The GNU/Linux operating system (also called simply "Linux") has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a "typical" intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally "open source software" and/or "free software" (where the "free" here means "freedom"). A program that is "open source software" or "free software" is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of "open source software" is available at OSI [1999], a more formal definition of "free software" is available at FSF [2000], and other general information about these topics is available at Wheeler [2000a]. Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is "open source software"/"free

software", and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. The only published data I've found (other than my own) was developed by Microsoft in the documents usually called "Halloween I" and "Halloween II" [\[Halloween I\]](#) [\[Halloween II\]](#). In a previous paper, I examined Red Hat Linux 6.2 and the numbers from the Halloween papers.

This paper updates my previous paper, showing estimates of the size of one of today's GNU/Linux distributions, and it estimates how much it would cost to rebuild this typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean. I have intentionally written this paper so that you do *not* need to read the previous version of this paper first.

For my purposes, I have selected as my "representative" Linux distribution Red Hat Linux version 7.1. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [\[Shankland 2000b\]](#). Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE at 15%. Not all Linux copies are "sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size for the same kind of functionality.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate "all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 7.1 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Note that Red Hat Linux 6.2 was released on March 2000, Red Hat Linux 7 was released on September 2000 (I have not counted its code), and Red Hat Linux 7.1 was released on April 2001. Thus, the differences between Red Hat Linux 7.1 and 6.2 show differences accrued over 13 months (approximately one year).

Clearly there is far more open source / free software available worldwide than is counted in this paper. However, the job of a distributor is to examine these various options and select software that they believe is both sufficiently mature and useful to their target market. Thus, examining a particular distribution results in a selective analysis of such software.

Section 2 briefly describes the approach used to estimate the "size" of this distribution (more details are in Appendix A). Section 3 discusses some of the results. Section 4 presents conclusions, followed by an appendix.

2. Approach

My basic approach was to:

1. install the source code files in uncompressed format; this requires carefully selecting the source code to be analyzed.
2. count the number of source lines of code (SLOC); this requires a careful definition of SLOC.
3. use an estimation model to estimate the effort and cost of developing the same system in a proprietary manner; this requires an estimation model.
4. determine the software licenses of each component and develop statistics based on these categories.

More detail on this approach is described in Appendix A. A few summary points are worth mentioning here, however.

2.1 Selecting Source Code

I included all software provided in the Red Hat distribution, but note that Red Hat no longer includes software packages that only apply to other CPU architectures (and thus packages not applying to the x86 family were excluded). I did not include

``old" versions of software, or ``beta" software where non-beta was available. I did include ``beta" software where there was no alternative, because some developers don't remove the ``beta" label even when it's widely used and perceived to be reliable.

I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once (as a tie-breaker, such files are assigned to the first build package it applies to in alphabetic order).

The code in makefiles and RPM package specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

2.2 Defining SLOC

The ``physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: ``a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the ``logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the ``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

Note that this required that every file be categorized by language type (so that the correct syntax for comments, strings, and so on could be applied). Also, automatically generated files had to be detected and ignored. Thankfully, my tool ``sloccount" does this automatically.

2.3 Estimation Models

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer ``COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions. Of particular note, basic COCOMO does not include the time to develop translations to other human languages (of documentation, data, and program messages) nor fonts.

There is reason to believe that these models, while imperfect, are still valid for estimating effort in open source / free software projects. Although many open source programs don't need management of human resources, they still require technical management, infrastructure maintenance, and so on. Design documentation is captured less formally in open source projects, but it's often captured by necessity because open source projects tend to have many developers separated geographically. Clearly, the systems must still be programmed. Testing is still done, although as with many of today's proprietary programs, a good deal of testing is done through alpha and beta releases. In addition, quality is enhanced in many open source projects through peer review of submitted code. The estimates may be lower than the actual values because they don't include estimates of human language translations and fonts.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed

average value for overhead, also called the ``wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired. These are the same values as used in my last report.

2.4 Determining Software Licenses

A software license determines how that software can be used and reused, and open source software licensing has been a subject of great debate. Well-known open source licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD license, and the Artistic license. The GPL and LGPL are termed ``copylefting" licenses, that is, the license is designed to prevent the code from becoming proprietary. See [Perens \[1999\]](#) for more information. Obvious questions include ``what license(s) are developers choosing when they release their software" and ``how much code has been released under the various licenses?"

An approximation of the amount of software using various licenses can be found for this particular distribution. Red Hat Linux uses the Red Hat Package Manager (RPM), and RPM supports capturing license data for each package (these are the ``Copyright" and ``License" fields in the specification file). I used this information to determine how much code was covered by each license. Since this field is simply a string of text, there were some variances in the data that I had to clean up, for example, some entries said ``GNU" while most said ``GPL". In some cases Red Hat did not include licensing information with a package. In that case, I wrote a program to attempt to determine the license by looking for certain conventional filenames and contents.

This is an imperfect approach. Some packages contain different pieces of code with different licenses applying to different pieces. Some packages are ``dual licensed", that is, they are released under more than one license. Sometimes these other licenses are noted, while at other times they aren't. There are actually two BSD licenses (the ``old" and ``new" licenses), but the specification files don't distinguish between them. Also, if the license wasn't one of a small set of common licenses, Red Hat tended to assign nondescriptive phrases such as ``distributable". My automated techniques were limited too, in particular, while some licenses (e.g., the GPL and LGPL) are easy to recognize automatically, BSD-like and MIT-like licenses vary the license text and so are more difficult to recognize automatically (and some changes to the license would render them non-open source, non-free software). Thus, when Red Hat did not identify a package's license, a program dual licensed under both the BSD and GPL license might only be labelled as having the GPL using these techniques. Nevertheless, this approach is sufficient to give some insight into the amount of software using various licenses. Future research could examine each license in turn and categorize them; such research might require several lawyers to determine when two licenses in certain circumstances are ``equal."

One program worth mentioning in this context is Python, which has had several different licenses. Version 1.6 and later (through 2.1) had more complex licenses that the Free Software Foundation (FSF) believes were incompatible with the GPL. Recently this was resolved by another change to the Python license to make Python fully compatible with the GPL. Red Hat Linux 7.1 includes an older version of Python (1.5.2), presumably because of these licensing issues. It can't be because Red Hat is unaware of later versions of Python; Red Hat uses Python in its installation program (which it developed and maintains). Hopefully, the recent resolution of license incompatibilities with the GPL license will enable Red Hat to include the latest versions of Python in the future. In any case, there are several different Python-specific licenses, all of which can legitimately be called the ``Python" license. Red Hat has labelled Python itself as having a ``Distributable" license, and package Distutils-1.0.1 is labelled with the ``Python" license; these labels are kept in this paper.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (sorted by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total counts grouped by their software licenses, section 3.6 presents total SLOC counts, and section 3.7 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 35 largest components (as measured by number of source lines of code), along with their licenses (see section 2.4 for how these license values were determined). In the language section, ``ansic" means C code, ``asm" is assembly, ``sh" is Bourne shell and related shells, and ``cpp" is C++.

SLOC	Directory	SLOC-by-Language (Sorted)
2437470	kernel-2.4.2	ansic=2285657,asm=144411,sh=3035,perl=2022,yacc=1147, tcl=576,lex=302,awk=248,sed=72 [GPL]
2065224	mozilla	cpp=1279902,ansic=739470,perl=21220,sh=13717,asm=5212, java=3107,yacc=1831,lex=470,csh=271,sed=24 [MPL]
1837608	XFree86-4.0.3	ansic=1750460,asm=35397,cpp=20725,sh=14666,tcl=9182, yacc=3360,perl=1675,lex=1608,awk=393,csh=85,sed=57 [MIT]
984076	gcc-2.96-20000731	ansic=789901,cpp=126738,yacc=19272,sh=17993,asm=14559, lisp=7161,fortran=3814,exp=3705,objc=479,sed=310,perl=144 [GPL]
967263	gdb+dejagnu-20010316	ansic=871288,exp=58422,sh=12054,cpp=8252,yacc=5906, asm=5031,tcl=4477,lisp=1403,sed=248,awk=170,java=7,fortran=5 [GPL]
690983	binutils-2.10.91.0.2	ansic=489993,asm=161236,exp=13234,sh=12835, yacc=5665,cpp=4777,lex=1488,perl=776,sed=561,lisp=394,awk=24 [GPL]
646692	glibc-2.2.2	ansic=548722,asm=88413,sh=6036,perl=2120,awk=1037, yacc=315,sed=49 [LGPL]
627626	emacs-20.7	lisp=453898,ansic=169956,sh=2622,perl=884,asm=253, csh=9,sed=4 [GPL]
474829	LAPACK	fortran=473590,ansic=1239 [Freely distributable]
455980	gimp-1.2.1	ansic=427967,perl=17482,lisp=9648,yacc=502,sh=381 [GPL, LGPL]
402799	mysql-3.23.36	ansic=249350,cpp=84068,perl=25088,tcl=18980,sh=18323, asm=3987,awk=1436,java=1149,sed=418 [LGPL]
395194	tcltk-8.3.1	ansic=291457,tcl=84322,sh=12259,exp=5742,yacc=876, awk=273,perl=265 [BSD]
345949	kdebase-2.1.1	cpp=181210,ansic=158682,sh=4880,perl=1155,python=22 [GPL]
323730	Mesa-3.4	ansic=286437,cpp=18189,asm=10002,sh=7611,objc=1184, python=307 [GPL/MIT]
321123	perl-5.6.0	perl=146755,ansic=118233,sh=49377,lisp=5739,yacc=996, java=23 [Artistic or GPL]
318430	libgcj	ansic=191432,cpp=56843,java=41716,sh=15581,asm=11262, exp=841,perl=731,awk=24 [GPL]
304819	teTeX-1.0	ansic=223491,perl=49789,sh=17634,cpp=9407,pascal=1546, yacc=1507,awk=622,lex=323,sed=314,asm=139,csh=47 [Distributable]
298742	qt-2.3.0	cpp=259310,ansic=34578,yacc=2444,sh=1493,lex=480, perl=422,lisp=15 [GPL]
286113	postgresql-7.0.3	ansic=237184,java=17540,yacc=9740,sh=8975,tcl=7751,

		lex=1810,perl=1276,python=959,cpp=801,asm=70,csch=5,sed=2 [BSD]
283785	kdelibs-2.1.1	cpp=261334,ansic=17578,sh=1887,java=1538,perl=731, yacc=607,lex=110 [LGPL]
277502	xemacs-21.1.14	ansic=199927,lisp=73366,sh=2948,perl=930,asm=247, csch=62,sed=22 [GPL]
264528	gs5.50	ansic=259471,cpp=2266,asm=968,sh=823,lisp=405,perl=336, yacc=201,lex=58 [GPL]
227354	krb5-1.2.2	ansic=197886,exp=19124,sh=5140,yacc=2474,perl=1529, awk=393,python=348,lex=190,csch=147,sed=123 [MIT]
215473	vnc_unixsrc	ansic=212766,cpp=848,asm=780,perl=648,sh=431 [GPL]
213818	koffice-2.0.1	cpp=197637,sh=7296,yacc=3791,ansic=3213,perl=1801, lex=80 [GPL]
202842	openssl-0.9.6	ansic=131874,cpp=25744,perl=14737,asm=12428,python=10171, yacc=3297,sh=2641,tcl=1583,lisp=224,objc=143 [BSD-like]
200908	Python-1.5.2	python=101017,ansic=96521,lisp=2353,sh=673,perl=342, sed=2 [Distributable]
194799	bind-9.1.0	ansic=173830,sh=12101,yacc=6025,perl=2830,tcl=13 [BSD-like]
192394	xpdf-0.92	cpp=167135,ansic=21621,sh=3638 [GPL]
191379	php-4.0.4pl1	ansic=173334,cpp=7033,sh=6591,lex=1867,yacc=1569, java=437,awk=367,perl=181 [PHP]
190950	pine4.33	ansic=190020,sh=838,csch=62,perl=30 [Freely distributable]
173492	abi	cpp=159595,ansic=12605,perl=725,sh=550,python=17 [GPL]
167663	kdemultimedia-2.1.1	cpp=140731,ansic=23844,tcl=1004,sh=800,asm=598, lex=578,perl=106,awk=2 [GPL]
163449	4Suite-0.10.1	python=91445,ansic=72004 [Apache-like]
159301	linuxconf-1.24r2	cpp=142970,perl=6738,sh=3821,java=3074,ansic=2613, python=85 [GPL]

Note that the operating system kernel (Linux) is the largest single component, at over 2.4 million lines of code (mostly in C); that compares to 1.5 million lines of code in Red Hat 6.2. See section 3.2 for a more detailed discussion about the Linux kernel.

The next largest component is Mozilla; this is large because it's really a suite of applications including a web browser, email reader, news reader, HTML editor, and so on. Mozilla is the basis for Netscape Navigator 6.0. Mozilla was not included at all in Red Hat Linux 6.2.

The next largest component is X windows, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to gain functionality and size), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, the symbolic debugger, a set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). Emacs is next largest, which should not be a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system.

Note that language implementations tend to be written in themselves, particularly for their libraries. Perl's implementation is written mostly in Perl, and Python is written mostly in Python. Intriguingly, this is not true for Tcl.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the Linux kernel (at over 2.4 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 1,400,000 lines (57% of the Linux kernel) was in the `drivers` subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of peripherals. No other subdirectory comes close to this size - the second largest is the `arch` directory (at over 446,000 SLOC, 18% of the kernel), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is over 168,000 SLOC.

Richard Stallman and others have argued that the resulting system often called `Linux` should instead be called `GNU/Linux` [Stallman 2000]. In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to `free software` (free as in freedom). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term `GNU/Linux` is that it is confusing if both the entire operating system and the operating system kernel are both called `Linux`. Using the term `Linux` is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either `Linux` or `GNU/Linux`. It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (`Linux`). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc, gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system `GNU/Linux` and not just `Linux`. For more information on the sizes of the Linux kernel components, see http://www.dwheeler.com/sloc/redhat71-v1/kernel_sloc.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code (using the naming conventions of sloccount, the program used to count SLOC):

Language	SLOC (%)
C	21461450 (71.18%)
C++	4575907 (15.18%)
Shell (Bourne-like)	793238 (2.63%)
Lisp	722430 (2.40%)
Assembly	565536 (1.88%)
Perl	562900 (1.87%)
Fortran	493297 (1.64%)
Python	285050 (0.95%)
Tcl	213014 (0.71%)
Java	147285 (0.49%)
yacc/bison	122325 (0.41%)
Expect	103701 (0.34%)
lex/flex	41967 (0.14%)
awk/gawk	17431 (0.06%)
Objective-C	14645 (0.05%)
Ada	13200 (0.04%)

C shell	10753 (0.04%)
Pascal	4045 (0.01%)
sed	3940 (0.01%)

Here you can see that C is pre-eminent (with over 71% of the code), followed by C++, shell, LISP, assembly, Perl, Fortran, and Python. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has about 4.5 million lines of code, a very respectable showing, but is far less than C (over 21 million SLOC). Still, there's increasing use of C++ code; in the last survey, C had 80.55% and C++ had 7.51%. There is slightly less C code in the total percentage of code, most of which is being taken by C++. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

LISP continues to place very highly, far more than Perl, Python, Fortran, or Java. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 87% (627626/722430) of the LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages. Perl includes 5739 lines of LISP, and Python includes another 2353 of LISP that is directly used to support elaborate Emacs modes for program editing. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

Some may be surprised at the number of different languages, but I believe this should be considered not a weakness but a strength. This Linux distribution supports a wide number of languages, enabling developers to choose the ``best tool for the job."

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 94 different lex/flex files, and 138 yacc/bison files. Some build directories use lex/flex or yacc/bison more than once.

Other insights can be gained from the file counts. There were 352,549 files, of which 130,488 were counted source code files (ignoring duplicate files and automatically generated ones). Not included in this count were 10,807 files which contained duplicate contents, and 1,587 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 21,461,450 SLOC contained in 78,676 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code. Intriguingly enough, Red Hat Linux 6.2 had essentially the same average number of physical lines of C code.

3.5 Total Counts by License

Here are the various license types, sorted by the SLOC in the packages with those licenses (see section 2.4 for how these license values were determined):

```
15185987 (50.36%) GPL
 2498084 ( 8.28%) MIT
 2305001 ( 7.64%) LGPL
 2065224 ( 6.85%) MPL
```



```

1826601 (6.06%) Distributable
1315348 (4.36%) BSD
 907867 (3.01%) BSD-like
 766859 (2.54%) Freely distributable
 692561 (2.30%) Free
 455980 (1.51%) GPL, LGPL
 323730 (1.07%) GPL/MIT
 321123 (1.07%) Artistic or GPL
 191379 (0.63%) PHP
 173161 (0.57%) Apache-like
 161451 (0.54%) OpenLDAP
 146647 (0.49%) LGPL/GPL
 103439 (0.34%) GPL (programs), relaxed LGPL (libraries),
                and public domain (docs)
 103291 (0.34%) Apache
  73650 (0.24%) W3C
  73356 (0.24%) IBM Public License
  66554 (0.22%) University of Washington's Free-Fork License
  59354 (0.20%) Public domain
  39828 (0.13%) GPL and Artistic
  31019 (0.10%) GPL or BSD
  25944 (0.09%) GPL/BSD
  20740 (0.07%) Not listed
  20722 (0.07%) MIT-like
  18353 (0.06%) GPL/LGPL
  12987 (0.04%) Distributable - most of it GPL
   8031 (0.03%) Python
   6234 (0.02%) GPL/distributable
   4894 (0.02%) Freely redistributable
   1977 (0.01%) Artistic
   1941 (0.01%) GPL (not Firmware)
    606 (0.00%) Proprietary

```

These can be grouped by totalling up SLOC for licenses containing certain key phrases:

```

16673212 (55.30%) GPL
 3029420 (10.05%) LGPL
 2842536 (9.43%) MIT
 2612681 (8.67%) distributable
 2280178 (7.56%) BSD
 2065224 (6.85%) MPL
  162793 (0.54%) public domain

```

From these numbers, you can determine that:

1. The GPL is far and away the most common license (by lines of code) of any single license. In fact, the category ``GPL" (packages with only this one license) all by itself accounts for 50.36% of the packages. By totalling the SLOC for all packages that include "GPL" in the license text, the total rises to 55%. No matter how you look at it, the GPL is the dominant single license in this distribution.
2. The next most common licenses were the MIT, LGPL, MPL, and BSD licenses (in order). This is in line with expectations: the most well-known and well-used open source licenses are the GPL, MIT, LGPL, and BSD licenses. Although the MPL does well in terms of SLOC, there is only one program in this distribution that uses it - Mozilla. There is some use of the ``Artistic" license, but its use is far less; note that papers such as [Perens \[1999\]](#) specifically recommend against using the the Artistic license due to its legal ambiguities.
3. Very little software is released as public domain software (``no copyright"). In this distribution, only 0.2% of the software is in packages labelled as ``public domain" (note that the 0.54% figure above includes the ``sane" package which has documentation in the public domain). There may be several factors that account for this. First, if a developer wishes to get credit for their work, this is a poor ``license;" by law anyone can claim ownership of ``public domain" software. Second, there may be a fear of litigation; both the MIT and BSD licenses permit essentially arbitrary use but forbid lawsuits. While licenses such as MIT's and BSD's are not proof against a lawsuit, they at least provide some

legal protection, while releasing software to the public domain provides absolutely no protection. Finally, any software released into the public domain can be re-licensed under any other license, so there's nothing that keeps public domain software in the public domain - any of the other licenses here can ``dominate" a public domain license.

4. There is a tiny amount of proprietary code, which is entirely in one component - Netscape Communicator / Navigator. This component uses the Motif toolkit (which is not open source) and has proprietary code mixed into it. As a result, almost none of the code for this package is included on the CD-ROM - only a small amount of ``placeholder" code is there. In the future it is expected that this component will be replaced by Mozilla.
5. The packages which are clearly MIT-like/BSD-like licenses (totalling the MIT, BSD, MIT-like, BSD-like, and none/public domain entries) total 4,742,021 SLOC (15.92%). It's worth noting that 1,837,608 of these lines (39%) is accounted for by the XFree86 X server, an infrastructure component used for Linux's graphical user interface (GUI).
6. If the license types "distributable", "freely distributable", "MPL", "Free", "Artistic", "Apache", "Apache-like", and "IBM Public license" software was also considered MIT-like/BSD-like, the total SLOC would be 7,954,474 (26%, down from 36%). Unfortunately, the information to determine which of these other packages are simply BSD-like/MIT-like licenses is not included in the specification files.
7. The packages which include copylefting licenses (GPL or LGPL) total 63%. Limiting to only those that are GPL, LGPL, or both yields 60%, the same percentage as in Red Hat Linux 6.2 and a clear majority.

It is quite clear that in this distribution the GPL is the dominant license and that copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses. This is a simple quantitative explanation why several visible projects (Mozilla, Troll Tech's Qt, and Python) have changed their licenses so that they're compatible with the GPL. When there is so much GPL software, GPL compatibility is critically important to the survival of many open source projects. See the Free Software Foundation's information on [Various Licenses and Comments about Them \[FSF 2001a\]](#) for information on GPL compatibility, and the [GPL FAQ \[FSF 2001b\]](#) for more information on the GPL in general.

The most common open source licenses in this distribution (by SLOC) are the GPL, MIT, LGPL, and BSD licenses (as well as the MPL, but note that it's only used by one project). Note that this is consistent with [Perens \[1999\]](#), who pleads that developers use an existing license instead of developing a new license where possible.

As of this writing, the GPL has received the most attention of these licenses, because Microsoft has specifically been attacking the GPL license. The GPL license permits commercial use of the program, but requires that distributors of modified versions of the program must also release the source code to their changes under the same terms. Therefore, software released under the GPL resists Microsoft's usual ``embrace and extend" approach to destroy competitors - Microsoft can use and change GPL'ed code, but it cannot make its changes to that code proprietary. As a counter-example, Kerberos (a security component released using an MIT license instead of the GPL) was recently incorporated by Microsoft into their products, and then extended in an incompatible way to prevent users from fully interoperating between products [\[Schneier 2000\]](#). Had Kerberos been released under a GPL or LGPL license, this would have been much more difficult. The presence of so many GPL and LGPL components should make Linux distributions more resistant to being ``embraced, extended, and extinguished."

3.6 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 30,152,114 physical source lines of code (SLOC); I will simplify this to ``over 30 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (as of 1998)	20 million
Red Hat Linux 6.2 (2000)	17 million

These numbers come from Bruce Schneier's *Crypto-Gram* [\[Schneier 2000\]](#), except for the Space Shuttle numbers which come from a National Academy of Sciences study [\[NAS 1996\]](#) and the Red Hat Linux 6.2 numbers which come from

[Wheeler 2001]. Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of most of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Note that a deployed ``minimal system" would have less code; see the paper analyzing Red Hat Linux 6.2 for more discussion about this [Wheeler 2001].

Note that the Red Hat Linux 7.1 system includes a number of applications - in many cases a choice for each category. There are two major desktop environments (GNOME and KDE), plus various lightweight options. There are two word processors (Abiword and KWord), two spreadsheets (Gnumeric and KSpread), two relational database systems (MySQL and Postgres), and two web servers (Apache and TUX). In short, Red Hat Linux 7.1 includes a large number of applications, many of which are not included in its Microsoft or Sun equivalents.

At first blush, this bundling of applications with the operating system might appear similar to Microsoft's policy of combining applications with operating systems (which got Microsoft into legal trouble). However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as ``secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, this distribution (and many others) include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with smaller SLOC counts can sometimes provide greater functionality than programs with larger SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessarily cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized proprietary systems.

3.7 Effort and Cost Estimates

Finally, given all the assumptions shown previously, the effort values are:

Total Physical Source Lines of Code (SLOC)	= 30152114
Estimated Development Effort in Person-Years (Person-Months)	= 7955.75 (95469)
(Basic COCOMO model, Person-Months = $2.4 * (KSLOC^{**1.05})$)	
Estimated Schedule in Years (Months)	= 6.53 (78.31)
(Basic COCOMO model, Months = $2.5 * (person-months^{**0.38})$)	
Total Estimated Cost to Develop	= \$ 1074713481
(average salary = \$56286/year, overhead = 2.4).	

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2 (which had been released about one year earlier). Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2).

Had this Linux distribution been developed by conventional proprietary means, it would have cost over \$1.08 billion (1,000 million) to develop in the U.S. (in year 2000 dollars). Compare this to the \$600 million estimate for version 6.2. Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is quite extraordinary, since this represents approximately one year.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), X-windows (infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). Note that Mozilla and LAPACK were not included in Red Hat 6.2 at all.

The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed. C is still the predominant language but less so, with C++ primarily taking the difference. This appears in part to reflect many programmers' choice of C++ over C for GUI development. The increased amount of Fortran is primarily due to the inclusion of LAPACK.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is in the public domain.

Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches. Back in 1976, Bill Gates published his "Open Letter to Hobbyists", claiming that if software was freely shared it would prevent the writing of good software. He asked rhetorically, "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?" He presumed these were unanswerable questions, and both he and others based an industry on this assumption [Moody 2001]. Now, however, there are thousands of developers who are writing their own excellent code, and then giving it away. Gates was fundamentally wrong: sharing source code, and allowing others to extend it, is indeed a practical approach to developing large-scale systems - and its products can be more reliable.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), and other open source systems (such as FreeBSD). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost.

As was noted in the previous paper, some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like "Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not "reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ".m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

More information is available at <http://www.dwheeler.com/sloc>.

Appendix A. Details of Approach

This appendix discusses some of the issues I had to deal with when performing the analysis, hopefully in enough detail that someone could repeat the effort.

In particular, installing the source code required two steps:

1. install the source code files (converting source RPM packages into "spec" files and compressed source files),
2. unpack the source code files (which generates uncompressed source code, and the licensing information),

I then ran `sloccount` version 1.9 to analyze the source code files, and examined the warning messages and fixed any serious problems that arose. This was not as easy as it sounds; the previous paper (analyzing Red Hat Linux 6.2) discusses this in more detail [Wheeler 2001]. I've since released the tools I used to count code as the program `sloccount`, available at <http://www.dwheeler.com/sloccount>.

One complication should be mentioned here. Although the Red Hat Linux 7.1 package comes with a CD-ROM labelled "Source Code", the "Source Code" CD-ROM doesn't contain all the source code. Skipping the source code on the "binary" CD-ROM would produce an invalid count, because 224 source code packages are placed there (including important packages like `ssl`, `perl`, `python`, and `samba`). It's likely this was done because of CD-ROM space needs - there is so much source code that, even when compressed, it doesn't fit on one CD-ROM.

I then searched for "old" versions of programs that were also included on the CD (so that the same program wouldn't be counted twice), or those required for non-Intel x86 operation (since these would not be fully counted anyway). I did this by examining any specification (in `/usr/src/redhat/SPECS`) with "compat" or "10" or "11" in its title (it turned out all of them were old and needed removing). I also examined anything ending in a digit or "x" followed by ".spec", which located `qt1x.spec`. Through this process I removed:

```
compat-egcs.spec  compat-glibc.spec  compat-libs.spec  kde1-compat.spec
gtk+10.spec  libxml10.spec  x86-compat-libs.spec  qt1x.spec
```

I also removed any "beta" software which had a non-beta version available (beta software was identified by searching for "beta" in the package or specification file name). This removed:

```
glib-gtkbeta.spec  gtk+-gtkbeta.spec  pango-gtkbeta.spec
```

I also removed `mysqlclient9.spec`. This specification contained the older MySQL client library version 3.23.22, as shipped with Red Hat Linux 7, for use with applications linked against it. I did include `mysql.spec`, which had the code for the newer version 3.23.36 of MySQL (a relational database package).

Note that unlike Red Hat Linux 6.2, Red Hat Linux 7.1 didn't have two versions of `bash` or `ncurses`, so I didn't have to remove old versions of them. I left `db1`, `db2`, and `db3` in, because it can be argued that none of these three necessarily replaces the other two.

One complication was in handling the graphical subsystem "XFree86". Version 4 of XFree86 was used for all client-side applications, but version 3 was used to implement X servers. The XFree86 developers completely rebuilt XFree86, and Red Hat chose to stick with the older servers and the newer clients. I looked at the XFree86 source package for version 4, and it turned out that server code was included in the package (although not used in the standard Red Hat distribution). Rather than have XFree86 counted essentially twice (once as version 3, and another as version 4), I only counted the code in version 4 of XFree86.

I then unpacked the source code by running code that in essence did this:

```
cd /usr/src/redhat/SPECS
rpm -bp *.spec
```

This uncompresses the source code and applies all patches used by the actual system. Since I wanted to count the amount of code actually included in the system, it was important to include the patches. The actual code to unpack the source code was more complex, because it also marked every unpacked directory (in the BUILD directory) to identify the spec file it came from and the license of the program. The license was determined by (1) looking at the "Copyright" and "License" fields of the spec file, and if that didn't work, (2) looking at various files in the build directory, such as "LICENSE", "COPYING*", and "Artistic". Unfortunately, MIT-like and BSD-like licenses can be harder to detect (because their text can be varied), but many licenses (such as the GPL and LGPL) can be detected with great confidence. I used the "spec" file as the primary

source, because this was placed by a human (who could better understand legal technicalities than a machine).

I actually had to repeat the unpacking more than once; the RPM system would notice a missing dependency for building the software and protest. This required installation of the missing component (in some cases I didn't have to install the program and could have forced installation, but I did not want to risk corrupting the results by failing to install a package).

A surprising development was that the packages "imap" and "samba" reported errors in unpacking. For imap, patch #5 (imap-4.7c2-flock.patch) and for samba, patch #21 (samba-ia64.patch of source/passdb/pass_check.c) would cause unpacking to halt. I unpacked the software and simply counted what was there; this appears to be what the original developers did.

I examined the reported license values, in particular for all code more than 100,000 source lines of code (as the largest components, wrong values for these components would be more likely to cause significant error). I found that Perl had been assigned "GPL" in its spec file, but this isn't the whole story; as documented in its README file, Perl can be used under either the GPL or Artistic license, so its license entry was changed to "GPL or Artistic". Mozilla's licensing situation is more complex; some portions of it are actually under a separate dual licensing scheme (licensed under both the GPL and Netscape Public License, i.e., NPL). However, labelling it as "MPL, NPL, and GPL" would probably overstate the amount of code licensed under the GPL, so I left its entry as the MPL license.

Note that the unpacked source files (including source code, fonts, documentation, and so on) totalled more than 4.4 Gigabytes.

I ran the analysis code as a normal user, so I first had to set the permissions for users to read the code. I then reverted to normal user account, and used sloccount version 1.9 to measure the source code, using the following bash command:

```
sloccount --multiproject /usr/src/redhat/BUILD > sloc-actions 2>&1 &
```

Note that I did `_not_` use the `--follow` option of sloccount. Some programs, notably pine, include a symbolic link to other directories such as `/usr/lib`. Thus, using `--follow` would have included files outside of the intended directory in the analysis.

I looked over various error reports and determined that none would fundamentally invalidate the results. For example, there were several errors in the XFree86 source code involving improperly formatted strings. It appears that these are syntax errors in the code that are preprocessed away (and thus not noticed by the compiler). I intend to report these problems to the XFree86 project. One program was a bash shell script that began with `"#!/usr/bin/env bash"`, which sloccount's heuristics could not handle at the time. I then modified sloccount to correctly determine its type (it's a bash shell script).

Note that sloccount creates a large number of small files. This isn't fundamentally a problem, but because of the large scale of the system I found that I ran out of inodes if I tried to store multiple copies of results. Those who try to duplicate this activity may want to specially format their filesystems to include more inodes.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/developpro.html>.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the "Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[FSF 2001a] Free Software Foundation (FSF). 2001. *Various Licenses and Comments about Them*. <http://www.gnu.org/philosophy/license-list.html>.

[FSF 2001b] Free Software Foundation (FSF). 2001. *General Public License (GPL) Frequently Asked Questions (FAQ)*. <http://www.gnu.org/copyleft/gpl-faq.html>.

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

- [Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>
- [Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf
- [Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>
- [Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.
- [Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.
- [Moody 2001] Moody, Glyn. 2001. *Rebel Code*. ISBN 0713995203.
- [NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>
- [OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.
- [Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>
- [Perens 1999] Perens, Bruce. January 1999. *Open Sources: Voices from the Open Source Revolution*. "The Open Source Definition". ISBN 1-56592-582-3. <http://www.oreilly.com/catalog/opensources/book/perens.html>
- [Raymond 1999] Raymond, Eric S. January 1999. "A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.
- [Schneier 2000] Schneier, Bruce. March 15, 2000. "Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>
- [Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.
- [Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>
- [Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...". <http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.
- [Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet. <http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>
- [Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*. http://www.dwheeler.com/oss_fs_refs.html.
- [Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.
- [Wheeler 2001]. Wheeler, David A. May 9, 2001 (minor update from November 6, 2000). *Estimating Linux's Size*. Version 1.04. <http://www.dwheeler.com/sloc>.
- [Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

Trademark owners own any trademarks mentioned.

This paper is (C) Copyright 2001 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. Talk to me to request republication rights. When referring to the paper, please refer to it as "More than a Gigabuck: Estimating GNU/Linux's Size" by David A. Wheeler, located at

<http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

More Than a Gigabuck: Estimating GNU/Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

June 20, 2001

Version 1.01

This paper analyzes the amount of source code in GNU/Linux, using Red Hat Linux 7.1 as a representative GNU/Linux distribution, and presents what I believe are interesting results.

In particular, it would cost over \$1 billion (\$1,000 million - a Gigabuck) to develop this Linux distribution by conventional proprietary means in the U.S. (in year 2000 U.S. dollars). Compare this to the \$600 million estimate for Red Hat Linux version 6.2 (which had been released about one year earlier). Also, Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2. Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2). Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is due to an increased number of mature and maturing open source / free software programs available worldwide.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), X-windows (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part or as an alternative, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is public domain.

This paper is an update of my previous paper on estimating GNU/Linux's size, which measured Red Hat Linux 6.2 [Wheeler 2001]. Since Red Hat Linux 6.2 was released in March 2000, and Red Hat Linux 7.1 was released in April 2001, this paper shows what's changed over approximately one year. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The GNU/Linux operating system (also called simply ``Linux") has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a ``typical" intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally ``open source software" and/or ``free software". A program that is ``open source software" or ``free software" is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of ``open source software" is available from the Open Source Initiative [OSI 1999], a more formal definition of ``free software" (as the term is used in this paper) is available from the Free Software Foundation [FSF 2000], and other general information about these topics is available at Wheeler [2000a]. Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is ``open source software"/``free software", and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular

vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. The only published data I've found (other than my own) was developed by Microsoft in the documents usually called ``Halloween I" and ``Halloween II" [\[Halloween I\]](#) [\[Halloween II\]](#). In a previous paper, I examined Red Hat Linux 6.2 and the numbers from the Halloween papers.

This paper updates my previous paper, showing estimates of the size of one of today's GNU/Linux distributions, and it estimates how much it would cost to rebuild this typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean. I have intentionally written this paper so that you do *not* need to read the previous version of this paper first.

For my purposes, I have selected as my ``representative" Linux distribution Red Hat Linux version 7.1. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [\[Shankland 2000b\]](#). Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE (a German distributor) at 15%. Not all Linux copies are ``sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size for the same kind of functionality.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate ``all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 7.1 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Note that Red Hat Linux 6.2 was released on March 2000, Red Hat Linux 7 was released on September 2000 (I have not counted its code), and Red Hat Linux 7.1 was released on April 2001. Thus, the differences between Red Hat Linux 7.1 and 6.2 show differences accrued over 13 months (approximately one year).

Clearly there is far more open source / free software available worldwide than is counted in this paper. However, the job of a distributor is to examine these various options and select software that they believe is both sufficiently mature and useful to their target market. Thus, examining a particular distribution results in a selective analysis of such software.

Section 2 briefly describes the approach used to estimate the ``size" of this distribution (more details are in Appendix A). Section 3 discusses some of the results. Section 4 presents conclusions, followed by an appendix.

2. Approach

My basic approach was to:

1. install the source code files in uncompressed format; this requires carefully selecting the source code to be analyzed.
2. count the number of source lines of code (SLOC); this requires a careful definition of SLOC.
3. use an estimation model to estimate the effort and cost of developing the same system in a proprietary manner; this requires an estimation model.
4. determine the software licenses of each component and develop statistics based on these categories.

More detail on this approach is described in Appendix A. A few summary points are worth mentioning here, however.

2.1 Selecting Source Code

I included all software provided in the Red Hat distribution, but note that Red Hat no longer includes software packages that only apply to other CPU architectures (and thus packages not applying to the x86 family were excluded). I did not include ``old" versions of software, or ``beta" software where non-beta was available. I did include ``beta" software where there was

no alternative, because some developers don't remove the ``beta" label even when it's widely used and perceived to be reliable.

I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once (as a tie-breaker, such files are assigned to the first build package it applies to in alphabetic order).

The code in makefiles and Red Hat Package Manager (RPM) specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

2.2 Defining SLOC

The ``physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: ``a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the ``logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the ``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

Note that this required that every file be categorized by language type (so that the correct syntax for comments, strings, and so on could be applied). Also, automatically generated files had to be detected and ignored. Thankfully, my tool ``sloccount" does this automatically.

2.3 Estimation Models

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer ``COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions. Of particular note, basic COCOMO does not include the time to develop translations to other human languages (of documentation, data, and program messages) nor fonts.

There is reason to believe that these models, while imperfect, are still valid for estimating effort in open source / free software projects. Although many open source programs don't need management of human resources, they still require technical management, infrastructure maintenance, and so on. Design documentation is captured less formally in open source projects, but it's often captured by necessity because open source projects tend to have many developers separated geographically. Clearly, the systems must still be programmed. Testing is still done, although as with many of today's proprietary programs, a good deal of testing is done through alpha and beta releases. In addition, quality is enhanced in many open source projects through peer review of submitted code. The estimates may be lower than the actual values because they don't include estimates of human language translations and fonts.

Each software source code package, once uncompressed, produced zero or more ``build directories" of source code. Some packages do not actually contain source code (e.g., they only contain configuration information), and some packages are collections of multiple separate pieces (each in different build directories), but in most cases each package uncompresses into

a single build directory containing the source code for that package. Each build directory had its effort estimation computed separately; the efforts of each were then totalled. This approach assumes that each build directory was developed essentially separately from the others, which in nearly all cases is quite accurate. This approach slightly underestimates the actual effort in the rare cases where the development of the code in separate build directories are actually highly interrelated; this effect is not expected to invalidate the overall results.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [DSMC]. This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired. These are the same values as used in my last report.

2.4 Determining Software Licenses

A software license determines how that software can be used and reused, and open source software licensing has been a subject of great debate. Well-known open source licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD license, and the Artistic license. The GPL and LGPL are termed "copylefting" licenses, that is, the license is designed to prevent the code from becoming proprietary. See [Perens \[1999\]](#) for more information. Obvious questions include "what license(s) are developers choosing when they release their software" and "how much code has been released under the various licenses?"

An approximation of the amount of software using various licenses can be found for this particular distribution. Red Hat Linux uses the Red Hat Package Manager (RPM), and RPM supports capturing license data for each package (these are the "Copyright" and "License" fields in the specification file). I used this information to determine how much code was covered by each license. Since this field is simply a string of text, there were some variances in the data that I had to clean up, for example, some entries said "GNU" while most said "GPL". In some cases Red Hat did not include licensing information with a package. In that case, I wrote a program to attempt to determine the license by looking for certain conventional filenames and contents.

This is an imperfect approach. Some packages contain different pieces of code with different licenses applying to different pieces. Some packages are "dual licensed", that is, they are released under more than one license. Sometimes these other licenses are noted, while at other times they aren't. There are actually two BSD licenses (the "old" and "new" licenses), but the specification files don't distinguish between them. Also, if the license wasn't one of a small set of common licenses, Red Hat tended to assign nondescriptive phrases such as "distributable". My automated techniques were limited too, in particular, while some licenses (e.g., the GPL and LGPL) are easy to recognize automatically, BSD-like and MIT-like licenses vary the license text and so are more difficult to recognize automatically (and some changes to the license would render them non-open source, non-free software). Thus, when Red Hat did not identify a package's license, a program dual licensed under both the BSD and GPL license might only be labelled as having the GPL using these techniques. Nevertheless, this approach is sufficient to give some insight into the amount of software using various licenses. Future research could examine each license in turn and categorize them; such research might require several lawyers to determine when two licenses in certain circumstances are "equal."

One program worth mentioning in this context is Python, which has had several different licenses. Version 1.6 and later (through 2.1) had more complex licenses that the Free Software Foundation (FSF) believes were incompatible with the GPL. Recently this was resolved by another change to the Python license to make Python fully compatible with the GPL. Red Hat Linux 7.1 includes an older version of Python (1.5.2), presumably because of these licensing issues. It can't be because Red Hat is unaware of later versions of Python; Red Hat uses Python in its installation program (which it developed and maintains). Hopefully, the recent resolution of license incompatibilities with the GPL license will enable Red Hat to include the latest versions of Python in the future. In any case, there are several different Python-specific licenses, all of which can legitimately be called the "Python" license. Red Hat has labelled Python itself as having a "Distributable" license, and package Distutils-1.0.1 is labelled with the "Python" license; these labels are kept in this paper.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (sorted by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total counts grouped by their software licenses, section 3.6 presents total SLOC counts, and section 3.7 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 35 largest components (as measured by number of source lines of code), along with their licenses (see section 2.4 for how these license values were determined). In the language section, ``ansic" means C code, ``asm" is assembly, ``sh" is Bourne shell and related shells, and ``cpp" is C++.

SLOC	Directory	SLOC-by-Language (Sorted)
2437470	kernel-2.4.2	ansic=2285657,asm=144411,sh=3035,perl=2022,yacc=1147, tcl=576,lex=302,awk=248,sed=72 [GPL]
2065224	mozilla	cpp=1279902,ansic=739470,perl=21220,sh=13717,asm=5212, java=3107,yacc=1831,lex=470,csch=271,sed=24 [MPL]
1837608	XFree86-4.0.3	ansic=1750460,asm=35397,cpp=20725,sh=14666,tcl=9182, yacc=3360,perl=1675,lex=1608,awk=393,csch=85,sed=57 [MIT]
984076	gcc-2.96-20000731	ansic=789901,cpp=126738,yacc=19272,sh=17993,asm=14559, lisp=7161,fortran=3814,exp=3705,objc=479,sed=310,perl=144 [GPL]
967263	gdb+dejagnum-20010316	ansic=871288,exp=58422,sh=12054,cpp=8252,yacc=5906, asm=5031,tcl=4477,lisp=1403,sed=248,awk=170,java=7,fortran=5 [GPL]
690983	binutils-2.10.91.0.2	ansic=489993,asm=161236,exp=13234,sh=12835, yacc=5665,cpp=4777,lex=1488,perl=776,sed=561,lisp=394,awk=24 [GPL]
646692	glibc-2.2.2	ansic=548722,asm=88413,sh=6036,perl=2120,awk=1037, yacc=315,sed=49 [LGPL]
627626	emacs-20.7	lisp=453898,ansic=169956,sh=2622,perl=884,asm=253, csch=9,sed=4 [GPL]
474829	LAPACK	fortran=473590,ansic=1239 [Freely distributable]
455980	gimp-1.2.1	ansic=427967,perl=17482,lisp=9648,yacc=502,sh=381 [GPL, LGPL]
402799	mysql-3.23.36	ansic=249350,cpp=84068,perl=25088,tcl=18980,sh=18323, asm=3987,awk=1436,java=1149,sed=418 [LGPL]
395194	tcltk-8.3.1	ansic=291457,tcl=84322,sh=12259,exp=5742,yacc=876, awk=273,perl=265 [BSD]
345949	kdebase-2.1.1	cpp=181210,ansic=158682,sh=4880,perl=1155,python=22 [GPL]
323730	Mesa-3.4	ansic=286437,cpp=18189,asm=10002,sh=7611,objc=1184, python=307 [GPL/MIT]
321123	perl-5.6.0	perl=146755,ansic=118233,sh=49377,lisp=5739,yacc=996, java=23 [Artistic or GPL]
318430	libgcj	ansic=191432,cpp=56843,java=41716,sh=15581,asm=11262, exp=841,perl=731,awk=24

		[GPL]
304819	teTeX-1.0	ansic=223491,perl=49789,sh=17634,cpp=9407,pascal=1546, yacc=1507,awk=622,lex=323,sed=314,asm=139,csch=47 [Distributable]
298742	qt-2.3.0	cpp=259310,ansic=34578,yacc=2444,sh=1493,lex=480, perl=422,lisp=15 [GPL]
286113	postgresql-7.0.3	ansic=237184,java=17540,yacc=9740,sh=8975,tcl=7751, lex=1810,perl=1276,python=959,cpp=801,asm=70,csch=5,sed=2 [BSD]
283785	kdelibs-2.1.1	cpp=261334,ansic=17578,sh=1887,java=1538,perl=731, yacc=607,lex=110 [LGPL]
277502	xemacs-21.1.14	ansic=199927,lisp=73366,sh=2948,perl=930,asm=247, csch=62,sed=22 [GPL]
264528	gs5.50	ansic=259471,cpp=2266,asm=968,sh=823,lisp=405,perl=336, yacc=201,lex=58 [GPL]
227354	krb5-1.2.2	ansic=197886,exp=19124,sh=5140,yacc=2474,perl=1529, awk=393,python=348,lex=190,csch=147,sed=123 [MIT]
215473	vnc_unixsrc	ansic=212766,cpp=848,asm=780,perl=648,sh=431 [GPL]
213818	koffice-2.0.1	cpp=197637,sh=7296,yacc=3791,ansic=3213,perl=1801, lex=80 [GPL]
202842	openssl-0.9.6	ansic=131874,cpp=25744,perl=14737,asm=12428,python=10171, yacc=3297,sh=2641,tcl=1583,lisp=224,objc=143 [BSD-like]
200908	Python-1.5.2	python=101017,ansic=96521,lisp=2353,sh=673,perl=342, sed=2 [Distributable]
194799	bind-9.1.0	ansic=173830,sh=12101,yacc=6025,perl=2830,tcl=13 [BSD-like]
192394	xpdf-0.92	cpp=167135,ansic=21621,sh=3638 [GPL]
191379	php-4.0.4pl1	ansic=173334,cpp=7033,sh=6591,lex=1867,yacc=1569, java=437,awk=367,perl=181 [PHP]
190950	pine4.33	ansic=190020,sh=838,csch=62,perl=30 [Freely distributable]
173492	abi	cpp=159595,ansic=12605,perl=725,sh=550,python=17 [GPL]
167663	kdemultimedia-2.1.1	cpp=140731,ansic=23844,tcl=1004,sh=800,asm=598, lex=578,perl=106,awk=2 [GPL]
163449	4Suite-0.10.1	python=91445,ansic=72004 [Apache-like]
159301	linuxconf-1.24r2	cpp=142970,perl=6738,sh=3821,java=3074,ansic=2613, python=85 [GPL]

Note that the operating system kernel (Linux) is the largest single component, at over 2.4 million lines of code (mostly in C); that compares to 1.5 million lines of code in Red Hat 6.2. See section 3.2 for a more detailed discussion about the Linux kernel.

The next largest component is Mozilla; this is large because it's really a suite of applications including a web browser, email reader, news reader, HTML editor, and so on. Mozilla is the basis for Netscape Navigator 6.0. Mozilla was not included at all in Red Hat Linux 6.2.

The next largest component is X windows, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to gain functionality and size), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, the symbolic debugger, a set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). Emacs is next largest, which should not be a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system.

Note that language implementations tend to be written in themselves, particularly for their libraries. Perl's implementation is written mostly in Perl, and Python is written mostly in Python. Intriguingly, this is not true for Tcl.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the Linux kernel (at over 2.4 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 1,400,000 lines (57% of the Linux kernel) was in the ``drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of peripherals. No other subdirectory comes close to this size - the second largest is the ``arch" directory (at over 446,000 SLOC, 18% of the kernel), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is over 168,000 SLOC.

Richard Stallman and others have argued that the resulting system often called ``Linux" should instead be called ``GNU/Linux" [\[Stallman 2000\]](#). In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to ``free software" (free as in the freedom to use, modify, and redistribute software for any purpose). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called ``Linux". Using the term ``Linux" is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux" or ``GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc, gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux" and not just ``Linux." For more information on the sizes of the Linux kernel components, see http://www.dwheeler.com/sloc/redhat71-v1/kernel_sloc.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code (using the naming conventions of sloccount, the program used to count SLOC):

Language	SLOC (%)
C	21461450 (71.18%)
C++	4575907 (15.18%)
Shell (Bourne-like)	793238 (2.63%)
Lisp	722430 (2.40%)
Assembly	565536 (1.88%)
Perl	562900 (1.87%)
Fortran	493297 (1.64%)
Python	285050 (0.95%)

Tcl	213014 (0.71%)
Java	147285 (0.49%)
yacc/bison	122325 (0.41%)
Expect	103701 (0.34%)
lex/flex	41967 (0.14%)
awk/gawk	17431 (0.06%)
Objective-C	14645 (0.05%)
Ada	13200 (0.04%)
C shell	10753 (0.04%)
Pascal	4045 (0.01%)
sed	3940 (0.01%)

Here you can see that C is pre-eminent (with over 71% of the code), followed by C++, shell, LISP, assembly, Perl, Fortran, and Python. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has about 4.5 million lines of code, a very respectable showing, but is far less than C (over 21 million SLOC). Still, there's increasing use of C++ code; in the last survey, C had 80.55% and C++ had 7.51%. There is slightly less C code in the total percentage of code, most of which is being taken by C++. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

LISP continues to place very highly, far more than Perl, Python, Fortran, or Java. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 87% (627626/722430) of the LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages. Perl includes 5739 lines of LISP, and Python includes another 2353 of LISP that is directly used to support elaborate Emacs modes for program editing. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

Some may be surprised at the number of different languages, but I believe this should be considered not a weakness but a strength. This Linux distribution supports a wide number of languages, enabling developers to choose the ``best tool for the job."

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 94 different lex/flex files, and 138 yacc/bison files. Some build directories use lex/flex or yacc/bison more than once.

Other insights can be gained from the file counts. There were 352,549 files, of which 130,488 were counted source code files (ignoring duplicate files and automatically generated ones). Not included in this count were 10,807 files which contained duplicate contents, and 1,587 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 21,461,450 SLOC contained in 78,676 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code. Intriguingly enough, Red Hat Linux 6.2 had essentially the same average number of physical lines of C code.

3.5 Total Counts by License

Here are the various license types, sorted by the SLOC in the packages with those licenses (see section 2.4 for how these license values were determined):

```

15185987 (50.36%) GPL
 2498084 (8.28%) MIT
 2305001 (7.64%) LGPL
 2065224 (6.85%) MPL
 1826601 (6.06%) Distributable
 1315348 (4.36%) BSD
   907867 (3.01%) BSD-like
   766859 (2.54%) Freely distributable
   692561 (2.30%) Free
   455980 (1.51%) GPL, LGPL
   323730 (1.07%) GPL/MIT
   321123 (1.07%) Artistic or GPL
   191379 (0.63%) PHP
   173161 (0.57%) Apache-like
   161451 (0.54%) OpenLDAP
   146647 (0.49%) LGPL/GPL
   103439 (0.34%) GPL (programs), relaxed LGPL (libraries),
                  and public domain (docs)
  103291 (0.34%) Apache
   73650 (0.24%) W3C
   73356 (0.24%) IBM Public License
   66554 (0.22%) University of Washington's Free-Fork License
   59354 (0.20%) Public domain
   39828 (0.13%) GPL and Artistic
   31019 (0.10%) GPL or BSD
   25944 (0.09%) GPL/BSD
   20740 (0.07%) Not listed
   20722 (0.07%) MIT-like
   18353 (0.06%) GPL/LGPL
   12987 (0.04%) Distributable - most of it GPL
    8031 (0.03%) Python
    6234 (0.02%) GPL/distributable
    4894 (0.02%) Freely redistributable
    1977 (0.01%) Artistic
    1941 (0.01%) GPL (not Firmware)
     606 (0.00%) Proprietary

```

These can be grouped by totalling up SLOC for licenses containing certain key phrases:

```

16673212 (55.30%) GPL
 3029420 (10.05%) LGPL
 2842536 (9.43%) MIT
 2612681 (8.67%) distributable
 2280178 (7.56%) BSD
 2065224 (6.85%) MPL
  162793 (0.54%) public domain

```

From these numbers, you can determine that:

1. The GPL is far and away the most common license (by lines of code) of any single license. In fact, the category ``GPL" (packages with only this one license) all by itself accounts for 50.36% of the packages. By totalling the SLOC for all packages that include "GPL" in the license text, the total rises to 55%. No matter how you look at it, the GPL is the dominant single license in this distribution.
2. The next most common licenses were the MIT, LGPL, MPL, and BSD licenses (in order). This is in line with expectations: the most well-known and well-used open source licenses are the GPL, MIT, LGPL, and BSD licenses.

Although the MPL does well in terms of SLOC, there is only one program in this distribution that uses it - Mozilla. There is some use of the ``Artistic" license, but its use is far less; note that papers such as [Perens \[1999\]](#) specifically recommend against using the the Artistic license due to its legal ambiguities.

3. Very little software is released as public domain software (``no copyright"). In this distribution, only 0.2% of the software is in packages labelled as ``public domain" (note that the 0.54% figure above includes the ``sane" package which has documentation in the public domain). There may be several factors that account for this. First, if a developer wishes to get credit for their work, this is a poor ``license;" by law anyone can claim ownership of ``public domain" software. Second, there may be a fear of litigation; both the MIT and BSD licenses permit essentially arbitrary use but forbid lawsuits. While licenses such as MIT's and BSD's are not proof against a lawsuit, they at least provide some legal protection, while releasing software to the public domain provides absolutely no protection. Finally, any software released into the public domain can be re-licensed under any other license, so there's nothing that keeps public domain software in the public domain - any of the other licenses here can ``dominate" a public domain license.
4. There is a tiny amount of proprietary code, which is entirely in one component - Netscape Communicator / Navigator. This component uses the Motif toolkit (which is not open source) and has proprietary code mixed into it. As a result, almost none of the code for this package is included on the CD-ROM - only a small amount of ``placeholder" code is there. In the future it is expected that this component will be replaced by Mozilla.
5. The packages which are clearly MIT-like/BSD-like licenses (totalling the MIT, BSD, MIT-like, BSD-like, and none/public domain entries) total 4,742,021 SLOC (15.92%). It's worth noting that 1,837,608 of these lines (39%) is accounted for by the XFree86 X server, an infrastructure component used for Linux's graphical user interface (GUI).
6. If the license types "distributable", "freely distributable", "MPL", "Free", "Artistic", "Apache", "Apache-like", and "IBM Public license" software was also considered MIT-like/BSD-like, the total SLOC would be 7,954,474 (26%, down from 36%). Unfortunately, the information to determine which of these other packages are simply BSD-like/MIT-like licenses is not included in the specification files.
7. The packages which include copylefting licenses (GPL or LGPL) total 63%. Limiting to only those that are GPL, LGPL, or both yields 60%, the same percentage as in Red Hat Linux 6.2 and a clear majority.

It is quite clear that in this distribution the GPL is the dominant license and that copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses. This is a simple quantitative explanation why several visible projects (Mozilla, Troll Tech's Qt, and Python) have changed their licenses so that they're compatible with the GPL. When there is so much GPL software, GPL compatibility is critically important to the survival of many open source projects. See the Free Software Foundation's information on [Various Licenses and Comments about Them \[FSF 2001a\]](#) for information on GPL compatibility, and the [GPL FAQ \[FSF 2001b\]](#) for more information on the GPL in general.

The most common open source licenses in this distribution (by SLOC) are the GPL, MIT, LGPL, and BSD licenses (as well as the MPL, but note that it's only used by one project). Note that this is consistent with [Perens \[1999\]](#), who pleads that developers use an existing license instead of developing a new license where possible.

As of this writing, the GPL has received the most attention of these licenses, because Microsoft has specifically been attacking the GPL license. The GPL license permits commercial use of the program, but requires that distributors of modified versions of the program must also release the source code to their changes under the same terms. Therefore, software released under the GPL resists Microsoft's usual ``embrace and extend" approach to destroy competitors - Microsoft can use and change GPL'ed code, but it cannot make its changes to that code proprietary. As a counter-example, Kerberos (a security component released using an MIT license instead of the GPL) was recently incorporated by Microsoft into their products, and then extended in an incompatible way to prevent users from fully interoperating between products [\[Schneier 2000\]](#). Had Kerberos been released under a GPL or LGPL license, this would have been much more difficult. The presence of so many GPL and LGPL components should make Linux distributions more resistant to being ``embraced, extended, and extinguished."

3.6 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 30,152,114 physical source lines of code (SLOC); I will simplify this to ``over 30 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million

Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (as of 1998)	20 million
Red Hat Linux 6.2 (2000)	17 million

These numbers come from Bruce Schneier's *Crypto-Gram* [Schneier 2000], except for the Space Shuttle numbers which come from a National Academy of Sciences study [NAS 1996] and the Red Hat Linux 6.2 numbers which come from [Wheeler 2001]. Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of most of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Note that a deployed ``minimal system" would have less code; see the paper analyzing Red Hat Linux 6.2 for more discussion about this [Wheeler 2001].

Note that the Red Hat Linux 7.1 system includes a number of applications - in many cases a choice for each category. There are two major desktop environments (GNOME and KDE), plus various lightweight options. There are two word processors (Abiword and KWord), two spreadsheets (Gnumeric and KSpread), two relational database systems (MySQL and Postgres), and two web servers (Apache and TUX). In short, Red Hat Linux 7.1 includes a large number of applications, many of which are not included in its Microsoft or Sun equivalents.

At first blush, this bundling of applications with the operating system might appear similar to Microsoft's policy of combining applications with operating systems (which got Microsoft into legal trouble). However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as ``secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, this distribution (and many others) include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with smaller SLOC counts can sometimes provide greater functionality than programs with larger SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessary cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized proprietary systems.

3.7 Effort and Cost Estimates

Finally, given all the assumptions shown previously, the effort values are:

```
Total Physical Source Lines of Code (SLOC)                = 30152114
Estimated Development Effort in Person-Years (Person-Months) = 7955.75 (95469)
  (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Estimated Schedule in Years (Months)                      = 6.53 (78.31)
  (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Total Estimated Cost to Develop                            = $ 1074713481
  (average salary = $56286/year, overhead = 2.4).
```

See appendix A for more data on how these effort values were calculated; you can retrieve more information from

<http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2 (which had been released about one year earlier). Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2).

Had this Linux distribution been developed by conventional proprietary means, it would have cost over \$1.08 billion (1,000 million) to develop in the U.S. (in year 2000 dollars). Compare this to the \$600 million estimate for version 6.2. Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is quite extraordinary, since this represents approximately one year.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), X-windows (infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). Note that Mozilla and LAPACK were not included in Red Hat 6.2 at all.

The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed. C is still the predominant language but less so, with C++ primarily taking the difference. This appears in part to reflect many programmers' choice of C++ over C for GUI development. The increased amount of Fortran is primarily due to the inclusion of LAPACK.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is in the public domain.

Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches. Back in 1976, Bill Gates published his "Open Letter to Hobbyists", claiming that if software was freely shared it would prevent the writing of good software. He asked rhetorically, "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?" He presumed these were unanswerable questions, and both he and others based an industry on this assumption [Moody 2001]. Now, however, there are thousands of developers who are writing their own excellent code, and then giving it away. Gates was fundamentally wrong: sharing source code, and allowing others to extend it, is indeed a practical approach to developing large-scale systems - and its products can be more reliable.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), and other open source systems (such as FreeBSD). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost.

As was noted in the previous paper, some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like "Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not "reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ".m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

More information is available at <http://www.dwheeler.com/sloc>.

Appendix A. Details of Approach

This appendix discusses some of the issues I had to deal with when performing the analysis, hopefully in enough detail that someone could repeat the effort.

In particular, installing the source code required two steps:

1. install the source code files (converting source RPM packages into "spec" files and compressed source files),
2. unpack the source code files (which generates uncompressed source code, and the licensing information),

I then ran `sloccount` version 1.9 to analyze the source code files, and examined the warning messages and fixed any serious problems that arose. This was not as easy as it sounds; the previous paper (analyzing Red Hat Linux 6.2) discusses this in more detail [Wheeler 2001]. I've since released the tools I used to count code as the program `sloccount`, available at <http://www.dwheeler.com/sloccount>.

One complication should be mentioned here. Although the Red Hat Linux 7.1 package comes with a CD-ROM labelled "Source Code", the "Source Code" CD-ROM doesn't contain all the source code. Skipping the source code on the "binary" CD-ROM would produce an invalid count, because 224 source code packages are placed there (including important packages like `ssl`, `perl`, `python`, and `samba`). It's likely this was done because of CD-ROM space needs - there is so much source code that, even when compressed, it doesn't fit on one CD-ROM.

I then searched for "old" versions of programs that were also included on the CD (so that the same program wouldn't be counted twice), or those required for non-Intel x86 operation (since these would not be fully counted anyway). I did this by examining any specification (in `/usr/src/redhat/SPECS`) with "compat" or "10" or "11" in its title (it turned out all of them were old and needed removing). I also examined anything ending in a digit or "x" followed by ".spec", which located `qt1x.spec`. Through this process I removed:

```
compat-egcs.spec  compat-glibc.spec  compat-libs.spec  kdel-compat.spec
gtk+10.spec  libxml10.spec  x86-compat-libs.spec  qt1x.spec
```

I also removed any "beta" software which had a non-beta version available (beta software was identified by searching for "beta" in the package or specification file name). This removed:

```
glib-gtkbeta.spec  gtk+-gtkbeta.spec  pango-gtkbeta.spec
```

I also removed "`mysqlclient9.spec`". This specification contained the older MySQL client library version 3.23.22, as shipped with Red Hat Linux 7, for use with applications linked against it. I did include "`mysql.spec`", which had the code for the newer version 3.23.36 of MySQL (a relational database package).

Note that unlike Red Hat Linux 6.2, Red Hat Linux 7.1 didn't have two versions of `bash` or `ncurses`, so I didn't have to remove old versions of them. I left `db1`, `db2`, and `db3` in, because it can be argued that none of these three necessarily replaces the other two.

One complication was in handling the graphical subsystem "XFree86". Version 4 of XFree86 was used for all client-side applications, but version 3 was used to implement X servers. The XFree86 developers completely rebuilt XFree86, and Red Hat chose to stick with the older servers and the newer clients. I looked at the XFree86 source package for version 4, and it turned out that server code was included in the package (although not used in the standard Red Hat distribution). Rather than have XFree86 counted essentially twice (once as version 3, and another as version 4), I only counted the code in version 4 of XFree86.

I then unpacked the source code by running code that in essence did this:

```
cd /usr/src/redhat/SPECS
rpm -bp *.spec
```

This uncompresses the source code and applies all patches used by the actual system. Since I wanted to count the amount of code actually included in the system, it was important to include the patches. The actual code to unpack the source code was more complex, because it also marked every unpacked directory (in the BUILD directory) to identify the spec file it came from and the license of the program. The license was determined by (1) looking at the "Copyright" and "License" fields of the spec file, and if that didn't work, (2) looking at various files in the build directory, such as "LICENSE", "COPYING*",

and "Artistic". Unfortunately, MIT-like and BSD-like licenses can be harder to detect (because their text can be varied), but many licenses (such as the GPL and LGPL) can be detected with great confidence. I used the "spec" file as the primary source, because this was placed by a human (who could better understand legal technicalities than a machine).

I actually had to repeat the unpacking more than once; the RPM system would notice a missing dependency for building the software and protest. This required installation of the missing component (in some cases I didn't have to install the program and could have forced installation, but I did not want to risk corrupting the results by failing to install a package).

A surprising development was that the packages "imap" and "samba" reported errors in unpacking. For imap, patch #5 (imap-4.7c2-flock.patch) and for samba, patch #21 (samba-ia64.patch of source/passdb/pass_check.c) would cause unpacking to halt. I unpacked the software and simply counted what was there; this appears to be what the original developers did.

I examined the reported license values, in particular for all code more than 100,000 source lines of code (as the largest components, wrong values for these components would be more likely to cause significant error). I found that Perl had been assigned "GPL" in its spec file, but this isn't the whole story; as documented in its README file, Perl can be used under either the GPL or Artistic license, so its license entry was changed to "GPL or Artistic". Mozilla's licensing situation is more complex; some portions of it are actually under a separate dual licensing scheme (licensed under both the GPL and Netscape Public License, i.e., NPL). However, labelling it as "MPL, NPL, and GPL" would probably overstate the amount of code licensed under the GPL, so I left its entry as the MPL license.

Note that the unpacked source files (including source code, fonts, documentation, and so on) totalled more than 4.4 Gigabytes.

I ran the analysis code as a normal user, so I first had to set the permissions for users to read the code. I then reverted to normal user account, and used sloccount version 1.9 to measure the source code, using the following bash command:

```
sloccount --multiproject /usr/src/redhat/BUILD > sloc-actions 2>&1 &
```

Note that I did not use the "--follow" option of sloccount. Some programs, notably pine, include a symbolic link to other directories such as /usr/lib. Thus, using --follow would have included files outside of the intended directory in the analysis.

I looked over various error reports and determined that none would fundamentally invalidate the results. For example, there were several errors in the XFree86 source code involving improperly formatted strings. It appears that these are syntax errors in the code that are preprocessed away (and thus not noticed by the compiler). I intend to report these problems to the XFree86 project. One program was a bash shell script that began with "#! /usr/bin/env bash", which sloccount's heuristics could not handle at the time. I then modified sloccount to correctly determine its type (it's a bash shell script).

Note that sloccount creates a large number of small files. This isn't fundamentally a problem, but because of the large scale of the system I found that I ran out of inodes if I tried to store multiple copies of results. Those who try to duplicate this activity may want to specially format their filesystems to include more inodes.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/develpro.html>.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the "Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[FSF 2001a] Free Software Foundation (FSF). 2001. *Various Licenses and Comments about Them*. <http://www.gnu.org/philosophy/license-list.html>.

[FSF 2001b] Free Software Foundation (FSF). 2001. *General Public License (GPL) Frequently Asked Questions (FAQ)* <http://www.gnu.org/copyleft/gpl-faq.html>.

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

[Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>

[Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf

[Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>

[Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.

[Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.

[Moody 2001] Moody, Glyn. 2001. *Rebel Code*. ISBN 0713995203.

[NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>

[OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.

[Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>

[Perens 1999] Perens, Bruce. January 1999. *Open Sources: Voices from the Open Source Revolution*. "The Open Source Definition". ISBN 1-56592-582-3. <http://www.oreilly.com/catalog/opensources/book/perens.html>

[Raymond 1999] Raymond, Eric S. January 1999. "A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.

[Schneier 2000] Schneier, Bruce. March 15, 2000. "Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...". <http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet. <http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*. http://www.dwheeler.com/oss_fs_refs.html.

[Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.

[Wheeler 2001]. Wheeler, David A. May 9, 2001 (minor update from November 6, 2000). *Estimating Linux's Size*. Version 1.04. <http://www.dwheeler.com/sloc>.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

Trademark owners own any trademarks mentioned.

This paper is (C) Copyright 2001 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. Talk to me to request republication rights. When referring to the paper, please refer to it as ``More than a Gigabuck: Estimating GNU/Linux's Size'' by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

More Than a Gigabuck: Estimating GNU/Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

June 21, 2001

Version 1.02

This paper analyzes the amount of source code in GNU/Linux, using Red Hat Linux 7.1 as a representative GNU/Linux distribution, and presents what I believe are interesting results.

In particular, it would cost over \$1 billion (\$1,000 million - a Gigabuck) to develop this Linux distribution by conventional proprietary means in the U.S. (in year 2000 U.S. dollars). Compare this to the \$600 million estimate for Red Hat Linux version 6.2 (which had been released about one year earlier). Also, Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2. Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2). Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is due to an increased number of mature and maturing open source / free software programs available worldwide.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), the X Window system (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part or as an alternative, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is public domain.

This paper is an update of my previous paper on estimating GNU/Linux's size, which measured Red Hat Linux 6.2 [Wheeler 2001]. Since Red Hat Linux 6.2 was released in March 2000, and Red Hat Linux 7.1 was released in April 2001, this paper shows what's changed over approximately one year. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The GNU/Linux operating system (also called simply ``Linux'') has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a ``typical'' intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally ``open source software'' and/or ``free software''. A program that is ``open source software'' or ``free software'' is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of ``open source software'' is available from the Open Source Initiative [OSI 1999], a more formal definition of ``free software'' (as the term is used in this paper) is available from the Free Software Foundation [FSF 2000], and other general information about these topics is available at Wheeler [2000a]. Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is ``open source software''/``free software'', and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular

vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. Microsoft unintentionally published some analysis data in the documents usually called ``Halloween I'' and ``Halloween II'' [[Halloween I](#)] [[Halloween II](#)]. Another study focused on the Linux kernel and its growth over time is by [Godfrey \[2000\]](#); this is an interesting study but it focuses solely on the Linux kernel (not the entire operating system). In a previous paper, I examined Red Hat Linux 6.2 and the numbers from the Halloween papers [[Wheeler 2001](#)].

This paper updates my previous paper, showing estimates of the size of one of today's GNU/Linux distributions, and it estimates how much it would cost to rebuild this typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean. I have intentionally written this paper so that you do *not* need to read the previous version of this paper first.

For my purposes, I have selected as my ``representative'' Linux distribution Red Hat Linux version 7.1. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [[Shankland 2000b](#)]. Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE (a German distributor) at 15%. Not all Linux copies are ``sold'' in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size for the same kind of functionality.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate ``all'' distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 7.1 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Note that Red Hat Linux 6.2 was released on March 2000, Red Hat Linux 7 was released on September 2000 (I have not counted its code), and Red Hat Linux 7.1 was released on April 2001. Thus, the differences between Red Hat Linux 7.1 and 6.2 show differences accrued over 13 months (approximately one year).

Clearly there is far more open source / free software available worldwide than is counted in this paper. However, the job of a distributor is to examine these various options and select software that they believe is both sufficiently mature and useful to their target market. Thus, examining a particular distribution results in a selective analysis of such software.

Section 2 briefly describes the approach used to estimate the ``size'' of this distribution (more details are in Appendix A). Section 3 discusses some of the results. Section 4 presents conclusions, followed by an appendix.

2. Approach

My basic approach was to:

1. install the source code files in uncompressed format; this requires carefully selecting the source code to be analyzed.
2. count the number of source lines of code (SLOC); this requires a careful definition of SLOC.
3. use an estimation model to estimate the effort and cost of developing the same system in a proprietary manner; this requires an estimation model.
4. determine the software licenses of each component and develop statistics based on these categories.

More detail on this approach is described in Appendix A. A few summary points are worth mentioning here, however.

2.1 Selecting Source Code

I included all software provided in the Red Hat distribution, but note that Red Hat no longer includes software packages that only apply to other CPU architectures (and thus packages not applying to the x86 family were excluded). I did not include ``old" versions of software, or ``beta" software where non-beta was available. I did include ``beta" software where there was no alternative, because some developers don't remove the ``beta" label even when it's widely used and perceived to be reliable.

I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once (as a tie-breaker, such files are assigned to the first build package it applies to in alphabetic order).

The code in makefiles and Red Hat Package Manager (RPM) specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

2.2 Defining SLOC

The ``physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: ``a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the ``logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the ``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

Note that this required that every file be categorized by language type (so that the correct syntax for comments, strings, and so on could be applied). Also, automatically generated files had to be detected and ignored. Thankfully, my tool ``sloccount" does this automatically.

2.3 Estimation Models

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer ``COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions. Of particular note, basic COCOMO does not include the time to develop translations to other human languages (of documentation, data, and program messages) nor fonts.

There is reason to believe that these models, while imperfect, are still valid for estimating effort in open source / free software projects. Although many open source programs don't need management of human resources, they still require technical management, infrastructure maintenance, and so on. Design documentation is captured less formally in open source projects, but it's often captured by necessity because open source projects tend to have many developers separated geographically. Clearly, the systems must still be programmed. Testing is still done, although as with many of today's proprietary programs, a good deal of testing is done through alpha and beta releases. In addition, quality is enhanced in many

open source projects through peer review of submitted code. The estimates may be lower than the actual values because they don't include estimates of human language translations and fonts.

Each software source code package, once uncompressed, produced zero or more "build directories" of source code. Some packages do not actually contain source code (e.g., they only contain configuration information), and some packages are collections of multiple separate pieces (each in different build directories), but in most cases each package uncompresses into a single build directory containing the source code for that package. Each build directory had its effort estimation computed separately; the efforts of each were then totalled. This approach assumes that each build directory was developed essentially separately from the others, which in nearly all cases is quite accurate. This approach slightly underestimates the actual effort in the rare cases where the development of the code in separate build directories are actually highly interrelated; this effect is not expected to invalidate the overall results.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [DSMC]. This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired. These are the same values as used in my last report.

2.4 Determining Software Licenses

A software license determines how that software can be used and reused, and open source software licensing has been a subject of great debate. Well-known open source licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD license, and the Artistic license. The GPL and LGPL are termed "copylefting" licenses, that is, the license is designed to prevent the code from becoming proprietary. See [Perens \[1999\]](#) for more information. Obvious questions include "what license(s) are developers choosing when they release their software" and "how much code has been released under the various licenses?"

An approximation of the amount of software using various licenses can be found for this particular distribution. Red Hat Linux uses the Red Hat Package Manager (RPM), and RPM supports capturing license data for each package (these are the "Copyright" and "License" fields in the specification file). I used this information to determine how much code was covered by each license. Since this field is simply a string of text, there were some variances in the data that I had to clean up, for example, some entries said "GNU" while most said "GPL". In some cases Red Hat did not include licensing information with a package. In that case, I wrote a program to attempt to determine the license by looking for certain conventional filenames and contents.

This is an imperfect approach. Some packages contain different pieces of code with different licenses applying to different pieces. Some packages are "dual licensed", that is, they are released under more than one license. Sometimes these other licenses are noted, while at other times they aren't. There are actually two BSD licenses (the "old" and "new" licenses), but the specification files don't distinguish between them. Also, if the license wasn't one of a small set of common licenses, Red Hat tended to assign nondescriptive phrases such as "distributable". My automated techniques were limited too, in particular, while some licenses (e.g., the GPL and LGPL) are easy to recognize automatically, BSD-like and MIT-like licenses vary the license text and so are more difficult to recognize automatically (and some changes to the license would render them non-open source, non-free software). Thus, when Red Hat did not identify a package's license, a program dual licensed under both the BSD and GPL license might only be labelled as having the GPL using these techniques. Nevertheless, this approach is sufficient to give some insight into the amount of software using various licenses. Future research could examine each license in turn and categorize them; such research might require several lawyers to determine when two licenses in certain circumstances are "equal."

One program worth mentioning in this context is Python, which has had several different licenses. Version 1.6 and later (through 2.1) had more complex licenses that the Free Software Foundation (FSF) believes were incompatible with the GPL. Recently this was resolved by another change to the Python license to make Python fully compatible with the GPL. Red Hat Linux 7.1 includes an older version of Python (1.5.2), presumably because of these licensing issues. It can't be because Red Hat is unaware of later versions of Python; Red Hat uses Python in its installation program (which it developed and maintains). Hopefully, the recent resolution of license incompatibilities with the GPL license will enable Red Hat to include the latest versions of Python in the future. In any case, there are several different Python-specific licenses, all of which can legitimately be called the "Python" license. Red Hat has labelled Python itself as having a "Distributable" license, and package Distutils-1.0.1 is labelled with the "Python" license; these labels are kept in this paper.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (sorted by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total counts grouped by their software licenses, section 3.6 presents total SLOC counts, and section 3.7 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 35 largest components (as measured by number of source lines of code), along with their licenses (see section 2.4 for how these license values were determined). In the language section, ``ansic" means C code, ``asm" is assembly, ``sh" is Bourne shell and related shells, and ``cpp" is C++.

SLOC	Directory	SLOC-by-Language (Sorted)
2437470	kernel-2.4.2	ansic=2285657,asm=144411,sh=3035,perl=2022,yacc=1147, tcl=576,lex=302,awk=248,sed=72 [GPL]
2065224	mozilla	cpp=1279902,ansic=739470,perl=21220,sh=13717,asm=5212, java=3107,yacc=1831,lex=470,csch=271,sed=24 [MPL]
1837608	XFree86-4.0.3	ansic=1750460,asm=35397,cpp=20725,sh=14666,tcl=9182, yacc=3360,perl=1675,lex=1608,awk=393,csch=85,sed=57 [MIT]
984076	gcc-2.96-20000731	ansic=789901,cpp=126738,yacc=19272,sh=17993,asm=14559, lisp=7161,fortran=3814,exp=3705,objc=479,sed=310,perl=144 [GPL]
967263	gdb+dejagnu-20010316	ansic=871288,exp=58422,sh=12054,cpp=8252,yacc=5906, asm=5031,tcl=4477,lisp=1403,sed=248,awk=170,java=7,fortran=5 [GPL]
690983	binutils-2.10.91.0.2	ansic=489993,asm=161236,exp=13234,sh=12835, yacc=5665,cpp=4777,lex=1488,perl=776,sed=561,lisp=394,awk=24 [GPL]
646692	glibc-2.2.2	ansic=548722,asm=88413,sh=6036,perl=2120,awk=1037, yacc=315,sed=49 [LGPL]
627626	emacs-20.7	lisp=453898,ansic=169956,sh=2622,perl=884,asm=253, csch=9,sed=4 [GPL]
474829	LAPACK	fortran=473590,ansic=1239 [Freely distributable]
455980	gimp-1.2.1	ansic=427967,perl=17482,lisp=9648,yacc=502,sh=381 [GPL, LGPL]
402799	mysql-3.23.36	ansic=249350,cpp=84068,perl=25088,tcl=18980,sh=18323, asm=3987,awk=1436,java=1149,sed=418 [LGPL]
395194	tcltk-8.3.1	ansic=291457,tcl=84322,sh=12259,exp=5742,yacc=876, awk=273,perl=265 [BSD]
345949	kdebase-2.1.1	cpp=181210,ansic=158682,sh=4880,perl=1155,python=22 [GPL]
323730	Mesa-3.4	ansic=286437,cpp=18189,asm=10002,sh=7611,objc=1184, python=307 [GPL/MIT]
321123	perl-5.6.0	perl=146755,ansic=118233,sh=49377,lisp=5739,yacc=996, java=23 [Artistic or GPL]
318430	libgcj	ansic=191432,cpp=56843,java=41716,sh=15581,asm=11262,

		exp=841,perl=731,awk=24 [GPL]
304819	teTeX-1.0	ansic=223491,perl=49789,sh=17634,cpp=9407,pascal=1546, yacc=1507,awk=622,lex=323,sed=314,asm=139,csch=47 [Distributable]
298742	qt-2.3.0	cpp=259310,ansic=34578,yacc=2444,sh=1493,lex=480, perl=422,lisp=15 [GPL]
286113	postgresql-7.0.3	ansic=237184,java=17540,yacc=9740,sh=8975,tcl=7751, lex=1810,perl=1276,python=959,cpp=801,asm=70,csch=5,sed=2 [BSD]
283785	kdelibs-2.1.1	cpp=261334,ansic=17578,sh=1887,java=1538,perl=731, yacc=607,lex=110 [LGPL]
277502	xemacs-21.1.14	ansic=199927,lisp=73366,sh=2948,perl=930,asm=247, csch=62,sed=22 [GPL]
264528	gs5.50	ansic=259471,cpp=2266,asm=968,sh=823,lisp=405,perl=336, yacc=201,lex=58 [GPL]
227354	krb5-1.2.2	ansic=197886,exp=19124,sh=5140,yacc=2474,perl=1529, awk=393,python=348,lex=190,csch=147,sed=123 [MIT]
215473	vnc_unixsrc	ansic=212766,cpp=848,asm=780,perl=648,sh=431 [GPL]
213818	koffice-2.0.1	cpp=197637,sh=7296,yacc=3791,ansic=3213,perl=1801, lex=80 [GPL]
202842	openssl-0.9.6	ansic=131874,cpp=25744,perl=14737,asm=12428,python=10171, yacc=3297,sh=2641,tcl=1583,lisp=224,objc=143 [BSD-like]
200908	Python-1.5.2	python=101017,ansic=96521,lisp=2353,sh=673,perl=342, sed=2 [Distributable]
194799	bind-9.1.0	ansic=173830,sh=12101,yacc=6025,perl=2830,tcl=13 [BSD-like]
192394	xpdf-0.92	cpp=167135,ansic=21621,sh=3638 [GPL]
191379	php-4.0.4pl1	ansic=173334,cpp=7033,sh=6591,lex=1867,yacc=1569, java=437,awk=367,perl=181 [PHP]
190950	pine4.33	ansic=190020,sh=838,csch=62,perl=30 [Freely distributable]
173492	abi	cpp=159595,ansic=12605,perl=725,sh=550,python=17 [GPL]
167663	kdemultimedia-2.1.1	cpp=140731,ansic=23844,tcl=1004,sh=800,asm=598, lex=578,perl=106,awk=2 [GPL]
163449	4Suite-0.10.1	python=91445,ansic=72004 [Apache-like]
159301	linuxconf-1.24r2	cpp=142970,perl=6738,sh=3821,java=3074,ansic=2613, python=85 [GPL]

Note that the operating system kernel (Linux) is the largest single component, at over 2.4 million lines of code (mostly in C); that compares to 1.5 million lines of code in Red Hat 6.2. See section 3.2 for a more detailed discussion about the Linux kernel.

The next largest component is Mozilla; this is large because it's really a suite of applications including a web browser, email reader, news reader, HTML editor, and so on. Mozilla is the basis for Netscape Navigator 6.0. Mozilla was not included at all

in Red Hat Linux 6.2.

The next largest component is the X Window system, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to gain functionality and size), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, the symbolic debugger, a set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). Emacs is next largest, which should not be a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system.

Note that language implementations tend to be written in themselves, particularly for their libraries. Perl's implementation is written mostly in Perl, and Python is written mostly in Python. Intriguingly, this is not true for Tcl.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the Linux kernel (at over 2.4 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 1,400,000 lines (57% of the Linux kernel) was in the ``drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of peripherals. No other subdirectory comes close to this size - the second largest is the ``arch" directory (at over 446,000 SLOC, 18% of the kernel), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is over 168,000 SLOC.

Richard Stallman and others have argued that the resulting system often called ``Linux" should instead be called ``GNU/Linux" [\[Stallman 2000\]](#). In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to ``free software" (free as in the freedom to use, modify, and redistribute software for any purpose). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called ``Linux". Using the term ``Linux" is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux" or ``GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc, gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux" and not just ``Linux." For more information on the sizes of the Linux kernel components, see http://www.dwheeler.com/sloc/redhat71-v1/kernel_sloc.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code (using the naming conventions of sloccount, the program used to count SLOC):

Language	SLOC (%)
C	21461450 (71.18%)
C++	4575907 (15.18%)
Shell (Bourne-like)	793238 (2.63%)
Lisp	722430 (2.40%)
Assembly	565536 (1.88%)
Perl	562900 (1.87%)
Fortran	493297 (1.64%)

Python	285050 (0.95%)
Tcl	213014 (0.71%)
Java	147285 (0.49%)
yacc/bison	122325 (0.41%)
Expect	103701 (0.34%)
lex/flex	41967 (0.14%)
awk/gawk	17431 (0.06%)
Objective-C	14645 (0.05%)
Ada	13200 (0.04%)
C shell	10753 (0.04%)
Pascal	4045 (0.01%)
sed	3940 (0.01%)

Here you can see that C is pre-eminent (with over 71% of the code), followed by C++, shell, LISP, assembly, Perl, Fortran, and Python. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has about 4.5 million lines of code, a very respectable showing, but is far less than C (over 21 million SLOC). Still, there's increasing use of C++ code; in the last survey, C had 80.55% and C++ had 7.51%. There is slightly less C code in the total percentage of code, most of which is being taken by C++. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

LISP continues to place very highly, far more than Perl, Python, Fortran, or Java. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 87% (627626/722430) of the LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages. Perl includes 5739 lines of LISP, and Python includes another 2353 of LISP that is directly used to support elaborate Emacs modes for program editing. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

Some may be surprised at the number of different languages, but I believe this should be considered not a weakness but a strength. This Linux distribution supports a wide number of languages, enabling developers to choose the ``best tool for the job."

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 94 different lex/flex files, and 138 yacc/bison files. Some build directories use lex/flex or yacc/bison more than once.

Other insights can be gained from the file counts. There were 352,549 files, of which 130,488 were counted source code files (ignoring duplicate files and automatically generated ones). Not included in this count were 10,807 files which contained duplicate contents, and 1,587 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 21,461,450 SLOC contained in 78,676 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code. Intriguingly enough, Red Hat Linux 6.2 had essentially the same average number of physical lines of C code.

3.5 Total Counts by License

Here are the various license types, sorted by the SLOC in the packages with those licenses (see section 2.4 for how these license values were determined):

```

15185987 (50.36%) GPL
 2498084 (8.28%) MIT
 2305001 (7.64%) LGPL
 2065224 (6.85%) MPL
 1826601 (6.06%) Distributable
 1315348 (4.36%) BSD
   907867 (3.01%) BSD-like
   766859 (2.54%) Freely distributable
   692561 (2.30%) Free
   455980 (1.51%) GPL, LGPL
   323730 (1.07%) GPL/MIT
   321123 (1.07%) Artistic or GPL
   191379 (0.63%) PHP
   173161 (0.57%) Apache-like
   161451 (0.54%) OpenLDAP
   146647 (0.49%) LGPL/GPL
   103439 (0.34%) GPL (programs), relaxed LGPL (libraries),
and public domain (docs)
 103291 (0.34%) Apache
   73650 (0.24%) W3C
   73356 (0.24%) IBM Public License
   66554 (0.22%) University of Washington's Free-Fork License
   59354 (0.20%) Public domain
   39828 (0.13%) GPL and Artistic
   31019 (0.10%) GPL or BSD
   25944 (0.09%) GPL/BSD
   20740 (0.07%) Not listed
   20722 (0.07%) MIT-like
   18353 (0.06%) GPL/LGPL
   12987 (0.04%) Distributable - most of it GPL
    8031 (0.03%) Python
    6234 (0.02%) GPL/distributable
    4894 (0.02%) Freely redistributable
    1977 (0.01%) Artistic
    1941 (0.01%) GPL (not Firmware)
     606 (0.00%) Proprietary

```

These can be grouped by totalling up SLOC for licenses containing certain key phrases:

```

16673212 (55.30%) GPL
 3029420 (10.05%) LGPL
 2842536 (9.43%) MIT
 2612681 (8.67%) distributable
 2280178 (7.56%) BSD
 2065224 (6.85%) MPL
 162793 (0.54%) public domain

```

From these numbers, you can determine that:

1. The GPL is far and away the most common license (by lines of code) of any single license. In fact, the category ``GPL" (packages with only this one license) all by itself accounts for 50.36% of the packages. By totalling the SLOC for all packages that include "GPL" in the license text, the total rises to 55%. No matter how you look at it, the GPL is the dominant single license in this distribution.
2. The next most common licenses were the MIT, LGPL, MPL, and BSD licenses (in order). This is in line with

expectations: the most well-known and well-used open source licenses are the GPL, MIT, LGPL, and BSD licenses. Although the MPL does well in terms of SLOC, there is only one program in this distribution that uses it - Mozilla. There is some use of the "Artistic" license, but its use is far less; note that papers such as [Perens \[1999\]](#) specifically recommend against using the the Artistic license due to its legal ambiguities.

3. Very little software is released as public domain software ("no copyright"). In this distribution, only 0.2% of the software is in packages labelled as "public domain" (note that the 0.54% figure above includes the "sane" package which has documentation in the public domain). There may be several factors that account for this. First, if a developer wishes to get credit for their work, this is a poor "license;" by law anyone can claim ownership of "public domain" software. Second, there may be a fear of litigation; both the MIT and BSD licenses permit essentially arbitrary use but forbid lawsuits. While licenses such as MIT's and BSD's are not proof against a lawsuit, they at least provide some legal protection, while releasing software to the public domain provides absolutely no protection. Finally, any software released into the public domain can be re-licensed under any other license, so there's nothing that keeps public domain software in the public domain - any of the other licenses here can "dominate" a public domain license.
4. There is a tiny amount of proprietary code, which is entirely in one component - Netscape Communicator / Navigator. This component uses the Motif toolkit (which is not open source) and has proprietary code mixed into it. As a result, almost none of the code for this package is included on the CD-ROM - only a small amount of "placeholder" code is there. In the future it is expected that this component will be replaced by Mozilla.
5. The packages which are clearly MIT-like/BSD-like licenses (totalling the MIT, BSD, MIT-like, BSD-like, and none/public domain entries) total 4,742,021 SLOC (15.92%). It's worth noting that 1,837,608 of these lines (39%) is accounted for by the XFree86 X server, an infrastructure component used for Linux's graphical user interface (GUI).
6. If the license types "distributable", "freely distributable", "MPL", "Free", "Artistic", "Apache", "Apache-like", and "IBM Public license" software was also considered MIT-like/BSD-like, the total SLOC would be 7,954,474 (26%, down from 36%). Unfortunately, the information to determine which of these other packages are simply BSD-like/MIT-like licenses is not included in the specification files.
7. The packages which include copylefting licenses (GPL or LGPL) total 63%. Limiting to only those that are GPL, LGPL, or both yields 60%, the same percentage as in Red Hat Linux 6.2 and a clear majority.

It is quite clear that in this distribution the GPL is the dominant license and that copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses. This is a simple quantitative explanation why several visible projects (Mozilla, Troll Tech's Qt, and Python) have changed their licenses so that they're compatible with the GPL. When there is so much GPL software, GPL compatibility is critically important to the survival of many open source projects. See the Free Software Foundation's information on [Various Licenses and Comments about Them \[FSF 2001a\]](#) for information on GPL compatibility, and the [GPL FAQ \[FSF 2001b\]](#) for more information on the GPL in general.

The most common open source licenses in this distribution (by SLOC) are the GPL, MIT, LGPL, and BSD licenses (as well as the MPL, but note that it's only used by one project). Note that this is consistent with [Perens \[1999\]](#), who pleads that developers use an existing license instead of developing a new license where possible.

As of this writing, the GPL has received the most attention of these licenses, because Microsoft has specifically been attacking the GPL license. The GPL license permits commercial use of the program, but requires that distributors of modified versions of the program must also release the source code to their changes under the same terms. Therefore, software released under the GPL resists Microsoft's usual "embrace and extend" approach to destroy competitors - Microsoft can use and change GPL'ed code, but it cannot make its changes to that code proprietary. As a counter-example, Kerberos (a security component released using an MIT license instead of the GPL) was recently incorporated by Microsoft into their products, and then extended in an incompatible way to prevent users from fully interoperating between products [\[Schneier 2000\]](#). Had Kerberos been released under a GPL or LGPL license, this would have been much more difficult. The presence of so many GPL and LGPL components should make Linux distributions more resistant to being "embraced, extended, and extinguished."

3.6 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 30,152,114 physical source lines of code (SLOC); I will simplify this to "over 30 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million

Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (as of 1998)	20 million
Red Hat Linux 6.2 (2000)	17 million

These numbers come from Bruce Schneier's *Crypto-Gram* [Schneier 2000], except for the Space Shuttle numbers which come from a National Academy of Sciences study [NAS 1996] and the Red Hat Linux 6.2 numbers which come from [Wheeler 2001]. Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of most of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Note that a deployed ``minimal system" would have less code; see the paper analyzing Red Hat Linux 6.2 for more discussion about this [Wheeler 2001].

Note that the Red Hat Linux 7.1 system includes a number of applications - in many cases a choice for each category. There are two major desktop environments (GNOME and KDE), plus various lightweight options. There are two word processors (Abiword and KWord), two spreadsheets (Gnumeric and KSpread), two relational database systems (MySQL and Postgres), and two web servers (Apache and TUX). In short, Red Hat Linux 7.1 includes a large number of applications, many of which are not included in its Microsoft or Sun equivalents.

At first blush, this bundling of applications with the operating system might appear similar to Microsoft's policy of combining applications with operating systems (which got Microsoft into legal trouble). However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as ``secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, this distribution (and many others) include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with smaller SLOC counts can sometimes provide greater functionality than programs with larger SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessary cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized proprietary systems.

3.7 Effort and Cost Estimates

Finally, given all the assumptions shown previously, the effort values are:

```

Total Physical Source Lines of Code (SLOC)                = 30152114
Estimated Development Effort in Person-Years (Person-Months) = 7955.75 (95469)
  (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Estimated Schedule in Years (Months)                      = 6.53 (78.31)
  (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Total Estimated Cost to Develop                            = $ 1074713481
  (average salary = $56286/year, overhead = 2.4).
```


See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2 (which had been released about one year earlier). Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2).

Had this Linux distribution been developed by conventional proprietary means, it would have cost over \$1.08 billion (1,000 million) to develop in the U.S. (in year 2000 dollars). Compare this to the \$600 million estimate for version 6.2. Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is quite extraordinary, since this represents approximately one year.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), the X Window system (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). Note that Mozilla and LAPACK were not included in Red Hat 6.2 at all.

The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed. C is still the predominant language but less so, with C++ primarily taking the difference. This appears in part to reflect many programmers' choice of C++ over C for GUI development. The increased amount of Fortran is primarily due to the inclusion of LAPACK.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is in the public domain.

Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches. Back in 1976, Bill Gates published his "Open Letter to Hobbyists", claiming that if software was freely shared it would prevent the writing of good software. He asked rhetorically, "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?" He presumed these were unanswerable questions, and both he and others based an industry on this assumption [Moody 2001]. Now, however, there are thousands of developers who are writing their own excellent code, and then giving it away. Gates was fundamentally wrong: sharing source code, and allowing others to extend it, is indeed a practical approach to developing large-scale systems - and its products can be more reliable.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), and other open source systems (such as FreeBSD). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost.

As was noted in the previous paper, some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like "Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not "reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ".m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

More information is available at <http://www.dwheeler.com/sloc>.

Appendix A. Details of Approach

This appendix discusses some of the issues I had to deal with when performing the analysis, hopefully in enough detail that someone could repeat the effort.

In particular, installing the source code required two steps:

1. install the source code files (converting source RPM packages into "spec" files and compressed source files),
2. unpack the source code files (which generates uncompressed source code, and the licensing information),

I then ran `sloccount` version 1.9 to analyze the source code files, and examined the warning messages and fixed any serious problems that arose. This was not as easy as it sounds; the previous paper (analyzing Red Hat Linux 6.2) discusses this in more detail [Wheeler 2001]. I've since released the tools I used to count code as the program `sloccount`, available at <http://www.dwheeler.com/sloccount>.

One complication should be mentioned here. Although the Red Hat Linux 7.1 package comes with a CD-ROM labelled "Source Code", the "Source Code" CD-ROM doesn't contain all the source code. Skipping the source code on the "binary" CD-ROM would produce an invalid count, because 224 source code packages are placed there (including important packages like `ssl`, `perl`, `python`, and `samba`). It's likely this was done because of CD-ROM space needs - there is so much source code that, even when compressed, it doesn't fit on one CD-ROM.

I then searched for "old" versions of programs that were also included on the CD (so that the same program wouldn't be counted twice), or those required for non-Intel x86 operation (since these would not be fully counted anyway). I did this by examining any specification (in `/usr/src/redhat/SPECS`) with "compat" or "10" or "11" in its title (it turned out all of them were old and needed removing). I also examined anything ending in a digit or "x" followed by ".spec", which located `qt1x.spec`. Through this process I removed:

```
compat-egcs.spec  compat-glibc.spec  compat-libs.spec  kde1-compat.spec
gtk+10.spec  libxml10.spec  x86-compat-libs.spec  qt1x.spec
```

I also removed any "beta" software which had a non-beta version available (beta software was identified by searching for "beta" in the package or specification file name). This removed:

```
glib-gtkbeta.spec  gtk+-gtkbeta.spec  pango-gtkbeta.spec
```

I also removed "mysqlclient9.spec". This specification contained the older MySQL client library version 3.23.22, as shipped with Red Hat Linux 7, for use with applications linked against it. I did include "mysql.spec", which had the code for the newer version 3.23.36 of MySQL (a relational database package).

Note that unlike Red Hat Linux 6.2, Red Hat Linux 7.1 didn't have two versions of `bash` or `ncurses`, so I didn't have to remove old versions of them. I left `db1`, `db2`, and `db3` in, because it can be argued that none of these three necessarily replaces the other two.

One complication was in handling the graphical subsystem "XFree86". Version 4 of XFree86 was used for all client-side applications, but version 3 was used to implement X servers. The XFree86 developers completely rebuilt XFree86, and Red Hat chose to stick with the older servers and the newer clients. I looked at the XFree86 source package for version 4, and it turned out that server code was included in the package (although not used in the standard Red Hat distribution). Rather than have XFree86 counted essentially twice (once as version 3, and another as version 4), I only counted the code in version 4 of XFree86.

I then unpacked the source code by running code that in essence did this:

```
cd /usr/src/redhat/SPECS
rpm -bp *.spec
```

This uncompresses the source code and applies all patches used by the actual system. Since I wanted to count the amount of code actually included in the system, it was important to include the patches. The actual code to unpack the source code was more complex, because it also marked every unpacked directory (in the `BUILD` directory) to identify the spec file it came

from and the license of the program. The license was determined by (1) looking at the "Copyright" and "License" fields of the spec file, and if that didn't work, (2) looking at various files in the build directory, such as "LICENSE", "COPYING*", and "Artistic". Unfortunately, MIT-like and BSD-like licenses can be harder to detect (because their text can be varied), but many licenses (such as the GPL and LGPL) can be detected with great confidence. I used the "spec" file as the primary source, because this was placed by a human (who could better understand legal technicalities than a machine).

I actually had to repeat the unpacking more than once; the RPM system would notice a missing dependency for building the software and protest. This required installation of the missing component (in some cases I didn't have to install the program and could have forced installation, but I did not want to risk corrupting the results by failing to install a package).

A surprising development was that the packages "imap" and "samba" reported errors in unpacking. For imap, patch #5 (imap-4.7c2-flock.patch) and for samba, patch #21 (samba-ia64.patch of source/passdb/pass_check.c) would cause unpacking to halt. I unpacked the software and simply counted what was there; this appears to be what the original developers did.

I examined the reported license values, in particular for all code more than 100,000 source lines of code (as the largest components, wrong values for these components would be more likely to cause significant error). I found that Perl had been assigned "GPL" in its spec file, but this isn't the whole story; as documented in its README file, Perl can be used under either the GPL or Artistic license, so its license entry was changed to "GPL or Artistic". Mozilla's licensing situation is more complex; some portions of it are actually under a separate dual licensing scheme (licensed under both the GPL and Netscape Public License, i.e., NPL). However, labelling it as "MPL, NPL, and GPL" would probably overstate the amount of code licensed under the GPL, so I left its entry as the MPL license.

Note that the unpacked source files (including source code, fonts, documentation, and so on) totalled more than 4.4 Gigabytes.

I ran the analysis code as a normal user, so I first had to set the permissions for users to read the code. I then reverted to normal user account, and used sloccount version 1.9 to measure the source code, using the following bash command:

```
sloccount --multiproject /usr/src/redhat/BUILD > sloc-actions 2>&1 &
```

Note that I did not use the "--follow" option of sloccount. Some programs, notably pine, include a symbolic link to other directories such as /usr/lib. Thus, using --follow would have included files outside of the intended directory in the analysis.

I looked over various error reports and determined that none would fundamentally invalidate the results. For example, there were several errors in the XFree86 source code involving improperly formatted strings. It appears that these are syntax errors in the code that are preprocessed away (and thus not noticed by the compiler). I intend to report these problems to the XFree86 project. One program was a bash shell script that began with "#! /usr/bin/env bash", which sloccount's heuristics could not handle at the time. I then modified sloccount to correctly determine its type (it's a bash shell script).

Note that sloccount creates a large number of small files. This isn't fundamentally a problem, but because of the large scale of the system I found that I ran out of inodes if I tried to store multiple copies of results. Those who try to duplicate this activity may want to specially format their filesystems to include more inodes.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/develpro.html>.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the "Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[FSF 2001a] Free Software Foundation (FSF). 2001. *Various Licenses and Comments about Them*. <http://www.gnu.org/philosophy/license-list.html>.

[FSF 2001b] Free Software Foundation (FSF). 2001. *General Public License (GPL) Frequently Asked Questions (FAQ)*

<http://www.gnu.org/copyleft/gpl-faq.html>.

[Godfrey 2000] Godfrey, Michael W., and Qiang Tu. Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo. ``Evolution in Open Source Software: A Case Study." *2000 Intl Conference on Software Maintenance*. <http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf>

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

[Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>

[Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf

[Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>

[Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.

[Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.

[Moody 2001] Moody, Glyn. 2001. *Rebel Code*. ISBN 0713995203.

[NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>

[OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.

[Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>

[Perens 1999] Perens, Bruce. January 1999. *Open Sources: Voices from the Open Source Revolution*. "The Open Source Definition". ISBN 1-56592-582-3. <http://www.oreilly.com/catalog/opensources/book/perens.html>

[Raymond 1999] Raymond, Eric S. January 1999. ``A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.

[Schneier 2000] Schneier, Bruce. March 15, 2000. ``Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...". <http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet. <http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*. http://www.dwheeler.com/oss_fs_refs.html.

[Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.

[Wheeler 2001]. Wheeler, David A. May 9, 2001 (minor update from November 6, 2000). *Estimating Linux's Size*. Version

1.04. <http://www.dwheeler.com/sloc>.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

Trademark owners own any trademarks mentioned.

This paper is (C) Copyright 2001 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. Talk to me to request republication rights. When referring to the paper, please refer to it as ``More than a Gigabuck: Estimating GNU/Linux's Size'' by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

More Than a Gigabuck: Estimating GNU/Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

June 21, 2001

Version 1.03

This paper analyzes the amount of source code in GNU/Linux, using Red Hat Linux 7.1 as a representative GNU/Linux distribution, and presents what I believe are interesting results.

In particular, it would cost over \$1 billion (\$1,000 million - a Gigabuck) to develop this Linux distribution by conventional proprietary means in the U.S. (in year 2000 U.S. dollars). Compare this to the \$600 million estimate for Red Hat Linux version 6.2 (which had been released about one year earlier). Also, Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2. Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2). Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is due to an increased number of mature and maturing open source / free software programs available worldwide.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), the X Window system (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part or as an alternative, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is public domain.

This paper is an update of my previous paper on estimating GNU/Linux's size, which measured Red Hat Linux 6.2 [Wheeler 2001]. Since Red Hat Linux 6.2 was released in March 2000, and Red Hat Linux 7.1 was released in April 2001, this paper shows what's changed over approximately one year. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The GNU/Linux operating system (also called simply ``Linux'') has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a ``typical'' intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally ``open source software'' and/or ``free software''. A program that is ``open source software'' or ``free software'' is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of ``open source software'' is available from the Open Source Initiative [OSI 1999], a more formal definition of ``free software'' (as the term is used in this paper) is available from the Free Software Foundation [FSF 2000], and other general information about these topics is available at Wheeler [2000a]. Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is ``open source software''/``free software'', and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular

vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. Microsoft unintentionally published some analysis data in the documents usually called "Halloween I" and "Halloween II" [[Halloween I](#)] [[Halloween II](#)]. Another study focused on the Linux kernel and its growth over time is by [Godfrey \[2000\]](#); this is an interesting study but it focuses solely on the Linux kernel (not the entire operating system). In a previous paper, I examined Red Hat Linux 6.2 and the numbers from the Halloween papers [[Wheeler 2001](#)].

This paper updates my previous paper, showing estimates of the size of one of today's GNU/Linux distributions, and it estimates how much it would cost to rebuild this typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean. I have intentionally written this paper so that you do *not* need to read the previous version of this paper first.

For my purposes, I have selected as my "representative" Linux distribution Red Hat Linux version 7.1. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [[Shankland 2000b](#)]. Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE (a German distributor) at 15%. Not all Linux copies are "sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on, or were originally developed from, a version of Red Hat Linux. This doesn't mean the other distributions are less capable, but it suggests that these other distributions are likely to have a similar set of components.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size for the same kind of functionality.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate "all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 7.1 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Note that Red Hat Linux 6.2 was released on March 2000, Red Hat Linux 7 was released on September 2000 (I have not counted its code), and Red Hat Linux 7.1 was released on April 2001. Thus, the differences between Red Hat Linux 7.1 and 6.2 show differences accrued over 13 months (approximately one year).

Clearly there is far more open source / free software available worldwide than is counted in this paper. However, the job of a distributor is to examine these various options and select software that they believe is both sufficiently mature and useful to their target market. Thus, examining a particular distribution results in a selective analysis of such software.

Section 2 briefly describes the approach used to estimate the "size" of this distribution (more details are in Appendix A). Section 3 discusses some of the results. Section 4 presents conclusions, followed by an appendix.

2. Approach

My basic approach was to:

1. install the source code files in uncompressed format; this requires carefully selecting the source code to be analyzed.
2. count the number of source lines of code (SLOC); this requires a careful definition of SLOC.
3. use an estimation model to estimate the effort and cost of developing the same system in a proprietary manner; this requires an estimation model.
4. determine the software licenses of each component and develop statistics based on these categories.

More detail on this approach is described in Appendix A. A few summary points are worth mentioning here, however.

2.1 Selecting Source Code

I included all software provided in the Red Hat distribution, but note that Red Hat no longer includes software packages that only apply to other CPU architectures (and thus packages not applying to the x86 family were excluded). I did not include ``old" versions of software, or ``beta" software where non-beta was available. I did include ``beta" software where there was no alternative, because some developers don't remove the ``beta" label even when it's widely used and perceived to be reliable.

I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once (as a tie-breaker, such files are assigned to the first build package it applies to in alphabetic order).

The code in makefiles and Red Hat Package Manager (RPM) specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

Since different languages have different syntaxes, I could only measure the SLOC for the languages that my tool (sloccount) could detect and handle. The languages sloccount could detect and handle are Ada, Assembly, awk, Bourne shell and variants, C, C++, C shell, Expect, Fortran, Java, lex/flex, LISP/Scheme, Makefile, Objective-C, Pascal, Perl, Python, sed, SQL, TCL, and Yacc/bison. Other languages are not counted; these include XUL (used in Mozilla), Javascript (also in Mozilla), PHP, and Objective Caml (an OO dialect of ML). Also code embedded in data is not counted (e.g., code embedded in HTML files). Some systems use their own built-in languages; in general code in these languages is not counted.

2.2 Defining SLOC

The ``physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: ``a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the ``logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the ``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

Note that this required that every file be categorized by language type (so that the correct syntax for comments, strings, and so on could be applied). Also, automatically generated files had to be detected and ignored. Thankfully, my tool ``sloccount" does this automatically.

2.3 Estimation Models

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer ``COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions. Of particular note, basic COCOMO does not include the time to develop translations to other human languages (of documentation, data, and program messages) nor fonts.

There is reason to believe that these models, while imperfect, are still valid for estimating effort in open source / free software projects. Although many open source programs don't need management of human resources, they still require technical management, infrastructure maintenance, and so on. Design documentation is captured less formally in open source projects, but it's often captured by necessity because open source projects tend to have many developers separated geographically. Clearly, the systems must still be programmed. Testing is still done, although as with many of today's proprietary programs, a good deal of testing is done through alpha and beta releases. In addition, quality is enhanced in many open source projects through peer review of submitted code. The estimates may be lower than the actual values because they don't include estimates of human language translations and fonts.

Each software source code package, once uncompressed, produced zero or more "build directories" of source code. Some packages do not actually contain source code (e.g., they only contain configuration information), and some packages are collections of multiple separate pieces (each in different build directories), but in most cases each package uncompresses into a single build directory containing the source code for that package. Each build directory had its effort estimation computed separately; the efforts of each were then totalled. This approach assumes that each build directory was developed essentially separately from the others, which in nearly all cases is quite accurate. This approach slightly underestimates the actual effort in the rare cases where the development of the code in separate build directories are actually highly interrelated; this effect is not expected to invalidate the overall results.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.81 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired. These are the same values as used in my last report.

2.4 Determining Software Licenses

A software license determines how that software can be used and reused, and open source software licensing has been a subject of great debate. Well-known open source licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD license, and the Artistic license. The GPL and LGPL are termed "copylefting" licenses, that is, the license is designed to prevent the code from becoming proprietary. See [Perens \[1999\]](#) for more information. Obvious questions include "what license(s) are developers choosing when they release their software" and "how much code has been released under the various licenses?"

An approximation of the amount of software using various licenses can be found for this particular distribution. Red Hat Linux uses the Red Hat Package Manager (RPM), and RPM supports capturing license data for each package (these are the "Copyright" and "License" fields in the specification file). I used this information to determine how much code was covered by each license. Since this field is simply a string of text, there were some variances in the data that I had to clean up, for example, some entries said "GNU" while most said "GPL". In some cases Red Hat did not include licensing information with a package. In that case, I wrote a program to attempt to determine the license by looking for certain conventional filenames and contents.

This is an imperfect approach. Some packages contain different pieces of code with different licenses applying to different pieces. Some packages are "dual licensed", that is, they are released under more than one license. Sometimes these other licenses are noted, while at other times they aren't. There are actually two BSD licenses (the "old" and "new" licenses), but the specification files don't distinguish between them. Also, if the license wasn't one of a small set of common licenses, Red Hat tended to assign nondescriptive phrases such as "distributable". My automated techniques were limited too, in particular, while some licenses (e.g., the GPL and LGPL) are easy to recognize automatically, BSD-like and MIT-like licenses vary the license text and so are more difficult to recognize automatically (and some changes to the license would render them non-open source, non-free software). Thus, when Red Hat did not identify a package's license, a program dual licensed under both the BSD and GPL license might only be labelled as having the GPL using these techniques. Nevertheless, this approach is sufficient to give some insight into the amount of software using various licenses. Future research could examine each license in turn and categorize them; such research might require several lawyers to determine when two licenses in certain circumstances are "equal."

One program worth mentioning in this context is Python, which has had several different licenses. Version 1.6 and later (through 2.1) had more complex licenses that the Free Software Foundation (FSF) believes were incompatible with the GPL.

Recently this was resolved by another change to the Python license to make Python fully compatible with the GPL. Red Hat Linux 7.1 includes an older version of Python (1.5.2), presumably because of these licensing issues. It can't be because Red Hat is unaware of later versions of Python; Red Hat uses Python in its installation program (which it developed and maintains). Hopefully, the recent resolution of license incompatibilities with the GPL license will enable Red Hat to include the latest versions of Python in the future. In any case, there are several different Python-specific licenses, all of which can legitimately be called the ``Python" license. Red Hat has labelled Python itself as having a ``Distributable" license, and package Distutils-1.0.1 is labelled with the ``Python" license; these labels are kept in this paper.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (sorted by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total counts grouped by their software licenses, section 3.6 presents total SLOC counts, and section 3.7 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 35 largest components (as measured by number of source lines of code), along with their licenses (see section 2.4 for how these license values were determined). In the language section, ``ansic" means C code, ``asm" is assembly, ``sh" is Bourne shell and related shells, and ``cpp" is C++.

SLOC	Directory	SLOC-by-Language (Sorted)
2437470	kernel-2.4.2	ansic=2285657,asm=144411,sh=3035,perl=2022,yacc=1147, tcl=576,lex=302,awk=248,sed=72 [GPL]
2065224	mozilla	cpp=1279902,ansic=739470,perl=21220,sh=13717,asm=5212, java=3107,yacc=1831,lex=470,csh=271,sed=24 [MPL]
1837608	XFree86-4.0.3	ansic=1750460,asm=35397,cpp=20725,sh=14666,tcl=9182, yacc=3360,perl=1675,lex=1608,awk=393,csh=85,sed=57 [MIT]
984076	gcc-2.96-20000731	ansic=789901,cpp=126738,yacc=19272,sh=17993,asm=14559, lisp=7161,fortran=3814,exp=3705,objc=479,sed=310,perl=144 [GPL]
967263	gdb+dejagnum-20010316	ansic=871288,exp=58422,sh=12054,cpp=8252,yacc=5906, asm=5031,tcl=4477,lisp=1403,sed=248,awk=170,java=7,fortran=5 [GPL]
690983	binutils-2.10.91.0.2	ansic=489993,asm=161236,exp=13234,sh=12835, yacc=5665,cpp=4777,lex=1488,perl=776,sed=561,lisp=394,awk=24 [GPL]
646692	glibc-2.2.2	ansic=548722,asm=88413,sh=6036,perl=2120,awk=1037, yacc=315,sed=49 [LGPL]
627626	emacs-20.7	lisp=453898,ansic=169956,sh=2622,perl=884,asm=253, csh=9,sed=4 [GPL]
474829	LAPACK	fortran=473590,ansic=1239 [Freely distributable]
455980	gimp-1.2.1	ansic=427967,perl=17482,lisp=9648,yacc=502,sh=381 [GPL, LGPL]
402799	mysql-3.23.36	ansic=249350,cpp=84068,perl=25088,tcl=18980,sh=18323, asm=3987,awk=1436,java=1149,sed=418 [LGPL]
395194	tcltk-8.3.1	ansic=291457,tcl=84322,sh=12259,exp=5742,yacc=876, awk=273,perl=265 [BSD]
345949	kdebase-2.1.1	cpp=181210,ansic=158682,sh=4880,perl=1155,python=22 [GPL]

323730	Mesa-3.4	ansic=286437,cpp=18189,asm=10002,sh=7611,objc=1184, python=307 [GPL/MIT]
321123	perl-5.6.0	perl=146755,ansic=118233,sh=49377,lisp=5739,yacc=996, java=23 [Artistic or GPL]
318430	libgcj	ansic=191432,cpp=56843,java=41716,sh=15581,asm=11262, exp=841,perl=731,awk=24 [GPL]
304819	teTeX-1.0	ansic=223491,perl=49789,sh=17634,cpp=9407,pascal=1546, yacc=1507,awk=622,lex=323,sed=314,asm=139,csch=47 [Distributable]
298742	qt-2.3.0	cpp=259310,ansic=34578,yacc=2444,sh=1493,lex=480, perl=422,lisp=15 [GPL]
286113	postgresql-7.0.3	ansic=237184,java=17540,yacc=9740,sh=8975,tcl=7751, lex=1810,perl=1276,python=959,cpp=801,asm=70,csch=5,sed=2 [BSD]
283785	kdelibs-2.1.1	cpp=261334,ansic=17578,sh=1887,java=1538,perl=731, yacc=607,lex=110 [LGPL]
277502	xemacs-21.1.14	ansic=199927,lisp=73366,sh=2948,perl=930,asm=247, csch=62,sed=22 [GPL]
264528	gs5.50	ansic=259471,cpp=2266,asm=968,sh=823,lisp=405,perl=336, yacc=201,lex=58 [GPL]
227354	krb5-1.2.2	ansic=197886,exp=19124,sh=5140,yacc=2474,perl=1529, awk=393,python=348,lex=190,csch=147,sed=123 [MIT]
215473	vnc_unixsrc	ansic=212766,cpp=848,asm=780,perl=648,sh=431 [GPL]
213818	koffice-2.0.1	cpp=197637,sh=7296,yacc=3791,ansic=3213,perl=1801, lex=80 [GPL]
202842	openssl-0.9.6	ansic=131874,cpp=25744,perl=14737,asm=12428,python=10171, yacc=3297,sh=2641,tcl=1583,lisp=224,objc=143 [BSD-like]
200908	Python-1.5.2	python=101017,ansic=96521,lisp=2353,sh=673,perl=342, sed=2 [Distributable]
194799	bind-9.1.0	ansic=173830,sh=12101,yacc=6025,perl=2830,tcl=13 [BSD-like]
192394	xpdf-0.92	cpp=167135,ansic=21621,sh=3638 [GPL]
191379	php-4.0.4pl1	ansic=173334,cpp=7033,sh=6591,lex=1867,yacc=1569, java=437,awk=367,perl=181 [PHP]
190950	pine4.33	ansic=190020,sh=838,csch=62,perl=30 [Freely distributable]
173492	abi	cpp=159595,ansic=12605,perl=725,sh=550,python=17 [GPL]
167663	kdemultimedia-2.1.1	cpp=140731,ansic=23844,tcl=1004,sh=800,asm=598, lex=578,perl=106,awk=2 [GPL]
163449	4Suite-0.10.1	python=91445,ansic=72004 [Apache-like]
159301	linuxconf-1.24r2	cpp=142970,perl=6738,sh=3821,java=3074,ansic=2613, python=85 [GPL]

Note that the operating system kernel (Linux) is the largest single component, at over 2.4 million lines of code (mostly in C); that compares to 1.5 million lines of code in Red Hat 6.2. See section 3.2 for a more detailed discussion about the Linux kernel.

The next largest component is Mozilla; this is large because it's really a suite of applications including a web browser, email reader, news reader, HTML editor, and so on. Mozilla is the basis for Netscape Navigator 6.0. Mozilla was not included at all in Red Hat Linux 6.2.

The next largest component is the X Window system, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to gain functionality and size), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, the symbolic debugger, a set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). Emacs is next largest, which should not be a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system.

Note that language implementations tend to be written in themselves, particularly for their libraries. Perl's implementation is written mostly in Perl, and Python is written mostly in Python. Intriguingly, this is not true for Tcl.

In many senses, what is the "largest" component is an artifact of packaging. GNOME and KDE are actually huge, but both are packaged as a set of components instead of being delivered as a single large component. The amount of C code in "kdebase" seemed suspiciously high to one KDE developer, but it turns out that Red Hat includes "lesstiflite" (a Motif clone) in kdebase, possibly to support Netscape plug-ins. My thanks to Waldo Bastian for pointing out this unusual situation and determining its cause.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the Linux kernel (at over 2.4 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 1,400,000 lines (57% of the Linux kernel) was in the "drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of peripherals. No other subdirectory comes close to this size - the second largest is the "arch" directory (at over 446,000 SLOC, 18% of the kernel), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is over 168,000 SLOC.

Richard Stallman and others have argued that the resulting system often called "Linux" should instead be called "GNU/Linux" [Stallman 2000]. In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to "free software" (free as in the freedom to use, modify, and redistribute software for any purpose). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term "GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called "Linux". Using the term "Linux" is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either "Linux" or "GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component ("Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc, gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system "GNU/Linux" and not just "Linux." For more information on the sizes of the Linux kernel components, see http://www.dwheeler.com/sloc/redhat71-v1/kernel_sloc.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code (using the naming conventions of `sloccount`, the program used to count SLOC):

Language	SLOC (%)
C	21461450 (71.18%)
C++	4575907 (15.18%)
Shell (Bourne-like)	793238 (2.63%)
Lisp	722430 (2.40%)
Assembly	565536 (1.88%)
Perl	562900 (1.87%)
Fortran	493297 (1.64%)
Python	285050 (0.95%)
Tcl	213014 (0.71%)
Java	147285 (0.49%)
yacc/bison	122325 (0.41%)
Expect	103701 (0.34%)
lex/flex	41967 (0.14%)
awk/gawk	17431 (0.06%)
Objective-C	14645 (0.05%)
Ada	13200 (0.04%)
C shell	10753 (0.04%)
Pascal	4045 (0.01%)
sed	3940 (0.01%)

Here you can see that C is pre-eminent (with over 71% of the code), followed by C++, shell, LISP, assembly, Perl, Fortran, and Python. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has about 4.5 million lines of code, a very respectable showing, but is far less than C (over 21 million SLOC). Still, there's increasing use of C++ code; in the last survey, C had 80.55% and C++ had 7.51%. There is slightly less C code in the total percentage of code, most of which is being taken by C++. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

LISP continues to place very highly, far more than Perl, Python, Fortran, or Java. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 87% (627626/722430) of the LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages. Perl includes 5739 lines of LISP, and Python includes another 2353 of LISP that is directly used to support elaborate Emacs modes for program editing. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a "control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

Some may be surprised at the number of different languages, but I believe this should be considered not a weakness but a strength. This Linux distribution supports a wide number of languages, enabling developers to choose the "best tool for the job."

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 94 different lex/flex files, and 138 yacc/bison files. Some build directories use lex/flex or yacc/bison more than once.

Other insights can be gained from the file counts. There were 352,549 files, of which 130,488 were counted source code files (ignoring duplicate files and automatically generated ones). Not included in this count were 10,807 files which contained duplicate contents, and 1,587 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 21,461,450 SLOC contained in 78,676 files, resulting in an "average" C file containing 273 (14218806/52088) physical source lines of code. Intriguingly enough, Red Hat Linux 6.2 had essentially the same average number of physical lines of C code.

3.5 Total Counts by License

Here are the various license types, sorted by the SLOC in the packages with those licenses (see section 2.4 for how these license values were determined):

```

15185987 (50.36%) GPL
 2498084 (8.28%) MIT
 2305001 (7.64%) LGPL
 2065224 (6.85%) MPL
 1826601 (6.06%) Distributable
 1315348 (4.36%) BSD
   907867 (3.01%) BSD-like
   766859 (2.54%) Freely distributable
   692561 (2.30%) Free
   455980 (1.51%) GPL, LGPL
   323730 (1.07%) GPL/MIT
   321123 (1.07%) Artistic or GPL
   191379 (0.63%) PHP
   173161 (0.57%) Apache-like
   161451 (0.54%) OpenLDAP
   146647 (0.49%) LGPL/GPL
   103439 (0.34%) GPL (programs), relaxed LGPL (libraries),
                  and public domain (docs)
  103291 (0.34%) Apache
    73650 (0.24%) W3C
    73356 (0.24%) IBM Public License
    66554 (0.22%) University of Washington's Free-Fork License
    59354 (0.20%) Public domain
    39828 (0.13%) GPL and Artistic
    31019 (0.10%) GPL or BSD
    25944 (0.09%) GPL/BSD
    20740 (0.07%) Not listed
    20722 (0.07%) MIT-like
    18353 (0.06%) GPL/LGPL
    12987 (0.04%) Distributable - most of it GPL
     8031 (0.03%) Python
     6234 (0.02%) GPL/distributable
     4894 (0.02%) Freely redistributable
     1977 (0.01%) Artistic
     1941 (0.01%) GPL (not Firmware)
      606 (0.00%) Proprietary

```


These can be grouped by totalling up SLOC for licenses containing certain key phrases:

```
16673212 (55.30%) GPL
 3029420 (10.05%) LGPL
 2842536 (9.43%) MIT
 2612681 (8.67%) distributable
 2280178 (7.56%) BSD
 2065224 (6.85%) MPL
  162793 (0.54%) public domain
```

From these numbers, you can determine that:

1. The GPL is far and away the most common license (by lines of code) of any single license. In fact, the category ``GPL" (packages with only this one license) all by itself accounts for 50.36% of the packages. By totalling the SLOC for all packages that include "GPL" in the license text, the total rises to 55%. No matter how you look at it, the GPL is the dominant single license in this distribution.
2. The next most common licenses were the LGPL, MIT, BSD, and MPL licenses (in order). This is in line with expectations: the most well-known and well-used open source licenses are the GPL, LGPL, MIT, and BSD licenses. Although the MPL does well in terms of SLOC, there is only one program in this distribution that uses it - Mozilla. There is some use of the ``Artistic" license, but its use is far less; note that papers such as [Perens \[1999\]](#) specifically recommend against using the the Artistic license due to its legal ambiguities.
3. Very little software is released as public domain software (``no copyright"). In this distribution, only 0.2% of the software is in packages labelled as ``public domain" (note that the 0.54% figure above includes the ``sane" package which has documentation in the public domain). There may be several factors that account for this. First, if a developer wishes to get credit for their work, this is a poor ``license;" by law anyone can claim ownership of ``public domain" software. Second, there may be a fear of litigation; both the MIT and BSD licenses permit essentially arbitrary use but forbid lawsuits. While licenses such as MIT's and BSD's are not proof against a lawsuit, they at least provide some legal protection, while releasing software to the public domain provides absolutely no protection. Finally, any software released into the public domain can be re-licensed under any other license, so there's nothing that keeps public domain software in the public domain - any of the other licenses here can ``dominate" a public domain license.
4. There is a tiny amount of proprietary code, which is entirely in one component - Netscape Communicator / Navigator. This component uses the Motif toolkit (which is not open source) and has proprietary code mixed into it. As a result, almost none of the code for this package is included on the CD-ROM - only a small amount of ``placeholder" code is there. In the future it is expected that this component will be replaced by Mozilla.
5. The packages which are clearly MIT-like/BSD-like licenses (totalling the MIT, BSD, MIT-like, BSD-like, and none/public domain entries) total 4,742,021 SLOC (15.92%). It's worth noting that 1,837,608 of these lines (39%) is accounted for by the XFree86 X server, an infrastructure component used for Linux's graphical user interface (GUI).
6. If the license types "distributable", "freely distributable", "MPL", "Free", "Artistic", "Apache", "Apache-like", and "IBM Public license" software was also considered MIT-like/BSD-like, the total SLOC would be 7,954,474 (26%, down from 36%). Unfortunately, the information to determine which of these other packages are simply BSD-like/MIT-like licenses is not included in the specification files.
7. The packages which include copylefting licenses (GPL or LGPL) total 63%. Limiting to only those that are GPL, LGPL, or both yields 60%, the same percentage as in Red Hat Linux 6.2 and a clear majority.

It is quite clear that in this distribution the GPL is the dominant license and that copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses. This is a simple quantitative explanation why several visible projects (Mozilla, Troll Tech's Qt, and Python) have changed their licenses so that they're compatible with the GPL. When there is so much GPL software, GPL compatibility is critically important to the survival of many open source projects. See the Free Software Foundation's information on [Various Licenses and Comments about Them \[FSF 2001a\]](#) for information on GPL compatibility, and the [GPL FAQ \[FSF 2001b\]](#) for more information on the GPL in general.

The most common open source licenses in this distribution (by SLOC) are the GPL, MIT, LGPL, and BSD licenses (as well as the MPL, but note that it's only used by one project). Note that this is consistent with [Perens \[1999\]](#), who pleads that developers use an existing license instead of developing a new license where possible.

As of this writing, the GPL has received the most attention of these licenses, because Microsoft has specifically been attacking the GPL license. The GPL license permits commercial use of the program, but requires that distributors of modified versions of the program must also release the source code to their changes under the same terms. Therefore, software released under the GPL resists Microsoft's usual ``embrace and extend" approach to destroy competitors - Microsoft can use and change GPL'ed code, but it cannot make its changes to that code proprietary. As a counter-example, Kerberos (a security

component released using an MIT license instead of the GPL) was recently incorporated by Microsoft into their products, and then extended in an incompatible way to prevent users from fully interoperating between products [\[Schneier 2000\]](#). Had Kerberos been released under a GPL or LGPL license, this would have been much more difficult. The presence of so many GPL and LGPL components should make Linux distributions more resistant to being ``embraced, extended, and extinguished."

3.6 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 30,152,114 physical source lines of code (SLOC); I will simplify this to ``over 30 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (as of 1998)	20 million
Red Hat Linux 6.2 (2000)	17 million

These numbers come from Bruce Schneier's *Crypto-Gram* [\[Schneier 2000\]](#), except for the Space Shuttle numbers which come from a National Academy of Sciences study [\[NAS 1996\]](#) and the Red Hat Linux 6.2 numbers which come from [\[Wheeler 2001\]](#). Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of most of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Note that a deployed ``minimal system" would have less code; see the paper analyzing Red Hat Linux 6.2 for more discussion about this [\[Wheeler 2001\]](#).

Note that the Red Hat Linux 7.1 system includes a number of applications - in many cases a choice for each category. There are two major desktop environments (GNOME and KDE), plus various lightweight options. There are two word processors (Abiword and KWord), two spreadsheets (Gnumeric and KSpread), two relational database systems (MySQL and Postgres), and two web servers (Apache and TUX). In short, Red Hat Linux 7.1 includes a large number of applications, many of which are not included in its Microsoft or Sun equivalents.

At first blush, this bundling of applications with the operating system might appear similar to Microsoft's policy of combining applications with operating systems (which got Microsoft into legal trouble). However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as ``secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, this distribution (and many others) include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with smaller SLOC counts can sometimes provide greater functionality than programs with larger SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessary cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer

review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized proprietary systems.

3.7 Effort and Cost Estimates

Finally, given all the assumptions shown previously, the effort values are:

Total Physical Source Lines of Code (SLOC)	= 30152114
Estimated Development Effort in Person-Years (Person-Months)	= 7955.75 (95469)
(Basic COCOMO model, Person-Months = $2.4 * (KSLOC^{**1.05})$)	
Estimated Schedule in Years (Months)	= 6.53 (78.31)
(Basic COCOMO model, Months = $2.5 * (person-months^{**0.38})$)	
Total Estimated Cost to Develop	= \$ 1074713481
(average salary = \$56286/year, overhead = 2.4).	

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux 7.1 includes over 30 million physical source lines of code (SLOC), compared to well over 17 million SLOC in version 6.2 (which had been released about one year earlier). Using the COCOMO cost model, this system is estimated to have required about 8,000 person-years of development time (as compared to 4,500 person-years to develop version 6.2).

Had this Linux distribution been developed by conventional proprietary means, it would have cost over \$1.08 billion (1,000 million) to develop in the U.S. (in year 2000 dollars). Compare this to the \$600 million estimate for version 6.2. Thus, Red Hat Linux 7.1 represents over a 60% increase in size, effort, and traditional development costs over Red Hat Linux 6.2. This is quite extraordinary, since this represents approximately one year.

This does not mean that all of the code added to the distribution in this thirteen month time period was actually written in that time period. In many cases, it represents the addition of whole new packages that have been in development for years, but have only now become sufficiently mature to include in the distribution. Also, many projects are developed over time and then released once testing is complete, and the time periods between releases can be more than a year. Still, from the user's point of view, this is a valid viewpoint - within one year of time much more functionality became available within their distribution.

Many other interesting statistics emerge. The largest components (in order) were the Linux kernel (including device drivers), Mozilla (Netscape's open source web system including a web browser, email client, and HTML editor), the X Window system (the infrastructure for the graphical user interface), gcc (a compilation system), gdb (for debugging), basic binary tools, emacs (a text editor and far more), LAPACK (a large Fortran library for numerical linear algebra), the Gimp (a bitmapped graphics editor), and MySQL (a relational database system). Note that Mozilla and LAPACK were not included in Red Hat 6.2 at all.

The languages used, sorted by the most lines of code, were C (71% - was 81%), C++ (15% - was 8%), shell (including ksh), Lisp, assembly, Perl, Fortran, Python, tcl, Java, yacc/bison, expect, lex/flex, awk, Objective-C, Ada, C shell, Pascal, and sed. C is still the predominant language but less so, with C++ primarily taking the difference. This appears in part to reflect many programmers' choice of C++ over C for GUI development. The increased amount of Fortran is primarily due to the inclusion of LAPACK.

The predominant software license is the GNU GPL. Slightly over half of the software is simply licensed using the GPL, and the software packages using the copylefting licenses (the GPL and LGPL), at least in part, accounted for 63% of the code. In all ways, the copylefting licenses (GPL and LGPL) are the dominant licenses in this Linux distribution. In contrast, only 0.2% of the software is in the public domain.

Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches. Back in 1976, Bill Gates published his "Open Letter to Hobbyists", claiming that if software was freely shared it would prevent the writing of good software. He asked rhetorically, "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?" He presumed these were unanswerable questions, and both he and others based an industry on this assumption [Moody 2001]. Now, however, there

are thousands of developers who are writing their own excellent code, and then giving it away. Gates was fundamentally wrong: sharing source code, and allowing others to extend it, is indeed a practical approach to developing large-scale systems - and its products can be more reliable.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), and other open source systems (such as FreeBSD). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost.

As was noted in the previous paper, some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like ``Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not ``reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ``.m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

More information is available at <http://www.dwheeler.com/sloc>.

Appendix A. Details of Approach

This appendix discusses some of the issues I had to deal with when performing the analysis, hopefully in enough detail that someone could repeat the effort.

In particular, installing the source code required two steps:

1. install the source code files (converting source RPM packages into "spec" files and compressed source files),
2. unpack the source code files (which generates uncompressed source code, and the licensing information),

I then ran sloccount version 1.9 to analyze the source code files, and examined the warning messages and fixed any serious problems that arose. This was not as easy as it sounds; the previous paper (analyzing Red Hat Linux 6.2) discusses this in more detail [[Wheeler 2001](#)]. I've since released the tools I used to count code as the program sloccount, available at <http://www.dwheeler.com/sloccount>.

One complication should be mentioned here. Although the Red Hat Linux 7.1 package comes with a CD-ROM labelled "Source Code", the "Source Code" CD-ROM doesn't contain all the source code. Skipping the source code on the "binary" CD-ROM would produce an invalid count, because 224 source code packages are placed there (including important packages like ssl, perl, python, and samba). It's likely this was done because of CD-ROM space needs - there is so much source code that, even when compressed, it doesn't fit on one CD-ROM.

I then searched for "old" versions of programs that were also included on the CD (so that the same program wouldn't be counted twice), or those required for non-Intel x86 operation (since these would not be fully counted anyway). I did this by examining any specification (in /usr/src/redhat/SPECS) with "compat" or "10" or "11" in its title (it turned out all of them were old and needed removing). I also examined anything ending in a digit or "x" followed by ".spec", which located qt1x.spec. Through this process I removed:

```
compat-egcs.spec  compat-glibc.spec  compat-libs.spec  kde1-compat.spec
gtk+10.spec      libxml10.spec  x86-compat-libs.spec  qt1x.spec
```

I also removed any ``beta" software which had a non-beta version available (beta software was identified by searching for ``beta" in the package or specification file name). This removed:

```
glib-gtkbeta.spec  gtk+-gtkbeta.spec  pango-gtkbeta.spec
```

I also removed "mysqlclient9.spec". This specification contained the older MySQL client library version 3.23.22, as shipped with Red Hat Linux 7, for use with applications linked against it. I did include "mysql.spec", which had the code for the newer version 3.23.36 of MySQL (a relational database package).

Note that unlike Red Hat Linux 6.2, Red Hat Linux 7.1 didn't have two versions of bash or ncurses, so I didn't have to remove old versions of them. I left db1, db2, and db3 in, because it can be argued that none of these three necessarily replaces the other two.

One complication was in handling the graphical subsystem "XFree86". Version 4 of XFree86 was used for all client-side applications, but Red Hat uses both version 3 and version 4 to implement various X servers. I looked at the XFree86 source package for version 4, and it turned out that server code was included in the package. Rather than have XFree86 counted essentially twice (once as version 3, and another as version 4), I only counted the code in version 4 of XFree86. This could be argued both ways; I understand that version 4 is a massive rewrite of much of the version 3 server, so counting it twice is actually not irrational. And unintentionally, I ended up counting a small amount of version 3 code through reuse by another program. It turns out that vnc_unixsrc includes (through reuse) portions of the X Window system version 3 code; in their words, "a cut-down version of the standard XFree86 distribution ("server only" distribution) without many of the later X extensions or hardware-specific code." VNC won't work without that code, and clearly there was effort to build version 3 and to rebuild version 4, so I let these counts stand.

I then unpacked the source code by running code that in essence did this:

```
cd /usr/src/redhat/SPECS
rpm -bp *.spec
```

This uncompresses the source code and applies all patches used by the actual system. Since I wanted to count the amount of code actually included in the system, it was important to include the patches. The actual code to unpack the source code was more complex, because it also marked every unpacked directory (in the BUILD directory) to identify the spec file it came from and the license of the program. The license was determined by (1) looking at the "Copyright" and "License" fields of the spec file, and if that didn't work, (2) looking at various files in the build directory, such as "LICENSE", "COPYING*", and "Artistic". Unfortunately, MIT-like and BSD-like licenses can be harder to detect (because their text can be varied), but many licenses (such as the GPL and LGPL) can be detected with great confidence. I used the "spec" file as the primary source, because this was placed by a human (who could better understand legal technicalities than a machine).

I actually had to repeat the unpacking more than once; the RPM system would notice a missing dependency for building the software and protest. This required installation of the missing component (in some cases I didn't have to install the program and could have forced installation, but I did not want to risk corrupting the results by failing to install a package).

A surprising development was that the packages "imap" and "samba" reported errors in unpacking. For imap, patch #5 (imap-4.7c2-flock.patch) and for samba, patch #21 (samba-ia64.patch of source/passdb/pass_check.c) would cause unpacking to halt. I unpacked the software and simply counted what was there; this appears to be what the original developers did.

I examined the reported license values, in particular for all code more than 100,000 source lines of code (as the largest components, wrong values for these components would be more likely to cause significant error). I found that Perl had been assigned "GPL" in its spec file, but this isn't the whole story; as documented in its README file, Perl can be used under either the GPL or Artistic license, so its license entry was changed to "GPL or Artistic". Mozilla's licensing situation is more complex; some portions of it are actually under a separate dual licensing scheme (licensed under both the GPL and Netscape Public License, i.e., NPL). However, labelling it as "MPL, NPL, and GPL" would probably overstate the amount of code licensed under the GPL, so I left its entry as the MPL license.

Note that the unpacked source files (including source code, fonts, documentation, and so on) totalled more than 4.4 Gigabytes.

I ran the analysis code as a normal user, so I first had to set the permissions for users to read the code. I then reverted to normal user account, and used sloccount version 1.9 to measure the source code, using the following bash command:

```
sloccount --multiproject /usr/src/redhat/BUILD > sloc-actions 2>&1 &
```

Note that I did not use the "--follow" option of sloccount. Some programs, notably pine, include a symbolic link to other directories such as /usr/lib. Thus, using --follow would have included files outside of the intended directory in the analysis.

I looked over various error reports and determined that none would fundamentally invalidate the results. For example, there were several errors in the XFree86 source code involving improperly formatted strings. It appears that these are syntax errors

in the code that are preprocessed away (and thus not noticed by the compiler). I intend to report these problems to the XFree86 project. One program was a bash shell script that began with "#! /usr/bin/env bash", which sloccount's heuristics could not handle at the time. I then modified sloccount to correctly determine its type (it's a bash shell script).

Note that sloccount creates a large number of small files. This isn't fundamentally a problem, but because of the large scale of the system I found that I ran out of inodes if I tried to store multiple copies of results. Those who try to duplicate this activity may want to specially format their filesystems to include more inodes.

For a complete list of all components and their SLOC counts, see <http://www.dwheeler.com/sloc/redhat71-v1/summary>.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/develpro.html>.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the ``Defense Acquisition Deskbook." <http://web2.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[FSF 2001a] Free Software Foundation (FSF). 2001. *Various Licenses and Comments about Them*. <http://www.gnu.org/philosophy/license-list.html>.

[FSF 2001b] Free Software Foundation (FSF). 2001. *General Public License (GPL) Frequently Asked Questions (FAQ)* <http://www.gnu.org/copyleft/gpl-faq.html>.

[Godfrey 2000] Godfrey, Michael W., and Qiang Tu. Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo. ``Evolution in Open Source Software: A Case Study." *2000 Intl Conference on Software Maintenance*. <http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf>

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

[Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>

[Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf

[Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>

[Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.

[Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.

[Moody 2001] Moody, Glyn. 2001. *Rebel Code*. ISBN 0713995203.

[NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>

[OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.

[Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report

CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>

[Perens 1999] Perens, Bruce. January 1999. *Open Sources: Voices from the Open Source Revolution*. "The Open Source Definition". ISBN 1-56592-582-3. <http://www.oreilly.com/catalog/opensources/book/perens.html>

[Raymond 1999] Raymond, Eric S. January 1999. "A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.

[Schneier 2000] Schneier, Bruce. March 15, 2000. "Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...". <http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet. <http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*. http://www.dwheeler.com/oss_fs_refs.html.

[Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.

[Wheeler 2001]. Wheeler, David A. May 9, 2001 (minor update from November 6, 2000). *Estimating Linux's Size*. Version 1.04. <http://www.dwheeler.com/sloc>.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

Trademark owners own any trademarks mentioned.

This paper is (C) Copyright 2001 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. Talk to me to request republication rights. When referring to the paper, please refer to it as "More than a Gigabuck: Estimating GNU/Linux's Size" by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

Estimating Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

July 26, 2001 (minor update from November 6, 2000)

Version 1.05

This paper presents size estimates (and their implications) of the source code of a distribution of the Linux operating system (OS), a combination often called GNU/Linux. The distribution used in this paper is Red Hat Linux version 6.2, including the kernel, software development tools, graphics interfaces, client applications, and so on. Other distributions and versions will have different sizes.

In total, this distribution includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars).

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), gcc (a compilation system), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. In this distribution the GPL is the dominant license, and copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses in terms of SLOC. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The Linux operating system (also called GNU/Linux) has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a "typical" intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally "open source software" and/or "free software" (where the "free" here means "freedom"). A program that is "open source software" or "free software" is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of "open source software" is available at OSI [1999], a more formal definition of "free software" is available at FSF [2000], and other general information about these topics is available at Wheeler [2000a]. Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is "open source software"/"free software", and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. The only published data I've found was developed by Microsoft in the documents usually called "Halloween I" and "Halloween II". Unfortunately, the meaning, derivation, and assumptions of their numbers is not explained, making the numbers hard to use and truly understand. Even worse, although the two documents were written by essentially the same people at the same time, the numbers in the documents appear (on their surface) to be contradictory. The so-called "Halloween I" document claimed that the Linux kernel (x86 only) was 500,000 lines of code, the Apache web server was 80,000 lines of code, the X-windows server was 1.5 million, and a full Linux distribution was about 10 million lines of code [Halloween I]. The "Halloween II" document seemed to contradict this, saying that "Linux" by 1998 included 1.5 million lines of code. Since "version 2.1.110" is identified as the version number,

presumably this only measures the Linux kernel, and it does note that this measure includes all Linux ports to various architectures [\[Halloween II\]](#). However, this asks as many questions as it answers - what exactly was being measured, and what assumptions were made? You could infer from these documents that the Linux kernel's support for other architectures took one million lines of code - but this appeared unlikely. Another study, [\[Dempsey 1999\]](#), did analyze open source programs, but it primarily focused on statistics about developers, and only reported information such as total file size report about the software.

This paper bridges this gap. In particular, it shows estimates of the size of Linux, and it estimates how much it would cost to rebuild a typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean.

For my purposes, I have selected as my "representative" Linux distribution Red Hat Linux version 6.2. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [\[Shankland 2000b\]](#). Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE at 15%. Not all Linux copies are "sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate "all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 6.2 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Section 2 briefly describes the approach used to estimate the "size" of this distribution (most of the details are in Appendix A). Section 3 discusses some of the results (with the details in Appendix B). Section 4 presents conclusions, followed by the two appendices.

2. Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; the steps and assumptions made are described in Appendix A.

A few summary points are worth mentioning here, however, for those who don't read appendix A. I included software for all architectures, not just the i386. I did not include "old" versions of software (with the one exception of bash, as discussed in appendix A). I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once. The code in makefiles and RPM package specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

The "physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: "a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the "logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the

``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer ``COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the ``wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total counts grouped by their software licenses, section 3.6 presents total SLOC counts, and section 3.7 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 25 largest components (as measured by number of source lines of code):

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,csh=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csh=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csh=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csh=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,

		awk=393,python=348,lex=190,csch=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	c++=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,sed=2
199982	gs5.50	ansic=195491,c++=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,c++=9407,perl=3795,pascal=1546,yacc=1507,awk=522,lex=323,sed=297,asm=139,csch=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,c++=1360,csch=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,c++=741,perl=243
138931	kdebase	c++=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csch=235,sed=35,lisp=12
133193	kaffe-1.0.5	java=65275,ansic=62125,c++=3923,perl=972,sh=814,asm=84
131372	jade-1.2.1	c++=120611,ansic=8228,sh=2150,perl=378,sed=5
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177

Note that the operating system kernel (linux) is the largest single component, at over 1.5 million lines of code (mostly in C). See section 3.2 for a more detailed discussion about the linux kernel.

The next largest component is the X windows server, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to accrete functionality), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, which is confusingly named ``egcs" instead. The naming conventions of gcc can be confusing, so a little explanation is in order. Officially, the compilation system is called ``gcc". Egcs was a project to experiment with a more open development model for gcc. Red Hat Linux 6.2 used one of the gcc releases from the egcs project, and called the release egcs-1.1.2 to avoid confusion with the official (at that time) gcc releases. The egcs experiment was a success; egcs as a separate project no longer exists, and current gcc development is based on the egcs code and development model. To sum it up, the compilation system is named ``gcc", and the version of gcc used here is a version developed by ``egcs".

Following this is the symbolic debugger and emacs. Emacs is probably not a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system. This is followed by the set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). This is followed by TCL/Tk (a combined language and widget set), PostgreSQL (a relational DBMS), and the GIMP (an excellent client application for editing bitmapped drawings).

Note that language implementations tend to be written in themselves, particularly for their libraries. Thus there is more Perl than any other single language in the Perl implementation, more Python than any other single language in Python, and more Java than any other single language in Kaffe (an implementation of the Java Virtual Machine and library).

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the linux kernel (at over 1.5 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 870,000 lines of this code was in the ``drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of hardware. The linux kernel's design is expressed in its source code directory structure, and no other directory comes close to this size - the second largest is the ``arch" directory (at over 230,000 SLOC), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is not quite 88,000 SLOC. See the appendix for more detail.

Richard Stallman and others have argued that the resulting system often called ``Linux" should instead be called ``GNU/Linux" [\[Stallman 2000\]](#). In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to ``free software" (free as in freedom). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system

kernel are both called ``Linux''. Using the term ``Linux'' is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux'' or ``GNU/Linux.'' It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux''). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc (packaged under the name ``egcs''), gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux'' and not just ``Linux.''

I also ran the CodeCount tools on the linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for the Linux kernel. When I removed all non-i86 code and re-ran the CodeCount tool on just the C code, a logical SLOC of 570,039 of C code was revealed. Since the Halloween I document reported 500,000 SLOC (when only including x86 code), it appeared very likely that the Halloween I paper counted logical SLOC (and only C code) when reporting measurements of the linux kernel. However, the other Halloween I measures appear to be physical SLOC measures: their estimate of 1.5 million SLOC for the X server is closer to the 1.2 million physical SLOC measured here, and their estimate of 80,000 SLOC for Apache is close to the 77,873 SLOC measured here (as shown in Appendix B). Note that the versions I am measuring are slightly different than the Halloween documents measured, and it is likely that some assumptions are different as well. Meanwhile, Halloween II reported a measure of 1.5 million lines of code for the Linux kernel, essentially the same value given here for physical SLOC.

Thus, it originally appeared that Halloween I used the ``logical SLOC'' measure when measuring the Linux kernel, while all other measures in Halloween I and II used physical SLOC as the measure.

I attempted to contact the Vinod Valloppillil (the author) to confirm this, and I received a reply on July 24, 2001 (long after the original version of this paper was posted). He commented that:

Actually, the way I counted was by excluding the device drivers files (device drivers share a very large % of code with each other and are therefore HIGHLY misleading w.r.t. LOC counts). The x86 vs. all archs diff is the inclusion of assembly + native machine C lang routines.

Vinod Valloppillil's concern is very valid. It's true that a number of the Linux kernel device driver files share large amounts of code with each other. In many cases, new device drivers are created by copying older code and modifying it (instead of trying to create single ``master'' files that handle all versions of software in a family). This is done intentionally; in many cases, it's difficult to find many testers with the old devices (and changing their device drivers without significant testing is risky), and doing this keeps the individual drivers simpler and more efficient.

However, while I believe this concern is valid, I don't agree with Valloppillil's approach - in fact, I believe not counting the device driver files is even more misleading. There are a vast number of different hardware devices, and one of the Linux kernel's main strengths is its support for a very large number of hardware devices. It's easily argued that the majority of the effort in kernel development was spent developing device drivers, so not counting this code is not an improvement.

In any case, this example clearly demonstrates the need to carefully identify the units of measure and assumptions made in any measurement of SLOC.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code:

ansic:	14218806	(80.55%)
c++:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)

```
python:      140725 (0.80%)
yacc:        97506 (0.55%)
java:        79656 (0.45%)
exp:         79605 (0.45%)
lex:         15334 (0.09%)
awk:         14705 (0.08%)
objc:        13619 (0.08%)
csh:         10803 (0.06%)
ada:         8217 (0.05%)
pascal:      4045 (0.02%)
sed:         2806 (0.02%)
fortran:     1707 (0.01%)
```

Here you can see that C is pre-eminent (with over 80% of the code), followed by C++, LISP, shell, and Perl. Note that the separation of Expect and TCL is somewhat artificial; if combined, they would be next (at 232115), followed by assembly. Following this in order are Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has over a million lines of code, a very respectable showing, and yet at least in this distribution it is far less than C. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

The fact that LISP places so highly (it's in third place) is a little surprising. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 80% (453647/565861) of the total amount of LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages: Perl includes 5584 lines of LISP, and Python includes another 2333 of LISP that is directly used to support elaborate Emacs modes for program editing. The ``psgml" package is solely an emacs mode for editing SGML documents. The components with the second and third largest amounts of LISP are xlipstat-3-52-17 and scheme-3.2, which are implementations of LISP and Scheme (a LISP dialect) respectively. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 57 different lex/flex files, and 110 yacc/bison files. Since some build directories use lex/flex or yacc/bison more than once, the count of build directories using these tools is smaller but still respectable: 38 different build directories use lex/flex, and 62 different build directories use yacc/bison.

Other insights can be gained from the file counts shown in appendix B. The number of source code files counted were 72,428. Not included in this count were 5,820 files which contained duplicate contents, and 817 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 14218806 SLOC contained in 52088 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code.

3.5 Total Counts by License

A software license determines how that software can be used and reused, and open source software licensing has been a subject of great debate. Well-known open source licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD license, and the Artistic license. The GPL and LGPL are termed "copylefting" licenses, that is, the license is designed to prevent the code from becoming proprietary. See [Perens \[1999\]](#) for more information. Obvious questions include "what license(s) are developers choosing when they release their software" and "how much code has been released under the various licenses?"

An approximation of the amount of software using various licenses can be found for this particular distribution. Red Hat Linux 6.2 uses the Red Hat Package Manager (RPM), and RPM supports capturing license data for each package (these are the "Copyright" and "License" fields in the specification file). I used this information to determine how much code was covered by each license. Since this field is simply a string of text, there were some variances in the data that I had to clean up, for example, some entries said "GNU" while most said "GPL".

This is an imperfect approach. Some packages contain different pieces of code with different licenses. Some packages are "dual licensed", that is, they are released under more than one license. Sometimes these other licenses are noted, while at other times they aren't. There are actually two BSD licenses (the "old" and "new" licenses), but the specification files don't distinguish between them. Also, if the license wasn't one of a small set of licenses, Red Hat tended to assign nondescriptive phrases such as "distributable". Nevertheless, this approach is sufficient to give some insight into the amount of software using various licenses. Future research could examine each license in turn and categorize them; such research might require lawyers to determine when two licenses in certain circumstances are "equal."

Here are the various license types, sorted by the SLOC in the packages with those licenses:

```
9350709 GPL
2865930 Distributable/Freely Distributable/Freeware
1927711 MIT (X)
1087757 LGPL
1060633 BSD
 383922 BSDish/Xish/MITish
278327 Miscellaneous (QPL, IBM, unknown)
273882 GPL/BSD
206237 Artistic or GPL
104721 LGPL/GPL
 62289 Artistic
 49851 None/Public Domain
   592 Proprietary (Netscape Communicator using Motif)
```

From these numbers, you can determine that:

1. The GPL is far and away the most common license (by lines of code) of any single license. In fact, the category "GPL" all by itself accounts for a simple majority of all code (53%), even when not including packages with multiple licenses (e.g., LGPL/GPL, GPL/BSD, Artistic or GPL, etc); adding these other packages would have made the total for the GPL even higher. Even if the single largest GPL component (the Linux kernel) is removed from this total, 44% of the software is specifically assigned solely to the GPL license -- and the system will not run without a kernel. No matter how you look at it, the GPL is the dominant single license in this distribution.
2. The "distributable" category comes in second. At least some of this code is released under essentially MIT/BSD-style licenses, but more precise information is not included in the RPM specification files.
3. The next most common licenses were the MIT, LGPL, and BSD licenses (in order). This is in line with expectations: the most well-known and well-used open source licenses are the GPL, MIT, LGPL, and BSD licenses. There is some use of the "Artistic" license, but its use is far less; note that papers such as [Perens \[1999\]](#) specifically recommend against using the the Artistic license due to its legal ambiguities.
4. Very little software is released as public domain software ("no copyright"). There may be several factors that account for this. First, if a developer wishes to get credit for their work, this is a poor "license;" by law anyone can claim ownership of "public domain" software. Second, there may be a fear of litigation; both the MIT and BSD licenses permit essentially arbitrary use but forbid lawsuits. While licenses such as MIT's and BSD's are not proof against a lawsuit, they at least provide some legal protection, while releasing software to the public domain provides absolutely no protection. Finally, any software released into the public domain can be re-licensed under any other license, so there's nothing that keeps public domain software in the public domain - any of the other licenses here can "dominate"

a public domain license.

5. There is a tiny amount of non-open-source code, which is entirely in one component - Netscape Communicator / Navigator. This component uses the Motif toolkit (which is not open source) and has proprietary code mixed into it. As a result, almost none of the code for this package is included on the CD-ROM - only a small amount of ``placeholder" code is there. In the future it is expected that this component will be replaced by the results of the Mozilla project.
6. The packages which are clearly MITish/BSDish licenses (totalling the MIT, BSD, BSDish, and none/public domain entries) total 3,422,117 SLOC, or 19%. It's worth noting that 1,291,745 of these lines (38%) is accounted for by the XFree86 X server, an infrastructure component used for Linux's graphical user interface (GUI). If the XFree86 X server didn't use the MIT license, the total SLOC clearly in this category (MITish/BSDish licenses) would go down to 2,130,372 SLOC (12% of the total system) -- and there are many systems which do not need or use an X server.
7. If all "distributable" and Artistic software was also considered MITish/BSDish, the total SLOC would be 6,350,336 (36%). Unfortunately, the information to determine which of these other packages are simply BSDish/Xish licenses is not included in the specification files.
8. The packages which are clearly copylefted (GPL, LGPL, LGPL/GPL) total 10,543,187 (60%) - a clear majority. Even if the largest copylefted component (the Linux kernel) was not counted as GPL'ed software (which it is), the total of copylefted software would be 51% - showing that copylefted software dominates in this distribution.

It is quite clear that in this distribution the GPL is the dominant license and that copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses. This is a simple quantitative explanation why several visible projects (Mozilla, Troll Tech's Qt, and Python) have more recently dual-licensed their software with the GPL or made other arrangements to be compatible with the GPL. When there is so much GPL software, GPL compatibility is critically important to the survival of many open source projects. The most common open source licenses in this distribution are the GPL, MIT, LGPL, and BSD licenses. Note that this is consistent with [Perens \[1999\]](#), who pleads that developers use an existing license instead of developing a new license where possible.

3.6 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 17,652,561 physical source lines of code (SLOC); I will simplify this to ``over 17 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (1998)	20 million

These numbers come from Bruce Schneier's *Crypto-Gram* [\[Schneier 2000\]](#), except for the Space Shuttle numbers which come from a National Academy of Sciences study [\[NAS 1996\]](#). Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Schneier also reports that ``Linux, even with the addition of X Windows and Apache, is still under 5 million lines of code". At first, this seems to be contradictory, since this paper counts over 17 million SLOC, but Schneier appears to be literally correct in the context of his statement. The phrasing of his sentence suggests that Schneier is considering some sort of ``minimal" system, since he considers ``even the addition of X Windows" as a significant addition. As shown in appendix section B.4, taking the minimal ``base" set of components in Red Hat Linux, and then adding the minimal set of components for graphical interaction (the X Windows's graphical server, library, configuration tool, and a graphics toolkit) and the Apache web server, the total is about 4.4 million physical SLOC - which is less than 5 million. This minimal system doesn't include some useful (but not strictly necessary) components, but a number of useful components could be added while still staying under a total of 5 million SLOC.

However, note the contrast. Many Linux distributions include with their operating systems many applications (e.g., bitmap editors) and development tools (for many different languages). As a result, the entire *delivered* system for such distributions

(including Red Hat Linux 6.2) is *much* larger than the 5 million SLOC stated by Schneier. In short, this distribution's size appears similar to the size of Windows 98 and Windows NT 5.0 in 1998.

Microsoft's recent legal battles with the U.S. Department of Justice (DoJ) also involve the bundling of applications with the operating system. However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as "secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, many Linux distributions include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with small SLOC counts can often provide greater functionality than programs with large SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessarily cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized systems.

3.7 Effort and Cost Estimates

Finally, given all the assumptions shown, are the effort values:

```
Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = $ 614421924.71
```

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux version 6.2 includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars).

Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches. Back in 1976, Bill Gates published his "Open Letter to Hobbyists", claiming that if software was freely shared it would prevent the writing of good software. He asked rhetorically, "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?" He presumed these were unanswerable questions, and both he and others based an industry on this assumption [Moody 2001]. Now, however, there are thousands of developers who are writing their own excellent code, and then giving it away. Gates was fundamentally wrong: sharing source code, and allowing others to extend it, is indeed a practical approach to developing large-scale systems - and its products can be more reliable.

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), gcc (a compilation system, with the package name of "egcs"), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including

Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Here you can see that C is pre-eminent (with over 80% of the code). In this distribution the GPL is the dominant license, and copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses in terms of SLOC. The most common open source licenses in this distribution are the GPL, MIT, LGPL, and BSD licenses. More information is available in the appendices and at <http://www.dwheeler.com/sloc>.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), other open source systems (such as FreeBSD), and other versions of Red Hat (such as Red Hat 7). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost. It's known that Red Hat 7 includes more source code; Red Hat 7 has had to add another CD-ROM to contain the binary programs, and adds such capabilities as a word processor (abiword) and secure shell (openssh).

Some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like ``Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not ``reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ``.m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

Appendix A. Details of Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; each step is described below. Some steps I describe in some detail, because it's sometimes hard to find the necessary information even when the actual steps are easy. Hopefully, this detail will make it easier for others to do similar activities or to repeat the experiment.

A.1 Installing Source Code

Installing the source code files turned out to be nontrivial. First, I inserted the CD-ROM containing all of the source files (in ``.src.rpm" format) and installed the packages (files) using:

```
mount /mnt/cdrom
cd /mnt/cdrom/SRPMS
rpm -ivh *.src.rpm
```

This installs ``spec" files and compressed source files; another rpm command (``rpm -bp") uses the spec files to uncompress the source files into ``build directories" (as well as apply any necessary patches). Unfortunately, the rpm tool does not enforce any naming consistency between the package names, the spec names, and the build directory names; for consistency this paper will use the names of the build directories, since all later tools based themselves on the build directories.

I decided to (in general) not count ``old" versions of software (usually placed there for compatibility reasons), since that

would be counting the same software more than once. Thus, the following components were not included: ``compat-binutils'', ``compat-egcs'', ``compat-glib'', ``compat-libs'', ``gtk+10'', ``libc-5.3.12'' (an old C library), ``libxml10'', ``ncurses3'', and ``qt1x''. I also didn't include egcs64-19980921 and netscape-sparc, which simply repeated something on another architecture that was available on the i386 in a different package. I did make one exception. I kept both bash-1.14.7 and bash2, two versions of the shell command processor, instead of only counting bash2. While bash2 is the later version of the shell available in the package, the main shell actually used by the Red Hat distribution was the older version of bash. The rationale for this decision appears to be backwards compatibility for older shell scripts; this is suggested by the Red Hat package documentation in both bash-1.14.7 and bash2. It seemed wrong to not include one of the most fundamental pieces of the system in the count, so I included it. At 47067 lines of code (ignoring duplicates), bash-1.14.7 is one of the smaller components anyway. Not including this older component would not substantively change the results presented here.

There are two directories, krb4-1.0 and krb5-1.1.1, which appear to violate this rule - but don't. krb5-1.1.1 is the build directory created by krb5.spec, which is in turn installed by the source RPM package krb5-1.1.1-9.src.rpm. This build directory contains Kerberos V5, a trusted-third-party authentication system. The source RPM package krb5-1.1.1-9.src.rpm eventually generates the binary RPM files krb5-configs-1.1.1-9, krb5-libs-1.1.1-9, and krb5-devel-1.1.1-9. You might guess that ``krb4-1.0'' is just the older version of Kerberos, but this build directory is created by the spec file krbafs.spec and not just an old version of the code. To quote its description, ``This is the Kerberos to AFS bridging library, built against Kerberos 5. krbafs is a shared library that allows programs to obtain AFS tokens using Kerberos IV credentials, without having to link with official AFS libraries which may not be available for a given platform." For this situation, I simply counted both packages, since their purposes are different.

I was then confronted with a fundamental question: should I count software that only works for another architecture? I was using an i86-type system, but some components are only for Alpha or Sparc systems. I decided that I should count them; even if I didn't use the code today, the ability to use these other architectures in the future was of value and certainly required effort to develop.

This caused complications for creating the build directories. If all installed packages fit the architecture, you can install the uncompressed software by typing:

```
cd /usr/src/redhat/SPECS and typing the command
rpm -bp *.spec
```

Unfortunately, the rpm tool notes that you're trying to load code for the ``wrong'' architecture, and (at least at the time) there was no simple ``override'' flag. Instead, I had to identify each package as belonging to SPARC or ALPHA, and then use the rpm option --target to forcibly load them. For example, I renamed all sparc-specific SPARC file files to end in ``.sparc'' and could then load them with:

```
rpm -bp --target sparc-redhat-linux *.spec.sparc
```

The following spec files were non-i86: (sparc) audioclt, elftoaout, ethtool, prtconf, silo, solemul, sparc32; (alpha) aboot, minlabel, quickstrip. In general, these were tools to aid in supporting some part of the boot process or for using system-specific hardware.

Note that not all packages create build directories. For example, ``anonftp'' is a package that, when installed, sets up an anonymous ftp system. This package doesn't actually install any software; it merely installs a specific configuration of another piece of software (and unsets the configuration when uninstalled). Such packages are not counted at all in this sizing estimate.

Simply loading all the source code requires a fair amount of disk space. Using ``du'' to measure the disk space requirements (with 1024 byte disk blocks), I obtained the following results:

```
$ du -s /usr/src/redhat/BUILD /usr/src/redhat/SOURCES /usr/src/redhat/SPECS
2375928 /usr/src/redhat/BUILD
592404 /usr/src/redhat/SOURCES
4592 /usr/src/redhat/SPECS
```

Thus, these three directories required 2972924 1K blocks - approximately 3 gigabytes of space. Much more space would be required to compile it all.

A.2 Categorizing Source Code

My next task was to identify all files containing source code (not including any automatically generated source code). This is a non-trivial problem; there are 181,679 ordinary files in the build directory, and I had no interest in doing this identification by hand.

In theory, one could just look at the file extensions (.c for C, .py for python), but this is not enough in practice. Some packages reuse extensions if the package doesn't use that kind of file (e.g., the ``.exp" extension of expect was used by some packages as ``export" files, and the ``.m" of objective-C was used by some packages for module information extracted from C code). Some files don't have extensions, particularly scripts. And finally, files automatically generated by another program should not be counted, since I wished to use the results to estimate effort.

I ended up writing a program of over 600 lines of Perl to perform this identification, which used a number of heuristics to categorize each file into categories. There is a category for each language, plus the categories non-programs, unknown (useful for scanning for problems), automatically generated program files, duplicate files (whose file contents duplicated other files), and zero-length files.

The program first checked for well-known extensions (such as .gif) that cannot be program files, and for a number of common generated filenames. It then peeked at the first line for "#!" followed by a legal script name. If that didn't work, it used the extension to try to determine the category. For a number of languages, the extension was not reliable, so for those languages the categorization program examined the file contents and used a set of heuristics to determine if the file actually belonged that category. If all else failed, the file was placed in the ``unknown" category for later analysis. I later looked at the ``unknown" items, checking the common extensions to ensure I had not missed any common types of code.

One complicating factor was that I wished to separate C, C++, and objective-C code, but a header file ending with ``.h" or ``.hpp" file could be any of them. I developed a number of heuristics to determine, for each file, what language it belonged to. For example, if a build directory has exactly one of these languages, determining the correct category for header files is easy. Similarly, if there is exactly one of these in the directory with the header file, it is presumed to be that kind. Finally, a header file with the keyword ``class" is almost certainly not a C header file, but a C++ header file.

Detecting automatically generated files was not easy, and it's quite conceivable I missed a number of them. The first 15 lines were examined, to determine if any of them included at the beginning of the line (after spaces and possible comment markers) one of the following phrases: ``generated automatically", ``automatically generated", ``this is a generated file", ``generated with the (something) utility", or ``do not edit". A number of filename conventions were used, too. For example, any ``configure" file is presumed to be automatically generated if there's a ``configure.in" file in the same directory.

To eliminate duplicates, the program kept md5 checksums of each program file. Any given md5 checksum would only be counted once. Build directories were processed alphabetically, so this meant that if the same file content was in both directories ``a" and ``b", it would be counted only once as being part of ``a". Thus, some packages with names later in the alphabet may appear smaller than would make sense at first glance. It is very difficult to eliminate ``almost identical" files (e.g., an older and newer version of the same code, included in two separate packages), because it is difficult to determine when ``similar" two files are essentially the ``same" file. Changes such as the use of pretty-printers and massive renaming of variables could make small changes seem large, while the many small files in the system could easily make different files seem the ``same." Thus, I did not try to make such a determination, and just considered files with different contents as different.

It's important to note that different rules could be used to ``count" lines of code. Some kinds of code were intentionally excluded from the count. Many RPM packages include a number of shell commands used to install and uninstall software; the estimate in this paper does not include the code in RPM packages. This estimate also does not include the code in Makefiles (which can be substantive). In both cases, the code in these cases is often cut and pasted from other similar files, so counting such code would probably overstate the actual development effort. In addition, Makefiles are often automatically generated.

On the other hand, this estimate does include some code that others might not count. This estimate includes test code included with the package, which isn't visible directly to users (other than hopefully higher quality of the executable program). It also includes code not used in this particular system, such as code for other architectures and OS's, bindings for languages not compiled into the binaries, and compilation-time options not chosen. I decided to include such code for two reasons. First, this code is validly represents the effort to build each component. Second, it does represent indirect value to the user, because the user can later use those components in other circumstances even if the user doesn't choose to do so by default.

So, after the work of categorizing the files, the following categories of files were created for each build directory (common extensions are shown in parentheses, and the name used in the data tables below are shown in brackets):

1. C (.c) [ansic]
2. C++ (.C, .cpp, .cxx, .cc) [cpp]
3. LISP (.el, .scm, .lsp, .jl) [lisp]
4. shell (.sh) [sh]
5. Perl (.pl, .pm, .perl) [perl]
6. Assembly (.s, .S, .asm) [asm]
7. TCL (.tcl, .tk, .itk) [tcl]

8. Python (.py) [python]
9. Yacc (.y) [yacc]
10. Java (.java) [java]
11. Expect (.exp) [exp]
12. lex (.l) [lex]
13. awk (.awk) [awk]
14. Objective-C (.m) [objc]
15. C shell (.csh) [csh]
16. Ada (.ada, .ads, .adb) [ada]
17. Pascal (.p) [pascal]
18. sed (.sed) [sed]
19. Fortran (.f) [fortran]

Note that we're counting Scheme as a dialect of LISP, and Expect is being counted separately from TCL. The command line shells Bourne shell, the Bourne-again shell (bash), and the K shell are all counted together as ``shell'', but the C shell (csh and tcsh) is counted separately.

A.3 Counting Lines of Code

Every language required its own counting scheme. This was more complex than I realized; there were a number of languages involved.

I originally tried to use USC's ``CodeCount'' tools to count the code. Unfortunately, this turned out to be buggy and did not handle most of the languages used in the system, so I eventually abandoned it for this task and wrote my own tools. Those who wish to use this tool are welcome to do so; you can learn more from its web site at <http://sunset.usc.edu/research/CODECOUNT>.

I did manage to use the CodeCount to compute the logical source lines of code for the C portions of the linux kernel. This came out to be 673,627 logical source lines of code, compared to the 1,462,165 lines of physical code (again, this ignores files with duplicate contents).

Since there were a large number of languages to count, I used the ``physical lines of code'' definition. In this definition, a line of code is a line (ending with newline or end-of-file) with at least one non-comment non-whitespace character. These are known as ``non-comment non-blank'' lines. If a line only had whitespace (tabs and spaces) it was not counted, even if it was in the middle of a data value (e.g., a multiline string). It is much easier to write programs to measure this value than to measure the ``logical'' lines of code, and this measure can be easily applied to widely different languages. Since I had to process a large number of different languages, it made sense to choose the measure that is easier to obtain.

[Park \[1992\]](#) presents a framework of issues to be decided when trying to count code. Using Park's framework, here is how code was counted in this paper:

1. Statement Type: I used a physical line-of-code as my basis. I included executable statements, declarations (e.g., data structure definitions), and compiler directives (e.g., preprocessor commands such as #define). I excluded all comments and blank lines.
2. How Produced: I included all programmed code, including any files that had been modified. I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. If a file was in the source package, I included it; if the file had been removed from a source package (including via a patch), I did not include it.
3. Origin: I included all code included in the package.
4. Usage: I included code in or part of the primary product; I did not include code external to the product (i.e., additional applications able to run on the system but not included with the system).
5. Delivery: I counted code delivered as source; not surprisingly, I didn't count code not delivered as source. I also didn't count undelivered code.
6. Functionality: I included both operative and inoperative code. An examples of intentionally ``inoperative'' code is code turned off by #ifdef commands; since it could be turned on for special purposes, it made sense to count it. An examples of unintentionally ``inoperative'' code is dead or unused code.
7. Replications: I included master (original) source statements. I also included ``physical replicates of master statements

stored in the master code". This is simply code cut and pasted from one place to another to reuse code; it's hard to tell where this happens, and since it has to be maintained separately, it's fair to include this in the measure. I excluded copies inserted, instantiated, or expanded when compiling or linking, and I excluded postproduction replicates (e.g., reparameterized systems).

8. Development Status: Since I only measured code included in the packages used to build the delivered system, I declared that all software I was measuring had (by definition) passed whatever "system tests" were required by that component's developers.
9. Languages: I included all languages, as identified earlier in section A.2.
10. Clarifications: I included all statement types. This included nulls, continues, no-ops, lone semicolons, statements that instantiate generics, lone curly braces ({ and }), and labels by themselves.

Park includes in his paper a "basic definition" of physical lines of code, defined using his framework. I adhered to Park's definition unless (1) it was impossible in my technique to do so, or (2) it would appear to make the result inappropriate for use in cost estimation (using COCOMO). COCOMO states that source code:

"includes all program instructions created by project personnel and processed into machine code by some combination of preprocessors, compilers, and assemblers. It excludes comment cards and unmodified utility software. It includes job control language, format statements, and data declarations. Instructions are defined as lines of code."

In summary, though in general I followed Park's definition, I didn't follow Park's "basic definition" in the following ways:

1. How Produced: I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. After all, COCOMO states that the only code that should be counted is code "produced by project personnel", whereas these kinds of files are instead the output of "preprocessors and compilers." If code is always maintained as the input to a code generator, and then the code generator is re-run, it's only the code generator input's size that validly measures the size of what is maintained. Note that while I attempted to exclude generated code, this exclusion is based on heuristics which may have missed some cases.
2. Origin: Normally physical SLOC doesn't include an unmodified "vendor-supplied language support library" nor a "vendor-supplied system or utility". However, in this case this was *exactly* what I was measuring, so I naturally included these as well.
3. Delivery: I didn't count code not delivered as source. After all, since I didn't have it, I couldn't count it.
4. Functionality: I included unintentionally inoperative code (e.g., dead or unused code). There might be such code, but it is very difficult to automatically detect in general for many languages. For example, a program not directly invoked by anything else nor installed by the installer is much more likely to be a test program, which I'm including in the count. Clearly, discerning human "intent" is hard to automate. Hopefully, unintentionally inoperative code is a small amount of the total delivered code.

Otherwise, I followed Park's "basic definition" of a physical line of code, even down to Park's language-specific definitions where Park defined them for a language.

One annoying problem was that one file wasn't syntactically correct and it affected the count. File `/usr/src/redhat/BUILD/cdrecord-1.8/mkiso` had an `#ifdef` not taken, and the road not taken had a missing double-quote mark before the word "cannot":

```
#ifdef  USE_LIBSCHILY
        comerr(Cannot open '%s'.\n", filename);
#endif
        perror ("fopen");
        exit (1);
#endif
```

I solved this by hand-patching the source code (for purposes of counting). There were also some files with intentionally erroneous code (e.g., compiler error tests), but these did not impact the SLOC count.

Several languages turn out to be non-trivial to count:

- In C, C++, and Java, comment markers should be ignored inside strings. Since they have multi-line comment markers this requirement should not be ignored, or a "/*" inside a string could cause most of the code to be erroneously uncounted.
- Officially, C doesn't have C++'s "/*" comment marker, but the gcc compiler accepts it and a great deal of C code uses it, so my counters accepted it.
- Perl permits in-line "perlpod" documents, "here" documents, and an `__END__` marker that complicate code-counting.

Perlpod documents are essentially comments, but a ``here" document may include text to generate them (in which case the perlpod document is data and should be counted). The `__END__` marker indicates the end of the file from Perl's viewpoint, even if there's more text afterwards.

- Python has a convention that, at the beginning of a definition (e.g., of a function, method, or class), an unassigned string can be placed to describe what's being defined. Since this is essentially a comment (though it doesn't syntactically look like one), the counter must avoid counting such strings, which may have multiple lines. To handle this, strings which started the beginning of a line were not counted. Python also has the ``triple quote" operator, permitting multiline strings; these needed to be handled specially. Triple quote strings were normally considered as data, regardless of content, unless they were used as a comment about a definition.
- Assembly languages vary greatly in the comment character they use, so my counter had to handle this variance. I wrote a program which first examined the file to determine if C-style ```/*"` comments and C preprocessor commands (e.g., ```#include"`) were used. If both ```/*"` and ```*/*"` were in the file, it was assumed that C-style comments were used, since it is unlikely that *both* would be used as something else (e.g., as string data) in the same assembly language file. Determining if a file used the C preprocessor was trickier, since many assembly files do use ```#"` as a comment character and some preprocessor directives are ordinary words that might be included in a human comment. The heuristic used was: if `#ifdef`, `#endif`, or `#include` are used, the preprocessor is used; if at least three lines have either `#define` or `#else`, then the preprocessor is used. No doubt other heuristics are possible, but this at least seemed to produce reasonable results. The program then determined what the comment character was, by identifying which punctuation mark (from a set of possible marks) was the most common non-space initial character on a line (ignoring ```/"` and ```#"` if C comments or preprocessor commands, respectively, were used). Once the comment character had been determined, and it had been determined if C-style comments were also allowed, the lines of code could be counted in the file.

Although their values are not used in estimating effort, I also counted the number of files; summaries of these values are included in appendix B.

Since the Linux kernel was the largest single component, and I had questions about the various inconsistencies in the ``Halloween" documents, I made additional measures of the Linux kernel.

Some have objected because the counting approach used here includes lines not compiled into code in this Linux distribution. However, the primary objective of these measures was to estimate total effort to develop all of these components. Even if some lines are not normally enabled on Linux, it still required effort to develop that code. Code for other architectures still has value, for example, because it enables users to port to other architectures while using the component. Even if that code is no longer being maintained (e.g., because the architecture has become less popular), nevertheless someone had to invest effort to create it, the results benefitted someone, and if it is needed again it's still there (at least for use as a starting point). Code that is only enabled by compile-time options still has value, because if the options were desired the user could enable them and recompile. Code that is only used for testing still has value, because its use improves the quality of the software directly run by users. It is possible that there is some ``dead code" (code that cannot be run under any circumstance), but it is expected that this amount of code is very small and would not significantly affect the results. Andi Kleen (of SuSE) noted that if you wanted to only count compiled and running code, one technique (for some languages) would be to use gcc's ```-g"` option and use the resulting .stabs debugging information with some filtering (to exclude duplicated inline functions). I determined this to be out-of-scope for this paper, but this approach could be used to make additional measurements of the system.

A.4 Estimating Effort and Costs

For each build directory, I totalled the source lines of code (SLOC) for each language, then totalled those values to determine the SLOC for each directory. I then used the basic Constructive Cost Model (COCOMO) to estimate effort. The basic model is the simplest (and least accurate) model, but I simply did not have the additional information necessary to use the more complex (and more accurate) models. COCOMO is described in depth by Boehm [1981].

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions.

In the basic COCOMO model, estimated man-months of effort, design through test, equals $2.4 \cdot (\text{KSLOC})^{1.05}$, where KSLOC is the total physical SLOC divided by 1000.

I assumed that each package was built completely independently and that there were no efforts necessary for

integration not represented in the code itself. This almost certainly underestimates the true costs, but for most packages it's actually true (many packages don't interact with each other at all). I wished to underestimate (instead of overestimate) the effort and costs, and having no better model, I assumed the simplest possible integration effort. This meant that I applied the model to each component, then summed the results, as opposed to applying the model once to the grand total of all software.

Note that the only input to this model is source lines of code, so some factors simply aren't captured. For example, creating some kinds of data (such as fonts) can be very time-consuming, but this isn't directly captured by this model. Some programs are intentionally designed to be data-driven, that is, they're designed as small programs which are driven by specialized data. Again, this data may be as complex to develop as code, but this is not counted.

Another example of uncaptured factors is the difficulty of writing kernel code. It's generally acknowledged that writing kernel-level code is more difficult than most other kinds of code, because this kind of code is subject to a subtle timing and race conditions, hardware interactions, a small stack, and none of the normal error protections. In this paper I do not attempt to account for this. You could try to use the Intermediate COCOMO model to try to account for this, but again this requires knowledge of other factors that can only be guessed at. Again, the effort estimation probably significantly underestimates the actual effort represented here.

It's worth noting that there is an update to COCOMO, COCOMO II. However, COCOMO II requires as its input logical (not physical) SLOC, and since this measure is much harder to obtain, I did not pursue it for this paper. More information about COCOMO II is available at the web site <http://sunset.usc.edu/research/COCOMOII/index.html>. A nice overview paper where you can learn more about software metrics is Masse [1997].

I assumed that an average U.S. programmer/analyst salary in the year 2000 was \$56,286 per year; this value was from the ComputerWorld, September 4, 2000's Salary Survey. Overhead is much harder to estimate; I did not find a definitive source for information on overheads. After informal discussions with several cost analysts, I determined that an overhead of 2.4 would be representative of the overhead sustained by a typical software development company. Should you disagree with these figures, I've provided all the information necessary to recalculate your own cost figures; just start with the effort estimates and recalculate cost yourself.

Appendix B. More Detailed Results

This appendix provides some more detailed results. B.1 lists the SLOC found in each build directory; B.2 shows counts of files for each category of file; B.3 presents some additional measures about the Linux kernel. B.4 presents some SLOC totals of putatively "minimal" systems. You can learn more at <http://www.dwheeler.com/sloc>.

B.1 SLOC in Build Directories

The following is a list of all build directories, and the source lines of code (SLOC) found in them, followed by a few statistics counting files (instead of SLOC).

Remember that duplicate files are only counted once, with the build directory "first in ASCII sort order" receiving any duplicates (to break ties). As a result, some build directories have a smaller number than might at first make sense. For example, the "kudzu" build directory does contain code, but all of it is also contained in the "Xconfigurator" build directory.. and since that directory sorts first, the kudzu package is considered to have "no code".

The columns are SLOC (total physical source lines of code), Directory (the name of the build directory, usually the same or similar to the package name), and SLOC-by-Language (Sorted). This last column lists languages by name and the number of SLOC in that language; zeros are not shown, and the list is sorted from largest to smallest in that build directory. Similarly, the directories are sorted from largest to smallest total SLOC.

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,cs=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5

625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csch=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csch=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csch=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,csch=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,sed=2
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,yacc=1507,awk=522,lex=323,sed=297,asm=139,csch=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,csch=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csch=235,sed=35,lisp=12
133193	kaffe-1.0.5	java=65275,ansic=62125,cpp=3923,perl=972,sh=814,asm=84
131372	jade-1.2.1	cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177
127536	pine4.21	ansic=126678,sh=766,csch=62,perl=30
121878	ImageMagick-4.2.9	ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
119613	lynx2-8-3	ansic=117385,sh=1860,perl=340,csch=28
116951	mc-4.5.42	ansic=114406,sh=1996,perl=345,awk=148,csch=56
116615	gnumeric-0.48	ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
113272	xlispstat-3-52-17	ansic=91484,lisp=21769,sh=18,csch=1
113241	vim-5.6	ansic=111724,awk=683,sh=469,perl=359,csch=6
109824	php-3.0.15	ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
104032	linuxconf-1.17r2	cpp=93139,perl=4570,sh=2984,java=2741,ansic=598
102674	libgr-2.0.13	ansic=99647,sh=2438,csch=589
100951	lam-6.3.1	ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187,csch=19
99066	krb4-1.0	ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765,yacc=1509,lex=236,awk=33
94637	xlockmore-4.15	ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
93940	kdenetwork	cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
92964	samba-2.0.6	ansic=88308,sh=3557,perl=831,awk=158,csch=110
91213	anaconda-6.2.2	ansic=74303,python=13657,sh=1583,yacc=810,lex=732,perl=128
89959	xscreensaver-3.23	ansic=88488,perl=1070,sh=401
88128	cvs-1.10.7	ansic=68303,sh=17909,perl=902,yacc=826,csch=181,lisp=7
87940	isd4k-utils	ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
85383	xpdf-0.90	cpp=60427,ansic=21400,sh=3556
81719	inn-2.2.2	ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547,lex=249,tcl=3
80343	kdelibs	cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116,sh=35

Estimating Linux's Size

```

79997 WindowMaker-0.61.1 ansic=77924,sh=1483,perl=371,lisp=219
78787 extace-1.2.15 ansic=66571,sh=9322,perl=2894
77873 apache_1.3.12 ansic=69191,sh=6781,perl=1846,cpp=55
75257 xpilot-4.1.0 ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68
73817 w3c-libwww-5.2.8 ansic=64754,sh=4678,cpp=3181,perl=1204
72726 ucd-snmp-4.1.1 ansic=64411,perl=5558,sh=2757
72425 gnome-core-1.0.55 ansic=72230,perl=141,sh=54
71810 jikes cpp=71452,java=358
70260 groff-1.15 cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397,
sh=265,sed=46
69265 fvwm-2.2.4 ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
69246 linux-86 ansic=63328,asm=5276,sh=642
68997 blt2.4g ansic=58630,tcl=10215,sh=152
68884 squid-2.3.STABLE1 ansic=66305,sh=1570,perl=1009
68560 bash-2.03 ansic=56758,sh=7264,yacc=2808,perl=1730
68453 kdegraphics cpp=34208,ansic=29347,sh=4898
65722 xntp3-5.93 ansic=60190,perl=3633,sh=1445,awk=417,asm=37
62922 ppp-2.3.11 ansic=61756,sh=996,exp=82,perl=44,csch=44
62137 sgml-tools-1.0.9 cpp=38543,ansic=19185,perl=2866,lex=560,sh=532,
lisp=309,awk=142
61688 imap-4.7 ansic=61628,sh=60
61324 ncurses-5.0 ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103,
sed=100
60429 kdesupport ansic=42421,cpp=17810,sh=173,awk=13,csch=12
60302 openldap-1.2.9 ansic=58078,sh=1393,perl=630,python=201
57217 xfig.3.2.3-beta-1 ansic=57212,csch=5
56093 lsof_4.47 ansic=50268,sh=4753,perl=856,awk=214,asm=2
55667 uucp-1.06.1 ansic=52078,sh=3400,perl=189
54935 gnupg-1.0.1 ansic=48884,asm=4586,sh=1465
54603 glade-0.5.5 ansic=49545,sh=5058
54431 svgalib-1.4.1 ansic=53725,asm=630,perl=54,sh=22
53141 AfterStep-1.8.0 ansic=50898,perl=1168,sh=842,cpp=233
52808 kdeutils cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
52574 nmh-1.0.3 ansic=50698,sh=1785,awk=74,sed=17
51813 freetype-1.3.1 ansic=48929,sh=2467,cpp=351,csch=53,perl=13
51592 enlightenment-0.15.5 ansic=51569,sh=23
50970 cdrecord-1.8 ansic=48595,sh=2177,perl=194,sed=4
49370 tin-1.4.2 ansic=47763,sh=908,yacc=699
49325 imlib-1.9.7 ansic=49260,sh=65
48223 kdemultimedia ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
47067 bash-1.14.7 ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
46312 tcsh-6.09.00 ansic=43544,sh=921,lisp=669,perl=593,csch=585
46159 unzip-5.40 ansic=40977,cpp=3778,asm=1271,sh=133
45811 mutt-1.0.1 ansic=45574,sh=237
45589 am-utils-6.0.3 ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
45485 guile-1.3 ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,csch=50
45378 gnuplot-3.7.1 ansic=43276,lisp=661,asm=539,objc=387,csch=297,perl=138,
sh=80
44323 mgetty-1.1.21 ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
42880 sendmail-8.9.3 ansic=40364,perl=1737,sh=779
42746 elm2.5.3 ansic=32931,sh=9774,awk=41
41388 p2c-1.22 ansic=38788,pascal=2499,perl=101
41205 gnome-games-1.0.51 ansic=31191,lisp=6966,cpp=3048
39861 rpm-3.0.4 ansic=36994,sh=1505,perl=1355,python=7
39160 util-linux-2.10f ansic=38627,sh=351,perl=65,csch=62,sed=55
38927 xmms-1.0.1 ansic=38366,asm=398,sh=163
38548 ORBit-0.5.0 ansic=35656,yacc=1750,sh=776,lex=366
38453 zsh-3.0.7 ansic=36208,sh=1763,perl=331,awk=145,sed=6
37515 ircii-4.4 ansic=36647,sh=852,lex=16
37360 tiff-v3.5.4 ansic=32734,sh=4054,cpp=572

```

Estimating Linux's Size

```

36338  textutils-2.0a  ansic=18949,sh=16111,perl=1218,sed=60
36243  exmh-2.1.1      tcl=35844,perl=316,sh=49,exp=34
36239  xllamp-0.9-alpha3  ansic=31686,sh=4200,asm=353
35812  xloadimage.4.1  ansic=35705,sh=107
35554  zip-2.3         ansic=32108,asm=3446
35397  gtk-engines-0.10  ansic=20636,sh=14761
35136  php-2.0.1      ansic=33991,sh=1056,awk=89
34882  pmake          ansic=34599,sh=184,awk=58,sed=41
34772  xpuzzles-5.4.1  ansic=34772
34768  fileutils-4.0p  ansic=31324,sh=2042,yacc=841,perl=561
33203  strace-4.2     ansic=30891,sh=1988,perl=280,lisp=44
32767  trn-3.6        ansic=25264,sh=6843,yacc=660
32277  pilot-link.0.9.3  ansic=26513,java=2162,cpp=1689,perl=971,yacc=660,
python=268,tcl=14
31994  korganizer      cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
31174  ncftp-3.0beta21  ansic=30347,cpp=595,sh=232
30438  gnome-pim-1.0.55  ansic=28665,yacc=1773
30122  scheme-3.2     lisp=19483,ansic=10515,sh=124
30061  tcpdump-3.4    ansic=29208,yacc=236,sh=211,lex=206,awk=184,csch=16
29730  screen-3.9.5   ansic=28156,sh=1574
29315  jed            ansic=29315
29091  xchat-1.4.0    ansic=28894,perl=121,python=53,sh=23
28897  ncpfs-2.2.0.17  ansic=28689,sh=182,tcl=26
28449  slrn-0.9.6.2   ansic=28438,sh=11
28261  xfish-tank-2.1tp  ansic=28261
28186  texinfo-4.0    ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
28169  e2fsprogs-1.18  ansic=27250,awk=437,sh=339,sed=121,perl=22
28118  slang          ansic=28118
27860  kdegames       cpp=27507,ansic=340,sh=13
27117  librepp-0.10   ansic=19381,lisp=5385,sh=2351
27040  mikmod-3.1.6   ansic=26975,sh=55,awk=10
27022  x3270-3.1.1    ansic=26456,sh=478,exp=88
26673  lout-3.17      ansic=26673
26608  Xaw3d-1.3     ansic=26235,yacc=247,lex=126
26363  gawk-3.0.4     ansic=19871,awk=2519,yacc=2046,sh=1927
26146  libxml-1.8.6   ansic=26069,sh=77
25994  xrn-9.02       ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31,
csch=13
25915  gv-3.5.8       ansic=25821,sh=94
25479  xpaint         ansic=25456,sh=23
25236  shadow-19990827  ansic=23464,sh=883,yacc=856,perl=33
24910  kdeadadmin     cpp=19919,sh=3936,perl=1055
24773  pdksh-5.2.14   ansic=23599,perl=945,sh=189,sed=40
24583  gmp-2.0.2      ansic=17888,asm=5252,sh=1443
24387  mars_nwe       ansic=24158,sh=229
24270  gnome-python-1.0.51  python=14331,ansic=9791,sh=148
23838  kterm-6.2.0    ansic=23838
23666  enscrip-1.6.1  ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
22373  sawmill-0.24   ansic=11038,lisp=8172,sh=3163
22279  make-3.78.1    ansic=19287,sh=2029,perl=963
22011  libpng-1.0.5   ansic=22011
21593  xboard-4.0.5   ansic=20640,lex=904,sh=41,csch=5,sed=3
21010  netkit-telnet-0.16  ansic=14796,cpp=6214
20433  pam-0.72       ansic=18936,yacc=634,sh=482,perl=321,lex=60
20125  ical-2.2       cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
20078  gdl-3         ansic=19946,perl=132
19971  wu-ftpd-2.6.0  ansic=17572,yacc=1774,sh=421,perl=204
19500  gnome-utils-1.0.50  ansic=18099,yacc=824,lisp=577
19065  joe           ansic=18841,asm=224
18885  X11R6-contrib-3.3.2  ansic=18616,lex=161,yacc=97,sh=11

```


18835	glib-1.2.6	ansic=18702,sh=133
18151	git-4.3.19	ansic=16166,sh=1985
18020	xboing	ansic=18006,sh=14
17939	sh-utils-2.0	ansic=13366,sh=3027,yacc=871,perl=675
17765	mtools-3.9.6	ansic=16155,sh=1602,sed=8
17750	gettext-0.10.35	ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9
17682	bc-1.05	ansic=9186,sh=7236,yacc=967,lex=293
17271	fetchmail-5.3.1	ansic=13441,python=1490,sh=1246,yacc=411,perl=321,lex=238,awk=124
17259	sox-12.16	ansic=16659,sh=600
16785	control-center-1.0.51	ansic=16659,sh=126
16266	dhcp-2.0	ansic=15328,sh=938
15967	SVGATextMode-1.9-src	ansic=15079,yacc=340,sh=294,lex=227,sed=15,asm=12
15868	kpilot-3.1b9	cpp=8613,ansic=5640,yacc=1615
15851	taper-6.9a	ansic=15851
15819	mpg123-0.59r	ansic=14900,asm=919
15691	transfig.3.2.1	ansic=15643,sh=38,csch=10
15638	mod_perl-1.21	perl=10278,ansic=5124,sh=236
15522	console-tools-0.3.3	ansic=13335,yacc=986,sh=800,lex=291,perl=110
15456	rpm2html-1.2	ansic=15334,perl=122
15143	gnotepad+-1.1.4	ansic=15143
15108	GXedit1.23	ansic=15019,sh=89
15087	mm2.7	ansic=8044,csch=6924,sh=119
14941	readline-2.2.1	ansic=11375,sh=1890,perl=1676
14912	ispell-3.1	ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385,csch=221,sh=157,perl=85,sed=15
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,csch=29
14774	flex-2.5.4	ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12
14587	multimedia	ansic=14577,sh=10
14516	libgtop-1.0.6	ansic=13768,perl=653,sh=64,asm=31
14427	mawk-1.2.2	ansic=12714,yacc=994,awk=629,sh=90
14363	automake-1.4	perl=10622,sh=3337,ansic=404
14350	rsync-2.4.1	ansic=13986,perl=179,sh=126,awk=59
14299	nfs-utils-0.1.6	ansic=14107,sh=165,perl=27
14269	rcs-5.7	ansic=12209,sh=2060
14255	tar-1.13.17	ansic=13014,lisp=592,sh=538,perl=111
14105	wmakerconf-2.1	ansic=13620,perl=348,sh=137
14039	less-346	ansic=14032,awk=7
13779	rxvt-2.6.1	ansic=13779
13586	wget-1.5.3	ansic=13509,perl=54,sh=23
13504	rp3-1.0.7	cpp=10416,ansic=2957,sh=131
13241	iproute2	ansic=12139,sh=1002,perl=100
13100	silos-0.9.8	ansic=10485,asm=2615
12657	macutils	ansic=12657
12639	libungif-4.1.0	ansic=12381,sh=204,perl=54
12633	minicom-1.83.0	ansic=12503,sh=130
12593	audiofile-0.1.9	sh=6440,ansic=6153
12463	gnome-objc-1.0.2	objc=12365,sh=86,ansic=12
12313	jpeg-6a	ansic=12313
12124	ypserv-1.3.9	ansic=11622,sh=460,perl=42
11790	lrzsz-0.12.20	ansic=9512,sh=1263,exp=1015
11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
11633	net-tools-1.54	ansic=11531,sh=102
11404	findutils-4.1	ansic=11160,sh=173,exp=71
11299	xmorph-1999dec12	ansic=10783,tcl=516
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
10914	diffutils-2.7	ansic=10914
10404	gnorpm-0.9	ansic=10404

Estimating Linux's Size

10271	gqview-0.7.0	ansic=10271
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
10187	dump-0.4b15	ansic=9422,sh=760,sed=5
10088	piranha	ansic=10048,sh=40
10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
9961	procps-2.0.6	ansic=9959,sh=2
9942	xpat2-1.04	ansic=9942
9927	procmail-3.14	ansic=8090,sh=1837
9873	nss_ldap-105	ansic=9784,perl=89
9801	man-1.5hl	ansic=7377,sh=1802,perl=317,awk=305
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6
9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
9699	bison-1.28	ansic=9650,sh=49
9666	ash-linux-0.2	ansic=9445,sh=221
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
9551	pwdb-0.61	ansic=9488,sh=63
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
9263	ctags-3.4	ansic=9240,sh=23
9138	gftp-2.0.6a	ansic=9138
8939	mkisofs-1.12b5	ansic=8939
8766	pxe-linux	cpp=4463,ansic=3622,asm=681
8572	psgml-1.2.1	lisp=8572
8540	xxgdb-1.12	ansic=8540
8491	gtop-1.0.5	ansic=8151,cpp=340
8356	gedit-0.6.1	ansic=8225,sh=131
8303	dip-3.3.7o	ansic=8207,sh=96
7859	libglade-0.11	ansic=5898,sh=1809,python=152
7826	xpm-3.4k	ansic=7750,sh=39,cpp=37
7740	sed-3.02	ansic=7301,sed=359,sh=80
7617	cpio-2.4.2	ansic=7598,sh=19
7615	esound-0.2.17	ansic=7387,sh=142,csch=86
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
7427	ed-0.2	ansic=7263,sh=164
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,cpp=3
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
7095	xgammon-0.98	ansic=6506,lex=589
7041	newt-0.50.8	ansic=6526,python=515
7030	ee-0.3.11	ansic=7007,sh=23
6976	aboot-0.5	ansic=6680,asm=296
6968	mailx-8.1.1	ansic=6963,sh=5
6877	lpr	ansic=6842,sh=35
6827	gnome-media-1.0.51	ansic=6827
6646	iputils	ansic=6646
6611	patch-2.5	ansic=6561,sed=50
6592	xosview-1.7.1	cpp=6205,ansic=367,awk=20
6550	byacc-1.9	ansic=5520,yacc=1030
6496	pidentd-3.0.10	ansic=6475,sh=21
6391	m4-1.4	ansic=5993,lisp=243,sh=155
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
6234	awesfx-0.4.3a	ansic=6234
6172	sash-3.4	ansic=6172
6116	lslk	ansic=5325,sh=791
6090	joystick-1.2.15	ansic=6086,sh=4
6072	kdcc	perl=6010,sh=45,cpp=17
6043	irda-utils-0.9.10	ansic=5697,sh=263,perl=83
6033	sysvinit-2.78	ansic=5256,sh=777
6025	pnm2ppa	ansic=5708,sh=317
6021	rpmfind-1.4	ansic=6021
5981	indent-2.2.5	ansic=5958,sh=23

```

5975      ytalk-3.1          ansic=5975
5960      isapnptools-1.21  ansic=4394,yacc=1383,perl=123,sh=60
5744      gdm-2.0beta2      ansic=5632,sh=112
5594      isdn-config       cpp=3058,sh=2228,perl=308
5526      efax-0.9          ansic=4570,sh=956
5383      acct-6.3.2        ansic=5016,cpp=287,sh=80
5115      libtool-1.3.4     sh=3374,ansic=1741
5111      netkit-ftp-0.16   ansic=5111
4996      bzip2-0.9.5d     ansic=4996
4895      xcpustate-2.5     ansic=4895
4792      libelf-0.6.4      ansic=3310,sh=1482
4780      make-3.78.1_pvm-0.5 ansic=4780
4542      gpgp-0.4          ansic=4441,sh=101
4430      gperf-2.7         cpp=2947,exp=745,ansic=695,sh=43
4367      aumix-1.30.1      ansic=4095,sh=179,sed=93
4087      zlib-1.1.3        ansic=2815,asm=712,cpp=560
4038      sysklogd-1.3-31   ansic=3741,perl=158,sh=139
4024      rep-gtk-0.8       ansic=2905,lisp=971,sh=148
3962      netkit-timed-0.16 ansic=3962
3929      initscripts-5.00  sh=2035,ansic=1866,csch=28
3896      ltrace-0.3.10     ansic=2986,sh=854,awk=56
3885      phhttpd-0.1.0     ansic=3859,sh=26
3860      xdaliclock-2.18   ansic=3837,sh=23
3855      pciutils-2.1.5    ansic=3800,sh=55
3804      quota-2.00-pre3   ansic=3795,sh=9
3675      dosfstools-2.2    ansic=3675
3654      tcp_wrappers_7.6  ansic=3654
3651      ipchains-1.3.9    ansic=2767,sh=884
3625      autofs-3.1.4      ansic=2862,sh=763
3588      netkit-rsh-0.16   ansic=3588
3438      yp-tools-2.4      ansic=3415,sh=23
3433      dialog-0.6        ansic=2834,perl=349,sh=250
3415      ext2ed-0.1        ansic=3415
3315      gdbm-1.8.0        ansic=3290,cpp=25
3245      ypbind-3.3        ansic=1793,sh=1452
3219      playmidi-2.4      ansic=3217,sed=2
3096      xtrojka123        ansic=3087,sh=9
3084      at-3.1.7          ansic=1442,sh=1196,yacc=362,lex=84
3051      dhcpcd-1.3.18-pl3 ansic=2771,sh=280
3012      apmd              ansic=2617,sh=395
2883      netkit-base-0.16  ansic=2883
2879      vixie-cron-3.0.1  ansic=2866,sh=13
2835      gkermi-1.0        ansic=2835
2810      kdetoys           cpp=2618,ansic=192
2791      xjewel-1.6        ansic=2791
2773      mpage-2.4         ansic=2704,sh=69
2758      autoconf-2.13     sh=2226,perl=283,exp=167,ansic=82
2705      autorun-2.61      sh=1985,cpp=720
2661      cdp-0.33          ansic=2661
2647      file-3.28         ansic=2601,perl=46
2645      libghttp-1.0.4    ansic=2645
2631      getty_ps-2.0.7j   ansic=2631
2597      pythonlib-1.23    python=2597
2580      magicdev-0.2.7    ansic=2580
2531      gnome-kberos-0.2  ansic=2531
2490      sndconfig-0.43    ansic=2490
2486      bug-buddy-0.7     ansic=2486
2459      usermode-1.20     ansic=2459
2455      fnlib-0.4         ansic=2432,sh=23
2447      sliplogin-2.1.1   ansic=2256,sh=143,perl=48

```

Estimating Linux's Size

```

2424    raidtools-0.90    ansic=2418,sh=6
2423    netkit-routed-0.16 ansic=2423
2407    nc                ansic=1670,sh=737
2324    up2date-1.13      python=2324
2270    memprof-0.3.0     ansic=2270
2268    which-2.9         ansic=1398,sh=870
2200    printtool         tcl=2200
2163    gnome-linuxconf-0.25 ansic=2163
2141    unarj-2.43        ansic=2141
2065    units-1.55        ansic=1963,perl=102
2048    netkit-ntalk-0.16 ansic=2048
1987    cracklib,2.7      ansic=1919,perl=46,sh=22
1984    cleanfeed-0.95.7b perl=1984
1977    wmconfig-0.9.8    ansic=1941,sh=36
1941    isicom            ansic=1898,sh=43
1883    slocate-2.1       ansic=1802,sh=81
1857    netkit-rusers-0.16 ansic=1857
1856    pump-0.7.8        ansic=1856
1842    cdecl-2.5          ansic=1002,yacc=765,lex=75
1765    fbset-2.1         ansic=1401,yacc=130,lex=121,perl=113
1653    adjtimex-1.9      ansic=1653
1634    netcfg-2.25       python=1632,sh=2
1630    psmisc             ansic=1624,sh=6
1621    urlview-0.7        ansic=1515,sh=106
1604    fortune-mod-9708  ansic=1604
1531    netkit-tftp-0.16   ansic=1531
1525    logrotate-3.3.2    ansic=1524,sh=1
1473    traceroute-1.4a5   ansic=1436,awk=37
1452    time-1.7          ansic=1395,sh=57
1435    ncompress-4.2.4    ansic=1435
1361    mt-st-0.5b         ansic=1361
1290    cxhextris         ansic=1290
1280    pam_krb5-1          ansic=1280
1272    bsd-finger-0.16    ansic=1272
1229    hdparm-3.6         ansic=1229
1226    procinfo-17        ansic=1145,perl=81
1194    passwd-0.64.1      ansic=1194
1182    auth_ldap-1.4.0    ansic=1182
1146    prtconf-1.3        ansic=1146
1143    anacron-2.1         ansic=1143
1129    xbill-2.0          cpp=1129
1099    popt-1.4           ansic=1039,sh=60
1088    nag                perl=1088
1076    stylesheets-0.13rh perl=888,sh=188
1075    authconfig-3.0.3    ansic=1075
1049    kpppload-1.04      cpp=1044,sh=5
1020    MAKEDEV-2.5.2       sh=1020
1013    trojka              ansic=1013
987    xmailbox-2.5        ansic=987
967    netkit-rwho-0.16    ansic=967
953    switchdesk-2.1     ansic=314,perl=287,cpp=233,sh=119
897    portmap_4          ansic=897
874    ldconfig-1999-02-21 ansic=874
844    jpeg-6b            sh=844
834    ElectricFence-2.1   ansic=834
830    mouseconfig-4.4     ansic=830
816    rpmlint-0.8         python=813,sh=3
809    kdpms-0.2.8        cpp=809
797    termcap-2.0.8       ansic=797
787    xsysinfo-1.7        ansic=787

```

Estimating Linux's Size

```

770    giftrans-1.12.2  ansic=770
742    setserial-2.15  ansic=742
728    tree-1.2        ansic=728
717    chkconfig-1.1.2 ansic=717
682    lpg              perl=682
657    eject-2.0.2     ansic=657
616    diffstat-1.27   ansic=616
592    netscape-4.72   sh=592
585    usernet-1.0.9   ansic=585
549    genromfs-0.3    ansic=549
548    tksysv-1.1      tcl=526,sh=22
537    minlabel-1.2    ansic=537
506    netkit-bootparamd-0.16 ansic=506
497    locale_config-0.2 ansic=497
491    helptool-2.4     perl=288,tcl=203
480    elftoaout-2.2    ansic=480
463    tmpwatch-2.2     ansic=311,sh=152
445    rhs-printfilters-1.63 sh=443,ansic=2
441    audioctl         ansic=441
404    control-panel-3.13 ansic=319,tcl=85
368    kbdconfig-1.9.2.4 ansic=368
368    vlock-1.3        ansic=368
367    timetool-2.7.3   tcl=367
347    kernelcfg-0.5    python=341,sh=6
346    timeconfig-3.0.3 ansic=318,sh=28
343    mingetty-0.9.4   ansic=343
343    chkfontpath-1.7  ansic=343
332    ethtool-1.0      ansic=332
314    mkbootdisk-1.2.5 sh=314
302    symlinks-1.2     ansic=302
301    xsri-1.0         ansic=301
294    netkit-rwall-0.16 ansic=294
290    biff+comsat-0.16 ansic=290
288    mkinitrd-2.4.1   sh=288
280    stat-1.5         ansic=280
265    sysreport-1.0    sh=265
261    bdflush-1.5      ansic=202,asm=59
255    ipvsadm-1.1      ansic=255
255    sag-0.6-html     perl=255
245    man-pages-1.28   sh=244,sed=1
240    open-1.4         ansic=240
236    xtoolwait-1.2    ansic=236
222    utempter-0.5.2   ansic=222
222    mkkickstart-2.1  sh=222
221    hellas          sh=179,perl=42
213    rhmask          ansic=213
159    quickstrip-1.1   ansic=159
132    rdate-1.0        ansic=132
131    statserial-1.1   ansic=121,sh=10
107    fwhois-1.00      ansic=107
85     mktemp-1.5       ansic=85
82     modemtool-1.21   python=73,sh=9
67     setup-1.2        ansic=67
56     shaper          ansic=56
52     sparc32-1.1      ansic=52
47     intimed-1.10     ansic=47
23     locale-ja-9      sh=23
16     AnotherLevel-1.0.1 sh=16
11     words-2          sh=11
7      trXFree86-2.1.2  tcl=7

```

```

0      install-guide-3.2.html (none)
0      caching-nameserver-6.2 (none)
0      XFree86-ISO8859-2-1.0 (none)
0      rootfiles              (none)
0      ghostscript-fonts-5.50 (none)
0      kudzu-0.36              (none)
0      wvdial-1.41             (none)
0      mailcap-2.0.6           (none)
0      desktop-backgrounds-1.1 (none)
0      redhat-logos            (none)
0      solemul-1.1             (none)
0      dev-2.7.18              (none)
0      urw-fonts-2.0           (none)
0      users-guide-1.0.72      (none)
0      sgml-common-0.1         (none)
0      setup-2.1.8             (none)
0      jadetex                  (none)
0      gnome-audio-1.0.0       (none)
0      specs-po-6.2            (none)
0      gimp-data-extras-1.0.0 (none)
0      docbook-3.1             (none)
0      indexhtml-6.2           (none)

```

```

ansic:      14218806 (80.55%)
cpp:        1326212 (7.51%)
lisp:       565861 (3.21%)
sh:         469950 (2.66%)
perl:       245860 (1.39%)
asm:        204634 (1.16%)
tcl:        152510 (0.86%)
python:     140725 (0.80%)
yacc:       97506 (0.55%)
java:       79656 (0.45%)
exp:        79605 (0.45%)
lex:        15334 (0.09%)
awk:        14705 (0.08%)
objc:       13619 (0.08%)
csh:        10803 (0.06%)
ada:        8217 (0.05%)
pascal:     4045 (0.02%)
sed:        2806 (0.02%)
fortran:    1707 (0.01%)

```

```

Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = $ 614421924.71

```

B.2 Counts of Files For Each Category

There were 181,679 ordinary files in the build directory. The following are counts of the number of files (*not* the SLOC) for each language:

```

ansic:      52088 (71.92%)
cpp:        8092 (11.17%)
sh:         3381 (4.67%)

```


asm:	1931	(2.67%)
perl:	1387	(1.92%)
lisp:	1168	(1.61%)
java:	1047	(1.45%)
python:	997	(1.38%)
tcl:	798	(1.10%)
exp:	472	(0.65%)
awk:	285	(0.39%)
objc:	260	(0.36%)
sed:	112	(0.15%)
yacc:	110	(0.15%)
cs:	94	(0.13%)
ada:	92	(0.13%)
lex:	57	(0.08%)
fortran:	50	(0.07%)
pascal:	7	(0.01%)

Total Number of Source Code Files = 72428

In addition, when counting the number of files (not SLOC), some files were identified as source code files but nevertheless were not counted for other reasons (and thus not included in the file counts above). Of these source code files, 5,820 files were identified as duplicating the contents of another file, 817 files were identified as files that had been automatically generated, and 65 files were identified as zero-length files.

B.3 Additional Measures of the Linux Kernel

I also made additional measures of the Linux kernel. This kernel is Linux kernel version 2.2.14 as patched by Red Hat. The Linux kernel's design is reflected in its directory structure. Only 8 lines of source code are in its main directory; the rest are in descendent directories. Counting the physical SLOC in each subdirectory (or its descendents) yielded the following:

BUILD/linux/Documentation/	765
BUILD/linux/arch/	236651
BUILD/linux/configs/	0
BUILD/linux/drivers/	876436
BUILD/linux/fs/	88667
BUILD/linux/ibcs/	16619
BUILD/linux/include/	136982
BUILD/linux/init/	1302
BUILD/linux/ipc/	1757
BUILD/linux/kernel/	7436
BUILD/linux/ksymoops-0.7c/	3271
BUILD/linux/lib/	1300
BUILD/linux/mm/	6771
BUILD/linux/net/	105549
BUILD/linux/pcmcia-cs-3.1.8/	34851
BUILD/linux/scripts/	8357

I separately ran the CodeCount tools on the entire linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for Linux.

However, this included non-i86 code. To make a more reasonable comparison with the Halloween documents, I needed to ignore non-i386 code.

First, I looked at the linux/arch directory, which contained architecture-specific code. This directory had the following subdirectories (architectures): alpha, arm, i386, m68k, mips, ppc, s390, sparc, sparc64. I then computed the total for all of ``arch'', which was 236651 SLOC, and subtracted out linux/arch/i386 code, which totalled to 26178 SLOC; this gave me a total of non-i386 code in linux/arch as 210473 physical SLOC. I then looked through the ``drivers'' directory to see

if there were sets of drivers which were non-i386. I identified the following directories, with the SLOC totals as shown:

linux/drivers/sbus/	22354
linux/drivers/macintosh/	6000
linux/drivers/sgi/	4402
linux/drivers/fc4/	3167
linux/drivers/nubus/	421
linux/drivers/acorn/	11850
linux/drivers/s390/	8653

Driver Total: 56847

Thus, I had a grand total on non-i86 code (including drivers and architecture-specific code) as 267320 physical SLOC. This is, of course, another approximation, since there's certainly other architecture-specific lines, but I believe that is most of it. Running the CodeCount tool on just the C code, once these architectural and driver directories are removed, reveals a logical SLOC of 570,039 of C code.

B.4 Minimum System SLOC

Most of this paper worries about counting an ``entire" system. However, what's the SLOC size of a ``minimal" system? Here's an attempt to answer that question.

Red Hat Linux 6.2, CD-ROM #1, file RedHat/base/comps, defines the ``base" (minimum) Red Hat Linux 6.2 installation as a set of packages. The following are the build directories corresponding to this base (minimum) installation, along with the SLOC counts (as shown above). Note that this creates a text-only system:

Component	SLOC
anacron-2.1	1143
apmd	3012
ash-linux-0.2	9666
at-3.1.7	3084
authconfig-3.0.3	1075
bash-1.14.7	47067
bc-1.05	17682
bdf flush-1.5	261
binutils-2.9.5.0.22	467120
bzip2-0.9.5d	4996
chkconfig-1.1.2	717
console-tools-0.3.3	15522
cpio-2.4.2	7617
cracklib, 2.7	1987
dev-2.7.18	0
diffutils-2.7	10914
dump-0.4b15	10187
e2fsprogs-1.18	28169
ed-0.2	7427
egcs-1.1.2	720112
eject-2.0.2	657
file-3.28	2647
fileutils-4.0p	34768
findutils-4.1	11404
gawk-3.0.4	26363
gd1.3	20078
gdbm-1.8.0	3315
getty_ps-2.0.7j	2631
glibc-2.1.3	415026
gmp-2.0.2	24583
gnupg-1.0.1	54935
gpm-1.18.1	9725
grep-2.4	10013

Estimating Linux's Size

groff-1.15	70260
gzip-1.2.4a	6306
hdparm-3.6	1229
initscripts-5.00	3929
isapnptools-1.21	5960
kbdconfig-1.9.2.4	368
kernelcfg-0.5	347
kudzu-0.36	0
ldconfig-1999-02-21	874
ld.so-1.9.5	9731
less-346	14039
lilo	7255
linuxconf-1.17r2	104032
logrotate-3.3.2	1525
mailcap-2.0.6	0
mailx-8.1.1	6968
MAKEDEV-2.5.2	1020
man-1.5hl	9801
mingetty-0.9.4	343
mkbootdisk-1.2.5	314
mkinitrd-2.4.1	288
mktemp-1.5	85
modutils-2.3.9	11775
mouseconfig-4.4	830
mt-st-0.5b	1361
ncompress-4.2.4	1435
ncurses-5.0	61324
net-tools-1.54	11633
newt-0.50.8	7041
pam-0.72	20433
passwd-0.64.1	1194
pciutils-2.1.5	3855
popt-1.4	1099
procmail-3.14	9927
procps-2.0.6	9961
psmisc	1630
pump-0.7.8	1856
pwdb-0.61	9551
quota-2.00-pre3	3804
raidtools-0.90	2424
readline-2.2.1	14941
redhat-logos	0
rootfiles	0
rpm-3.0.4	39861
sash-3.4	6172
sed-3.02	7740
sendmail-8.9.3	42880
setserial-2.15	742
setup-1.2	67
setup-2.1.8	0
shadow-19990827	25236
sh-utils-2.0	17939
slang	28118
slocate-2.1	1883
stat-1.5	280
sysklogd-1.3-31	4038
sysvinit-2.78	6033
tar-1.13.17	14255
termcap-2.0.8	797
texinfo-4.0	28186

textutils-2.0a	36338
time-1.7	1452
timeconfig-3.0.3	346
tmpwatch-2.2	463
utempter-0.5.2	222
util-linux-2.10f	39160
vim-5.6	113241
vixie-cron-3.0.1	2879
which-2.9	2268
zlib-1.1.3	4087

Thus, the contents of the build directories corresponding to the ``base" (minimum) installation totals to 2,819,334 SLOC.

A few notes are in order about this build directory total:

1. Some of the packages listed by a traditional package list aren't shown here because they don't contain any code. Package "basesystem" is a pseudo-package for dependency purposes. Package redhat-release is just a package for keeping track of the base system's version number. Package "filesystem" contains a directory layout.
2. ntsysv's source is in chkconfig-1.1.2; kernel-utils and kernel-pcmcia-cs are part of "linux". Package shadow-utils is in build directory shadow-19990827. Build directory util-linux includes losetup and mount. "dump" is included to include rmt.
3. Sometimes the build directories contain more code than is necessary to create just the parts for the ``base" system; this is a side-effect of how things are packaged. ``info" is included in the base, so we count all of texinfo. The build directory termcap is counted, because libtermcap is in the base. Possibly most important, gcc (egcs) is there because libstdc++ is in the base.
4. Sometimes a large component is included in the base, even though most of the time little of its functionality is used. In particular, the mail transfer agent ``sendmail" is in the base, even though for many users most of sendmail's functionality isn't used. However, for this paper's purposes this isn't a problem. After all, even if sendmail's functionality is often underused, clearly that functionality took time to develop and that functionality is available to those who want it.
5. My tools intentionally eliminated duplicates; it may be that a few files aren't counted here because they're considered duplicates of another build directory not included here. I do not expect this factor to materially change the total.
6. Red Hat Linux is not optimized to be a ``small as possible" distribution; their emphasis is on functionality, not small size. A working Linux distribution could include much less code, depending on its intended application. For example, ``linuxconf" simplifies system configuration, but the system can be configured by editing its system configuration files directly, which would reduce the base system's size. This also includes vim, a full-featured text editor - a simpler editor with fewer functions would be smaller as well.

Many people prefer some sort of graphical interface; here is a minimal configuration of a graphical system, adding the X server, a window manager, and a few tools:

Component	SLOC
XFree86-3.3.6	1291745
Xconfigurator-4.3.5	9741
fvwm-2.2.4	69265
X11R6-contrib-3.3.2	18885

These additional graphical components add 1,389,636 SLOC. Due to oddities of the way the initialization system xinitrc is built, it isn't shown here in the total, but xinitrc has so little code that its omission does not significantly affect the total.

Adding these numbers together, we now have a total of 4,208,970 SLOC for a ``minimal graphical system." Many people would want to add more components. For example, this doesn't include a graphical toolkit (necessary for running most graphical applications). We could add gtk+-1.2.6 (a toolkit needed for running GTK+ based applications), adding 138,118 SLOC. This would now total 4,347,088 for a ``basic graphical system," one able to run basic GTK+ applications.

Let's add a web server to the mix. Adding apache_1.3.12 adds only 77,873 SLOC. We now have 4,424,961 physical SLOC for a basic graphical system plus a web server.

We could then add a graphical desktop environment, but there are so many different options and possibilities that trying to identify a ``minimal" system is hard to do without knowing the specific uses intended for the system. Red Hat defines a standard ``GNOME" and ``KDE" desktop, but these are intended to be highly functional (not ``minimal"). Thus, we'll stop here, with a total of 2.8 million physical SLOC for a minimal text-based system, and total of 4.4 million physical SLOC for a basic graphical system plus a web server.

References

- [Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.
- [Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/develpro.html>.
- [DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the ``Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.
- [FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.
- [Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.
- [Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>
- [Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf
- [Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>
- [Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.
- [Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.
- [Moody 2001] Moody, Glyn. 2001. *Rebel Code*. ISBN 0713995203.
- [NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>
- [OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.
- [Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>
- [Perens 1999] Perens, Bruce. January 1999. *Open Sources: Voices from the Open Source Revolution*. "The Open Source Definition". ISBN 1-56592-582-3. <http://www.oreilly.com/catalog/opensources/book/perens.html>
- [Raymond 1999] Raymond, Eric S. January 1999. ``A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.
- [Schneier 2000] Schneier, Bruce. March 15, 2000. ``Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>
- [Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...".
<http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet.
<http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*.
http://www.dwheeler.com/oss_fs_refs.html.

[Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

This paper is (C) Copyright 2000 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. When referring to the paper, please refer to it as ``Estimating GNU/Linux's Size'' by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

- 2000-7-26 David A. Wheeler <dwheeler@dwheeler.com>
* Added clarification from Vinod Valloppillil.
- 2000-11-6 David A. Wheeler <dwheeler@dwheeler.com>
* Added information on each license. GPL is the most common.
- 2000-11-2 David A. Wheeler <dwheeler@dwheeler.com>
* Clarified that "gcc" is the compilation system & "egcs" was a project to extend egcs, based on feedback from Per Bothner.
* Added a reference to [Dempsey 1999] and to my list of quantitative evidences.
- 2000-10-31 David A. Wheeler <dwheeler@dwheeler.com>
* Removed claim that SuSE was derived from Red Hat. SuSE uses Red Hat's RPM, but SuSE predated Red Hat.
* Included the complete list of languages in the abstract and conclusions, so people wouldn't have to rummage through the paper just to find that.
* Added a note that I didn't try to eliminate ``nearly duplicate'' files, because that's hard to do.
* Added a note explaining why I counted ALL code, not just the code generated into object code.
* Showed the disk space used by the source code.

Estimating Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

October 30, 2000

Version 1.0

This paper presents size estimates (and their implications) of the source code of a distribution of the Linux operating system (OS), a combination often called GNU/Linux. The distribution used in this paper is Red Hat Linux version 6.2, including the kernel, software development tools, graphics interfaces, client applications, and so on. Other distributions and versions will have different sizes.

In total, this distribution includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars).

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), egcs (a compilation system), and emacs (a text editor and far more). The languages with the most lines of code were (in order): C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), and perl; many other languages were also represented. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The Linux operating system (also called GNU/Linux) has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a "typical" intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally "open source software" and/or "free software" (where the "free" here means "freedom"). A program that is "open source software" or "free software" is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of "open source software" is available at OSI [1999], a more formal definition of "free software" is available at FSF [2000], and other general information about these topics is available at Wheeler [2000]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is "open source software"/"free software", and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. The only published data I've found was developed by Microsoft in the documents usually called "Halloween I" and "Halloween II". Unfortunately, the meaning, derivation, and assumptions of their numbers is not explained, making the numbers hard to use and truly understand. Even worse, although the two documents were written by essentially the same people at the same time, the numbers in the documents appear (on their surface) to be contradictory. The so-called "Halloween I" document claimed that the Linux kernel (x86 only) was 500,000 lines of code, the Apache web server was 80,000 lines of code, the X-windows server was 1.5 million, and a full Linux distribution was about 10 million lines of code [Halloween I]. The "Halloween II" document seemed to contradict this, saying that "Linux" by 1998 included 1.5 million lines of code. Since "version 2.1.110" is identified as the version number, presumably this only measures the Linux kernel, and it does note that this measure includes all Linux ports to various architectures [Halloween II]. However, this asks as many questions as it answers - what exactly was being measured, and what assumptions were made? For example, is the Linux kernel support for other architectures *really* one million lines of

code?

This paper bridges this gap. In particular, it shows estimates of the size of Linux, and it estimates how much it would cost to rebuild a typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean.

For my purposes, I have selected as my ``representative" Linux distribution Red Hat Linux version 6.2. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [[Shankland 2000b](#)]. Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE at 15%. Not all Linux copies are ``sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many major distributions (such as SuSE and Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate ``all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 6.2 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Section 2 briefly describes the approach used to estimate the ``size" of this distribution (most of the details are in Appendix A). Section 3 discusses some of the results (with the details in Appendix B). Section 4 presents conclusions, followed by the two appendices.

2. Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; the steps and assumptions made are described in Appendix A.

A few summary points are worth mentioning here, however, for those who don't read appendix A. I included software for all architectures, not just the i386. I did not include ``old" versions of software (with the one exception of bash, as discussed in appendix A). I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once. The code in makefiles and RPM package specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

The ``physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: ``a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the ``logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the ``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC

measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer "COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total SLOC counts, and section 3.6 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 25 largest components (as measured by number of source lines of code):

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,csh=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csh=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csh=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csh=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,csh=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,

```

sed=2
199982  gs5.50      ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916  teTeX-1.0      ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,
yacc=1507,awk=522,lex=323,sed=297,asm=139,csh=47,lisp=29
155035  bind-8.2.2_P5  ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,
csh=848,awk=753,lex=222
140130  AfterStep-APPS-20000124 ansic=135806,sh=3340,cpp=741,perl=243
138931  kdebase      cpp=113971,ansic=23016,perl=1326,sh=618
138118  gtk+-1.2.6   ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024  gated-3-5-11 ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csh=235,
sed=35,lisp=12
133193  kaffe-1.0.5   java=65275,ansic=62125,cpp=3923,perl=972,sh=814,
asm=84
131372  jade-1.2.1    cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672  gnome-libs-1.0.55 ansic=125373,sh=2178,perl=667,awk=277,lisp=177

```

Note that the operating system kernel (linux) is the largest single component, at over 1.5 million lines of code (mostly in C). See section 3.2 for a more discussion discussion of the linux kernel.

The next largest component is the X windows server, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to accrete functionality), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the compilation system, including the C and C++ compilers. Following this is the symbolic debugger and emacs. Emacs is probably not a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system. This is followed by the set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). This is followed by TCL/Tk (a combined language and widget set), PostgreSQL (a relational DBMS), and the GIMP (an excellent client application for editing bitmapped drawings).

Note that language implementations tend to be written in themselves, particularly for their libraries. Thus there is more Perl than any other single language in the Perl implementation, more Python than any other single language in Python, and more Java than any other single language in Kaffe (an implementation of the Java Virtual Machine and library).

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the linux kernel (at over 1.5 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 870,000 lines of this code was in the ``drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of hardware. The linux kernel's design is expressed in its source code directory structure, and no other directory comes close to this size - the second largest is the ``arch" directory (at over 230,000 SLOC), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is not quite 88,000 SLOC. See the appendix for more detail.

Richard Stallman and others have argued that the resulting system often called ``Linux" should instead be called ``GNU/Linux" [\[Stallman 2000\]](#). In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to ``free software" (free as in freedom). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called ``Linux". Using the term ``Linux" is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux" or ``GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: egcs (gcc), gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux" and not just ``Linux."

These measurements at least debunk one possible explanation of the Halloween documents' measures. Since Halloween I claimed that the x86-only code for the Linux kernel measured 500,000 SLOC, while Halloween II claimed that the kernel (all architectures) was 1.5 million SLOC, one explanation of this difference would be that the code for non-x86 systems was 1 million SLOC. This isn't so; I computed a grand total of 267,320 physical SLOC of non-i86 code (including drivers and architecture-specific code). It seems unlikely that over 700,000 lines of code would have been removed (not added) in the intervening time.

However, other measures (and explanations) are more promising. I also ran the CodeCount tools on the linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for the Linux kernel. When I removed all non-i86 code and re-ran the CodeCount tool on just the C code, a logical SLOC of 570,039 of C code was revealed. Since the Halloween I document reported 500,000 SLOC (when only including x86 code), it appears very likely that the Halloween I paper counted logical SLOC (and only C code) when reporting measurements of the linux kernel. However, the other Halloween I measures appear to be physical SLOC measures: their estimate of 1.5 million SLOC for the X server is closer to the 1.2 million physical SLOC measured here, and their estimate of 80,000 SLOC for Apache is close to the 77,873 SLOC measured here (as shown in Appendix B). These variations in measurements should be expected, since the versions I am measuring are slightly different than the ones they measured, and it is likely that some assumptions are different as well. Meanwhile, Halloween II reported a measure of 1.5 million lines of code for the linux kernel, essentially the same value given here for physical SLOC.

In short, it appears that Halloween I used the "logical SLOC" measure when measuring the Linux kernel, while all other measures in Halloween I and II used physical SLOC as the measure. I have attempted to contact the Microsoft author to confirm this, but as of yet I have not received such confirmation. In any case, this example clearly demonstrates the need to carefully identify the units of measure and assumptions made in any measurement of SLOC.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code:

ansic:	14218806	(80.55%)
c++:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)
python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Here you can see that C is pre-eminent (with over 80% of the code), followed by C++, LISP, shell, and Perl. Note that the separation of Expect and TCL is somewhat artificial; if combined, they would be next (at 232115), followed by assembly. Python, yacc, Java, lex, and awk make respectable showings as well. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has over a million lines of code, a very respectable showing, and yet at least in this distribution it is far less than C. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to

switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

The fact that LISP places so highly (it's in third place) is a little surprising. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 80% (453647/565861) of the total amount of LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages: Perl includes 5584 lines of LISP, and Python includes another 2333 of LISP that is directly used to support elaborate Emacs modes for program editing. The ``psgml" package is solely an emacs mode for editing SGML documents. The components with the second and third largest amounts of LISP are xlipstat-3-52-17 and scheme-3.2, which are implementations of LISP and Scheme (a LISP dialect) respectively. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 57 different lex/flex files, and 110 yacc/bison files. Since some build directories use lex/flex or yacc/bison more than once, the count of build directories using these tools is smaller but still respectable: 38 different build directories use lex/flex, and 62 different build directories use yacc/bison.

Other insights can be gained from the file counts shown in appendix B. The number of source code files counted were 72,428. Not included in this count were 5,820 files which contained duplicate contents, and 817 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 14218806 SLOC contained in 52088 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code.

3.5 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 17,652,561 physical source lines of code (SLOC); I will simplify this to ``over 17 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (1998)	20 million

These numbers come from Bruce Schneier's *Crypto-Gram* [\[Schneier 2000\]](#), except for the Space Shuttle numbers which come from a National Academy of Sciences study [\[NAS 1996\]](#). Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Schneier also reports that ``Linux, even with the addition of X Windows and Apache, is still under 5 million lines of code". At first, this seems to be contradictory, since this paper counts over 17 million SLOC, but Schneier appears to be literally correct in the context of his statement. The phrasing of his sentence suggests that Schneier is considering some sort of ``minimal" system, since he considers ``even the addition of X Windows" as a significant addition. As shown in appendix section B.4, taking the minimal ``base" set of components in Red Hat Linux, and then adding the minimal set of components for graphical interaction (the X Windows's graphical server, library, configuration tool, and a graphics toolkit) and the Apache web server, the total is about 4.4 million physical SLOC - which is less than 5 million. This minimal system doesn't include some useful

(but not strictly necessary) components, but a number of useful components could be added while still staying under a total of 5 million SLOC.

However, note the contrast. Many Linux distributions include with their operating systems many applications (e.g., bitmap editors) and development tools (for many different languages). As a result, the entire *delivered* system for such distributions (including Red Hat Linux 6.2) is *much* larger than the 5 million SLOC stated by Schneier. In short, this distribution's size appears similar to the size of Windows 98 and Windows NT 5.0 in 1998.

Microsoft's recent legal battles with the U.S. Department of Justice (DoJ) also involve the bundling of applications with the operating system. However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as "secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, many Linux distributions include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with small SLOC counts can often provide greater functionality than programs with large SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessarily cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized systems.

3.6 Effort and Cost Estimates

Finally, given all the assumptions shown, are the effort values:

```
Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = $ 614421924.71
```

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux version 6.2 includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars). Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches.

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), egcs (a compilation system), and emacs (a text editor and far more). The languages with the most lines of code were (in order): C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), and perl. More information is available in the appendices and at <http://www.dwheeler.com/sloc>.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), other open source systems (such as FreeBSD), and other versions of Red Hat (such as Red Hat 7). SuSE and Debian, for example, by policy

include many more packages, and would probably produce significantly larger estimates of effort and development cost. It's known that Red Hat 7 includes more source code; Red Hat 7 has had to add another CD-ROM to contain the binary programs, and adds such capabilities as a word processor (abiword) and secure shell (openssh).

Some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like ``Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not ``reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ``.m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

Appendix A. Details of Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; each step is described below. Some steps I describe in some detail, because it's sometimes hard to find the necessary information even when the actual steps are easy. Hopefully, this detail will make it easier for others to do similar activities or to repeat the experiment.

A.1 Installing Source Code

Installing the source code files turned out to be nontrivial. First, I inserted the CD-ROM containing all of the source files (in ``.src.rpm" format) and installed the packages (files) using:

```
mount /mnt/cdrom
cd /mnt/cdrom/SRPMS
rpm -ivh *.src.rpm
```

This installs ``spec" files and compressed source files; another rpm command (``rpm -bp") uses the spec files to uncompress the source files into ``build directories" (as well as apply any necessary patches). Unfortunately, the rpm tool does not enforce any naming consistency between the package names, the spec names, and the build directory names; for consistency this paper will use the names of the build directories, since all later tools based themselves on the build directories.

I decided to (in general) not count ``old" versions of software (usually placed there for compatibility reasons), since that would be counting the same software more than once. Thus, the following components were not included: ``compat-binutils", ``compat-egcs", ``compat-glib", ``compat-libs", ``gtk+10", ``libc-5.3.12" (an old C library), ``libxml10", ``ncurses3", and ``qt1x". I also didn't include egcs64-19980921 and netscape-sparc, which simply repeated something on another architecture that was available on the i386 in a different package. I did make one exception. I kept both bash-1.14.7 and bash2, two versions of the shell command processor, instead of only counting bash2. While bash2 is the later version of the shell available in the package, the main shell actually used by the Red Hat distribution was the older version of bash. The rationale for this decision appears to be backwards compatibility for older shell scripts; this is suggested by the Red Hat package documentation in both bash-1.14.7 and bash2. It seemed wrong to not include one of the most fundamental pieces of the

system in the count, so I included it. At 47067 lines of code (ignoring duplicates), bash-1.14.7 is one of the smaller components anyway. Not including this older component would not substantively change the results presented here.

There are two directories, krb4-1.0 and krb5-1.1.1, which appear to violate this rule - but don't. krb5-1.1.1 is the build directory created by krb5.spec, which is in turn installed by the source RPM package krb5-1.1.1-9.src.rpm. This build directory contains Kerberos V5, a trusted-third-party authentication system. The source RPM package krb5-1.1.1-9.src.rpm eventually generates the binary RPM files krb5-configs-1.1.1-9, krb5-libs-1.1.1-9, and krb5-devel-1.1.1-9. You might guess that ``krb4-1.0" is just the older version of Kerberos, but this build directory is created by the spec file krbafs.spec and not just an old version of the code. To quote its description, ``This is the Kerberos to AFS bridging library, built against Kerberos 5. krbafs is a shared library that allows programs to obtain AFS tokens using Kerberos IV credentials, without having to link with official AFS libraries which may not be available for a given platform." For this situation, I simply counted both packages, since their purposes are different.

I was then confronted with a fundamental question: should I count software that only works for another architecture? I was using an i86-type system, but some components are only for Alpha or Sparc systems. I decided that I should count them; even if I didn't use the code today, the ability to use these other architectures in the future was of value and certainly required effort to develop.

This caused complications for creating the build directories. If all installed packages fit the architecture, you can install the uncompressed software by typing:

```
cd /usr/src/redhat/SPECS and typing the command
rpm -bp *.spec
```

Unfortunately, the rpm tool notes that you're trying to load code for the ``wrong" architecture, and (at least at the time) there was no simple ``override" flag. Instead, I had to identify each package as belonging to SPARC or ALPHA, and then use the rpm option --target to forcibly load them. For example, I renamed all sparc-specific SPARC file files to end in ``.sparc" and could then load them with:

```
rpm -bp --target sparc-redhat-linux *.spec.sparc
```

The following spec files were non-i86: (sparc) audiocctl, elftoaout, ethtool, prtconf, silo, solemul, sparc32; (alpha) aboot, minlabel, quickstrip. In general, these were tools to aid in supporting some part of the boot process or for using system-specific hardware.

Note that not all packages create build directories. For example, ``anonftp" is a package that, when installed, sets up an anonymous ftp system. This package doesn't actually install any software; it merely installs a specific configuration of another piece of software (and unsets the configuration when uninstalled). Such packages are not counted at all in this sizing estimate.

A.2 Categorizing Source Code

My next task was to identify all files containing source code (not including any automatically generated source code). This is a non-trivial problem; there are 181,679 ordinary files in the build directory, and I had no interest in doing this identification by hand.

In theory, one could just look at the file extensions (.c for C, .py for python), but this is not enough in practice. Some packages reuse extensions if the package doesn't use that kind of file (e.g., the ``.exp" extension of expect was used by some packages as ``export" files, and the ``.m" of objective-C was used by some packages for module information extracted from C code). Some files don't have extensions, particularly scripts. And finally, files automatically generated by another program should not be counted, since I wished to use the results to estimate effort.

I ended up writing a program of over 600 lines of Perl to perform this identification, which used a number of heuristics to categorize each file into categories. There is a category for each language, plus the categories non-programs, unknown (useful for scanning for problems), automatically generated program files, duplicate files (whose file contents duplicated other files), and zero-length files.

The program first checked for well-known extensions (such as .gif) that cannot be program files, and for a number of common generated filenames. It then peeked at the first line for "#!" followed by a legal script name. If that didn't work, it used the extension to try to determine the category. For a number of languages, the extension was not reliable, so for those languages the categorization program examined the file contents and used a set of heuristics to determine if the file actually belonged that category. If all else failed, the file was placed in the ``unknown" category for later analysis. I later looked at the ``unknown" items, checking the common extensions to ensure I had not missed any common types of code.

One complicating factor was that I wished to separate C, C++, and objective-C code, but a header file ending with ```.h"` or ```.hpp"` file could be any of them. I developed a number of heuristics to determine, for each file, what language it belonged to. For example, if a build directory has exactly one of these languages, determining the correct category for header files is easy. Similarly, if there is exactly one of these in the directory with the header file, it is presumed to be that kind. Finally, a header file with the keyword ```class"` is almost certainly not a C header file, but a C++ header file.

Detecting automatically generated files was not easy, and it's quite conceivable I missed a number of them. The first 15 lines were examined, to determine if any of them included at the beginning of the line (after spaces and possible comment markers) one of the following phrases: ```generated automatically"`, ```automatically generated"`, ```this is a generated file"`, ```generated with the (something) utility"`, or ```do not edit"`. A number of filename conventions were used, too. For example, any ```configure"` file is presumed to be automatically generated if there's a ```configure.in"` file in the same directory.

To eliminate duplicates, the program kept md5 checksums of each program file. Any given md5 checksum would only be counted once. Build directories were processed alphabetically, so this meant that if the same file content was in both directories ```a"` and ```b"`, it would be counted only once as being part of ```a"`. Thus, some packages with names later in the alphabet may appear smaller than would make sense at first glance.

It's important to note that different rules could be used to ```count"` lines of code. Some kinds of code were intentionally excluded from the count. Many RPM packages include a number of shell commands used to install and uninstall software; the estimate in this paper does not include the code in RPM packages. This estimate also does not include the code in Makefiles (which can be substantive). In both cases, the code in these cases is often cut and pasted from other similar files, so counting such code would probably overstate the actual development effort. In addition, Makefiles are often automatically generated.

On the other hand, this estimate does include some code that others might not count. This estimate includes test code included with the package, which isn't visible directly to users (other than hopefully higher quality of the executable program). It also includes code not used in this particular system, such as code for other architectures and OS's, bindings for languages not compiled into the binaries, and compilation-time options not chosen. I decided to include such code for two reasons. First, this code is validly represents the effort to build each component. Second, it does represent indirect value to the user, because the user can later use those components in other circumstances even if the user doesn't choose to do so by default.

So, after the work of categorizing the files, the following categories of files were created for each build directory (common extensions are shown in parentheses, and the name used in the data tables below are shown in brackets):

1. C (.c) [ansic]
2. C++ (.C, .cpp, .cxx, .cc) [cpp]
3. LISP (.el, .scm, .lsp, .jl) [lisp]
4. shell (.sh) [sh]
5. Perl (.pl, .pm, .perl) [perl]
6. Assembly (.s, .S, .asm) [asm]
7. TCL (.tcl, .tk, .itk) [tcl]
8. Python (.py) [python]
9. Yacc (.y) [yacc]
10. Java (.java) [java]
11. Expect (.exp) [exp]
12. lex (.l) [lex]
13. awk (.awk) [awk]
14. Objective-C (.m) [objc]
15. C shell (.csh) [csh]
16. Ada (.ada, .ads, .adb) [ada]
17. Pascal (.p) [pascal]
18. sed (.sed) [sed]
19. Fortran (.f) [fortran]

Note that we're counting Scheme as a dialect of LISP, and Expect is being counted separately from TCL. The command line shells Bourne shell, the Bourne-again shell (bash), and the K shell are all counted together as ```shell"`, but the C shell (csh and tcsh) is counted separately.

A.3 Counting Lines of Code

Every language required its own counting scheme. This was more complex than I realized; there were a number of languages involved.

I originally tried to use USC's ``CodeCount" tools to count the code. Unfortunately, this turned out to be buggy and did not handle most of the languages used in the system, so I eventually abandoned it for this task and wrote my own tools. Those who wish to use this tool are welcome to do so; you can learn more from its web site at <http://sunset.usc.edu/research/CODECOUNT>.

I did manage to use the CodeCount to compute the logical source lines of code for the C portions of the linux kernel. This came out to be 673,627 logical source lines of code, compared to the 1,462,165 lines of physical code (again, this ignores files with duplicate contents).

Since there were a large number of languages to count, I used the ``physical lines of code" definition. In this definition, a line of code is a line (ending with newline or end-of-file) with at least one non-comment non-whitespace character. These are known as ``non-comment non-blank" lines. If a line only had whitespace (tabs and spaces) it was not counted, even if it was in the middle of a data value (e.g., a multiline string). It is much easier to write programs to measure this value than to measure the ``logical" lines of code, and this measure can be easily applied to widely different languages. Since I had to process a large number of different languages, it made sense to choose the measure that is easier to obtain.

[Park \[1992\]](#) presents a framework of issues to be decided when trying to count code. Using Park's framework, here is how code was counted in this paper:

1. Statement Type: I used a physical line-of-code as my basis. I included executable statements, declarations (e.g., data structure definitions), and compiler directives (e.g., preprocessor commands such as #define). I excluded all comments and blank lines.
2. How Produced: I included all programmed code, including any files that had been modified. I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. If a file was in the source package, I included it; if the file had been removed from a source package (including via a patch), I did not include it.
3. Origin: I included all code included in the package.
4. Usage: I included code in or part of the primary product; I did not include code external to the product (i.e., additional applications able to run on the system but not included with the system).
5. Delivery: I counted code delivered as source; not surprisingly, I didn't count code not delivered as source. I also didn't count undelivered code.
6. Functionality: I included both operative and inoperative code. An examples of intentionally ``inoperative" code is code turned off by #ifdef commands; since it could be turned on for special purposes, it made sense to count it. An examples of unintentionally ``inoperative" code is dead or unused code.
7. Replications: I included master (original) source statements. I also included ``physical replicates of master statements stored in the master code". This is simply code cut and pasted from one place to another to reuse code; it's hard to tell where this happens, and since it has to be maintained separately, it's fair to include this in the measure. I excluded copies inserted, instantiated, or expanded when compiling or linking, and I excluded postproduction replicates (e.g., reparameterized systems).
8. Development Status: Since I only measured code included in the packages used to build the delivered system, I declared that all software I was measuring had (by definition) passed whatever ``system tests" were required by that component's developers.
9. Languages: I included all languages, as identified earlier in section A.2.
10. Clarifications: I included all statement types. This included nulls, continues, no-ops, lone semicolons, statements that instantiate generics, lone curly braces ({ and }), and labels by themselves.

Park includes in his paper a ``basic definition" of physical lines of code, defined using his framework. I adhered to Park's definition unless (1) it was impossible in my technique to do so, or (2) it would appear to make the result inappropriate for use in cost estimation (using COCOMO). COCOMO states that source code:

``includes all program instructions created by project personnel and processed into machine code by some combination of preprocessors, compilers, and assemblers. It excludes comment cards and unmodified utility software. It includes job control language, format statements, and data declarations. Instructions are defined as lines of code."

In summary, though in general I followed Park's definition, I didn't follow Park's ``basic definition" in the following ways:

1. How Produced: I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. After all, COCOMO states that the only code that should be counted is code ``produced by project personnel'', whereas these kinds of files are instead the output of ``preprocessors and compilers.'' If code is always maintained as the input to a code generator, and then the code generator is re-run, it's only the code generator input's size that validly measures the size of what is maintained. Note that while I attempted to exclude generated code, this exclusion is based on heuristics which may have missed some cases.
2. Origin: Normally physical SLOC doesn't include an unmodified ``vendor-supplied language support library'' nor a ``vendor-supplied system or utility''. However, in this case this was *exactly* what I was measuring, so I naturally included these as well.
3. Delivery: I didn't count code not delivered as source. After all, since I didn't have it, I couldn't count it.
4. Functionality: I included unintentionally inoperative code (e.g., dead or unused code). There might be such code, but it is very difficult to automatically detect in general for many languages. For example, a program not directly invoked by anything else nor installed by the installer is much more likely to be a test program, which I'm including in the count. Clearly, discerning human ``intent'' is hard to automate. Hopefully, unintentionally inoperative code is a small amount of the total delivered code.

Otherwise, I followed Park's ``basic definition'' of a physical line of code, even down to Park's language-specific definitions where Park defined them for a language.

One annoying problem was that one file wasn't syntactically correct and it affected the count. File `/usr/src/redhat/BUILD/cdrecord-1.8/mkiso` had an `#ifdef` not taken, and the road not taken had a missing double-quote mark before the word ``cannot'':

```
#ifdef  USE_LIBSCHILY
        comerr(Cannot open '%s'.\n", filename);
#endif
        perror ("fopen");
        exit (1);
#endif
```

I solved this by hand-patching the source code (for purposes of counting). There were also some files with intentionally erroneous code (e.g., compiler error tests), but these did not impact the SLOC count.

Several languages turn out to be non-trivial to count:

- In C, C++, and Java, comment markers should be ignored inside strings. Since they have multi-line comment markers this requirement should not be ignored, or a ``/*'' inside a string could cause most of the code to be erroneously uncounted.
- Officially, C doesn't have C++'s ``/*'' comment marker, but the gcc compiler accepts it and a great deal of C code uses it, so my counters accepted it.
- Perl permits in-line ``perlpod'' documents, ``here'' documents, and an `__END__` marker that complicate code-counting. Perlpod documents are essentially comments, but a ``here'' document may include text to generate them (in which case the perlpod document is data and should be counted). The `__END__` marker indicates the end of the file from Perl's viewpoint, even if there's more text afterwards.
- Python has a convention that, at the beginning of a definition (e.g., of a function, method, or class), an unassigned string can be placed to describe what's being defined. Since this is essentially a comment (though it doesn't syntactically look like one), the counter must avoid counting such strings, which may have multiple lines. To handle this, strings which started the beginning of a line were not counted. Python also has the ``triple quote'' operator, permitting multiline strings; these needed to be handled specially. Triple quote strings were normally considered as data, regardless of content, unless they were used as a comment about a definition.
- Assembly languages vary greatly in the comment character they use, so my counter had to handle this variance. I wrote a program which first examined the file to determine if C-style ``/*'' comments and C preprocessor commands (e.g., `#include`) were used. If both ``/*'' and ``*/'' were in the file, it was assumed that C-style comments were used, since it is unlikely that *both* would be used as something else (e.g., as string data) in the same assembly language file. Determining if a file used the C preprocessor was trickier, since many assembly files do use ``#'' as a comment character and some preprocessor directives are ordinary words that might be included in a human comment. The heuristic used was: if `#ifdef`, `#endif`, or `#include` are used, the preprocessor is used; if at least three lines have either `#define` or `#else`, then the preprocessor is used. No doubt other heuristics are possible, but this at least seemed to produce reasonable results. The program then determined what the comment character was, by identifying which punctuation mark (from a set of possible marks) was the most common non-space initial character on a line (ignoring ``/*'' and ``#'' if C comments or preprocessor commands, respectively, were used). Once the comment character had been

determined, and it had been determined if C-style comments were also allowed, the lines of code could be counted in the file.

Although their values are not used in estimating effort, I also counted the number of files; summaries of these values are included in appendix B.

Since the linux kernel was the largest single component, and I had questions about the various inconsistencies in the "Halloween" documents, I made additional measures of the Linux kernel.

A.4 Estimating Effort and Costs

For each build directory, I totalled the source lines of code (SLOC) for each language, then totalled those values to determine the SLOC for each directory. I then used the basic Constructive Cost Model (COCOMO) to estimate effort. The basic model is the simplest (and least accurate) model, but I simply did not have the additional information necessary to use the more complex (and more accurate) models. COCOMO is described in depth by Boehm [1981].

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions.

In the basic COCOMO model, estimated man-months of effort, design through test, equals $2.4 \cdot (KSLOC)^{1.05}$, where KSLOC is the total physical SLOC divided by 1000.

I assumed that each package was built completely independently and that there were no efforts necessary for integration not represented in the code itself. This almost certainly underestimates the true costs, but for most packages it's actually true (many packages don't interact with each other at all). I wished to underestimate (instead of overestimate) the effort and costs, and having no better model, I assumed the simplest possible integration effort. This meant that I applied the model to each component, then summed the results, as opposed to applying the model once to the grand total of all software.

Note that the only input to this model is source lines of code, so some factors simply aren't captured. For example, creating some kinds of data (such as fonts) can be very time-consuming, but this isn't directly captured by this model. Some programs are intentionally designed to be data-driven, that is, they're designed as small programs which are driven by specialized data. Again, this data may be as complex to develop as code, but this is not counted.

Another example of uncaptured factors is the difficulty of writing kernel code. It's generally acknowledged that writing kernel-level code is more difficult than most other kinds of code, because this kind of code is subject to a subtle timing and race conditions, hardware interactions, a small stack, and none of the normal error protections. In this paper I do not attempt to account for this. You could try to use the Intermediate COCOMO model to try to account for this, but again this requires knowledge of other factors that can only be guessed at. Again, the effort estimation probably significantly underestimates the actual effort represented here.

It's worth noting that there is an update to COCOMO, COCOMO II. However, COCOMO II requires as its input logical (not physical) SLOC, and since this measure is much harder to obtain, I did not pursue it for this paper. More information about COCOMO II is available at the web site <http://sunset.usc.edu/research/COCOMOII/index.html>. A nice overview paper where you can learn more about software metrics is Masse [1997].

I assumed that an average U.S. programmer/analyst salary in the year 2000 was \$56,286 per year; this value was from the ComputerWorld, September 4, 2000's Salary Survey. Overhead is much harder to estimate; I did not find a definitive source for information on overheads. After informal discussions with several cost analysts, I determined that an overhead of 2.4 would be representative of the overhead sustained by a typical software development company. Should you disagree with these figures, I've provided all the information necessary to recalculate your own cost figures; just start with the effort estimates and recalculate cost yourself.

Appendix B. More Detailed Results

This appendix provides some more detailed results. B.1 lists the SLOC found in each build directory; B.2 shows counts of files for each category of file; B.3 presents some additional measures about the Linux kernel. B.4 presents some SLOC totals of putatively "minimal" systems. You can learn more at <http://www.dwheeler.com/sloc>.

B.1 SLOC in Build Directories

The following is a list of all build directories, and the source lines of code (SLOC) found in them, followed by a few statistics counting files (instead of SLOC).

Remember that duplicate files are only counted once, with the build directory "first in ASCII sort order" receiving any duplicates (to break ties). As a result, some build directories have a smaller number than might at first make sense. For example, the "kudzu" build directory does contain code, but all of it is also contained in the "Xconfigurator" build directory.. and since that directory sorts first, the kudzu package is considered to have "no code".

The columns are SLOC (total physical source lines of code), Directory (the name of the build directory, usually the same or similar to the package name), and SLOC-by-Language (Sorted). This last column lists languages by name and the number of SLOC in that language; zeros are not shown, and the list is sorted from largest to smallest in that build directory. Similarly, the directories are sorted from largest to smallest total SLOC.

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414, yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358, yacc=2710,perl=711,awk=393,lex=383,sed=57,cs=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988, lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139, yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253, cs=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606, cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910, perl=464,sed=16,cs=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762, awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399, lex=1642,perl=1206,python=959,cpp=746,asm=70,cs=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528, awk=393,python=348,lex=190,cs=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464, perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342, sed=2
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546, yacc=1507,awk=522,lex=323,sed=297,asm=139,cs=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360, cs=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,cs=235, sed=35,lisp=12

133193	kaffe-1.0.5	java=65275,ansic=62125,cpp=3923,perl=972,sh=814, asm=84
131372	jade-1.2.1	cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177
127536	pine4.21	ansic=126678,sh=766,csch=62,perl=30
121878	ImageMagick-4.2.9	ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
119613	lynx2-8-3	ansic=117385,sh=1860,perl=340,csch=28
116951	mc-4.5.42	ansic=114406,sh=1996,perl=345,awk=148,csch=56
116615	gnumeric-0.48	ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
113272	xlispstat-3-52-17	ansic=91484,lisp=21769,sh=18,csch=1
113241	vim-5.6	ansic=111724,awk=683,sh=469,perl=359,csch=6
109824	php-3.0.15	ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
104032	linuxconf-1.17r2	cpp=93139,perl=4570,sh=2984,java=2741,ansic=598
102674	libgr-2.0.13	ansic=99647,sh=2438,csch=589
100951	lam-6.3.1	ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187, csch=19
99066	krb4-1.0	ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765, yacc=1509,lex=236,awk=33
94637	xlockmore-4.15	ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
93940	kdenetwork	cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
92964	samba-2.0.6	ansic=88308,sh=3557,perl=831,awk=158,csch=110
91213	anaconda-6.2.2	ansic=74303,python=13657,sh=1583,yacc=810,lex=732, perl=128
89959	xscreensaver-3.23	ansic=88488,perl=1070,sh=401
88128	cvs-1.10.7	ansic=68303,sh=17909,perl=902,yacc=826,csch=181,lisp=7
87940	isd4k-utils	ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
85383	xpdf-0.90	cpp=60427,ansic=21400,sh=3556
81719	inn-2.2.2	ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547, lex=249,tcl=3
80343	kdelibs	cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116, sh=35
79997	WindowMaker-0.61.1	ansic=77924,sh=1483,perl=371,lisp=219
78787	extace-1.2.15	ansic=66571,sh=9322,perl=2894
77873	apache_1.3.12	ansic=69191,sh=6781,perl=1846,cpp=55
75257	xpilot-4.1.0	ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68
73817	w3c-libwww-5.2.8	ansic=64754,sh=4678,cpp=3181,perl=1204
72726	ucd-snmp-4.1.1	ansic=64411,perl=5558,sh=2757
72425	gnome-core-1.0.55	ansic=72230,perl=141,sh=54
71810	jikes	cpp=71452,java=358
70260	groff-1.15	cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397, sh=265,sed=46
69265	fvwm-2.2.4	ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
69246	linux-86	ansic=63328,asm=5276,sh=642
68997	blt2.4g	ansic=58630,tcl=10215,sh=152
68884	squid-2.3.STABLE1	ansic=66305,sh=1570,perl=1009
68560	bash-2.03	ansic=56758,sh=7264,yacc=2808,perl=1730
68453	kdegraphics	cpp=34208,ansic=29347,sh=4898
65722	xntp3-5.93	ansic=60190,perl=3633,sh=1445,awk=417,asm=37
62922	ppp-2.3.11	ansic=61756,sh=996,exp=82,perl=44,csch=44
62137	sgml-tools-1.0.9	cpp=38543,ansic=19185,perl=2866,lex=560,sh=532, lisp=309,awk=142
61688	imap-4.7	ansic=61628,sh=60
61324	ncurses-5.0	ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103, sed=100
60429	kdesupport	ansic=42421,cpp=17810,sh=173,awk=13,csch=12
60302	openldap-1.2.9	ansic=58078,sh=1393,perl=630,python=201
57217	xfig.3.2.3-beta-1	ansic=57212,csch=5
56093	lsof_4.47	ansic=50268,sh=4753,perl=856,awk=214,asm=2
55667	uucp-1.06.1	ansic=52078,sh=3400,perl=189
54935	gnupg-1.0.1	ansic=48884,asm=4586,sh=1465

Estimating Linux's Size

54603	glade-0.5.5	ansic=49545,sh=5058
54431	svglib-1.4.1	ansic=53725,asm=630,perl=54,sh=22
53141	AfterStep-1.8.0	ansic=50898,perl=1168,sh=842,cpp=233
52808	kdeutils	cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
52574	nmh-1.0.3	ansic=50698,sh=1785,awk=74,sed=17
51813	freetype-1.3.1	ansic=48929,sh=2467,cpp=351,cs=53,perl=13
51592	enlightenment-0.15.5	ansic=51569,sh=23
50970	cdrecord-1.8	ansic=48595,sh=2177,perl=194,sed=4
49370	tin-1.4.2	ansic=47763,sh=908,yacc=699
49325	imlib-1.9.7	ansic=49260,sh=65
48223	kdemultimedia	ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
47067	bash-1.14.7	ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
46312	tcsh-6.09.00	ansic=43544,sh=921,lisp=669,perl=593,cs=585
46159	unzip-5.40	ansic=40977,cpp=3778,asm=1271,sh=133
45811	mutt-1.0.1	ansic=45574,sh=237
45589	am-utils-6.0.3	ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
45485	guile-1.3	ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,cs=50
45378	gnuplot-3.7.1	ansic=43276,lisp=661,asm=539,objc=387,cs=297,perl=138,sh=80
44323	mgetty-1.1.21	ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
42880	sendmail-8.9.3	ansic=40364,perl=1737,sh=779
42746	elm2.5.3	ansic=32931,sh=9774,awk=41
41388	p2c-1.22	ansic=38788,pascal=2499,perl=101
41205	gnome-games-1.0.51	ansic=31191,lisp=6966,cpp=3048
39861	rpm-3.0.4	ansic=36994,sh=1505,perl=1355,python=7
39160	util-linux-2.10f	ansic=38627,sh=351,perl=65,cs=62,sed=55
38927	xmms-1.0.1	ansic=38366,asm=398,sh=163
38548	ORBit-0.5.0	ansic=35656,yacc=1750,sh=776,lex=366
38453	zsh-3.0.7	ansic=36208,sh=1763,perl=331,awk=145,sed=6
37515	ircii-4.4	ansic=36647,sh=852,lex=16
37360	tiff-v3.5.4	ansic=32734,sh=4054,cpp=572
36338	textutils-2.0a	ansic=18949,sh=16111,perl=1218,sed=60
36243	exmh-2.1.1	tcl=35844,perl=316,sh=49,exp=34
36239	x11amp-0.9-alpha3	ansic=31686,sh=4200,asm=353
35812	xloadimage.4.1	ansic=35705,sh=107
35554	zip-2.3	ansic=32108,asm=3446
35397	gtk-engines-0.10	ansic=20636,sh=14761
35136	php-2.0.1	ansic=33991,sh=1056,awk=89
34882	pmake	ansic=34599,sh=184,awk=58,sed=41
34772	xpuzzles-5.4.1	ansic=34772
34768	fileutils-4.0p	ansic=31324,sh=2042,yacc=841,perl=561
33203	strace-4.2	ansic=30891,sh=1988,perl=280,lisp=44
32767	trn-3.6	ansic=25264,sh=6843,yacc=660
32277	pilot-link.0.9.3	ansic=26513,java=2162,cpp=1689,perl=971,yacc=660,python=268,tcl=14
31994	korganizer	cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
31174	ncftp-3.0beta21	ansic=30347,cpp=595,sh=232
30438	gnome-pim-1.0.55	ansic=28665,yacc=1773
30122	scheme-3.2	lisp=19483,ansic=10515,sh=124
30061	tcpdump-3.4	ansic=29208,yacc=236,sh=211,lex=206,awk=184,cs=16
29730	screen-3.9.5	ansic=28156,sh=1574
29315	jed	ansic=29315
29091	xchat-1.4.0	ansic=28894,perl=121,python=53,sh=23
28897	ncpfs-2.2.0.17	ansic=28689,sh=182,tcl=26
28449	slrn-0.9.6.2	ansic=28438,sh=11
28261	xfishtank-2.1tp	ansic=28261
28186	texinfo-4.0	ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
28169	e2fsprogs-1.18	ansic=27250,awk=437,sh=339,sed=121,perl=22
28118	slang	ansic=28118
27860	kdegames	cpp=27507,ansic=340,sh=13

Estimating Linux's Size

27117	librep-0.10	ansic=19381,lisp=5385,sh=2351
27040	mikmod-3.1.6	ansic=26975,sh=55,awk=10
27022	x3270-3.1.1	ansic=26456,sh=478,exp=88
26673	lout-3.17	ansic=26673
26608	Xaw3d-1.3	ansic=26235,yacc=247,lex=126
26363	gawk-3.0.4	ansic=19871,awk=2519,yacc=2046,sh=1927
26146	libxml-1.8.6	ansic=26069,sh=77
25994	xrn-9.02	ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31, csh=13
25915	gv-3.5.8	ansic=25821,sh=94
25479	xpaint	ansic=25456,sh=23
25236	shadow-19990827	ansic=23464,sh=883,yacc=856,perl=33
24910	kdeadmin	cpp=19919,sh=3936,perl=1055
24773	pdksh-5.2.14	ansic=23599,perl=945,sh=189,sed=40
24583	gmp-2.0.2	ansic=17888,asm=5252,sh=1443
24387	mars_nwe	ansic=24158,sh=229
24270	gnome-python-1.0.51	python=14331,ansic=9791,sh=148
23838	kterm-6.2.0	ansic=23838
23666	enscript-1.6.1	ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
22373	sawmill-0.24	ansic=11038,lisp=8172,sh=3163
22279	make-3.78.1	ansic=19287,sh=2029,perl=963
22011	libpng-1.0.5	ansic=22011
21593	xboard-4.0.5	ansic=20640,lex=904,sh=41,csh=5,sed=3
21010	netkit-telnet-0.16	ansic=14796,cpp=6214
20433	pam-0.72	ansic=18936,yacc=634,sh=482,perl=321,lex=60
20125	ical-2.2	cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
20078	gd1.3	ansic=19946,perl=132
19971	wu-ftpd-2.6.0	ansic=17572,yacc=1774,sh=421,perl=204
19500	gnome-utils-1.0.50	ansic=18099,yacc=824,lisp=577
19065	joe	ansic=18841,asm=224
18885	X11R6-contrib-3.3.2	ansic=18616,lex=161,yacc=97,sh=11
18835	glib-1.2.6	ansic=18702,sh=133
18151	git-4.3.19	ansic=16166,sh=1985
18020	xboing	ansic=18006,sh=14
17939	sh-utils-2.0	ansic=13366,sh=3027,yacc=871,perl=675
17765	mtools-3.9.6	ansic=16155,sh=1602,sed=8
17750	gettext-0.10.35	ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9
17682	bc-1.05	ansic=9186,sh=7236,yacc=967,lex=293
17271	fetchmail-5.3.1	ansic=13441,python=1490,sh=1246,yacc=411,perl=321, lex=238,awk=124
17259	sox-12.16	ansic=16659,sh=600
16785	control-center-1.0.51	ansic=16659,sh=126
16266	dhcp-2.0	ansic=15328,sh=938
15967	SVGATextMode-1.9-src	ansic=15079,yacc=340,sh=294,lex=227,sed=15, asm=12
15868	kpilot-3.1b9	cpp=8613,ansic=5640,yacc=1615
15851	taper-6.9a	ansic=15851
15819	mpg123-0.59r	ansic=14900,asm=919
15691	transfig.3.2.1	ansic=15643,sh=38,csh=10
15638	mod_perl-1.21	perl=10278,ansic=5124,sh=236
15522	console-tools-0.3.3	ansic=13335,yacc=986,sh=800,lex=291,perl=110
15456	rpm2html-1.2	ansic=15334,perl=122
15143	gnotepad+-1.1.4	ansic=15143
15108	GXedit1.23	ansic=15019,sh=89
15087	mm2.7	ansic=8044,csh=6924,sh=119
14941	readline-2.2.1	ansic=11375,sh=1890,perl=1676
14912	ispell-3.1	ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385, csh=221,sh=157,perl=85,sed=15
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,csh=29
14774	flex-2.5.4	ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12

Estimating Linux's Size

14587	multimedia	ansic=14577,sh=10
14516	libgtop-1.0.6	ansic=13768,perl=653,sh=64,asm=31
14427	mawk-1.2.2	ansic=12714,yacc=994,awk=629,sh=90
14363	automake-1.4	perl=10622,sh=3337,ansic=404
14350	rsync-2.4.1	ansic=13986,perl=179,sh=126,awk=59
14299	nfs-utils-0.1.6	ansic=14107,sh=165,perl=27
14269	rcs-5.7	ansic=12209,sh=2060
14255	tar-1.13.17	ansic=13014,lisp=592,sh=538,perl=111
14105	wmakerconf-2.1	ansic=13620,perl=348,sh=137
14039	less-346	ansic=14032,awk=7
13779	rxvt-2.6.1	ansic=13779
13586	wget-1.5.3	ansic=13509,perl=54,sh=23
13504	rp3-1.0.7	cpp=10416,ansic=2957,sh=131
13241	iproute2	ansic=12139,sh=1002,perl=100
13100	silo-0.9.8	ansic=10485,asm=2615
12657	macutils	ansic=12657
12639	libungif-4.1.0	ansic=12381,sh=204,perl=54
12633	minicom-1.83.0	ansic=12503,sh=130
12593	audiofile-0.1.9	sh=6440,ansic=6153
12463	gnome-objc-1.0.2	objc=12365,sh=86,ansic=12
12313	jpeg-6a	ansic=12313
12124	ypserv-1.3.9	ansic=11622,sh=460,perl=42
11790	lrzsz-0.12.20	ansic=9512,sh=1263,exp=1015
11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
11633	net-tools-1.54	ansic=11531,sh=102
11404	findutils-4.1	ansic=11160,sh=173,exp=71
11299	xmorphism-1999dec12	ansic=10783,tcl=516
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
10914	diffutils-2.7	ansic=10914
10404	gnorpm-0.9	ansic=10404
10271	gqview-0.7.0	ansic=10271
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
10187	dump-0.4b15	ansic=9422,sh=760,sed=5
10088	piranha	ansic=10048,sh=40
10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
9961	procps-2.0.6	ansic=9959,sh=2
9942	xpat2-1.04	ansic=9942
9927	procmail-3.14	ansic=8090,sh=1837
9873	nss_ldap-105	ansic=9784,perl=89
9801	man-1.5hl	ansic=7377,sh=1802,perl=317,awk=305
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6
9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
9699	bison-1.28	ansic=9650,sh=49
9666	ash-linux-0.2	ansic=9445,sh=221
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
9551	pwdb-0.61	ansic=9488,sh=63
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
9263	ctags-3.4	ansic=9240,sh=23
9138	gftp-2.0.6a	ansic=9138
8939	mkisofs-1.12b5	ansic=8939
8766	pxe-linux	cpp=4463,ansic=3622,asm=681
8572	psgml-1.2.1	lisp=8572
8540	xxgdb-1.12	ansic=8540
8491	gtop-1.0.5	ansic=8151,cpp=340
8356	gedit-0.6.1	ansic=8225,sh=131
8303	dip-3.3.7o	ansic=8207,sh=96
7859	libglade-0.11	ansic=5898,sh=1809,python=152
7826	xpm-3.4k	ansic=7750,sh=39,cpp=37

Estimating Linux's Size

7740	sed-3.02	ansic=7301,sed=359,sh=80
7617	cpio-2.4.2	ansic=7598,sh=19
7615	esound-0.2.17	ansic=7387,sh=142,csch=86
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
7427	ed-0.2	ansic=7263,sh=164
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,cpp=3
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
7095	xgammon-0.98	ansic=6506,lex=589
7041	newt-0.50.8	ansic=6526,python=515
7030	ee-0.3.11	ansic=7007,sh=23
6976	aboot-0.5	ansic=6680,asm=296
6968	mailx-8.1.1	ansic=6963,sh=5
6877	lpr	ansic=6842,sh=35
6827	gnome-media-1.0.51	ansic=6827
6646	iputils	ansic=6646
6611	patch-2.5	ansic=6561,sed=50
6592	xosview-1.7.1	cpp=6205,ansic=367,awk=20
6550	byacc-1.9	ansic=5520,yacc=1030
6496	pidentd-3.0.10	ansic=6475,sh=21
6391	m4-1.4	ansic=5993,lisp=243,sh=155
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
6234	awesfx-0.4.3a	ansic=6234
6172	sash-3.4	ansic=6172
6116	lslk	ansic=5325,sh=791
6090	joystick-1.2.15	ansic=6086,sh=4
6072	kdoc	perl=6010,sh=45,cpp=17
6043	irda-utils-0.9.10	ansic=5697,sh=263,perl=83
6033	sysvinit-2.78	ansic=5256,sh=777
6025	pnm2ppa	ansic=5708,sh=317
6021	rpmfind-1.4	ansic=6021
5981	indent-2.2.5	ansic=5958,sh=23
5975	ytalk-3.1	ansic=5975
5960	isapnptools-1.21	ansic=4394,yacc=1383,perl=123,sh=60
5744	gdm-2.0beta2	ansic=5632,sh=112
5594	isdn-config	cpp=3058,sh=2228,perl=308
5526	efax-0.9	ansic=4570,sh=956
5383	acct-6.3.2	ansic=5016,cpp=287,sh=80
5115	libtool-1.3.4	sh=3374,ansic=1741
5111	netkit-ftp-0.16	ansic=5111
4996	bzip2-0.9.5d	ansic=4996
4895	xcpustate-2.5	ansic=4895
4792	libelf-0.6.4	ansic=3310,sh=1482
4780	make-3.78.1_pvm-0.5	ansic=4780
4542	gpgp-0.4	ansic=4441,sh=101
4430	gperf-2.7	cpp=2947,exp=745,ansic=695,sh=43
4367	aumix-1.30.1	ansic=4095,sh=179,sed=93
4087	zlib-1.1.3	ansic=2815,asm=712,cpp=560
4038	sysklogd-1.3-31	ansic=3741,perl=158,sh=139
4024	rep-gtk-0.8	ansic=2905,lisp=971,sh=148
3962	netkit-timed-0.16	ansic=3962
3929	initscripts-5.00	sh=2035,ansic=1866,csch=28
3896	ltracex-0.3.10	ansic=2986,sh=854,awk=56
3885	phhttpd-0.1.0	ansic=3859,sh=26
3860	xdaliclock-2.18	ansic=3837,sh=23
3855	pciutils-2.1.5	ansic=3800,sh=55
3804	quota-2.00-pre3	ansic=3795,sh=9
3675	dosfstools-2.2	ansic=3675
3654	tcp_wrappers_7.6	ansic=3654
3651	ipchains-1.3.9	ansic=2767,sh=884
3625	autofs-3.1.4	ansic=2862,sh=763

```

3588 netkit-rsh-0.16 ansic=3588
3438 yp-tools-2.4 ansic=3415,sh=23
3433 dialog-0.6 ansic=2834,perl=349,sh=250
3415 ext2ed-0.1 ansic=3415
3315 gdbm-1.8.0 ansic=3290,cpp=25
3245 ypbind-3.3 ansic=1793,sh=1452
3219 playmidi-2.4 ansic=3217,sed=2
3096 xtrojka123 ansic=3087,sh=9
3084 at-3.1.7 ansic=1442,sh=1196,yacc=362,lex=84
3051 dhcpcd-1.3.18-pl3 ansic=2771,sh=280
3012 apmd ansic=2617,sh=395
2883 netkit-base-0.16 ansic=2883
2879 vixie-cron-3.0.1 ansic=2866,sh=13
2835 gkernit-1.0 ansic=2835
2810 kdetoys cpp=2618,ansic=192
2791 xjewel-1.6 ansic=2791
2773 mpage-2.4 ansic=2704,sh=69
2758 autoconf-2.13 sh=2226,perl=283,exp=167,ansic=82
2705 autorun-2.61 sh=1985,cpp=720
2661 cdp-0.33 ansic=2661
2647 file-3.28 ansic=2601,perl=46
2645 libghttp-1.0.4 ansic=2645
2631 getty_ps-2.0.7j ansic=2631
2597 pythonlib-1.23 python=2597
2580 magicdev-0.2.7 ansic=2580
2531 gnome-kerberos-0.2 ansic=2531
2490 sndconfig-0.43 ansic=2490
2486 bug-buddy-0.7 ansic=2486
2459 usermode-1.20 ansic=2459
2455 fnlib-0.4 ansic=2432,sh=23
2447 sliplogin-2.1.1 ansic=2256,sh=143,perl=48
2424 raidtools-0.90 ansic=2418,sh=6
2423 netkit-routed-0.16 ansic=2423
2407 nc ansic=1670,sh=737
2324 up2date-1.13 python=2324
2270 memprof-0.3.0 ansic=2270
2268 which-2.9 ansic=1398,sh=870
2200 printtool tcl=2200
2163 gnome-linuxconf-0.25 ansic=2163
2141 unarj-2.43 ansic=2141
2065 units-1.55 ansic=1963,perl=102
2048 netkit-ntalk-0.16 ansic=2048
1987 cracklib,2.7 ansic=1919,perl=46,sh=22
1984 cleanfeed-0.95.7b perl=1984
1977 wmconfig-0.9.8 ansic=1941,sh=36
1941 isicom ansic=1898,sh=43
1883 slocate-2.1 ansic=1802,sh=81
1857 netkit-rusers-0.16 ansic=1857
1856 pump-0.7.8 ansic=1856
1842 cdecl-2.5 ansic=1002,yacc=765,lex=75
1765 fbset-2.1 ansic=1401,yacc=130,lex=121,perl=113
1653 adjtimex-1.9 ansic=1653
1634 netcfg-2.25 python=1632,sh=2
1630 psmisc ansic=1624,sh=6
1621 urlview-0.7 ansic=1515,sh=106
1604 fortune-mod-9708 ansic=1604
1531 netkit-tftp-0.16 ansic=1531
1525 logrotate-3.3.2 ansic=1524,sh=1
1473 traceroute-1.4a5 ansic=1436,awk=37
1452 time-1.7 ansic=1395,sh=57

```


Estimating Linux's Size

1435	ncompress-4.2.4	ansic=1435
1361	mt-st-0.5b	ansic=1361
1290	cxhextris	ansic=1290
1280	pam_krb5-1	ansic=1280
1272	bsd-finger-0.16	ansic=1272
1229	hdparm-3.6	ansic=1229
1226	procinfo-17	ansic=1145,perl=81
1194	passwd-0.64.1	ansic=1194
1182	auth_ldap-1.4.0	ansic=1182
1146	prtconf-1.3	ansic=1146
1143	anacron-2.1	ansic=1143
1129	xbill-2.0	cpp=1129
1099	popt-1.4	ansic=1039,sh=60
1088	nag	perl=1088
1076	stylesheets-0.13rh	perl=888,sh=188
1075	authconfig-3.0.3	ansic=1075
1049	kpppload-1.04	cpp=1044,sh=5
1020	MAKEDEV-2.5.2	sh=1020
1013	trojka	ansic=1013
987	xmailbox-2.5	ansic=987
967	netkit-rwho-0.16	ansic=967
953	switchdesk-2.1	ansic=314,perl=287,cpp=233,sh=119
897	portmap_4	ansic=897
874	ldconfig-1999-02-21	ansic=874
844	jpeg-6b	sh=844
834	ElectricFence-2.1	ansic=834
830	mouseconfig-4.4	ansic=830
816	rpmlint-0.8	python=813,sh=3
809	kdpms-0.2.8	cpp=809
797	termcap-2.0.8	ansic=797
787	xsysinfo-1.7	ansic=787
770	giftrans-1.12.2	ansic=770
742	setserial-2.15	ansic=742
728	tree-1.2	ansic=728
717	chkconfig-1.1.2	ansic=717
682	lpg	perl=682
657	eject-2.0.2	ansic=657
616	diffstat-1.27	ansic=616
592	netscape-4.72	sh=592
585	usernet-1.0.9	ansic=585
549	genromfs-0.3	ansic=549
548	tksysv-1.1	tcl=526,sh=22
537	minlabel-1.2	ansic=537
506	netkit-bootparamd-0.16	ansic=506
497	locale_config-0.2	ansic=497
491	helptool-2.4	perl=288,tcl=203
480	elftoaout-2.2	ansic=480
463	tmpwatch-2.2	ansic=311,sh=152
445	rhs-printfilters-1.63	sh=443,ansic=2
441	audioctl	ansic=441
404	control-panel-3.13	ansic=319,tcl=85
368	kbdconfig-1.9.2.4	ansic=368
368	vlock-1.3	ansic=368
367	timetool-2.7.3	tcl=367
347	kernelcfg-0.5	python=341,sh=6
346	timeconfig-3.0.3	ansic=318,sh=28
343	mingetty-0.9.4	ansic=343
343	chkfontpath-1.7	ansic=343
332	ethtool-1.0	ansic=332
314	mkbootdisk-1.2.5	sh=314

Estimating Linux's Size

```

302      symlinks-1.2      ansic=302
301      xsri-1.0          ansic=301
294      netkit-rwall-0.16 ansic=294
290      biff+comsat-0.16 ansic=290
288      mkinitrd-2.4.1   sh=288
280      stat-1.5          ansic=280
265      sysreport-1.0     sh=265
261      bdflush-1.5       ansic=202,asm=59
255      ipvsadm-1.1       ansic=255
255      sag-0.6-html      perl=255
245      man-pages-1.28    sh=244,sed=1
240      open-1.4          ansic=240
236      xtoolwait-1.2     ansic=236
222      utempter-0.5.2    ansic=222
222      mkkickstart-2.1   sh=222
221      hellas            sh=179,perl=42
213      rhmask            ansic=213
159      quickstrip-1.1    ansic=159
132      rdate-1.0         ansic=132
131      statserial-1.1    ansic=121,sh=10
107      fwhois-1.00       ansic=107
85       mktemp-1.5        ansic=85
82       modemtool-1.21    python=73,sh=9
67       setup-1.2         ansic=67
56       shaper            ansic=56
52       sparc32-1.1       ansic=52
47       intimed-1.10      ansic=47
23       locale-ja-9       sh=23
16       AnotherLevel-1.0.1 sh=16
11       words-2           sh=11
7        trXFree86-2.1.2   tcl=7
0        install-guide-3.2.html (none)
0        caching-nameserver-6.2 (none)
0        XFree86-ISO8859-2-1.0 (none)
0        rootfiles        (none)
0        ghostscript-fonts-5.50 (none)
0        kudzu-0.36        (none)
0        wvdial-1.41       (none)
0        mailcap-2.0.6     (none)
0        desktop-backgrounds-1.1 (none)
0        redhat-logos      (none)
0        solemul-1.1       (none)
0        dev-2.7.18        (none)
0        urw-fonts-2.0     (none)
0        users-guide-1.0.72 (none)
0        sgml-common-0.1   (none)
0        setup-2.1.8       (none)
0        jadetex           (none)
0        gnome-audio-1.0.0 (none)
0        specsps-6.2       (none)
0        gimp-data-extras-1.0.0 (none)
0        docbook-3.1       (none)
0        indexhtml-6.2     (none)

```

```

ansic:      14218806 (80.55%)
cpp:        1326212 (7.51%)
lisp:       565861 (3.21%)
sh:         469950 (2.66%)
perl:       245860 (1.39%)

```

asm:	204634	(1.16%)
tcl:	152510	(0.86%)
python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Total Physical Source Lines of Code (SLOC) = 17652561

Total Estimated Person-Years of Development = 4548.36

Average Programmer Annual Salary = 56286

Overhead Multiplier = 2.4

Total Estimated Cost to Develop = \$ 614421924.71

B.2 Counts of Files For Each Category

There were 181,679 ordinary files in the build directory. The following are counts of the number of files (*not* the SLOC) for each language:

ansic:	52088	(71.92%)
cpp:	8092	(11.17%)
sh:	3381	(4.67%)
asm:	1931	(2.67%)
perl:	1387	(1.92%)
lisp:	1168	(1.61%)
java:	1047	(1.45%)
python:	997	(1.38%)
tcl:	798	(1.10%)
exp:	472	(0.65%)
awk:	285	(0.39%)
objc:	260	(0.36%)
sed:	112	(0.15%)
yacc:	110	(0.15%)
csh:	94	(0.13%)
ada:	92	(0.13%)
lex:	57	(0.08%)
fortran:	50	(0.07%)
pascal:	7	(0.01%)

Total Number of Source Code Files = 72428

In addition, when counting the number of files (not SLOC), some files were identified as source code files but nevertheless were not counted for other reasons (and thus not included in the file counts above). Of these source code files, 5,820 files were identified as duplicating the contents of another file, 817 files were identified as files that had been automatically generated, and 65 files were identified as zero-length files.

B.3 Additional Measures of the Linux Kernel

I also made additional measures of the Linux kernel. This kernel is Linux kernel version 2.2.14 as patched by Red Hat. The Linux kernel's design is reflected in its directory structure. Only 8 lines of source code are in its main directory; the rest are in descendent directories. Counting the physical SLOC in each subdirectory (or its descendents) yielded the following:

BUILD/linux/Documentation/	765
BUILD/linux/arch/	236651
BUILD/linux/configs/	0
BUILD/linux/drivers/	876436
BUILD/linux/fs/	88667
BUILD/linux/ibcs/	16619
BUILD/linux/include/	136982
BUILD/linux/init/	1302
BUILD/linux/ipc/	1757
BUILD/linux/kernel/	7436
BUILD/linux/ksymoops-0.7c/	3271
BUILD/linux/lib/	1300
BUILD/linux/mm/	6771
BUILD/linux/net/	105549
BUILD/linux/pcmcia-cs-3.1.8/	34851
BUILD/linux/scripts/	8357

I separately ran the CodeCount tools on the entire linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for Linux.

However, this included non-i86 code. To make a more reasonable comparison with the Halloween documents, I needed to ignore non-i386 code.

First, I looked at the linux/arch directory, which contained architecture-specific code. This directory had the following subdirectories (architectures): alpha, arm, i386, m68k, mips, ppc, s390, sparc, sparc64. I then computed the total for all of ``arch'', which was 236651 SLOC, and subtracted out linux/arch/i386 code, which totalled to 26178 SLOC; this gave me a total of non-i386 code in linux/arch as 210473 physical SLOC. I then looked through the ``drivers'' directory to see if there were sets of drivers which were non-i386. I identified the following directories, with the SLOC totals as shown:

linux/drivers/sbus/	22354
linux/drivers/macintosh/	6000
linux/drivers/sgi/	4402
linux/drivers/fc4/	3167
linux/drivers/nubus/	421
linux/drivers/acorn/	11850
linux/drivers/s390/	8653

Driver Total: 56847

Thus, I had a grand total on non-i86 code (including drivers and architecture-specific code) as 267320 physical SLOC. This is, of course, another approximation, since there's certainly other architecture-specific lines, but I believe that is most of it. Running the CodeCount tool on just the C code, once these architectural and driver directories are removed, reveals a logical SLOC of 570,039 of C code.

B.4 Minimum System SLOC

Most of this paper worries about counting an ``entire'' system. However, what's the SLOC size of a ``minimal'' system? Here's an attempt to answer that question.

Red Hat Linux 6.2, CD-ROM #1, file RedHat/base/comps, defines the ``base'' (minimum) Red Hat Linux 6.2 installation as a set of packages. The following are the build directories corresponding to this base (minimum)

installation, along with the SLOC counts (as shown above). Note that this creates a text-only system:

Component	SLOC
anacron-2.1	1143
apmd	3012
ash-linux-0.2	9666
at-3.1.7	3084
authconfig-3.0.3	1075
bash-1.14.7	47067
bc-1.05	17682
bdflush-1.5	261
binutils-2.9.5.0.22	467120
bzip2-0.9.5d	4996
chkconfig-1.1.2	717
console-tools-0.3.3	15522
cpio-2.4.2	7617
cracklib, 2.7	1987
dev-2.7.18	0
diffutils-2.7	10914
dump-0.4b15	10187
e2fsprogs-1.18	28169
ed-0.2	7427
egcs-1.1.2	720112
eject-2.0.2	657
file-3.28	2647
fileutils-4.0p	34768
findutils-4.1	11404
gawk-3.0.4	26363
gd1.3	20078
gdbm-1.8.0	3315
getty_ps-2.0.7j	2631
glibc-2.1.3	415026
gmp-2.0.2	24583
gnupg-1.0.1	54935
gpm-1.18.1	9725
grep-2.4	10013
groff-1.15	70260
gzip-1.2.4a	6306
hdparm-3.6	1229
initscripts-5.00	3929
isapnptools-1.21	5960
kbdconfig-1.9.2.4	368
kernelcfg-0.5	347
kudzu-0.36	0
ldconfig-1999-02-21	874
ld.so-1.9.5	9731
less-346	14039
lilo	7255
linuxconf-1.17r2	104032
logrotate-3.3.2	1525
mailcap-2.0.6	0
mailx-8.1.1	6968
MAKEDEV-2.5.2	1020
man-1.5hl	9801
mingetty-0.9.4	343
mkbootdisk-1.2.5	314
mkinitrd-2.4.1	288
mktemp-1.5	85
modutils-2.3.9	11775
mouseconfig-4.4	830

Estimating Linux's Size

mt-st-0.5b	1361
ncompress-4.2.4	1435
ncurses-5.0	61324
net-tools-1.54	11633
newt-0.50.8	7041
pam-0.72	20433
passwd-0.64.1	1194
pciutils-2.1.5	3855
popt-1.4	1099
procmail-3.14	9927
procps-2.0.6	9961
psmisc	1630
pump-0.7.8	1856
pwdb-0.61	9551
quota-2.00-pre3	3804
raidtools-0.90	2424
readline-2.2.1	14941
redhat-logos	0
rootfiles	0
rpm-3.0.4	39861
sash-3.4	6172
sed-3.02	7740
sendmail-8.9.3	42880
setserial-2.15	742
setup-1.2	67
setup-2.1.8	0
shadow-19990827	25236
sh-utils-2.0	17939
slang	28118
slocate-2.1	1883
stat-1.5	280
sysklogd-1.3-31	4038
sysvinit-2.78	6033
tar-1.13.17	14255
termcap-2.0.8	797
texinfo-4.0	28186
textutils-2.0a	36338
time-1.7	1452
timeconfig-3.0.3	346
tmpwatch-2.2	463
utempter-0.5.2	222
util-linux-2.10f	39160
vim-5.6	113241
vixie-cron-3.0.1	2879
which-2.9	2268
zlib-1.1.3	4087

Thus, the contents of the build directories corresponding to the ``base" (minimum) installation totals to 2,819,334 SLOC.

A few notes are in order about this build directory total:

1. Some of the packages listed by a traditional package list aren't shown here because they don't contain any code. Package "basesystem" is a pseudo-package for dependency purposes. Package redhat-release is just a package for keeping track of the base system's version number. Package "filesystem" contains a directory layout.
2. ntsysv's source is in chkconfig-1.1.2; kernel-utils and kernel-pcmcia-cs are part of "linux". Package shadow-utils is in build directory shadow-19990827. Build directory util-linux includes losetup and mount. "dump" is included to include rmt.
3. Sometimes the build directories contain more code than is necessary to create just the parts for the ``base" system; this is a side-effect of how things are packaged. ``info" is included in the base, so we count all of texinfo. The build directory termcap is counted, because libtermcap is in the base. Possibly most important, egcs is there

because libstdc++ is in the base.

4. Sometimes a large component is included in the base, even though most of the time little of its functionality is used. In particular, the mail transfer agent ``sendmail" is in the base, even though for many users most of sendmail's functionality isn't used. However, for this paper's purposes this isn't a problem. After all, even if sendmail's functionality is often underused, clearly that functionality took time to develop and that functionality is available to those who want it.
5. My tools intentionally eliminated duplicates; it may be that a few files aren't counted here because they're considered duplicates of another build directory not included here. I do not expect this factor to materially change the total.
6. Red Hat Linux is not optimized to be a ``small as possible" distribution; their emphasis is on functionality, not small size. A working Linux distribution could include much less code, depending on its intended application. For example, ``linuxconf" simplifies system configuration, but the system can be configured by editing its system configuration files directly, which would reduce the base system's size. This also includes vim, a full-featured text editor - a simpler editor with fewer functions would be smaller as well.

Many people prefer some sort of graphical interface; here is a minimal configuration of a graphical system, adding the X server, a window manager, and a few tools:

Component	SLOC
XFree86-3.3.6	1291745
Xconfigurator-4.3.5	9741
fvwm-2.2.4	69265
X11R6-contrib-3.3.2	18885

These additional graphical components add 1,389,636 SLOC. Due to oddities of the way the initialization system xinitrc is built, it isn't shown here in the total, but xinitrc has so little code that its omission does not significantly affect the total.

Adding these numbers together, we now have a total of 4,208,970 SLOC for a ``minimal graphical system." Many people would want to add more components. For example, this doesn't include a graphical toolkit (necessary for running most graphical applications). We could add gtk+-1.2.6 (a toolkit needed for running GTK+ based applications), adding 138,118 SLOC. This would now total 4,347,088 for a ``basic graphical system," one able to run basic GTK+ applications.

Let's add a web server to the mix. Adding apache_1.3.12 adds only 77,873 SLOC. We now have 4,424,961 physical SLOC for a basic graphical system plus a web server.

We could then add a graphical desktop environment, but there are so many different options and possibilities that trying to identify a ``minimal" system is hard to do without knowing the specific uses intended for the system. Red Hat defines a standard ``GNOME" and ``KDE" desktop, but these are intended to be highly functional (not ``minimal"). Thus, we'll stop here, with a total of 2.8 million physical SLOC for a minimal text-based system, and total of 4.4 million physical SLOC for a basic graphical system plus a web server.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the ``Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

[Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>

[Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to

the 3rd Annual REVIC User's Group Conference, January 10-12, 1990.

http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf

[Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter.

<http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>

[Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.

[Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.

[NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*.

<http://www.nap.edu/html/statsoft/chap2.html>

[OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.

[Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>

[Raymond 1999] Raymond, Eric S. January 1999. "A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.

[Schneier 2000] Schneier, Bruce. March 15, 2000. "Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name..."

<http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet.

<http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000] Wheeler, David A. 2000. *Open Source Software / Free Software References*.

http://www.dwheeler.com/oss_fs_refs.html.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

This paper is (C) Copyright 2000 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. When referring to the paper, please refer to it as "Estimating GNU/Linux's Size" by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

Estimating Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

November 1, 2000

Version 1.01

This paper presents size estimates (and their implications) of the source code of a distribution of the Linux operating system (OS), a combination often called GNU/Linux. The distribution used in this paper is Red Hat Linux version 6.2, including the kernel, software development tools, graphics interfaces, client applications, and so on. Other distributions and versions will have different sizes.

In total, this distribution includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars).

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), egcs (a compilation system), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The Linux operating system (also called GNU/Linux) has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a "typical" intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally "open source software" and/or "free software" (where the "free" here means "freedom"). A program that is "open source software" or "free software" is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of "open source software" is available at OSI [1999], a more formal definition of "free software" is available at FSF [2000], and other general information about these topics is available at Wheeler [2000]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is "open source software"/"free software", and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. The only published data I've found was developed by Microsoft in the documents usually called "Halloween I" and "Halloween II". Unfortunately, the meaning, derivation, and assumptions of their numbers is not explained, making the numbers hard to use and truly understand. Even worse, although the two documents were written by essentially the same people at the same time, the numbers in the documents appear (on their surface) to be contradictory. The so-called "Halloween I" document claimed that the Linux kernel (x86 only) was 500,000 lines of code, the Apache web server was 80,000 lines of code, the X-windows server was 1.5 million, and a full Linux distribution was about 10 million lines of code [Halloween I]. The "Halloween II" document seemed to contradict this, saying that "Linux" by 1998 included 1.5 million lines of code. Since "version 2.1.110" is identified as the version number, presumably this only measures the Linux kernel, and it does note that this measure includes all Linux ports to various architectures [Halloween II]. However, this asks as many questions as it answers - what exactly was being measured, and what assumptions were made? For example, is the Linux kernel support for other architectures *really* one million lines of

code?

This paper bridges this gap. In particular, it shows estimates of the size of Linux, and it estimates how much it would cost to rebuild a typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean.

For my purposes, I have selected as my ``representative" Linux distribution Red Hat Linux version 6.2. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [[Shankland 2000b](#)]. Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE at 15%. Not all Linux copies are ``sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate ``all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 6.2 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Section 2 briefly describes the approach used to estimate the ``size" of this distribution (most of the details are in Appendix A). Section 3 discusses some of the results (with the details in Appendix B). Section 4 presents conclusions, followed by the two appendices.

2. Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; the steps and assumptions made are described in Appendix A.

A few summary points are worth mentioning here, however, for those who don't read appendix A. I included software for all architectures, not just the i386. I did not include ``old" versions of software (with the one exception of bash, as discussed in appendix A). I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once. The code in makefiles and RPM package specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

The ``physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: ``a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the ``logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the ``count of all terminating semicolons in a C file." The ``physical" SLOC was chosen instead of the ``logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the ``physical" SLOC

measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer "COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total SLOC counts, and section 3.6 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 25 largest components (as measured by number of source lines of code):

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,csh=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csh=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csh=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csh=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,csh=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,


```

sed=2
199982  gs5.50      ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916  teTeX-1.0      ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,
yacc=1507,awk=522,lex=323,sed=297,asm=139,csh=47,lisp=29
155035  bind-8.2.2_P5   ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,
csh=848,awk=753,lex=222
140130  AfterStep-APPS-20000124 ansic=135806,sh=3340,cpp=741,perl=243
138931  kdebase         cpp=113971,ansic=23016,perl=1326,sh=618
138118  gtk+-1.2.6      ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024  gated-3-5-11    ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csh=235,
sed=35,lisp=12
133193  kaffe-1.0.5     java=65275,ansic=62125,cpp=3923,perl=972,sh=814,
asm=84
131372  jade-1.2.1      cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672  gnome-libs-1.0.55 ansic=125373,sh=2178,perl=667,awk=277,lisp=177

```

Note that the operating system kernel (linux) is the largest single component, at over 1.5 million lines of code (mostly in C). See section 3.2 for a more discussion discussion of the linux kernel.

The next largest component is the X windows server, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to accrete functionality), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the compilation system, including the C and C++ compilers. Following this is the symbolic debugger and emacs. Emacs is probably not a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system. This is followed by the set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). This is followed by TCL/Tk (a combined language and widget set), PostgreSQL (a relational DBMS), and the GIMP (an excellent client application for editing bitmapped drawings).

Note that language implementations tend to be written in themselves, particularly for their libraries. Thus there is more Perl than any other single language in the Perl implementation, more Python than any other single language in Python, and more Java than any other single language in Kaffe (an implementation of the Java Virtual Machine and library).

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the linux kernel (at over 1.5 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 870,000 lines of this code was in the ``drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of hardware. The linux kernel's design is expressed in its source code directory structure, and no other directory comes close to this size - the second largest is the ``arch" directory (at over 230,000 SLOC), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is not quite 88,000 SLOC. See the appendix for more detail.

Richard Stallman and others have argued that the resulting system often called ``Linux" should instead be called ``GNU/Linux" [\[Stallman 2000\]](#). In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to ``free software" (free as in freedom). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called ``Linux". Using the term ``Linux" is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux" or ``GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: egcs (gcc), gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux" and not just ``Linux."

These measurements at least debunk one possible explanation of the Halloween documents' measures. Since Halloween I claimed that the x86-only code for the Linux kernel measured 500,000 SLOC, while Halloween II claimed that the kernel (all architectures) was 1.5 million SLOC, one explanation of this difference would be that the code for non-x86 systems was 1 million SLOC. This isn't so; I computed a grand total of 267,320 physical SLOC of non-i86 code (including drivers and architecture-specific code). It seems unlikely that over 700,000 lines of code would have been removed (not added) in the intervening time.

However, other measures (and explanations) are more promising. I also ran the CodeCount tools on the linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for the Linux kernel. When I removed all non-i86 code and re-ran the CodeCount tool on just the C code, a logical SLOC of 570,039 of C code was revealed. Since the Halloween I document reported 500,000 SLOC (when only including x86 code), it appears very likely that the Halloween I paper counted logical SLOC (and only C code) when reporting measurements of the linux kernel. However, the other Halloween I measures appear to be physical SLOC measures: their estimate of 1.5 million SLOC for the X server is closer to the 1.2 million physical SLOC measured here, and their estimate of 80,000 SLOC for Apache is close to the 77,873 SLOC measured here (as shown in Appendix B). These variations in measurements should be expected, since the versions I am measuring are slightly different than the ones they measured, and it is likely that some assumptions are different as well. Meanwhile, Halloween II reported a measure of 1.5 million lines of code for the linux kernel, essentially the same value given here for physical SLOC.

In short, it appears that Halloween I used the "logical SLOC" measure when measuring the Linux kernel, while all other measures in Halloween I and II used physical SLOC as the measure. I have attempted to contact the Microsoft author to confirm this, but as of yet I have not received such confirmation. In any case, this example clearly demonstrates the need to carefully identify the units of measure and assumptions made in any measurement of SLOC.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code:

ansic:	14218806	(80.55%)
c++:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)
python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Here you can see that C is pre-eminent (with over 80% of the code), followed by C++, LISP, shell, and Perl. Note that the separation of Expect and TCL is somewhat artificial; if combined, they would be next (at 232115), followed by assembly. Following this in order are Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has over a million lines of code, a very respectable showing, and yet at least in this distribution it is far less than C. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of

understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

The fact that LISP places so highly (it's in third place) is a little surprising. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 80% (453647/565861) of the total amount of LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages: Perl includes 5584 lines of LISP, and Python includes another 2333 of LISP that is directly used to support elaborate Emacs modes for program editing. The ``psgml" package is solely an emacs mode for editing SGML documents. The components with the second and third largest amounts of LISP are xlipstat-3-52-17 and scheme-3.2, which are implementations of LISP and Scheme (a LISP dialect) respectively. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 57 different lex/flex files, and 110 yacc/bison files. Since some build directories use lex/flex or yacc/bison more than once, the count of build directories using these tools is smaller but still respectable: 38 different build directories use lex/flex, and 62 different build directories use yacc/bison.

Other insights can be gained from the file counts shown in appendix B. The number of source code files counted were 72,428. Not included in this count were 5,820 files which contained duplicate contents, and 817 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 14218806 SLOC contained in 52088 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code.

3.5 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 17,652,561 physical source lines of code (SLOC); I will simplify this to ``over 17 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (1998)	20 million

These numbers come from Bruce Schneier's *Crypto-Gram* [\[Schneier 2000\]](#), except for the Space Shuttle numbers which come from a National Academy of Sciences study [\[NAS 1996\]](#). Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Schneier also reports that ``Linux, even with the addition of X Windows and Apache, is still under 5 million lines of code". At first, this seems to be contradictory, since this paper counts over 17 million SLOC, but Schneier appears to be literally correct in the context of his statement. The phrasing of his sentence suggests that Schneier is considering some sort of ``minimal" system, since he considers ``even the addition of X Windows" as a significant addition. As shown in appendix section B.4, taking the minimal ``base" set of components in Red Hat Linux, and then adding the minimal set of components for graphical interaction (the X Windows's graphical server, library, configuration tool, and a graphics toolkit) and the Apache web server,

the total is about 4.4 million physical SLOC - which is less than 5 million. This minimal system doesn't include some useful (but not strictly necessary) components, but a number of useful components could be added while still staying under a total of 5 million SLOC.

However, note the contrast. Many Linux distributions include with their operating systems many applications (e.g., bitmap editors) and development tools (for many different languages). As a result, the entire *delivered* system for such distributions (including Red Hat Linux 6.2) is *much* larger than the 5 million SLOC stated by Schneier. In short, this distribution's size appears similar to the size of Windows 98 and Windows NT 5.0 in 1998.

Microsoft's recent legal battles with the U.S. Department of Justice (DoJ) also involve the bundling of applications with the operating system. However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as "secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, many Linux distributions include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with small SLOC counts can often provide greater functionality than programs with large SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessarily cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized systems.

3.6 Effort and Cost Estimates

Finally, given all the assumptions shown, are the effort values:

```
Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = $ 614421924.71
```

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux version 6.2 includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars). Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches.

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), egcs (a compilation system), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Here you can see that C is pre-eminent (with over 80% of the code), More information is available in the appendices and at <http://www.dwheeler.com/sloc>.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), other open source systems (such as FreeBSD), and other versions of Red Hat (such as Red Hat 7). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost. It's known that Red Hat 7 includes more source code; Red Hat 7 has had to add another CD-ROM to contain the binary programs, and adds such capabilities as a word processor (abiword) and secure shell (openssh).

Some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like ``Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not ``reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ``.m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

Appendix A. Details of Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; each step is described below. Some steps I describe in some detail, because it's sometimes hard to find the necessary information even when the actual steps are easy. Hopefully, this detail will make it easier for others to do similar activities or to repeat the experiment.

A.1 Installing Source Code

Installing the source code files turned out to be nontrivial. First, I inserted the CD-ROM containing all of the source files (in ``src.rpm" format) and installed the packages (files) using:

```
mount /mnt/cdrom
cd /mnt/cdrom/SRPMS
rpm -ivh *.src.rpm
```

This installs ``spec" files and compressed source files; another rpm command (`rpm -bp") uses the spec files to uncompress the source files into ``build directories" (as well as apply any necessary patches). Unfortunately, the rpm tool does not enforce any naming consistency between the package names, the spec names, and the build directory names; for consistency this paper will use the names of the build directories, since all later tools based themselves on the build directories.

I decided to (in general) not count ``old" versions of software (usually placed there for compatibility reasons), since that would be counting the same software more than once. Thus, the following components were not included: ``compat-binutils", ``compat-egcs", ``compat-glib", ``compat-libs", ``gtk+10", ``libc-5.3.12" (an old C library), ``libxml10", ``ncurses3", and ``qt1x". I also didn't include egcs64-19980921 and netscape-sparc, which simply repeated something on another architecture that was available on the i386 in a different package. I did make one exception. I kept both bash-1.14.7 and bash2, two versions of the shell command processor, instead of only counting bash2. While bash2 is the later version of the shell

available in the package, the main shell actually used by the Red Hat distribution was the older version of bash. The rationale for this decision appears to be backwards compatibility for older shell scripts; this is suggested by the Red Hat package documentation in both bash-1.14.7 and bash2. It seemed wrong to not include one of the most fundamental pieces of the system in the count, so I included it. At 47067 lines of code (ignoring duplicates), bash-1.14.7 is one of the smaller components anyway. Not including this older component would not substantively change the results presented here.

There are two directories, krb4-1.0 and krb5-1.1.1, which appear to violate this rule - but don't. krb5-1.1.1 is the build directory created by krb5.spec, which is in turn installed by the source RPM package krb5-1.1.1-9.src.rpm. This build directory contains Kerberos V5, a trusted-third-party authentication system. The source RPM package krb5-1.1.1-9.src.rpm eventually generates the binary RPM files krb5-configs-1.1.1-9, krb5-libs-1.1.1-9, and krb5-devel-1.1.1-9. You might guess that ``krb4-1.0" is just the older version of Kerberos, but this build directory is created by the spec file krbafs.spec and not just an old version of the code. To quote its description, ``This is the Kerberos to AFS bridging library, built against Kerberos 5. krbafs is a shared library that allows programs to obtain AFS tokens using Kerberos IV credentials, without having to link with official AFS libraries which may not be available for a given platform." For this situation, I simply counted both packages, since their purposes are different.

I was then confronted with a fundamental question: should I count software that only works for another architecture? I was using an i86-type system, but some components are only for Alpha or Sparc systems. I decided that I should count them; even if I didn't use the code today, the ability to use these other architectures in the future was of value and certainly required effort to develop.

This caused complications for creating the build directories. If all installed packages fit the architecture, you can install the uncompressed software by typing:

```
cd /usr/src/redhat/SPECS and typing the command
rpm -bp *.spec
```

Unfortunately, the rpm tool notes that you're trying to load code for the ``wrong" architecture, and (at least at the time) there was no simple ``override" flag. Instead, I had to identify each package as belonging to SPARC or ALPHA, and then use the rpm option --target to forcibly load them. For example, I renamed all sparc-specific SPARC file files to end in ``.sparc" and could then load them with:

```
rpm -bp --target sparc-redhat-linux *.spec.sparc
```

The following spec files were non-i86: (sparc) audioclt, elftoaout, ethtool, prtconf, silo, solemul, sparc32; (alpha) aboot, minlabel, quickstrip. In general, these were tools to aid in supporting some part of the boot process or for using system-specific hardware.

Note that not all packages create build directories. For example, ``anonftp" is a package that, when installed, sets up an anonymous ftp system. This package doesn't actually install any software; it merely installs a specific configuration of another piece of software (and unsets the configuration when uninstalled). Such packages are not counted at all in this sizing estimate.

Simply loading all the source code requires a fair amount of disk space. Using ``du" to measure the disk space requirements (with 1024 byte disk blocks), I obtained the following results:

```
$ du -s /usr/src/redhat/BUILD /usr/src/redhat/SOURCES /usr/src/redhat/SPECS
2375928 /usr/src/redhat/BUILD
592404 /usr/src/redhat/SOURCES
4592 /usr/src/redhat/SPECS
```

Thus, these three directories required 2972924 1K blocks - approximately 3 gigabytes of space. Much more space would be required to compile it all.

A.2 Categorizing Source Code

My next task was to identify all files containing source code (not including any automatically generated source code). This is a non-trivial problem; there are 181,679 ordinary files in the build directory, and I had no interest in doing this identification by hand.

In theory, one could just look at the file extensions (.c for C, .py for python), but this is not enough in practice. Some packages reuse extensions if the package doesn't use that kind of file (e.g., the ``.exp" extension of expect was used by some packages as ``export" files, and the ``.m" of objective-C was used by some packages for module information extracted from C code). Some files don't have extensions, particularly scripts. And finally, files automatically generated by another program

should not be counted, since I wished to use the results to estimate effort.

I ended up writing a program of over 600 lines of Perl to perform this identification, which used a number of heuristics to categorize each file into categories. There is a category for each language, plus the categories non-programs, unknown (useful for scanning for problems), automatically generated program files, duplicate files (whose file contents duplicated other files), and zero-length files.

The program first checked for well-known extensions (such as .gif) that cannot be program files, and for a number of common generated filenames. It then peeked at the first line for "#!" followed by a legal script name. If that didn't work, it used the extension to try to determine the category. For a number of languages, the extension was not reliable, so for those languages the categorization program examined the file contents and used a set of heuristics to determine if the file actually belonged that category. If all else failed, the file was placed in the "unknown" category for later analysis. I later looked at the "unknown" items, checking the common extensions to ensure I had not missed any common types of code.

One complicating factor was that I wished to separate C, C++, and objective-C code, but a header file ending with ".h" or ".hpp" file could be any of them. I developed a number of heuristics to determine, for each file, what language it belonged to. For example, if a build directory has exactly one of these languages, determining the correct category for header files is easy. Similarly, if there is exactly one of these in the directory with the header file, it is presumed to be that kind. Finally, a header file with the keyword "class" is almost certainly not a C header file, but a C++ header file.

Detecting automatically generated files was not easy, and it's quite conceivable I missed a number of them. The first 15 lines were examined, to determine if any of them included at the beginning of the line (after spaces and possible comment markers) one of the following phrases: "generated automatically", "automatically generated", "this is a generated file", "generated with the (something) utility", or "do not edit". A number of filename conventions were used, too. For example, any "configure" file is presumed to be automatically generated if there's a "configure.in" file in the same directory.

To eliminate duplicates, the program kept md5 checksums of each program file. Any given md5 checksum would only be counted once. Build directories were processed alphabetically, so this meant that if the same file content was in both directories "a" and "b", it would be counted only once as being part of "a". Thus, some packages with names later in the alphabet may appear smaller than would make sense at first glance. It is very difficult to eliminate "almost identical" files (e.g., an older and newer version of the same code, included in two separate packages), because it is difficult to determine when "similar" two files are essentially the "same" file. Changes such as the use of pretty-printers and massive renaming of variables could make small changes seem large, while the many small files in the system could easily make different files seem the "same." Thus, I did not try to make such a determination, and just considered files with different contents as different.

It's important to note that different rules could be used to "count" lines of code. Some kinds of code were intentionally excluded from the count. Many RPM packages include a number of shell commands used to install and uninstall software; the estimate in this paper does not include the code in RPM packages. This estimate also does not include the code in Makefiles (which can be substantive). In both cases, the code in these cases is often cut and pasted from other similar files, so counting such code would probably overstate the actual development effort. In addition, Makefiles are often automatically generated.

On the other hand, this estimate does include some code that others might not count. This estimate includes test code included with the package, which isn't visible directly to users (other than hopefully higher quality of the executable program). It also includes code not used in this particular system, such as code for other architectures and OS's, bindings for languages not compiled into the binaries, and compilation-time options not chosen. I decided to include such code for two reasons. First, this code is validly represents the effort to build each component. Second, it does represent indirect value to the user, because the user can later use those components in other circumstances even if the user doesn't choose to do so by default.

So, after the work of categorizing the files, the following categories of files were created for each build directory (common extensions are shown in parentheses, and the name used in the data tables below are shown in brackets):

1. C (.c) [ansic]
2. C++ (.C, .cpp, .cxx, .cc) [cpp]
3. LISP (.el, .scm, .lsp, .jl) [lisp]
4. shell (.sh) [sh]
5. Perl (.pl, .pm, .perl) [perl]
6. Assembly (.s, .S, .asm) [asm]
7. TCL (.tcl, .tk, .itk) [tcl]
8. Python (.py) [python]
9. Yacc (.y) [yacc]
10. Java (.java) [java]

11. Expect (.exp) [exp]
12. lex (.l) [lex]
13. awk (.awk) [awk]
14. Objective-C (.m) [objc]
15. C shell (.csh) [csh]
16. Ada (.ada, .ads, .adb) [ada]
17. Pascal (.p) [pascal]
18. sed (.sed) [sed]
19. Fortran (.f) [fortran]

Note that we're counting Scheme as a dialect of LISP, and Expect is being counted separately from TCL. The command line shells Bourne shell, the Bourne-again shell (bash), and the K shell are all counted together as ``shell'', but the C shell (csh and tcsh) is counted separately.

A.3 Counting Lines of Code

Every language required its own counting scheme. This was more complex than I realized; there were a number of languages involved.

I originally tried to use USC's ``CodeCount'' tools to count the code. Unfortunately, this turned out to be buggy and did not handle most of the languages used in the system, so I eventually abandoned it for this task and wrote my own tools. Those who wish to use this tool are welcome to do so; you can learn more from its web site at

<http://sunset.usc.edu/research/CODECOUNT>.

I did manage to use the CodeCount to compute the logical source lines of code for the C portions of the linux kernel. This came out to be 673,627 logical source lines of code, compared to the 1,462,165 lines of physical code (again, this ignores files with duplicate contents).

Since there were a large number of languages to count, I used the ``physical lines of code'' definition. In this definition, a line of code is a line (ending with newline or end-of-file) with at least one non-comment non-whitespace character. These are known as ``non-comment non-blank'' lines. If a line only had whitespace (tabs and spaces) it was not counted, even if it was in the middle of a data value (e.g., a multiline string). It is much easier to write programs to measure this value than to measure the ``logical'' lines of code, and this measure can be easily applied to widely different languages. Since I had to process a large number of different languages, it made sense to choose the measure that is easier to obtain.

[Park \[1992\]](#) presents a framework of issues to be decided when trying to count code. Using Park's framework, here is how code was counted in this paper:

1. Statement Type: I used a physical line-of-code as my basis. I included executable statements, declarations (e.g., data structure definitions), and compiler directives (e.g., preprocessor commands such as #define). I excluded all comments and blank lines.
2. How Produced: I included all programmed code, including any files that had been modified. I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. If a file was in the source package, I included it; if the file had been removed from a source package (including via a patch), I did not include it.
3. Origin: I included all code included in the package.
4. Usage: I included code in or part of the primary product; I did not include code external to the product (i.e., additional applications able to run on the system but not included with the system).
5. Delivery: I counted code delivered as source; not surprisingly, I didn't count code not delivered as source. I also didn't count undelivered code.
6. Functionality: I included both operative and inoperative code. An examples of intentionally ``inoperative'' code is code turned off by #ifdef commands; since it could be turned on for special purposes, it made sense to count it. An examples of unintentionally ``inoperative'' code is dead or unused code.
7. Replications: I included master (original) source statements. I also included ``physical replicates of master statements stored in the master code''. This is simply code cut and pasted from one place to another to reuse code; it's hard to tell where this happens, and since it has to be maintained separately, it's fair to include this in the measure. I excluded copies inserted, instantiated, or expanded when compiling or linking, and I excluded postproduction replicates (e.g., reparameterized systems).

8. Development Status: Since I only measured code included in the packages used to build the delivered system, I declared that all software I was measuring had (by definition) passed whatever "system tests" were required by that component's developers.
9. Languages: I included all languages, as identified earlier in section A.2.
10. Clarifications: I included all statement types. This included nulls, continues, no-ops, lone semicolons, statements that instantiate generics, lone curly braces ({ and }), and labels by themselves.

Park includes in his paper a "basic definition" of physical lines of code, defined using his framework. I adhered to Park's definition unless (1) it was impossible in my technique to do so, or (2) it would appear to make the result inappropriate for use in cost estimation (using COCOMO). COCOMO states that source code:

"includes all program instructions created by project personnel and processed into machine code by some combination of preprocessors, compilers, and assemblers. It excludes comment cards and unmodified utility software. It includes job control language, format statements, and data declarations. Instructions are defined as lines of code."

In summary, though in general I followed Park's definition, I didn't follow Park's "basic definition" in the following ways:

1. How Produced: I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. After all, COCOMO states that the only code that should be counted is code "produced by project personnel", whereas these kinds of files are instead the output of "preprocessors and compilers." If code is always maintained as the input to a code generator, and then the code generator is re-run, it's only the code generator input's size that validly measures the size of what is maintained. Note that while I attempted to exclude generated code, this exclusion is based on heuristics which may have missed some cases.
2. Origin: Normally physical SLOC doesn't include an unmodified "vendor-supplied language support library" nor a "vendor-supplied system or utility". However, in this case this was *exactly* what I was measuring, so I naturally included these as well.
3. Delivery: I didn't count code not delivered as source. After all, since I didn't have it, I couldn't count it.
4. Functionality: I included unintentionally inoperative code (e.g., dead or unused code). There might be such code, but it is very difficult to automatically detect in general for many languages. For example, a program not directly invoked by anything else nor installed by the installer is much more likely to be a test program, which I'm including in the count. Clearly, discerning human "intent" is hard to automate. Hopefully, unintentionally inoperative code is a small amount of the total delivered code.

Otherwise, I followed Park's "basic definition" of a physical line of code, even down to Park's language-specific definitions where Park defined them for a language.

One annoying problem was that one file wasn't syntactically correct and it affected the count. File `/usr/src/redhat/BUILD/cdrecord-1.8/mkiso` had an `#ifdef` not taken, and the road not taken had a missing double-quote mark before the word "cannot":

```
#ifdef  USE_LIBSCHILY
        comerr(Cannot open '%s'.\n", filename);
#endif
        perror ("fopen");
        exit (1);
#endif
```

I solved this by hand-patching the source code (for purposes of counting). There were also some files with intentionally erroneous code (e.g., compiler error tests), but these did not impact the SLOC count.

Several languages turn out to be non-trivial to count:

- In C, C++, and Java, comment markers should be ignored inside strings. Since they have multi-line comment markers this requirement should not be ignored, or a `"/*` inside a string could cause most of the code to be erroneously uncounted.
- Officially, C doesn't have C++'s `"/` comment marker, but the gcc compiler accepts it and a great deal of C code uses it, so my counters accepted it.
- Perl permits in-line "perlpod" documents, "here" documents, and an `__END__` marker that complicate code-counting. Perlpod documents are essentially comments, but a "here" document may include text to generate them (in which case the perlpod document is data and should be counted). The `__END__` marker indicates the end of the file from Perl's viewpoint, even if there's more text afterwards.
- Python has a convention that, at the beginning of a definition (e.g., of a function, method, or class), an unassigned

string can be placed to describe what's being defined. Since this is essentially a comment (though it doesn't syntactically look like one), the counter must avoid counting such strings, which may have multiple lines. To handle this, strings which started the beginning of a line were not counted. Python also has the `"""triple quote"""` operator, permitting multiline strings; these needed to be handled specially. Triple quote strings were normally considered as data, regardless of content, unless they were used as a comment about a definition.

- Assembly languages vary greatly in the comment character they use, so my counter had to handle this variance. I wrote a program which first examined the file to determine if C-style `/*` comments and C preprocessor commands (e.g., `#include`) were used. If both `/*` and `*/` were in the file, it was assumed that C-style comments were used, since it is unlikely that *both* would be used as something else (e.g., as string data) in the same assembly language file. Determining if a file used the C preprocessor was trickier, since many assembly files do use `#` as a comment character and some preprocessor directives are ordinary words that might be included in a human comment. The heuristic used was: if `#ifdef`, `#endif`, or `#include` are used, the preprocessor is used; if at least three lines have either `#define` or `#else`, then the preprocessor is used. No doubt other heuristics are possible, but this at least seemed to produce reasonable results. The program then determined what the comment character was, by identifying which punctuation mark (from a set of possible marks) was the most common non-space initial character on a line (ignoring `/*` and `#` if C comments or preprocessor commands, respectively, were used). Once the comment character had been determined, and it had been determined if C-style comments were also allowed, the lines of code could be counted in the file.

Although their values are not used in estimating effort, I also counted the number of files; summaries of these values are included in appendix B.

Since the linux kernel was the largest single component, and I had questions about the various inconsistencies in the "Halloween" documents, I made additional measures of the Linux kernel.

Some have objected because the counting approach used here includes lines not compiled into code in this Linux distribution. However, the primary objective of these measures was to estimate total effort to develop all of these components. Even if some lines are not normally enabled on Linux, it still required effort to develop that code. Code for other architectures still has value, for example, because it enables users to port to other architectures while using the component. Even if that code is no longer being maintained (e.g., because the architecture has become less popular), nevertheless someone had to invest effort to create it, the results benefitted someone, and if it is needed again it's still there (at least for use as a starting point). Code that is only enabled by compile-time options still has value, because if the options were desired the user could enable them and recompile. Code that is only used for testing still has value, because its use improves the quality of the software directly run by users. It is possible that there is some "dead code" (code that cannot be run under any circumstance), but it is expected that this amount of code is very small and would not significantly affect the results. Andi Kleen (of SuSE) noted that if you wanted to only count compiled and running code, one technique (for some languages) would be to use gcc's `-g` option and use the resulting .stabs debugging information with some filtering (to exclude duplicated inline functions). I determined this to be out-of-scope for this paper, but this approach could be used to make additional measurements of the system.

A.4 Estimating Effort and Costs

For each build directory, I totalled the source lines of code (SLOC) for each language, then totalled those values to determine the SLOC for each directory. I then used the basic Constructive Cost Model (COCOMO) to estimate effort. The basic model is the simplest (and least accurate) model, but I simply did not have the additional information necessary to use the more complex (and more accurate) models. COCOMO is described in depth by Boehm [1981].

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions.

In the basic COCOMO model, estimated man-months of effort, design through test, equals $2.4 \cdot (\text{KSLOC})^{1.05}$, where KSLOC is the total physical SLOC divided by 1000.

I assumed that each package was built completely independently and that there were no efforts necessary for integration not represented in the code itself. This almost certainly underestimates the true costs, but for most packages it's actually true (many packages don't interact with each other at all). I wished to underestimate (instead of overestimate) the effort and costs, and having no better model, I assumed the simplest possible integration effort. This meant that I applied the model to each component, then summed the results, as opposed to applying the model once to

the grand total of all software.

Note that the only input to this model is source lines of code, so some factors simply aren't captured. For example, creating some kinds of data (such as fonts) can be very time-consuming, but this isn't directly captured by this model. Some programs are intentionally designed to be data-driven, that is, they're designed as small programs which are driven by specialized data. Again, this data may be as complex to develop as code, but this is not counted.

Another example of uncaptured factors is the difficulty of writing kernel code. It's generally acknowledged that writing kernel-level code is more difficult than most other kinds of code, because this kind of code is subject to a subtle timing and race conditions, hardware interactions, a small stack, and none of the normal error protections. In this paper I do not attempt to account for this. You could try to use the Intermediate COCOMO model to try to account for this, but again this requires knowledge of other factors that can only be guessed at. Again, the effort estimation probably significantly underestimates the actual effort represented here.

It's worth noting that there is an update to COCOMO, COCOMO II. However, COCOMO II requires as its input logical (not physical) SLOC, and since this measure is much harder to obtain, I did not pursue it for this paper. More information about COCOMO II is available at the web site <http://sunset.usc.edu/research/COCOMOII/index.html>. A nice overview paper where you can learn more about software metrics is Masse [1997].

I assumed that an average U.S. programmer/analyst salary in the year 2000 was \$56,286 per year; this value was from the ComputerWorld, September 4, 2000's Salary Survey. Overhead is much harder to estimate; I did not find a definitive source for information on overheads. After informal discussions with several cost analysts, I determined that an overhead of 2.4 would be representative of the overhead sustained by a typical software development company. Should you disagree with these figures, I've provided all the information necessary to recalculate your own cost figures; just start with the effort estimates and recalculate cost yourself.

Appendix B. More Detailed Results

This appendix provides some more detailed results. B.1 lists the SLOC found in each build directory; B.2 shows counts of files for each category of file; B.3 presents some additional measures about the Linux kernel. B.4 presents some SLOC totals of putatively ``minimal" systems. You can learn more at <http://www.dwheeler.com/sloc>.

B.1 SLOC in Build Directories

The following is a list of all build directories, and the source lines of code (SLOC) found in them, followed by a few statistics counting files (instead of SLOC).

Remember that duplicate files are only counted once, with the build directory ``first in ASCII sort order" receiving any duplicates (to break ties). As a result, some build directories have a smaller number than might at first make sense. For example, the ``kudzu" build directory does contain code, but all of it is also contained in the ``Xconfigurator" build directory.. and since that directory sorts first, the kudzu package is considered to have ``no code".

The columns are SLOC (total physical source lines of code), Directory (the name of the build directory, usually the same or similar to the package name), and SLOC-by-Language (Sorted). This last column lists languages by name and the number of SLOC in that language; zeros are not shown, and the list is sorted from largest to smallest in that build directory. Similarly, the directories are sorted from largest to smallest total SLOC.

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,csh=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csh=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16

Estimating Linux's Size

415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csch=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csch=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,csch=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,sed=2
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,yacc=1507,awk=522,lex=323,sed=297,asm=139,csch=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,csch=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csch=235,sed=35,lisp=12
133193	kaffe-1.0.5	java=65275,ansic=62125,cpp=3923,perl=972,sh=814,asm=84
131372	jade-1.2.1	cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177
127536	pine4.21	ansic=126678,sh=766,csch=62,perl=30
121878	ImageMagick-4.2.9	ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
119613	lynx2-8-3	ansic=117385,sh=1860,perl=340,csch=28
116951	mc-4.5.42	ansic=114406,sh=1996,perl=345,awk=148,csch=56
116615	gnumeric-0.48	ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
113272	xlispstat-3-52-17	ansic=91484,lisp=21769,sh=18,csch=1
113241	vim-5.6	ansic=111724,awk=683,sh=469,perl=359,csch=6
109824	php-3.0.15	ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
104032	linuxconf-1.17r2	cpp=93139,perl=4570,sh=2984,java=2741,ansic=598
102674	libgr-2.0.13	ansic=99647,sh=2438,csch=589
100951	lam-6.3.1	ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187,csch=19
99066	krb4-1.0	ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765,yacc=1509,lex=236,awk=33
94637	xlockmore-4.15	ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
93940	kdenetwork	cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
92964	samba-2.0.6	ansic=88308,sh=3557,perl=831,awk=158,csch=110
91213	anaconda-6.2.2	ansic=74303,python=13657,sh=1583,yacc=810,lex=732,perl=128
89959	xscreensaver-3.23	ansic=88488,perl=1070,sh=401
88128	cvs-1.10.7	ansic=68303,sh=17909,perl=902,yacc=826,csch=181,lisp=7
87940	isd4k-utils	ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
85383	xpdf-0.90	cpp=60427,ansic=21400,sh=3556
81719	inn-2.2.2	ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547,lex=249,tcl=3
80343	kdelibs	cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116,sh=35
79997	WindowMaker-0.61.1	ansic=77924,sh=1483,perl=371,lisp=219
78787	extace-1.2.15	ansic=66571,sh=9322,perl=2894
77873	apache_1.3.12	ansic=69191,sh=6781,perl=1846,cpp=55
75257	xpilot-4.1.0	ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68

Estimating Linux's Size

```

73817  w3c-libwww-5.2.8  ansic=64754,sh=4678,cpp=3181,perl=1204
72726  ucd-snmp-4.1.1    ansic=64411,perl=5558,sh=2757
72425  gnome-core-1.0.55 ansic=72230,perl=141,sh=54
71810  jikes              cpp=71452,java=358
70260  groff-1.15        cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397,
                        sh=265,sed=46
69265  fvwm-2.2.4        ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
69246  linux-86          ansic=63328,asm=5276,sh=642
68997  blt2.4g           ansic=58630,tcl=10215,sh=152
68884  squid-2.3.STABLE1 ansic=66305,sh=1570,perl=1009
68560  bash-2.03         ansic=56758,sh=7264,yacc=2808,perl=1730
68453  kdeggraphics      cpp=34208,ansic=29347,sh=4898
65722  xntp3-5.93        ansic=60190,perl=3633,sh=1445,awk=417,asm=37
62922  ppp-2.3.11        ansic=61756,sh=996,exp=82,perl=44,csch=44
62137  sgml-tools-1.0.9  cpp=38543,ansic=19185,perl=2866,lex=560,sh=532,
                        lisp=309,awk=142
61688  imap-4.7          ansic=61628,sh=60
61324  ncurses-5.0       ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103,
                        sed=100
60429  kdesupport        ansic=42421,cpp=17810,sh=173,awk=13,csch=12
60302  openldap-1.2.9    ansic=58078,sh=1393,perl=630,python=201
57217  xfig.3.2.3-beta-1 ansic=57212,csch=5
56093  lsof_4.47         ansic=50268,sh=4753,perl=856,awk=214,asm=2
55667  uucp-1.06.1       ansic=52078,sh=3400,perl=189
54935  gnupg-1.0.1       ansic=48884,asm=4586,sh=1465
54603  glade-0.5.5       ansic=49545,sh=5058
54431  svgalib-1.4.1     ansic=53725,asm=630,perl=54,sh=22
53141  AfterStep-1.8.0   ansic=50898,perl=1168,sh=842,cpp=233
52808  kdeutils          cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
52574  nmh-1.0.3         ansic=50698,sh=1785,awk=74,sed=17
51813  freetype-1.3.1    ansic=48929,sh=2467,cpp=351,csch=53,perl=13
51592  enlightenment-0.15.5 ansic=51569,sh=23
50970  cdrecord-1.8      ansic=48595,sh=2177,perl=194,sed=4
49370  tin-1.4.2         ansic=47763,sh=908,yacc=699
49325  imlib-1.9.7       ansic=49260,sh=65
48223  kdemultimedia     ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
47067  bash-1.14.7       ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
46312  tcsh-6.09.00      ansic=43544,sh=921,lisp=669,perl=593,csch=585
46159  unzip-5.40        ansic=40977,cpp=3778,asm=1271,sh=133
45811  mutt-1.0.1        ansic=45574,sh=237
45589  am-utils-6.0.3    ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
45485  guile-1.3         ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,csch=50
45378  gnuplot-3.7.1     ansic=43276,lisp=661,asm=539,objc=387,csch=297,perl=138,
                        sh=80
44323  mgetty-1.1.21     ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
42880  sendmail-8.9.3    ansic=40364,perl=1737,sh=779
42746  elm2.5.3          ansic=32931,sh=9774,awk=41
41388  p2c-1.22          ansic=38788,pascal=2499,perl=101
41205  gnome-games-1.0.51 ansic=31191,lisp=6966,cpp=3048
39861  rpm-3.0.4         ansic=36994,sh=1505,perl=1355,python=7
39160  util-linux-2.10f  ansic=38627,sh=351,perl=65,csch=62,sed=55
38927  xms-1.0.1         ansic=38366,asm=398,sh=163
38548  ORBit-0.5.0       ansic=35656,yacc=1750,sh=776,lex=366
38453  zsh-3.0.7         ansic=36208,sh=1763,perl=331,awk=145,sed=6
37515  ircii-4.4         ansic=36647,sh=852,lex=16
37360  tiff-v3.5.4       ansic=32734,sh=4054,cpp=572
36338  textutils-2.0a    ansic=18949,sh=16111,perl=1218,sed=60
36243  exmh-2.1.1        tcl=35844,perl=316,sh=49,exp=34
36239  x11amp-0.9-alpha3 ansic=31686,sh=4200,asm=353
35812  xloadimage.4.1    ansic=35705,sh=107

```

```

35554 zip-2.3 ansic=32108,asm=3446
35397 gtk-engines-0.10 ansic=20636,sh=14761
35136 php-2.0.1 ansic=33991,sh=1056,awk=89
34882 pmake ansic=34599,sh=184,awk=58,sed=41
34772 xpuzzles-5.4.1 ansic=34772
34768 fileutils-4.0p ansic=31324,sh=2042,yacc=841,perl=561
33203 strace-4.2 ansic=30891,sh=1988,perl=280,lisp=44
32767 trn-3.6 ansic=25264,sh=6843,yacc=660
32277 pilot-link.0.9.3 ansic=26513,java=2162,cpp=1689,perl=971,yacc=660,
python=268,tcl=14
31994 korganizer cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
31174 ncftp-3.0beta21 ansic=30347,cpp=595,sh=232
30438 gnome-pim-1.0.55 ansic=28665,yacc=1773
30122 scheme-3.2 lisp=19483,ansic=10515,sh=124
30061 tcpdump-3.4 ansic=29208,yacc=236,sh=211,lex=206,awk=184,csch=16
29730 screen-3.9.5 ansic=28156,sh=1574
29315 jed ansic=29315
29091 xchat-1.4.0 ansic=28894,perl=121,python=53,sh=23
28897 ncpfs-2.2.0.17 ansic=28689,sh=182,tcl=26
28449 slrn-0.9.6.2 ansic=28438,sh=11
28261 xfishtank-2.1tp ansic=28261
28186 texinfo-4.0 ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
28169 e2fsprogs-1.18 ansic=27250,awk=437,sh=339,sed=121,perl=22
28118 slang ansic=28118
27860 kdegames cpp=27507,ansic=340,sh=13
27117 librep-0.10 ansic=19381,lisp=5385,sh=2351
27040 mikmod-3.1.6 ansic=26975,sh=55,awk=10
27022 x3270-3.1.1 ansic=26456,sh=478,exp=88
26673 lout-3.17 ansic=26673
26608 Xaw3d-1.3 ansic=26235,yacc=247,lex=126
26363 gawk-3.0.4 ansic=19871,awk=2519,yacc=2046,sh=1927
26146 libxml-1.8.6 ansic=26069,sh=77
25994 xrn-9.02 ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31,
csch=13
25915 gv-3.5.8 ansic=25821,sh=94
25479 xpaint ansic=25456,sh=23
25236 shadow-19990827 ansic=23464,sh=883,yacc=856,perl=33
24910 kdeadadmin cpp=19919,sh=3936,perl=1055
24773 pdksh-5.2.14 ansic=23599,perl=945,sh=189,sed=40
24583 gmp-2.0.2 ansic=17888,asm=5252,sh=1443
24387 mars_nwe ansic=24158,sh=229
24270 gnome-python-1.0.51 python=14331,ansic=9791,sh=148
23838 kterm-6.2.0 ansic=23838
23666 enscrip-1.6.1 ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
22373 sawmill-0.24 ansic=11038,lisp=8172,sh=3163
22279 make-3.78.1 ansic=19287,sh=2029,perl=963
22011 libpng-1.0.5 ansic=22011
21593 xboard-4.0.5 ansic=20640,lex=904,sh=41,csch=5,sed=3
21010 netkit-telnet-0.16 ansic=14796,cpp=6214
20433 pam-0.72 ansic=18936,yacc=634,sh=482,perl=321,lex=60
20125 ical-2.2 cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
20078 gd1.3 ansic=19946,perl=132
19971 wu-ftpd-2.6.0 ansic=17572,yacc=1774,sh=421,perl=204
19500 gnome-utils-1.0.50 ansic=18099,yacc=824,lisp=577
19065 joe ansic=18841,asm=224
18885 X11R6-contrib-3.3.2 ansic=18616,lex=161,yacc=97,sh=11
18835 glib-1.2.6 ansic=18702,sh=133
18151 git-4.3.19 ansic=16166,sh=1985
18020 xboing ansic=18006,sh=14
17939 sh-utils-2.0 ansic=13366,sh=3027,yacc=871,perl=675

```

17765	mtools-3.9.6	ansic=16155,sh=1602,sed=8
17750	gettext-0.10.35	ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9
17682	bc-1.05	ansic=9186,sh=7236,yacc=967,lex=293
17271	fetchmail-5.3.1	ansic=13441,python=1490,sh=1246,yacc=411,perl=321,lex=238,awk=124
17259	sox-12.16	ansic=16659,sh=600
16785	control-center-1.0.51	ansic=16659,sh=126
16266	dhcp-2.0	ansic=15328,sh=938
15967	SVGATextMode-1.9-src	ansic=15079,yacc=340,sh=294,lex=227,sed=15,asm=12
15868	kpilot-3.1b9	cpp=8613,ansic=5640,yacc=1615
15851	taper-6.9a	ansic=15851
15819	mpg123-0.59r	ansic=14900,asm=919
15691	transfig.3.2.1	ansic=15643,sh=38,csch=10
15638	mod_perl-1.21	perl=10278,ansic=5124,sh=236
15522	console-tools-0.3.3	ansic=13335,yacc=986,sh=800,lex=291,perl=110
15456	rpm2html-1.2	ansic=15334,perl=122
15143	gnotepad+-1.1.4	ansic=15143
15108	GXedit1.23	ansic=15019,sh=89
15087	mm2.7	ansic=8044,csch=6924,sh=119
14941	readline-2.2.1	ansic=11375,sh=1890,perl=1676
14912	ispell-3.1	ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385,csch=221,sh=157,perl=85,sed=15
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,csch=29
14774	flex-2.5.4	ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12
14587	multimedia	ansic=14577,sh=10
14516	libgtop-1.0.6	ansic=13768,perl=653,sh=64,asm=31
14427	mawk-1.2.2	ansic=12714,yacc=994,awk=629,sh=90
14363	automake-1.4	perl=10622,sh=3337,ansic=404
14350	rsync-2.4.1	ansic=13986,perl=179,sh=126,awk=59
14299	nfs-utils-0.1.6	ansic=14107,sh=165,perl=27
14269	rcs-5.7	ansic=12209,sh=2060
14255	tar-1.13.17	ansic=13014,lisp=592,sh=538,perl=111
14105	wmakerconf-2.1	ansic=13620,perl=348,sh=137
14039	less-346	ansic=14032,awk=7
13779	rxvt-2.6.1	ansic=13779
13586	wget-1.5.3	ansic=13509,perl=54,sh=23
13504	rp3-1.0.7	cpp=10416,ansic=2957,sh=131
13241	iproute2	ansic=12139,sh=1002,perl=100
13100	silo-0.9.8	ansic=10485,asm=2615
12657	macutils	ansic=12657
12639	libungif-4.1.0	ansic=12381,sh=204,perl=54
12633	minicom-1.83.0	ansic=12503,sh=130
12593	audiofile-0.1.9	sh=6440,ansic=6153
12463	gnome-objc-1.0.2	objc=12365,sh=86,ansic=12
12313	jpeg-6a	ansic=12313
12124	ypserv-1.3.9	ansic=11622,sh=460,perl=42
11790	lrzsz-0.12.20	ansic=9512,sh=1263,exp=1015
11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
11633	net-tools-1.54	ansic=11531,sh=102
11404	findutils-4.1	ansic=11160,sh=173,exp=71
11299	xmorph-1999dec12	ansic=10783,tcl=516
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
10914	diffutils-2.7	ansic=10914
10404	gnorpm-0.9	ansic=10404
10271	gqview-0.7.0	ansic=10271
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
10187	dump-0.4b15	ansic=9422,sh=760,sed=5
10088	piranha	ansic=10048,sh=40

10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
9961	procps-2.0.6	ansic=9959,sh=2
9942	xpat2-1.04	ansic=9942
9927	procmail-3.14	ansic=8090,sh=1837
9873	nss_ldap-105	ansic=9784,perl=89
9801	man-1.5h1	ansic=7377,sh=1802,perl=317,awk=305
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6
9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
9699	bison-1.28	ansic=9650,sh=49
9666	ash-linux-0.2	ansic=9445,sh=221
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
9551	pwdb-0.61	ansic=9488,sh=63
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
9263	ctags-3.4	ansic=9240,sh=23
9138	gftp-2.0.6a	ansic=9138
8939	mkisofs-1.12b5	ansic=8939
8766	pxe-linux	c++=4463,ansic=3622,asm=681
8572	psgml-1.2.1	lisp=8572
8540	xxgdb-1.12	ansic=8540
8491	gtop-1.0.5	ansic=8151,c++=340
8356	gedit-0.6.1	ansic=8225,sh=131
8303	dip-3.3.7o	ansic=8207,sh=96
7859	libglade-0.11	ansic=5898,sh=1809,python=152
7826	xpm-3.4k	ansic=7750,sh=39,c++=37
7740	sed-3.02	ansic=7301,sed=359,sh=80
7617	cpio-2.4.2	ansic=7598,sh=19
7615	esound-0.2.17	ansic=7387,sh=142,csh=86
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
7427	ed-0.2	ansic=7263,sh=164
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,c++=3
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
7095	xgammon-0.98	ansic=6506,lex=589
7041	newt-0.50.8	ansic=6526,python=515
7030	ee-0.3.11	ansic=7007,sh=23
6976	aboot-0.5	ansic=6680,asm=296
6968	mailx-8.1.1	ansic=6963,sh=5
6877	lpr	ansic=6842,sh=35
6827	gnome-media-1.0.51	ansic=6827
6646	iputils	ansic=6646
6611	patch-2.5	ansic=6561,sed=50
6592	xosview-1.7.1	c++=6205,ansic=367,awk=20
6550	byacc-1.9	ansic=5520,yacc=1030
6496	pidentd-3.0.10	ansic=6475,sh=21
6391	m4-1.4	ansic=5993,lisp=243,sh=155
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
6234	awesfx-0.4.3a	ansic=6234
6172	sash-3.4	ansic=6172
6116	lslk	ansic=5325,sh=791
6090	joystick-1.2.15	ansic=6086,sh=4
6072	kdoc	perl=6010,sh=45,c++=17
6043	irda-utils-0.9.10	ansic=5697,sh=263,perl=83
6033	sysvinit-2.78	ansic=5256,sh=777
6025	pnm2ppa	ansic=5708,sh=317
6021	rpmfind-1.4	ansic=6021
5981	indent-2.2.5	ansic=5958,sh=23
5975	ytalk-3.1	ansic=5975
5960	isapnptools-1.21	ansic=4394,yacc=1383,perl=123,sh=60
5744	gdm-2.0beta2	ansic=5632,sh=112
5594	isdn-config	c++=3058,sh=2228,perl=308

5526	efax-0.9	ansic=4570,sh=956
5383	acct-6.3.2	ansic=5016,cpp=287,sh=80
5115	libtool-1.3.4	sh=3374,ansic=1741
5111	netkit-ftp-0.16	ansic=5111
4996	bzip2-0.9.5d	ansic=4996
4895	xcpustate-2.5	ansic=4895
4792	libelf-0.6.4	ansic=3310,sh=1482
4780	make-3.78.1_pvm-0.5	ansic=4780
4542	gpgp-0.4	ansic=4441,sh=101
4430	gperf-2.7	cpp=2947,exp=745,ansic=695,sh=43
4367	aumix-1.30.1	ansic=4095,sh=179,sed=93
4087	zlib-1.1.3	ansic=2815,asm=712,cpp=560
4038	sysklogd-1.3-31	ansic=3741,perl=158,sh=139
4024	rep-gtk-0.8	ansic=2905,lisp=971,sh=148
3962	netkit-timed-0.16	ansic=3962
3929	initscripts-5.00	sh=2035,ansic=1866,csh=28
3896	ltracex-0.3.10	ansic=2986,sh=854,awk=56
3885	phhttpd-0.1.0	ansic=3859,sh=26
3860	xdaliclock-2.18	ansic=3837,sh=23
3855	pciutils-2.1.5	ansic=3800,sh=55
3804	quota-2.00-pre3	ansic=3795,sh=9
3675	dosfstools-2.2	ansic=3675
3654	tcp_wrappers_7.6	ansic=3654
3651	ipchains-1.3.9	ansic=2767,sh=884
3625	autofs-3.1.4	ansic=2862,sh=763
3588	netkit-rsh-0.16	ansic=3588
3438	yp-tools-2.4	ansic=3415,sh=23
3433	dialog-0.6	ansic=2834,perl=349,sh=250
3415	ext2ed-0.1	ansic=3415
3315	gdbm-1.8.0	ansic=3290,cpp=25
3245	ypbind-3.3	ansic=1793,sh=1452
3219	playmidi-2.4	ansic=3217,sed=2
3096	xtrojka123	ansic=3087,sh=9
3084	at-3.1.7	ansic=1442,sh=1196,yacc=362,lex=84
3051	dhcpcd-1.3.18-pl3	ansic=2771,sh=280
3012	apmd	ansic=2617,sh=395
2883	netkit-base-0.16	ansic=2883
2879	vixie-cron-3.0.1	ansic=2866,sh=13
2835	gkermi-1.0	ansic=2835
2810	kdetoys	cpp=2618,ansic=192
2791	xjewel-1.6	ansic=2791
2773	mpage-2.4	ansic=2704,sh=69
2758	autoconf-2.13	sh=2226,perl=283,exp=167,ansic=82
2705	autorun-2.61	sh=1985,cpp=720
2661	cdp-0.33	ansic=2661
2647	file-3.28	ansic=2601,perl=46
2645	libghttp-1.0.4	ansic=2645
2631	getty_ps-2.0.7j	ansic=2631
2597	pythonlib-1.23	python=2597
2580	magicdev-0.2.7	ansic=2580
2531	gnome-kerberos-0.2	ansic=2531
2490	sndconfig-0.43	ansic=2490
2486	bug-buddy-0.7	ansic=2486
2459	usermode-1.20	ansic=2459
2455	fnlib-0.4	ansic=2432,sh=23
2447	sliplogin-2.1.1	ansic=2256,sh=143,perl=48
2424	raidtools-0.90	ansic=2418,sh=6
2423	netkit-routed-0.16	ansic=2423
2407	nc	ansic=1670,sh=737
2324	up2date-1.13	python=2324

Estimating Linux's Size

2270	memprof-0.3.0	ansic=2270
2268	which-2.9	ansic=1398,sh=870
2200	printtool	tcl=2200
2163	gnome-linuxconf-0.25	ansic=2163
2141	unarj-2.43	ansic=2141
2065	units-1.55	ansic=1963,perl=102
2048	netkit-ntalk-0.16	ansic=2048
1987	cracklib,2.7	ansic=1919,perl=46,sh=22
1984	cleanfeed-0.95.7b	perl=1984
1977	wmconfig-0.9.8	ansic=1941,sh=36
1941	isicom	ansic=1898,sh=43
1883	slocate-2.1	ansic=1802,sh=81
1857	netkit-rusers-0.16	ansic=1857
1856	pump-0.7.8	ansic=1856
1842	cdecl-2.5	ansic=1002,yacc=765,lex=75
1765	fbset-2.1	ansic=1401,yacc=130,lex=121,perl=113
1653	adjtimex-1.9	ansic=1653
1634	netcfg-2.25	python=1632,sh=2
1630	psmisc	ansic=1624,sh=6
1621	urlview-0.7	ansic=1515,sh=106
1604	fortune-mod-9708	ansic=1604
1531	netkit-tftp-0.16	ansic=1531
1525	logrotate-3.3.2	ansic=1524,sh=1
1473	traceroute-1.4a5	ansic=1436,awk=37
1452	time-1.7	ansic=1395,sh=57
1435	ncompress-4.2.4	ansic=1435
1361	mt-st-0.5b	ansic=1361
1290	cxhextris	ansic=1290
1280	pam_krb5-1	ansic=1280
1272	bsd-finger-0.16	ansic=1272
1229	hdparm-3.6	ansic=1229
1226	procinfo-17	ansic=1145,perl=81
1194	passwd-0.64.1	ansic=1194
1182	auth_ldap-1.4.0	ansic=1182
1146	prtconf-1.3	ansic=1146
1143	anacron-2.1	ansic=1143
1129	xbill-2.0	cpp=1129
1099	popt-1.4	ansic=1039,sh=60
1088	nag	perl=1088
1076	stylesheets-0.13rh	perl=888,sh=188
1075	authconfig-3.0.3	ansic=1075
1049	kpppload-1.04	cpp=1044,sh=5
1020	MAKEDEV-2.5.2	sh=1020
1013	trojka	ansic=1013
987	xmailbox-2.5	ansic=987
967	netkit-rwho-0.16	ansic=967
953	switchdesk-2.1	ansic=314,perl=287,cpp=233,sh=119
897	portmap_4	ansic=897
874	ldconfig-1999-02-21	ansic=874
844	jpeg-6b	sh=844
834	ElectricFence-2.1	ansic=834
830	mouseconfig-4.4	ansic=830
816	rpmlint-0.8	python=813,sh=3
809	kdpms-0.2.8	cpp=809
797	termcap-2.0.8	ansic=797
787	xsysinfo-1.7	ansic=787
770	giftrans-1.12.2	ansic=770
742	setserial-2.15	ansic=742
728	tree-1.2	ansic=728
717	chkconfig-1.1.2	ansic=717

Estimating Linux's Size

682	lpg	perl=682
657	eject-2.0.2	ansic=657
616	diffstat-1.27	ansic=616
592	netscape-4.72	sh=592
585	usenet-1.0.9	ansic=585
549	genromfs-0.3	ansic=549
548	tksysv-1.1	tcl=526,sh=22
537	minlabel-1.2	ansic=537
506	netkit-bootparamd-0.16	ansic=506
497	locale_config-0.2	ansic=497
491	helptool-2.4	perl=288,tcl=203
480	elftoaout-2.2	ansic=480
463	tmpwatch-2.2	ansic=311,sh=152
445	rhs-printfilters-1.63	sh=443,ansic=2
441	audioctl	ansic=441
404	control-panel-3.13	ansic=319,tcl=85
368	kbdconfig-1.9.2.4	ansic=368
368	vlock-1.3	ansic=368
367	timetool-2.7.3	tcl=367
347	kernelcfg-0.5	python=341,sh=6
346	timeconfig-3.0.3	ansic=318,sh=28
343	mingetty-0.9.4	ansic=343
343	chkfontpath-1.7	ansic=343
332	ethtool-1.0	ansic=332
314	mkbootdisk-1.2.5	sh=314
302	symlinks-1.2	ansic=302
301	xsri-1.0	ansic=301
294	netkit-rwall-0.16	ansic=294
290	biff+comsat-0.16	ansic=290
288	mkinitrd-2.4.1	sh=288
280	stat-1.5	ansic=280
265	sysreport-1.0	sh=265
261	bdflush-1.5	ansic=202,asm=59
255	ipvsadm-1.1	ansic=255
255	sag-0.6-html	perl=255
245	man-pages-1.28	sh=244,sed=1
240	open-1.4	ansic=240
236	xtoolwait-1.2	ansic=236
222	utempter-0.5.2	ansic=222
222	mkkickstart-2.1	sh=222
221	hellas	sh=179,perl=42
213	rhmask	ansic=213
159	quickstrip-1.1	ansic=159
132	rdate-1.0	ansic=132
131	statserial-1.1	ansic=121,sh=10
107	fwhois-1.00	ansic=107
85	mktemp-1.5	ansic=85
82	modemtool-1.21	python=73,sh=9
67	setup-1.2	ansic=67
56	shaper	ansic=56
52	sparc32-1.1	ansic=52
47	intimed-1.10	ansic=47
23	locale-ja-9	sh=23
16	AnotherLevel-1.0.1	sh=16
11	words-2	sh=11
7	trXFree86-2.1.2	tcl=7
0	install-guide-3.2.html	(none)
0	caching-nameserver-6.2	(none)
0	XFree86-ISO8859-2-1.0	(none)
0	rootfiles	(none)

```

0      ghostscript-fonts-5.50 (none)
0      kudzu-0.36             (none)
0      wvdial-1.41            (none)
0      mailcap-2.0.6          (none)
0      desktop-backgrounds-1.1 (none)
0      redhat-logos           (none)
0      solemul-1.1            (none)
0      dev-2.7.18             (none)
0      urw-fonts-2.0          (none)
0      users-guide-1.0.72     (none)
0      sgml-common-0.1        (none)
0      setup-2.1.8            (none)
0      jadetex                 (none)
0      gnome-audio-1.0.0      (none)
0      specs-po-6.2           (none)
0      gimp-data-extras-1.0.0 (none)
0      docbook-3.1            (none)
0      indexhtml-6.2          (none)

```

```

ansic:      14218806 (80.55%)
cpp:        1326212 (7.51%)
lisp:       565861 (3.21%)
sh:         469950 (2.66%)
perl:       245860 (1.39%)
asm:        204634 (1.16%)
tcl:        152510 (0.86%)
python:     140725 (0.80%)
yacc:       97506 (0.55%)
java:       79656 (0.45%)
exp:        79605 (0.45%)
lex:        15334 (0.09%)
awk:        14705 (0.08%)
objc:       13619 (0.08%)
csh:        10803 (0.06%)
ada:        8217 (0.05%)
pascal:     4045 (0.02%)
sed:        2806 (0.02%)
fortran:    1707 (0.01%)

```

Total Physical Source Lines of Code (SLOC) = 17652561
 Total Estimated Person-Years of Development = 4548.36
 Average Programmer Annual Salary = 56286
 Overhead Multiplier = 2.4
 Total Estimated Cost to Develop = \$ 614421924.71

B.2 Counts of Files For Each Category

There were 181,679 ordinary files in the build directory. The following are counts of the number of files (*not* the SLOC) for each language:

```

ansic:      52088 (71.92%)
cpp:        8092 (11.17%)
sh:         3381 (4.67%)
asm:        1931 (2.67%)
perl:       1387 (1.92%)
lisp:       1168 (1.61%)
java:       1047 (1.45%)

```

python:	997	(1.38%)
tcl:	798	(1.10%)
exp:	472	(0.65%)
awk:	285	(0.39%)
objc:	260	(0.36%)
sed:	112	(0.15%)
yacc:	110	(0.15%)
csch:	94	(0.13%)
ada:	92	(0.13%)
lex:	57	(0.08%)
fortran:	50	(0.07%)
pascal:	7	(0.01%)

Total Number of Source Code Files = 72428

In addition, when counting the number of files (not SLOC), some files were identified as source code files but nevertheless were not counted for other reasons (and thus not included in the file counts above). Of these source code files, 5,820 files were identified as duplicating the contents of another file, 817 files were identified as files that had been automatically generated, and 65 files were identified as zero-length files.

B.3 Additional Measures of the Linux Kernel

I also made additional measures of the Linux kernel. This kernel is Linux kernel version 2.2.14 as patched by Red Hat. The Linux kernel's design is reflected in its directory structure. Only 8 lines of source code are in its main directory; the rest are in descendent directories. Counting the physical SLOC in each subdirectory (or its descendents) yielded the following:

BUILD/linux/Documentation/	765
BUILD/linux/arch/	236651
BUILD/linux/configs/	0
BUILD/linux/drivers/	876436
BUILD/linux/fs/	88667
BUILD/linux/ibcs/	16619
BUILD/linux/include/	136982
BUILD/linux/init/	1302
BUILD/linux/ipc/	1757
BUILD/linux/kernel/	7436
BUILD/linux/ksymoops-0.7c/	3271
BUILD/linux/lib/	1300
BUILD/linux/mm/	6771
BUILD/linux/net/	105549
BUILD/linux/pcmcia-cs-3.1.8/	34851
BUILD/linux/scripts/	8357

I separately ran the CodeCount tools on the entire linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for Linux.

However, this included non-i86 code. To make a more reasonable comparison with the Halloween documents, I needed to ignore non-i386 code.

First, I looked at the linux/arch directory, which contained architecture-specific code. This directory had the following subdirectories (architectures): alpha, arm, i386, m68k, mips, ppc, s390, sparc, sparc64. I then computed the total for all of ``arch'', which was 236651 SLOC, and subtracted out linux/arch/i386 code, which totalled to 26178 SLOC; this gave me a total of non-i386 code in linux/arch as 210473 physical SLOC. I then looked through the ``drivers'' directory to see if there were sets of drivers which were non-i386. I identified the following directories, with the SLOC totals as shown:

linux/drivers/sbus/	22354
---------------------	-------

linux/drivers/macintosh/	6000
linux/drivers/sgi/	4402
linux/drivers/fc4/	3167
linux/drivers/nubus/	421
linux/drivers/acorn/	11850
linux/drivers/s390/	8653

Driver Total: 56847

Thus, I had a grand total on non-i86 code (including drivers and architecture-specific code) as 267320 physical SLOC. This is, of course, another approximation, since there's certainly other architecture-specific lines, but I believe that is most of it. Running the CodeCount tool on just the C code, once these architectural and driver directories are removed, reveals a logical SLOC of 570,039 of C code.

B.4 Minimum System SLOC

Most of this paper worries about counting an "entire" system. However, what's the SLOC size of a "minimal" system? Here's an attempt to answer that question.

Red Hat Linux 6.2, CD-ROM #1, file RedHat/base/comps, defines the "base" (minimum) Red Hat Linux 6.2 installation as a set of packages. The following are the build directories corresponding to this base (minimum) installation, along with the SLOC counts (as shown above). Note that this creates a text-only system:

Component	SLOC
anacron-2.1	1143
apmd	3012
ash-linux-0.2	9666
at-3.1.7	3084
authconfig-3.0.3	1075
bash-1.14.7	47067
bc-1.05	17682
bdf flush-1.5	261
binutils-2.9.5.0.22	467120
bzip2-0.9.5d	4996
chkconfig-1.1.2	717
console-tools-0.3.3	15522
cpio-2.4.2	7617
cracklib, 2.7	1987
dev-2.7.18	0
diffutils-2.7	10914
dump-0.4b15	10187
e2fsprogs-1.18	28169
ed-0.2	7427
egcs-1.1.2	720112
eject-2.0.2	657
file-3.28	2647
fileutils-4.0p	34768
findutils-4.1	11404
gawk-3.0.4	26363
gd1.3	20078
gdbm-1.8.0	3315
getty_ps-2.0.7j	2631
glibc-2.1.3	415026
gmp-2.0.2	24583
gnupg-1.0.1	54935
gpm-1.18.1	9725
grep-2.4	10013
groff-1.15	70260
gzip-1.2.4a	6306
hdparm-3.6	1229

Estimating Linux's Size

initscripts-5.00	3929
isapnptools-1.21	5960
kbdconfig-1.9.2.4	368
kernelcfg-0.5	347
kudzu-0.36	0
ldconfig-1999-02-21	874
ld.so-1.9.5	9731
less-346	14039
lilo	7255
linuxconf-1.17r2	104032
logrotate-3.3.2	1525
mailcap-2.0.6	0
mailx-8.1.1	6968
MAKEDEV-2.5.2	1020
man-1.5hl	9801
mingetty-0.9.4	343
mkbootdisk-1.2.5	314
mkinitrd-2.4.1	288
mktemp-1.5	85
modutils-2.3.9	11775
mouseconfig-4.4	830
mt-st-0.5b	1361
ncompress-4.2.4	1435
ncurses-5.0	61324
net-tools-1.54	11633
newt-0.50.8	7041
pam-0.72	20433
passwd-0.64.1	1194
pciutils-2.1.5	3855
popt-1.4	1099
procmail-3.14	9927
procps-2.0.6	9961
psmisc	1630
pump-0.7.8	1856
pwdb-0.61	9551
quota-2.00-pre3	3804
raidtools-0.90	2424
readline-2.2.1	14941
redhat-logos	0
rootfiles	0
rpm-3.0.4	39861
sash-3.4	6172
sed-3.02	7740
sendmail-8.9.3	42880
setserial-2.15	742
setup-1.2	67
setup-2.1.8	0
shadow-19990827	25236
sh-utils-2.0	17939
slang	28118
slocate-2.1	1883
stat-1.5	280
sysklogd-1.3-31	4038
sysvinit-2.78	6033
tar-1.13.17	14255
termcap-2.0.8	797
texinfo-4.0	28186
textutils-2.0a	36338
time-1.7	1452
timeconfig-3.0.3	346

tmpwatch-2.2	463
utempter-0.5.2	222
util-linux-2.10f	39160
vim-5.6	113241
vixie-cron-3.0.1	2879
which-2.9	2268
zlib-1.1.3	4087

Thus, the contents of the build directories corresponding to the ``base" (minimum) installation totals to 2,819,334 SLOC.

A few notes are in order about this build directory total:

1. Some of the packages listed by a traditional package list aren't shown here because they don't contain any code. Package "basesystem" is a pseudo-package for dependency purposes. Package redhat-release is just a package for keeping track of the base system's version number. Package "filesystem" contains a directory layout.
2. ntsysv's source is in chkconfig-1.1.2; kernel-utils and kernel-pcmcia-cs are part of "linux". Package shadow-utils is in build directory shadow-19990827. Build directory util-linux includes losetup and mount. "dump" is included to include rmt.
3. Sometimes the build directories contain more code than is necessary to create just the parts for the ``base" system; this is a side-effect of how things are packaged. ``info" is included in the base, so we count all of texinfo. The build directory termcap is counted, because libtermcap is in the base. Possibly most important, egcs is there because libstdc++ is in the base.
4. Sometimes a large component is included in the base, even though most of the time little of its functionality is used. In particular, the mail transfer agent ``sendmail" is in the base, even though for many users most of sendmail's functionality isn't used. However, for this paper's purposes this isn't a problem. After all, even if sendmail's functionality is often underused, clearly that functionality took time to develop and that functionality is available to those who want it.
5. My tools intentionally eliminated duplicates; it may be that a few files aren't counted here because they're considered duplicates of another build directory not included here. I do not expect this factor to materially change the total.
6. Red Hat Linux is not optimized to be a ``small as possible" distribution; their emphasis is on functionality, not small size. A working Linux distribution could include much less code, depending on its intended application. For example, ``linuxconf" simplifies system configuration, but the system can be configured by editing its system configuration files directly, which would reduce the base system's size. This also includes vim, a full-featured text editor - a simpler editor with fewer functions would be smaller as well.

Many people prefer some sort of graphical interface; here is a minimal configuration of a graphical system, adding the X server, a window manager, and a few tools:

Component	SLOC
XFree86-3.3.6	1291745
Xconfigurator-4.3.5	9741
fvwm-2.2.4	69265
X11R6-contrib-3.3.2	18885

These additional graphical components add 1,389,636 SLOC. Due to oddities of the way the initialization system xinitrc is built, it isn't shown here in the total, but xinitrc has so little code that its omission does not significantly affect the total.

Adding these numbers together, we now have a total of 4,208,970 SLOC for a ``minimal graphical system." Many people would want to add more components. For example, this doesn't include a graphical toolkit (necessary for running most graphical applications). We could add gtk+-1.2.6 (a toolkit needed for running GTK+ based applications), adding 138,118 SLOC. This would now total 4,347,088 for a ``basic graphical system," one able to run basic GTK+ applications.

Let's add a web server to the mix. Adding apache_1.3.12 adds only 77,873 SLOC. We now have 4,424,961 physical SLOC for a basic graphical system plus a web server.

We could then add a graphical desktop environment, but there are so many different options and possibilities that trying to identify a ``minimal" system is hard to do without knowing the specific uses intended for the system. Red Hat defines a standard ``GNOME" and ``KDE" desktop, but these are intended to be highly functional (not ``minimal").

Thus, we'll stop here, with a total of 2.8 million physical SLOC for a minimal text-based system, and total of 4.4 million physical SLOC for a basic graphical system plus a web server.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the ``Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

[Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>

[Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf

[Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>

[Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.

[Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.

[NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>

[OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.

[Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>

[Raymond 1999] Raymond, Eric S. January 1999. ``A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.

[Schneier 2000] Schneier, Bruce. March 15, 2000. ``Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...". <http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet. <http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000] Wheeler, David A. 2000. *Open Source Software / Free Software References*. http://www.dwheeler.com/oss_fs_refs.html.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

This paper is (C) Copyright 2000 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. When referring to the paper, please refer to it as ``Estimating GNU/Linux's Size'' by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

Estimating Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

November 3, 2000

Version 1.02

This paper presents size estimates (and their implications) of the source code of a distribution of the Linux operating system (OS), a combination often called GNU/Linux. The distribution used in this paper is Red Hat Linux version 6.2, including the kernel, software development tools, graphics interfaces, client applications, and so on. Other distributions and versions will have different sizes.

In total, this distribution includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars).

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), gcc (a compilation system), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The Linux operating system (also called GNU/Linux) has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a "typical" intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally "open source software" and/or "free software" (where the "free" here means "freedom"). A program that is "open source software" or "free software" is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of "open source software" is available at OSI [1999], a more formal definition of "free software" is available at FSF [2000], and other general information about these topics is available at Wheeler [2000a]. Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is "open source software"/"free software", and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. The only published data I've found was developed by Microsoft in the documents usually called "Halloween I" and "Halloween II". Unfortunately, the meaning, derivation, and assumptions of their numbers is not explained, making the numbers hard to use and truly understand. Even worse, although the two documents were written by essentially the same people at the same time, the numbers in the documents appear (on their surface) to be contradictory. The so-called "Halloween I" document claimed that the Linux kernel (x86 only) was 500,000 lines of code, the Apache web server was 80,000 lines of code, the X-windows server was 1.5 million, and a full Linux distribution was about 10 million lines of code [Halloween I]. The "Halloween II" document seemed to contradict this, saying that "Linux" by 1998 included 1.5 million lines of code. Since "version 2.1.110" is identified as the version number, presumably this only measures the Linux kernel, and it does note that this measure includes all Linux ports to various architectures [Halloween II]. However, this asks as many questions as it answers - what exactly was being measured, and

what assumptions were made? You could infer from these documents that the Linux kernel's support for other architectures took one million lines of code - but this appeared unlikely. Another study, [\[Dempsey 1999\]](#), did analyze open source programs, but it primarily focused on statistics about developers and used basic filesize numbers to report about the software.

This paper bridges this gap. In particular, it shows estimates of the size of Linux, and it estimates how much it would cost to rebuild a typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean.

For my purposes, I have selected as my "representative" Linux distribution Red Hat Linux version 6.2. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [\[Shankland 2000b\]](#). Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE at 15%. Not all Linux copies are "sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate "all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 6.2 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Section 2 briefly describes the approach used to estimate the "size" of this distribution (most of the details are in Appendix A). Section 3 discusses some of the results (with the details in Appendix B). Section 4 presents conclusions, followed by the two appendices.

2. Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; the steps and assumptions made are described in Appendix A.

A few summary points are worth mentioning here, however, for those who don't read appendix A. I included software for all architectures, not just the i386. I did not include "old" versions of software (with the one exception of bash, as discussed in appendix A). I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once. The code in makefiles and RPM package specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

The "physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: "a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the "logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the "count of all terminating semicolons in a C file." The "physical" SLOC was chosen instead of the "logical" SLOC because there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I

had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the "physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer "COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total SLOC counts, and section 3.6 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 25 largest components (as measured by number of source lines of code):

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,csh=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csh=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csh=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csh=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,csh=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,


```

perl=417
200628 Python-1.5.2 python=100935,ansic=96323,lisp=2353,sh=673,perl=342,
sed=2
199982 gs5.50 ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916 teTeX-1.0 ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,
yacc=1507,awk=522,lex=323,sed=297,asm=139,csh=47,lisp=29
155035 bind-8.2.2_P5 ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,
csh=848,awk=753,lex=222
140130 AfterStep-APPS-20000124 ansic=135806,sh=3340,cpp=741,perl=243
138931 kdbase cpp=113971,ansic=23016,perl=1326,sh=618
138118 gtk+-1.2.6 ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024 gated-3-5-11 ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csh=235,
sed=35,lisp=12
133193 kaffe-1.0.5 java=65275,ansic=62125,cpp=3923,perl=972,sh=814,
asm=84
131372 jade-1.2.1 cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672 gnome-libs-1.0.55 ansic=125373,sh=2178,perl=667,awk=277,lisp=177

```

Note that the operating system kernel (linux) is the largest single component, at over 1.5 million lines of code (mostly in C). See section 3.2 for a more discussion discussion of the linux kernel.

The next largest component is the X windows server, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to accrete functionality), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, which is confusingly named ``egcs" instead. The naming conventions of gcc can be confusing, so a little explanation is in order. Officially, the compilation system is called ``gcc". Egcs was a project to experiment with a more open development model for gcc. Red Hat Linux 6.2 used one of the gcc releases from the egcs project, and called the release egcs-1.1.2 to avoid confusion with the official (at that time) gcc releases. The egcs experiment was a success; egcs as a separate project no longer exists, and current gcc development is based on the egcs code and development model. To sum it up, the compilation system is named ``gcc", and the version of gcc used here is a version developed by ``egcs".

Following this is the symbolic debugger and emacs. Emacs is probably not a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system. This is followed by the set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). This is followed by TCL/Tk (a combined language and widget set), PostgreSQL (a relational DBMS), and the GIMP (an excellent client application for editing bitmapped drawings).

Note that language implementations tend to be written in themselves, particularly for their libraries. Thus there is more Perl than any other single language in the Perl implementation, more Python than any other single language in Python, and more Java than any other single language in Kaffe (an implementation of the Java Virtual Machine and library).

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the linux kernel (at over 1.5 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 870,000 lines of this code was in the ``drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of hardware. The linux kernel's design is expressed in its source code directory structure, and no other directory comes close to this size - the second largest is the ``arch" directory (at over 230,000 SLOC), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is not quite 88,000 SLOC. See the appendix for more detail.

Richard Stallman and others have argued that the resulting system often called ``Linux" should instead be called ``GNU/Linux" [\[Stallman 2000\]](#). In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to ``free software" (free as in freedom). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called ``Linux". Using the term ``Linux" is particularly bizarre for GNU/Hurd, which takes the Debian GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux" or ``GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc (packaged under the name ``egcs"), gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux" and not just ``Linux."

These measurements at least debunk one possible explanation of the Halloween documents' measures. Since Halloween I claimed that the x86-only code for the Linux kernel measured 500,000 SLOC, while Halloween II claimed that the kernel (all architectures) was 1.5 million SLOC, one explanation of this difference would be that the code for non-x86 systems was 1 million SLOC. This isn't so; I computed a grand total of 267,320 physical SLOC of non-i86 code (including drivers and architecture-specific code). It seems unlikely that over 700,000 lines of code would have been removed (not added) in the intervening time.

However, other measures (and explanations) are more promising. I also ran the CodeCount tools on the linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for the Linux kernel. When I removed all non-i86 code and re-ran the CodeCount tool on just the C code, a logical SLOC of 570,039 of C code was revealed. Since the Halloween I document reported 500,000 SLOC (when only including x86 code), it appears very likely that the Halloween I paper counted logical SLOC (and only C code) when reporting measurements of the linux kernel. However, the other Halloween I measures appear to be physical SLOC measures: their estimate of 1.5 million SLOC for the X server is closer to the 1.2 million physical SLOC measured here, and their estimate of 80,000 SLOC for Apache is close to the 77,873 SLOC measured here (as shown in Appendix B). These variations in measurements should be expected, since the versions I am measuring are slightly different than the ones they measured, and it is likely that some assumptions are different as well. Meanwhile, Halloween II reported a measure of 1.5 million lines of code for the linux kernel, essentially the same value given here for physical SLOC.

In short, it appears that Halloween I used the ``logical SLOC" measure when measuring the Linux kernel, while all other measures in Halloween I and II used physical SLOC as the measure. I have attempted to contact the Microsoft author to confirm this, but as of yet I have not received such confirmation. In any case, this example clearly demonstrates the need to carefully identify the units of measure and assumptions made in any measurement of SLOC.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code:

ansic:	14218806	(80.55%)
cpp:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)
python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Here you can see that C is pre-eminent (with over 80% of the code), followed by C++, LISP, shell, and Perl. Note that the

separation of Expect and TCL is somewhat artificial; if combined, they would be next (at 232115), followed by assembly. Following this in order are Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has over a million lines of code, a very respectable showing, and yet at least in this distribution it is far less than C. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

The fact that LISP places so highly (it's in third place) is a little surprising. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 80% (453647/565861) of the total amount of LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages: Perl includes 5584 lines of LISP, and Python includes another 2333 of LISP that is directly used to support elaborate Emacs modes for program editing. The ``psgml" package is solely an emacs mode for editing SGML documents. The components with the second and third largest amounts of LISP are xlipstat-3-52-17 and scheme-3.2, which are implementations of LISP and Scheme (a LISP dialect) respectively. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 57 different lex/flex files, and 110 yacc/bison files. Since some build directories use lex/flex or yacc/bison more than once, the count of build directories using these tools is smaller but still respectable: 38 different build directories use lex/flex, and 62 different build directories use yacc/bison.

Other insights can be gained from the file counts shown in appendix B. The number of source code files counted were 72,428. Not included in this count were 5,820 files which contained duplicate contents, and 817 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 14218806 SLOC contained in 52088 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code.

3.5 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 17,652,561 physical source lines of code (SLOC); I will simplify this to ``over 17 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (1998)	20 million

These numbers come from Bruce Schneier's *Crypto-Gram* [Schneier 2000], except for the Space Shuttle numbers which come from a National Academy of Sciences study [NAS 1996]. Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Schneier also reports that "Linux, even with the addition of X Windows and Apache, is still under 5 million lines of code". At first, this seems to be contradictory, since this paper counts over 17 million SLOC, but Schneier appears to be literally correct in the context of his statement. The phrasing of his sentence suggests that Schneier is considering some sort of "minimal" system, since he considers "even the addition of X Windows" as a significant addition. As shown in appendix section B.4, taking the minimal "base" set of components in Red Hat Linux, and then adding the minimal set of components for graphical interaction (the X Windows's graphical server, library, configuration tool, and a graphics toolkit) and the Apache web server, the total is about 4.4 million physical SLOC - which is less than 5 million. This minimal system doesn't include some useful (but not strictly necessary) components, but a number of useful components could be added while still staying under a total of 5 million SLOC.

However, note the contrast. Many Linux distributions include with their operating systems many applications (e.g., bitmap editors) and development tools (for many different languages). As a result, the entire *delivered* system for such distributions (including Red Hat Linux 6.2) is *much* larger than the 5 million SLOC stated by Schneier. In short, this distribution's size appears similar to the size of Windows 98 and Windows NT 5.0 in 1998.

Microsoft's recent legal battles with the U.S. Department of Justice (DoJ) also involve the bundling of applications with the operating system. However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as "secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, many Linux distributions include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with small SLOC counts can often provide greater functionality than programs with large SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessarily cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized systems.

3.6 Effort and Cost Estimates

Finally, given all the assumptions shown, are the effort values:

```
Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = $ 614421924.71
```

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux version 6.2 includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars). Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches.

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), gcc (a compilation system, with the package name of ``egcs''), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Here you can see that C is pre-eminent (with over 80% of the code), More information is available in the appendices and at <http://www.dwheeler.com/sloc>.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), other open source systems (such as FreeBSD), and other versions of Red Hat (such as Red Hat 7). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost. It's known that Red Hat 7 includes more source code; Red Hat 7 has had to add another CD-ROM to contain the binary programs, and adds such capabilities as a word processor (abiword) and secure shell (openssh).

Some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like ``Automatically generated"). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not ``reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ``.m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

Appendix A. Details of Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; each step is described below. Some steps I describe in some detail, because it's sometimes hard to find the necessary information even when the actual steps are easy. Hopefully, this detail will make it easier for others to do similar activities or to repeat the experiment.

A.1 Installing Source Code

Installing the source code files turned out to be nontrivial. First, I inserted the CD-ROM containing all of the source files (in ``.src.rpm" format) and installed the packages (files) using:

```
mount /mnt/cdrom
```

```
cd /mnt/cdrom/SRPMs
rpm -ivh *.src.rpm
```

This installs ``spec" files and compressed source files; another rpm command (`rpm -bp") uses the spec files to uncompress the source files into ``build directories" (as well as apply any necessary patches). Unfortunately, the rpm tool does not enforce any naming consistency between the package names, the spec names, and the build directory names; for consistency this paper will use the names of the build directories, since all later tools based themselves on the build directories.

I decided to (in general) not count ``old" versions of software (usually placed there for compatibility reasons), since that would be counting the same software more than once. Thus, the following components were not included: ``compat-binutils", ``compat-egcs", ``compat-glib", ``compat-libs", ``gtk+10", ``libc-5.3.12" (an old C library), ``libxml10", ``ncurses3", and ``qt1x". I also didn't include egcs64-19980921 and netscape-sparc, which simply repeated something on another architecture that was available on the i386 in a different package. I did make one exception. I kept both bash-1.14.7 and bash2, two versions of the shell command processor, instead of only counting bash2. While bash2 is the later version of the shell available in the package, the main shell actually used by the Red Hat distribution was the older version of bash. The rationale for this decision appears to be backwards compatibility for older shell scripts; this is suggested by the Red Hat package documentation in both bash-1.14.7 and bash2. It seemed wrong to not include one of the most fundamental pieces of the system in the count, so I included it. At 47067 lines of code (ignoring duplicates), bash-1.14.7 is one of the smaller components anyway. Not including this older component would not substantively change the results presented here.

There are two directories, krb4-1.0 and krb5-1.1.1, which appear to violate this rule - but don't. krb5-1.1.1 is the build directory created by krb5.spec, which is in turn installed by the source RPM package krb5-1.1.1-9.src.rpm. This build directory contains Kerberos V5, a trusted-third-party authentication system. The source RPM package krb5-1.1.1-9.src.rpm eventually generates the binary RPM files krb5-configs-1.1.1-9, krb5-libs-1.1.1-9, and krb5-devel-1.1.1-9. You might guess that ``krb4-1.0" is just the older version of Kerberos, but this build directory is created by the spec file krbafs.spec and not just an old version of the code. To quote its description, ``This is the Kerberos to AFS bridging library, built against Kerberos 5. krbafs is a shared library that allows programs to obtain AFS tokens using Kerberos IV credentials, without having to link with official AFS libraries which may not be available for a given platform." For this situation, I simply counted both packages, since their purposes are different.

I was then confronted with a fundamental question: should I count software that only works for another architecture? I was using an i86-type system, but some components are only for Alpha or Sparc systems. I decided that I should count them; even if I didn't use the code today, the ability to use these other architectures in the future was of value and certainly required effort to develop.

This caused complications for creating the build directories. If all installed packages fit the architecture, you can install the uncompressed software by typing:

```
cd /usr/src/redhat/SPECS and typing the command
rpm -bp *.spec
```

Unfortunately, the rpm tool notes that you're trying to load code for the ``wrong" architecture, and (at least at the time) there was no simple ``override" flag. Instead, I had to identify each package as belonging to SPARC or ALPHA, and then use the rpm option --target to forcibly load them. For example, I renamed all sparc-specific SPARC file files to end in ``.sparc" and could then load them with:

```
rpm -bp --target sparc-redhat-linux *.spec.sparc
```

The following spec files were non-i86: (sparc) audiocctl, elftoaout, ethtool, prtconf, silo, solemul, sparc32; (alpha) aboot, minlabel, quickstrip. In general, these were tools to aid in supporting some part of the boot process or for using system-specific hardware.

Note that not all packages create build directories. For example, ``anonftp" is a package that, when installed, sets up an anonymous ftp system. This package doesn't actually install any software; it merely installs a specific configuration of another piece of software (and unsets the configuration when uninstalled). Such packages are not counted at all in this sizing estimate.

Simply loading all the source code requires a fair amount of disk space. Using ``du" to measure the disk space requirements (with 1024 byte disk blocks), I obtained the following results:

```
$ du -s /usr/src/redhat/BUILD /usr/src/redhat/SOURCES /usr/src/redhat/SPECS
2375928 /usr/src/redhat/BUILD
592404 /usr/src/redhat/SOURCES
4592 /usr/src/redhat/SPECS
```

Thus, these three directories required 2972924 1K blocks - approximately 3 gigabytes of space. Much more space would be required to compile it all.

A.2 Categorizing Source Code

My next task was to identify all files containing source code (not including any automatically generated source code). This is a non-trivial problem; there are 181,679 ordinary files in the build directory, and I had no interest in doing this identification by hand.

In theory, one could just look at the file extensions (.c for C, .py for python), but this is not enough in practice. Some packages reuse extensions if the package doesn't use that kind of file (e.g., the ``.exp" extension of expect was used by some packages as ``export" files, and the ``.m" of objective-C was used by some packages for module information extracted from C code). Some files don't have extensions, particularly scripts. And finally, files automatically generated by another program should not be counted, since I wished to use the results to estimate effort.

I ended up writing a program of over 600 lines of Perl to perform this identification, which used a number of heuristics to categorize each file into categories. There is a category for each language, plus the categories non-programs, unknown (useful for scanning for problems), automatically generated program files, duplicate files (whose file contents duplicated other files), and zero-length files.

The program first checked for well-known extensions (such as .gif) that cannot be program files, and for a number of common generated filenames. It then peeked at the first line for "#!" followed by a legal script name. If that didn't work, it used the extension to try to determine the category. For a number of languages, the extension was not reliable, so for those languages the categorization program examined the file contents and used a set of heuristics to determine if the file actually belonged that category. If all else failed, the file was placed in the ``unknown" category for later analysis. I later looked at the ``unknown" items, checking the common extensions to ensure I had not missed any common types of code.

One complicating factor was that I wished to separate C, C++, and objective-C code, but a header file ending with ``.h" or ``.hpp" file could be any of them. I developed a number of heuristics to determine, for each file, what language it belonged to. For example, if a build directory has exactly one of these languages, determining the correct category for header files is easy. Similarly, if there is exactly one of these in the directory with the header file, it is presumed to be that kind. Finally, a header file with the keyword ``class" is almost certainly not a C header file, but a C++ header file.

Detecting automatically generated files was not easy, and it's quite conceivable I missed a number of them. The first 15 lines were examined, to determine if any of them included at the beginning of the line (after spaces and possible comment markers) one of the following phrases: ``generated automatically", ``automatically generated", ``this is a generated file", ``generated with the (something) utility", or ``do not edit". A number of filename conventions were used, too. For example, any ``configure" file is presumed to be automatically generated if there's a ``configure.in" file in the same directory.

To eliminate duplicates, the program kept md5 checksums of each program file. Any given md5 checksum would only be counted once. Build directories were processed alphabetically, so this meant that if the same file content was in both directories ``a" and ``b", it would be counted only once as being part of ``a". Thus, some packages with names later in the alphabet may appear smaller than would make sense at first glance. It is very difficult to eliminate ``almost identical" files (e.g., an older and newer version of the same code, included in two separate packages), because it is difficult to determine when ``similar" two files are essentially the ``same" file. Changes such as the use of pretty-printers and massive renaming of variables could make small changes seem large, while the many small files in the system could easily make different files seem the ``same." Thus, I did not try to make such a determination, and just considered files with different contents as different.

It's important to note that different rules could be used to ``count" lines of code. Some kinds of code were intentionally excluded from the count. Many RPM packages include a number of shell commands used to install and uninstall software; the estimate in this paper does not include the code in RPM packages. This estimate also does not include the code in Makefiles (which can be substantive). In both cases, the code in these cases is often cut and pasted from other similar files, so counting such code would probably overstate the actual development effort. In addition, Makefiles are often automatically generated.

On the other hand, this estimate does include some code that others might not count. This estimate includes test code included with the package, which isn't visible directly to users (other than hopefully higher quality of the executable program). It also includes code not used in this particular system, such as code for other architectures and OS's, bindings for languages not compiled into the binaries, and compilation-time options not chosen. I decided to include such code for two reasons. First, this code is validly represents the effort to build each component. Second, it does represent indirect value to the user, because the user can later use those components in other circumstances even if the user doesn't choose to do so by default.

So, after the work of categorizing the files, the following categories of files were created for each build directory (common extensions are shown in parentheses, and the name used in the data tables below are shown in brackets):

1. C (.c) [ansic]
2. C++ (.C, .cpp, .cxx, .cc) [cpp]
3. LISP (.el, .scm, .lsp, .jl) [lisp]
4. shell (.sh) [sh]
5. Perl (.pl, .pm, .perl) [perl]
6. Assembly (.s, .S, .asm) [asm]
7. TCL (.tcl, .tk, .itk) [tcl]
8. Python (.py) [python]
9. Yacc (.y) [yacc]
10. Java (.java) [java]
11. Expect (.exp) [exp]
12. lex (.l) [lex]
13. awk (.awk) [awk]
14. Objective-C (.m) [objc]
15. C shell (.csh) [csh]
16. Ada (.ada, .ads, .adb) [ada]
17. Pascal (.p) [pascal]
18. sed (.sed) [sed]
19. Fortran (.f) [fortran]

Note that we're counting Scheme as a dialect of LISP, and Expect is being counted separately from TCL. The command line shells Bourne shell, the Bourne-again shell (bash), and the K shell are all counted together as ``shell'', but the C shell (csh and tcsh) is counted separately.

A.3 Counting Lines of Code

Every language required its own counting scheme. This was more complex than I realized; there were a number of languages involved.

I originally tried to use USC's ``CodeCount'' tools to count the code. Unfortunately, this turned out to be buggy and did not handle most of the languages used in the system, so I eventually abandoned it for this task and wrote my own tools. Those who wish to use this tool are welcome to do so; you can learn more from its web site at <http://sunset.usc.edu/research/CODECOUNT>.

I did manage to use the CodeCount to compute the logical source lines of code for the C portions of the linux kernel. This came out to be 673,627 logical source lines of code, compared to the 1,462,165 lines of physical code (again, this ignores files with duplicate contents).

Since there were a large number of languages to count, I used the ``physical lines of code'' definition. In this definition, a line of code is a line (ending with newline or end-of-file) with at least one non-comment non-whitespace character. These are known as ``non-comment non-blank'' lines. If a line only had whitespace (tabs and spaces) it was not counted, even if it was in the middle of a data value (e.g., a multiline string). It is much easier to write programs to measure this value than to measure the ``logical'' lines of code, and this measure can be easily applied to widely different languages. Since I had to process a large number of different languages, it made sense to choose the measure that is easier to obtain.

[Park \[1992\]](#) presents a framework of issues to be decided when trying to count code. Using Park's framework, here is how code was counted in this paper:

1. Statement Type: I used a physical line-of-code as my basis. I included executable statements, declarations (e.g., data structure definitions), and compiler directives (e.g., preprocessor commands such as #define). I excluded all comments and blank lines.
2. How Produced: I included all programmed code, including any files that had been modified. I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. If a file was in the source package, I included it; if the file had been removed from a source package (including via a patch), I did not include it.
3. Origin: I included all code included in the package.

4. Usage: I included code in or part of the primary product; I did not include code external to the product (i.e., additional applications able to run on the system but not included with the system).
5. Delivery: I counted code delivered as source; not surprisingly, I didn't count code not delivered as source. I also didn't count undelivered code.
6. Functionality: I included both operative and inoperative code. An examples of intentionally ``inoperative" code is code turned off by `#ifdef` commands; since it could be turned on for special purposes, it made sense to count it. An examples of unintentionally ``inoperative" code is dead or unused code.
7. Replications: I included master (original) source statements. I also included ``physical replicates of master statements stored in the master code". This is simply code cut and pasted from one place to another to reuse code; it's hard to tell where this happens, and since it has to be maintained separately, it's fair to include this in the measure. I excluded copies inserted, instantiated, or expanded when compiling or linking, and I excluded postproduction replicates (e.g., reparameterized systems).
8. Development Status: Since I only measured code included in the packages used to build the delivered system, I declared that all software I was measuring had (by definition) passed whatever ``system tests" were required by that component's developers.
9. Languages: I included all languages, as identified earlier in section A.2.
10. Clarifications: I included all statement types. This included nulls, continues, no-ops, lone semicolons, statements that instantiate generics, lone curly braces (`{` and `}`), and labels by themselves.

Park includes in his paper a ``basic definition" of physical lines of code, defined using his framework. I adhered to Park's definition unless (1) it was impossible in my technique to do so, or (2) it would appear to make the result inappropriate for use in cost estimation (using COCOMO). COCOMO states that source code:

``includes all program instructions created by project personnel and processed into machine code by some combination of preprocessors, compilers, and assemblers. It excludes comment cards and unmodified utility software. It includes job control language, format statements, and data declarations. Instructions are defined as lines of code."

In summary, though in general I followed Park's definition, I didn't follow Park's ``basic definition" in the following ways:

1. How Produced: I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. After all, COCOMO states that the only code that should be counted is code ``produced by project personnel", whereas these kinds of files are instead the output of ``preprocessors and compilers." If code is always maintained as the input to a code generator, and then the code generator is re-run, it's only the code generator input's size that validly measures the size of what is maintained. Note that while I attempted to exclude generated code, this exclusion is based on heuristics which may have missed some cases.
2. Origin: Normally physical SLOC doesn't include an unmodified ``vendor-supplied language support library" nor a ``vendor-supplied system or utility". However, in this case this was *exactly* what I was measuring, so I naturally included these as well.
3. Delivery: I didn't count code not delivered as source. After all, since I didn't have it, I couldn't count it.
4. Functionality: I included unintentionally inoperative code (e.g., dead or unused code). There might be such code, but it is very difficult to automatically detect in general for many languages. For example, a program not directly invoked by anything else nor installed by the installer is much more likely to be a test program, which I'm including in the count. Clearly, discerning human ``intent" is hard to automate. Hopefully, unintentionally inoperative code is a small amount of the total delivered code.

Otherwise, I followed Park's ``basic definition" of a physical line of code, even down to Park's language-specific definitions where Park defined them for a language.

One annoying problem was that one file wasn't syntactically correct and it affected the count. File `/usr/src/redhat/BUILD/cdrecord-1.8/mkiso` had an `#ifdef` not taken, and the road not taken had a missing double-quote mark before the word ``cannot":

```
#ifdef  USE_LIBSCHILY
        comerr(Cannot open '%s'.\n", filename);
#endif
        perror ("fopen");
        exit (1);
#endif
```

I solved this by hand-patching the source code (for purposes of counting). There were also some files with intentionally erroneous code (e.g., compiler error tests), but these did not impact the SLOC count.

Several languages turn out to be non-trivial to count:

- In C, C++, and Java, comment markers should be ignored inside strings. Since they have multi-line comment markers this requirement should not be ignored, or a ```/*` inside a string could cause most of the code to be erroneously uncounted.
- Officially, C doesn't have C++'s `///
` here" documents, and an __END__ marker that complicate code-counting. Perlpod documents are essentially comments, but a ``here document may include text to generate them (in which case the perlpod document is data and should be counted). The __END__ marker indicates the end of the file from Perl's viewpoint, even if there's more text afterwards.`
- Python has a convention that, at the beginning of a definition (e.g., of a function, method, or class), an unassigned string can be placed to describe what's being defined. Since this is essentially a comment (though it doesn't syntactically look like one), the counter must avoid counting such strings, which may have multiple lines. To handle this, strings which started the beginning of a line were not counted. Python also has the ```triple quote` operator, permitting multiline strings; these needed to be handled specially. Triple quote strings were normally considered as data, regardless of content, unless they were used as a comment about a definition.
- Assembly languages vary greatly in the comment character they use, so my counter had to handle this variance. I wrote a program which first examined the file to determine if C-style ```/*` comments and C preprocessor commands (e.g., ```#include`) were used. If both ```/*` and ```*/` were in the file, it was assumed that C-style comments were used, since it is unlikely that *both* would be used as something else (e.g., as string data) in the same assembly language file. Determining if a file used the C preprocessor was trickier, since many assembly files do use ```#` as a comment character and some preprocessor directives are ordinary words that might be included in a human comment. The heuristic used was: if `#ifdef`, `#endif`, or `#include` are used, the preprocessor is used; if at least three lines have either `#define` or `#else`, then the preprocessor is used. No doubt other heuristics are possible, but this at least seemed to produce reasonable results. The program then determined what the comment character was, by identifying which punctuation mark (from a set of possible marks) was the most common non-space initial character on a line (ignoring ```/` and ```#` if C comments or preprocessor commands, respectively, were used). Once the comment character had been determined, and it had been determined if C-style comments were also allowed, the lines of code could be counted in the file.

Although their values are not used in estimating effort, I also counted the number of files; summaries of these values are included in appendix B.

Since the linux kernel was the largest single component, and I had questions about the various inconsistencies in the ```Halloween` documents, I made additional measures of the Linux kernel.

Some have objected because the counting approach used here includes lines not compiled into code in this Linux distribution. However, the primary objective of these measures was to estimate total effort to develop all of these components. Even if some lines are not normally enabled on Linux, it still required effort to develop that code. Code for other architectures still has value, for example, because it enables users to port to other architectures while using the component. Even if that code is no longer being maintained (e.g., because the architecture has become less popular), nevertheless someone had to invest effort to create it, the results benefitted someone, and if it is needed again it's still there (at least for use as a starting point). Code that is only enabled by compile-time options still has value, because if the options were desired the user could enable them and recompile. Code that is only used for testing still has value, because its use improves the quality of the software directly run by users. It is possible that there is some ```dead code` (code that cannot be run under any circumstance), but it is expected that this amount of code is very small and would not significantly affect the results. Andi Kleen (of SuSE) noted that if you wanted to only count compiled and running code, one technique (for some languages) would be to use gcc's ```-g` option and use the resulting `.stabs` debugging information with some filtering (to exclude duplicated inline functions). I determined this to be out-of-scope for this paper, but this approach could be used to make additional measurements of the system.

A.4 Estimating Effort and Costs

For each build directory, I totalled the source lines of code (SLOC) for each language, then totalled those values to determine the SLOC for each directory. I then used the basic Constructive Cost Model (COCOMO) to estimate effort. The basic model is the simplest (and least accurate) model, but I simply did not have the additional information necessary to use the more complex (and more accurate) models. COCOMO is described in depth by Boehm [1981].

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been

developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions.

In the basic COCOMO model, estimated man-months of effort, design through test, equals $2.4 \cdot (\text{KSLOC})^{1.05}$, where KSLOC is the total physical SLOC divided by 1000.

I assumed that each package was built completely independently and that there were no efforts necessary for integration not represented in the code itself. This almost certainly underestimates the true costs, but for most packages it's actually true (many packages don't interact with each other at all). I wished to underestimate (instead of overestimate) the effort and costs, and having no better model, I assumed the simplest possible integration effort. This meant that I applied the model to each component, then summed the results, as opposed to applying the model once to the grand total of all software.

Note that the only input to this model is source lines of code, so some factors simply aren't captured. For example, creating some kinds of data (such as fonts) can be very time-consuming, but this isn't directly captured by this model. Some programs are intentionally designed to be data-driven, that is, they're designed as small programs which are driven by specialized data. Again, this data may be as complex to develop as code, but this is not counted.

Another example of uncaptured factors is the difficulty of writing kernel code. It's generally acknowledged that writing kernel-level code is more difficult than most other kinds of code, because this kind of code is subject to a subtle timing and race conditions, hardware interactions, a small stack, and none of the normal error protections. In this paper I do not attempt to account for this. You could try to use the Intermediate COCOMO model to try to account for this, but again this requires knowledge of other factors that can only be guessed at. Again, the effort estimation probably significantly underestimates the actual effort represented here.

It's worth noting that there is an update to COCOMO, COCOMO II. However, COCOMO II requires as its input logical (not physical) SLOC, and since this measure is much harder to obtain, I did not pursue it for this paper. More information about COCOMO II is available at the web site <http://sunset.usc.edu/research/COCOMOII/index.html>. A nice overview paper where you can learn more about software metrics is Masse [1997].

I assumed that an average U.S. programmer/analyst salary in the year 2000 was \$56,286 per year; this value was from the ComputerWorld, September 4, 2000's Salary Survey. Overhead is much harder to estimate; I did not find a definitive source for information on overheads. After informal discussions with several cost analysts, I determined that an overhead of 2.4 would be representative of the overhead sustained by a typical software development company. Should you disagree with these figures, I've provided all the information necessary to recalculate your own cost figures; just start with the effort estimates and recalculate cost yourself.

Appendix B. More Detailed Results

This appendix provides some more detailed results. B.1 lists the SLOC found in each build directory; B.2 shows counts of files for each category of file; B.3 presents some additional measures about the Linux kernel. B.4 presents some SLOC totals of putatively "minimal" systems. You can learn more at <http://www.dwheeler.com/sloc>.

B.1 SLOC in Build Directories

The following is a list of all build directories, and the source lines of code (SLOC) found in them, followed by a few statistics counting files (instead of SLOC).

Remember that duplicate files are only counted once, with the build directory "first in ASCII sort order" receiving any duplicates (to break ties). As a result, some build directories have a smaller number than might at first make sense. For example, the "kudzu" build directory does contain code, but all of it is also contained in the "Xconfigurator" build directory.. and since that directory sorts first, the kudzu package is considered to have "no code".

The columns are SLOC (total physical source lines of code), Directory (the name of the build directory, usually the same or similar to the package name), and SLOC-by-Language (Sorted). This last column lists languages by name and the number of SLOC in that language; zeros are not shown, and the list is sorted from largest to smallest in that build directory. Similarly, the directories are sorted from largest to smallest total SLOC.

SLOC	Directory	SLOC-by-Language (Sorted)
------	-----------	---------------------------

1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414, yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358, yacc=2710,perl=711,awk=393,lex=383,sed=57,csch=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988, lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139, yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253, csch=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606, cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910, perl=464,sed=16,csch=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762, awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399, lex=1642,perl=1206,python=959,cpp=746,asm=70,csch=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528, awk=393,python=348,lex=190,csch=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464, perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342, sed=2
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546, yacc=1507,awk=522,lex=323,sed=297,asm=139,csch=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360, csch=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csch=235, sed=35,lisp=12
133193	kaffe-1.0.5	java=65275,ansic=62125,cpp=3923,perl=972,sh=814, asm=84
131372	jade-1.2.1	cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177
127536	pine4.21	ansic=126678,sh=766,csch=62,perl=30
121878	ImageMagick-4.2.9	ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
119613	lynx2-8-3	ansic=117385,sh=1860,perl=340,csch=28
116951	mc-4.5.42	ansic=114406,sh=1996,perl=345,awk=148,csch=56
116615	gnumeric-0.48	ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
113272	xlispstat-3-52-17	ansic=91484,lisp=21769,sh=18,csch=1
113241	vim-5.6	ansic=111724,awk=683,sh=469,perl=359,csch=6
109824	php-3.0.15	ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
104032	linuxconf-1.17r2	cpp=93139,perl=4570,sh=2984,java=2741,ansic=598
102674	libgr-2.0.13	ansic=99647,sh=2438,csch=589
100951	lam-6.3.1	ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187, csch=19
99066	krb4-1.0	ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765, yacc=1509,lex=236,awk=33
94637	xlockmore-4.15	ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
93940	kdenetwork	cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
92964	samba-2.0.6	ansic=88308,sh=3557,perl=831,awk=158,csch=110
91213	anaconda-6.2.2	ansic=74303,python=13657,sh=1583,yacc=810,lex=732, perl=128

89959	xscreensaver-3.23	ansic=88488,perl=1070,sh=401
88128	cvs-1.10.7	ansic=68303,sh=17909,perl=902,yacc=826,csch=181,lisp=7
87940	isd4k-utils	ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
85383	xpdf-0.90	cpp=60427,ansic=21400,sh=3556
81719	inn-2.2.2	ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547, lex=249,tcl=3
80343	kdelibs	cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116, sh=35
79997	WindowMaker-0.61.1	ansic=77924,sh=1483,perl=371,lisp=219
78787	extace-1.2.15	ansic=66571,sh=9322,perl=2894
77873	apache_1.3.12	ansic=69191,sh=6781,perl=1846,cpp=55
75257	xpilot-4.1.0	ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68
73817	w3c-libwww-5.2.8	ansic=64754,sh=4678,cpp=3181,perl=1204
72726	ucd-snmp-4.1.1	ansic=64411,perl=5558,sh=2757
72425	gnome-core-1.0.55	ansic=72230,perl=141,sh=54
71810	jikes	cpp=71452,java=358
70260	groff-1.15	cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397, sh=265,sed=46
69265	fvwm-2.2.4	ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
69246	linux-86	ansic=63328,asm=5276,sh=642
68997	blt2.4g	ansic=58630,tcl=10215,sh=152
68884	squid-2.3.STABLE1	ansic=66305,sh=1570,perl=1009
68560	bash-2.03	ansic=56758,sh=7264,yacc=2808,perl=1730
68453	kdegraphics	cpp=34208,ansic=29347,sh=4898
65722	xntp3-5.93	ansic=60190,perl=3633,sh=1445,awk=417,asm=37
62922	ppp-2.3.11	ansic=61756,sh=996,exp=82,perl=44,csch=44
62137	sgml-tools-1.0.9	cpp=38543,ansic=19185,perl=2866,lex=560,sh=532, lisp=309,awk=142
61688	imap-4.7	ansic=61628,sh=60
61324	ncurses-5.0	ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103, sed=100
60429	kdesupport	ansic=42421,cpp=17810,sh=173,awk=13,csch=12
60302	openldap-1.2.9	ansic=58078,sh=1393,perl=630,python=201
57217	xfig.3.2.3-beta-1	ansic=57212,csch=5
56093	lsof_4.47	ansic=50268,sh=4753,perl=856,awk=214,asm=2
55667	uucp-1.06.1	ansic=52078,sh=3400,perl=189
54935	gnupg-1.0.1	ansic=48884,asm=4586,sh=1465
54603	glade-0.5.5	ansic=49545,sh=5058
54431	svglib-1.4.1	ansic=53725,asm=630,perl=54,sh=22
53141	AfterStep-1.8.0	ansic=50898,perl=1168,sh=842,cpp=233
52808	kdeutils	cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
52574	nmh-1.0.3	ansic=50698,sh=1785,awk=74,sed=17
51813	freetype-1.3.1	ansic=48929,sh=2467,cpp=351,csch=53,perl=13
51592	enlightenment-0.15.5	ansic=51569,sh=23
50970	cdrecord-1.8	ansic=48595,sh=2177,perl=194,sed=4
49370	tin-1.4.2	ansic=47763,sh=908,yacc=699
49325	imlib-1.9.7	ansic=49260,sh=65
48223	kdemultimedia	ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
47067	bash-1.14.7	ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
46312	tcsh-6.09.00	ansic=43544,sh=921,lisp=669,perl=593,csch=585
46159	unzip-5.40	ansic=40977,cpp=3778,asm=1271,sh=133
45811	mutt-1.0.1	ansic=45574,sh=237
45589	am-utils-6.0.3	ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
45485	guile-1.3	ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,csch=50
45378	gnuplot-3.7.1	ansic=43276,lisp=661,asm=539,objc=387,csch=297,perl=138, sh=80
44323	mgetty-1.1.21	ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
42880	sendmail-8.9.3	ansic=40364,perl=1737,sh=779
42746	elm2.5.3	ansic=32931,sh=9774,awk=41
41388	p2c-1.22	ansic=38788,pascal=2499,perl=101

```

41205  gnome-games-1.0.51  ansic=31191,lisp=6966,cpp=3048
39861  rpm-3.0.4            ansic=36994,sh=1505,perl=1355,python=7
39160  util-linux-2.10f     ansic=38627,sh=351,perl=65,csch=62,sed=55
38927  xmmms-1.0.1          ansic=38366,asm=398,sh=163
38548  ORBit-0.5.0          ansic=35656,yacc=1750,sh=776,lex=366
38453  zsh-3.0.7            ansic=36208,sh=1763,perl=331,awk=145,sed=6
37515  ircii-4.4            ansic=36647,sh=852,lex=16
37360  tiff-v3.5.4          ansic=32734,sh=4054,cpp=572
36338  textutils-2.0a       ansic=18949,sh=16111,perl=1218,sed=60
36243  exmh-2.1.1           tcl=35844,perl=316,sh=49,exp=34
36239  xllamp-0.9-alpha3    ansic=31686,sh=4200,asm=353
35812  xloadimage.4.1       ansic=35705,sh=107
35554  zip-2.3              ansic=32108,asm=3446
35397  gtk-engines-0.10     ansic=20636,sh=14761
35136  php-2.0.1            ansic=33991,sh=1056,awk=89
34882  pmake               ansic=34599,sh=184,awk=58,sed=41
34772  xpuzzles-5.4.1       ansic=34772
34768  fileutils-4.0p       ansic=31324,sh=2042,yacc=841,perl=561
33203  strace-4.2           ansic=30891,sh=1988,perl=280,lisp=44
32767  trn-3.6              ansic=25264,sh=6843,yacc=660
32277  pilot-link.0.9.3     ansic=26513,java=2162,cpp=1689,perl=971,yacc=660,
                        python=268,tcl=14
31994  korganizer           cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
31174  ncftp-3.0beta21     ansic=30347,cpp=595,sh=232
30438  gnome-pim-1.0.55     ansic=28665,yacc=1773
30122  scheme-3.2           lisp=19483,ansic=10515,sh=124
30061  tcpdump-3.4          ansic=29208,yacc=236,sh=211,lex=206,awk=184,csch=16
29730  screen-3.9.5         ansic=28156,sh=1574
29315  jed                  ansic=29315
29091  xchat-1.4.0          ansic=28894,perl=121,python=53,sh=23
28897  ncpfs-2.2.0.17       ansic=28689,sh=182,tcl=26
28449  slrn-0.9.6.2         ansic=28438,sh=11
28261  xfish-tank-2.1tp     ansic=28261
28186  texinfo-4.0          ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
28169  e2fsprogs-1.18       ansic=27250,awk=437,sh=339,sed=121,perl=22
28118  slang                ansic=28118
27860  kdegames             cpp=27507,ansic=340,sh=13
27117  librep-0.10          ansic=19381,lisp=5385,sh=2351
27040  mikmod-3.1.6         ansic=26975,sh=55,awk=10
27022  x3270-3.1.1          ansic=26456,sh=478,exp=88
26673  lout-3.17            ansic=26673
26608  Xaw3d-1.3            ansic=26235,yacc=247,lex=126
26363  gawk-3.0.4           ansic=19871,awk=2519,yacc=2046,sh=1927
26146  libxml-1.8.6         ansic=26069,sh=77
25994  xrn-9.02             ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31,
                        csch=13
25915  gv-3.5.8             ansic=25821,sh=94
25479  xpaint              ansic=25456,sh=23
25236  shadow-19990827      ansic=23464,sh=883,yacc=856,perl=33
24910  kdeadadmin          cpp=19919,sh=3936,perl=1055
24773  pdksh-5.2.14         ansic=23599,perl=945,sh=189,sed=40
24583  gmp-2.0.2           ansic=17888,asm=5252,sh=1443
24387  mars_nwe            ansic=24158,sh=229
24270  gnome-python-1.0.51  python=14331,ansic=9791,sh=148
23838  kterm-6.2.0          ansic=23838
23666  enscript-1.6.1       ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
22373  sawmill-0.24         ansic=11038,lisp=8172,sh=3163
22279  make-3.78.1          ansic=19287,sh=2029,perl=963
22011  libpng-1.0.5         ansic=22011
21593  xboard-4.0.5         ansic=20640,lex=904,sh=41,csch=5,sed=3

```

```

21010 netkit-telnet-0.16 ansic=14796,cpp=6214
20433 pam-0.72 ansic=18936,yacc=634,sh=482,perl=321,lex=60
20125 ical-2.2 cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
20078 gdl.3 ansic=19946,perl=132
19971 wu-ftpd-2.6.0 ansic=17572,yacc=1774,sh=421,perl=204
19500 gnome-utils-1.0.50 ansic=18099,yacc=824,lisp=577
19065 joe ansic=18841,asm=224
18885 X11R6-contrib-3.3.2 ansic=18616,lex=161,yacc=97,sh=11
18835 glib-1.2.6 ansic=18702,sh=133
18151 git-4.3.19 ansic=16166,sh=1985
18020 xboing ansic=18006,sh=14
17939 sh-utils-2.0 ansic=13366,sh=3027,yacc=871,perl=675
17765 mtools-3.9.6 ansic=16155,sh=1602,sed=8
17750 gettext-0.10.35 ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9
17682 bc-1.05 ansic=9186,sh=7236,yacc=967,lex=293
17271 fetchmail-5.3.1 ansic=13441,python=1490,sh=1246,yacc=411,perl=321,
lex=238,awk=124
17259 sox-12.16 ansic=16659,sh=600
16785 control-center-1.0.51 ansic=16659,sh=126
16266 dhcp-2.0 ansic=15328,sh=938
15967 SVGATextMode-1.9-src ansic=15079,yacc=340,sh=294,lex=227,sed=15,
asm=12
15868 kpilot-3.1b9 cpp=8613,ansic=5640,yacc=1615
15851 taper-6.9a ansic=15851
15819 mpg123-0.59r ansic=14900,asm=919
15691 transfig.3.2.1 ansic=15643,sh=38,cs=10
15638 mod_perl-1.21 perl=10278,ansic=5124,sh=236
15522 console-tools-0.3.3 ansic=13335,yacc=986,sh=800,lex=291,perl=110
15456 rpm2html-1.2 ansic=15334,perl=122
15143 gnotepad+-1.1.4 ansic=15143
15108 GXedit1.23 ansic=15019,sh=89
15087 mm2.7 ansic=8044,cs=6924,sh=119
14941 readline-2.2.1 ansic=11375,sh=1890,perl=1676
14912 ispell-3.1 ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385,
cs=221,sh=157,perl=85,sed=15
14871 gnuchess-4.0.pl80 ansic=14584,sh=258,cs=29
14774 flex-2.5.4 ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12
14587 multimedia ansic=14577,sh=10
14516 libgtop-1.0.6 ansic=13768,perl=653,sh=64,asm=31
14427 mawk-1.2.2 ansic=12714,yacc=994,awk=629,sh=90
14363 automake-1.4 perl=10622,sh=3337,ansic=404
14350 rsync-2.4.1 ansic=13986,perl=179,sh=126,awk=59
14299 nfs-utils-0.1.6 ansic=14107,sh=165,perl=27
14269 rcs-5.7 ansic=12209,sh=2060
14255 tar-1.13.17 ansic=13014,lisp=592,sh=538,perl=111
14105 wmakerconf-2.1 ansic=13620,perl=348,sh=137
14039 less-346 ansic=14032,awk=7
13779 rxvt-2.6.1 ansic=13779
13586 wget-1.5.3 ansic=13509,perl=54,sh=23
13504 rp3-1.0.7 cpp=10416,ansic=2957,sh=131
13241 iproute2 ansic=12139,sh=1002,perl=100
13100 silo-0.9.8 ansic=10485,asm=2615
12657 macutils ansic=12657
12639 libungif-4.1.0 ansic=12381,sh=204,perl=54
12633 minicom-1.83.0 ansic=12503,sh=130
12593 audiofile-0.1.9 sh=6440,ansic=6153
12463 gnome-objc-1.0.2 objc=12365,sh=86,ansic=12
12313 jpeg-6a ansic=12313
12124 ypserv-1.3.9 ansic=11622,sh=460,perl=42
11790 lrzsz-0.12.20 ansic=9512,sh=1263,exp=1015

```

11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
11633	net-tools-1.54	ansic=11531,sh=102
11404	findutils-4.1	ansic=11160,sh=173,exp=71
11299	xmorphism-1999dec12	ansic=10783,tcl=516
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
10914	diffutils-2.7	ansic=10914
10404	gnorpm-0.9	ansic=10404
10271	gqview-0.7.0	ansic=10271
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
10187	dump-0.4b15	ansic=9422,sh=760,sed=5
10088	piranha	ansic=10048,sh=40
10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
9961	procps-2.0.6	ansic=9959,sh=2
9942	xpat2-1.04	ansic=9942
9927	procmail-3.14	ansic=8090,sh=1837
9873	nss_ldap-105	ansic=9784,perl=89
9801	man-1.5hl	ansic=7377,sh=1802,perl=317,awk=305
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6
9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
9699	bison-1.28	ansic=9650,sh=49
9666	ash-linux-0.2	ansic=9445,sh=221
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
9551	pwdb-0.61	ansic=9488,sh=63
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
9263	ctags-3.4	ansic=9240,sh=23
9138	gftp-2.0.6a	ansic=9138
8939	mkisofs-1.12b5	ansic=8939
8766	pxe-linux	cpp=4463,ansic=3622,asm=681
8572	psgml-1.2.1	lisp=8572
8540	xxgdb-1.12	ansic=8540
8491	gtop-1.0.5	ansic=8151,cpp=340
8356	gedit-0.6.1	ansic=8225,sh=131
8303	dip-3.3.7o	ansic=8207,sh=96
7859	libglade-0.11	ansic=5898,sh=1809,python=152
7826	xpm-3.4k	ansic=7750,sh=39,cpp=37
7740	sed-3.02	ansic=7301,sed=359,sh=80
7617	cpio-2.4.2	ansic=7598,sh=19
7615	esound-0.2.17	ansic=7387,sh=142,cs=86
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
7427	ed-0.2	ansic=7263,sh=164
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,cpp=3
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
7095	xgammon-0.98	ansic=6506,lex=589
7041	newt-0.50.8	ansic=6526,python=515
7030	ee-0.3.11	ansic=7007,sh=23
6976	aboot-0.5	ansic=6680,asm=296
6968	mailx-8.1.1	ansic=6963,sh=5
6877	lpr	ansic=6842,sh=35
6827	gnome-media-1.0.51	ansic=6827
6646	iputils	ansic=6646
6611	patch-2.5	ansic=6561,sed=50
6592	xosview-1.7.1	cpp=6205,ansic=367,awk=20
6550	byacc-1.9	ansic=5520,yacc=1030
6496	pidntd-3.0.10	ansic=6475,sh=21
6391	m4-1.4	ansic=5993,lisp=243,sh=155
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
6234	awesfx-0.4.3a	ansic=6234
6172	sash-3.4	ansic=6172

Estimating Linux's Size

```

6116      lslk                ansic=5325,sh=791
6090      joystick-1.2.15   ansic=6086,sh=4
6072      kdoc              perl=6010,sh=45,cpp=17
6043      irda-utils-0.9.10 ansic=5697,sh=263,perl=83
6033      sysvinit-2.78     ansic=5256,sh=777
6025      pnm2ppa          ansic=5708,sh=317
6021      rpmsfind-1.4      ansic=6021
5981      indent-2.2.5     ansic=5958,sh=23
5975      ytalk-3.1        ansic=5975
5960      isapnptools-1.21 ansic=4394,yacc=1383,perl=123,sh=60
5744      gdm-2.0beta2     ansic=5632,sh=112
5594      isdn-config      cpp=3058,sh=2228,perl=308
5526      efax-0.9         ansic=4570,sh=956
5383      acct-6.3.2       ansic=5016,cpp=287,sh=80
5115      libtool-1.3.4    sh=3374,ansic=1741
5111      netkit-ftp-0.16  ansic=5111
4996      bzip2-0.9.5d     ansic=4996
4895      xcpustate-2.5    ansic=4895
4792      libelf-0.6.4     ansic=3310,sh=1482
4780      make-3.78.1_pvm-0.5 ansic=4780
4542      gpgp-0.4         ansic=4441,sh=101
4430      gperf-2.7        cpp=2947,exp=745,ansic=695,sh=43
4367      aumix-1.30.1     ansic=4095,sh=179,sed=93
4087      zlib-1.1.3       ansic=2815,asm=712,cpp=560
4038      sysklogd-1.3-31  ansic=3741,perl=158,sh=139
4024      rep-gtk-0.8      ansic=2905,lisp=971,sh=148
3962      netkit-timed-0.16 ansic=3962
3929      initscripts-5.00 sh=2035,ansic=1866,csch=28
3896      ltrace-0.3.10    ansic=2986,sh=854,awk=56
3885      phhttpd-0.1.0    ansic=3859,sh=26
3860      xdaliclock-2.18  ansic=3837,sh=23
3855      pciutils-2.1.5   ansic=3800,sh=55
3804      quota-2.00-pre3  ansic=3795,sh=9
3675      dosfstools-2.2   ansic=3675
3654      tcp_wrappers_7.6 ansic=3654
3651      ipchains-1.3.9   ansic=2767,sh=884
3625      autofs-3.1.4     ansic=2862,sh=763
3588      netkit-rsh-0.16  ansic=3588
3438      yp-tools-2.4     ansic=3415,sh=23
3433      dialog-0.6       ansic=2834,perl=349,sh=250
3415      ext2ed-0.1       ansic=3415
3315      gdbm-1.8.0       ansic=3290,cpp=25
3245      ypbind-3.3       ansic=1793,sh=1452
3219      playmidi-2.4     ansic=3217,sed=2
3096      xtrojka123       ansic=3087,sh=9
3084      at-3.1.7         ansic=1442,sh=1196,yacc=362,lex=84
3051      dhcpcd-1.3.18-pl3 ansic=2771,sh=280
3012      apmd             ansic=2617,sh=395
2883      netkit-base-0.16 ansic=2883
2879      vixie-cron-3.0.1 ansic=2866,sh=13
2835      gkermmit-1.0     ansic=2835
2810      kdetoys          cpp=2618,ansic=192
2791      xjewel-1.6       ansic=2791
2773      mpage-2.4        ansic=2704,sh=69
2758      autoconf-2.13    sh=2226,perl=283,exp=167,ansic=82
2705      autorun-2.61     sh=1985,cpp=720
2661      cdp-0.33         ansic=2661
2647      file-3.28        ansic=2601,perl=46
2645      libghttp-1.0.4   ansic=2645
2631      getty_ps-2.0.7j  ansic=2631

```

Estimating Linux's Size

```

2597 pythonlib-1.23 python=2597
2580 magicdev-0.2.7 ansic=2580
2531 gnome-kerberos-0.2 ansic=2531
2490 sndconfig-0.43 ansic=2490
2486 bug-buddy-0.7 ansic=2486
2459 usermode-1.20 ansic=2459
2455 fnlib-0.4 ansic=2432,sh=23
2447 sliplogin-2.1.1 ansic=2256,sh=143,perl=48
2424 raidtools-0.90 ansic=2418,sh=6
2423 netkit-routed-0.16 ansic=2423
2407 nc ansic=1670,sh=737
2324 up2date-1.13 python=2324
2270 memprof-0.3.0 ansic=2270
2268 which-2.9 ansic=1398,sh=870
2200 printtool tcl=2200
2163 gnome-linuxconf-0.25 ansic=2163
2141 unarj-2.43 ansic=2141
2065 units-1.55 ansic=1963,perl=102
2048 netkit-ntalk-0.16 ansic=2048
1987 cracklib,2.7 ansic=1919,perl=46,sh=22
1984 cleanfeed-0.95.7b perl=1984
1977 wmconfig-0.9.8 ansic=1941,sh=36
1941 isicom ansic=1898,sh=43
1883 slocate-2.1 ansic=1802,sh=81
1857 netkit-rusers-0.16 ansic=1857
1856 pump-0.7.8 ansic=1856
1842 cdecl-2.5 ansic=1002,yacc=765,lex=75
1765 fbset-2.1 ansic=1401,yacc=130,lex=121,perl=113
1653 adjtimex-1.9 ansic=1653
1634 netcfg-2.25 python=1632,sh=2
1630 psmisc ansic=1624,sh=6
1621 urlview-0.7 ansic=1515,sh=106
1604 fortune-mod-9708 ansic=1604
1531 netkit-tftp-0.16 ansic=1531
1525 logrotate-3.3.2 ansic=1524,sh=1
1473 traceroute-1.4a5 ansic=1436,awk=37
1452 time-1.7 ansic=1395,sh=57
1435 ncompress-4.2.4 ansic=1435
1361 mt-st-0.5b ansic=1361
1290 cxhextris ansic=1290
1280 pam_krb5-1 ansic=1280
1272 bsd-finger-0.16 ansic=1272
1229 hdparm-3.6 ansic=1229
1226 procinfo-17 ansic=1145,perl=81
1194 passwd-0.64.1 ansic=1194
1182 auth_ldap-1.4.0 ansic=1182
1146 prtconf-1.3 ansic=1146
1143 anacron-2.1 ansic=1143
1129 xbill-2.0 cpp=1129
1099 popt-1.4 ansic=1039,sh=60
1088 nag perl=1088
1076 stylesheets-0.13rh perl=888,sh=188
1075 authconfig-3.0.3 ansic=1075
1049 kpppload-1.04 cpp=1044,sh=5
1020 MAKEDEV-2.5.2 sh=1020
1013 trojka ansic=1013
987 xmailbox-2.5 ansic=987
967 netkit-rwho-0.16 ansic=967
953 switchdesk-2.1 ansic=314,perl=287,cpp=233,sh=119
897 portmap_4 ansic=897

```

```

874      ldconfig-1999-02-21 ansic=874
844      jpeg-6b           sh=844
834      ElectricFence-2.1 ansic=834
830      mouseconfig-4.4   ansic=830
816      rpmlint-0.8       python=813,sh=3
809      kdpms-0.2.8      cpp=809
797      termcap-2.0.8     ansic=797
787      xsysinfo-1.7      ansic=787
770      giftrans-1.12.2   ansic=770
742      setserial-2.15    ansic=742
728      tree-1.2         ansic=728
717      chkconfig-1.1.2   ansic=717
682      lpg              perl=682
657      eject-2.0.2      ansic=657
616      diffstat-1.27     ansic=616
592      netscape-4.72     sh=592
585      usernet-1.0.9     ansic=585
549      genromfs-0.3      ansic=549
548      tksysv-1.1       tcl=526,sh=22
537      minlabel-1.2      ansic=537
506      netkit-bootparamd-0.16 ansic=506
497      locale_config-0.2 ansic=497
491      helptool-2.4      perl=288,tcl=203
480      elftoaout-2.2     ansic=480
463      tmpwatch-2.2      ansic=311,sh=152
445      rhs-printfilters-1.63 sh=443,ansic=2
441      audioctl         ansic=441
404      control-panel-3.13 ansic=319,tcl=85
368      kbdconfig-1.9.2.4 ansic=368
368      vlock-1.3        ansic=368
367      timetool-2.7.3    tcl=367
347      kernelcfg-0.5     python=341,sh=6
346      timeconfig-3.0.3  ansic=318,sh=28
343      mingetty-0.9.4    ansic=343
343      chkfontpath-1.7   ansic=343
332      ethtool-1.0       ansic=332
314      mkbootdisk-1.2.5  sh=314
302      symlinks-1.2      ansic=302
301      xsri-1.0         ansic=301
294      netkit-rwall-0.16 ansic=294
290      biff+comsat-0.16  ansic=290
288      mkinitrd-2.4.1    sh=288
280      stat-1.5         ansic=280
265      sysreport-1.0     sh=265
261      bdflush-1.5       ansic=202,asm=59
255      ipvsadm-1.1       ansic=255
255      sag-0.6-html      perl=255
245      man-pages-1.28    sh=244,sed=1
240      open-1.4         ansic=240
236      xtoolwait-1.2     ansic=236
222      utempter-0.5.2    ansic=222
222      mkkickstart-2.1   sh=222
221      hellas           sh=179,perl=42
213      rhmask           ansic=213
159      quickstrip-1.1    ansic=159
132      rdate-1.0        ansic=132
131      statserial-1.1    ansic=121,sh=10
107      fwhois-1.00      ansic=107
85      mktemp-1.5        ansic=85
82      modemtool-1.21    python=73,sh=9

```

Estimating Linux's Size

```

67      setup-1.2          ansic=67
56      shaper             ansic=56
52      sparc32-1.1        ansic=52
47      intimed-1.10       ansic=47
23      locale-ja-9        sh=23
16      AnotherLevel-1.0.1 sh=16
11      words-2            sh=11
7       trXFree86-2.1.2    tcl=7
0       install-guide-3.2.html (none)
0       caching-nameserver-6.2 (none)
0       XFree86-ISO8859-2-1.0 (none)
0       rootfiles          (none)
0       ghostscript-fonts-5.50 (none)
0       kudzu-0.36          (none)
0       wvdial-1.41         (none)
0       mailcap-2.0.6       (none)
0       desktop-backgrounds-1.1 (none)
0       redhat-logos        (none)
0       solemul-1.1         (none)
0       dev-2.7.18          (none)
0       urw-fonts-2.0        (none)
0       users-guide-1.0.72  (none)
0       sgml-common-0.1     (none)
0       setup-2.1.8         (none)
0       jadetex             (none)
0       gnome-audio-1.0.0   (none)
0       specsps-6.2         (none)
0       gimp-data-extras-1.0.0 (none)
0       docbook-3.1         (none)
0       indexhtml-6.2       (none)

```

```

ansic:    14218806 (80.55%)
cpp:      1326212 (7.51%)
lisp:     565861 (3.21%)
sh:       469950 (2.66%)
perl:     245860 (1.39%)
asm:      204634 (1.16%)
tcl:      152510 (0.86%)
python:   140725 (0.80%)
yacc:     97506 (0.55%)
java:     79656 (0.45%)
exp:      79605 (0.45%)
lex:      15334 (0.09%)
awk:      14705 (0.08%)
objc:     13619 (0.08%)
csh:      10803 (0.06%)
ada:      8217 (0.05%)
pascal:   4045 (0.02%)
sed:      2806 (0.02%)
fortran:  1707 (0.01%)

```

```

Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = $ 614421924.71

```

B.2 Counts of Files For Each Category

There were 181,679 ordinary files in the build directory. The following are counts of the number of files (*not* the SLOC) for each language:

ansic:	52088	(71.92%)
cpp:	8092	(11.17%)
sh:	3381	(4.67%)
asm:	1931	(2.67%)
perl:	1387	(1.92%)
lisp:	1168	(1.61%)
java:	1047	(1.45%)
python:	997	(1.38%)
tcl:	798	(1.10%)
exp:	472	(0.65%)
awk:	285	(0.39%)
objc:	260	(0.36%)
sed:	112	(0.15%)
yacc:	110	(0.15%)
csh:	94	(0.13%)
ada:	92	(0.13%)
lex:	57	(0.08%)
fortran:	50	(0.07%)
pascal:	7	(0.01%)

Total Number of Source Code Files = 72428

In addition, when counting the number of files (not SLOC), some files were identified as source code files but nevertheless were not counted for other reasons (and thus not included in the file counts above). Of these source code files, 5,820 files were identified as duplicating the contents of another file, 817 files were identified as files that had been automatically generated, and 65 files were identified as zero-length files.

B.3 Additional Measures of the Linux Kernel

I also made additional measures of the Linux kernel. This kernel is Linux kernel version 2.2.14 as patched by Red Hat. The Linux kernel's design is reflected in its directory structure. Only 8 lines of source code are in its main directory; the rest are in descendent directories. Counting the physical SLOC in each subdirectory (or its descendents) yielded the following:

BUILD/linux/Documentation/	765
BUILD/linux/arch/	236651
BUILD/linux/configs/	0
BUILD/linux/drivers/	876436
BUILD/linux/fs/	88667
BUILD/linux/ibcs/	16619
BUILD/linux/include/	136982
BUILD/linux/init/	1302
BUILD/linux/ipc/	1757
BUILD/linux/kernel/	7436
BUILD/linux/ksymoops-0.7c/	3271
BUILD/linux/lib/	1300
BUILD/linux/mm/	6771
BUILD/linux/net/	105549
BUILD/linux/pcmcia-cs-3.1.8/	34851
BUILD/linux/scripts/	8357

I separately ran the CodeCount tools on the entire linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C.

This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for Linux.

However, this included non-i86 code. To make a more reasonable comparison with the Halloween documents, I needed to ignore non-i386 code.

First, I looked at the linux/arch directory, which contained architecture-specific code. This directory had the following subdirectories (architectures): alpha, arm, i386, m68k, mips, ppc, s390, sparc, sparc64. I then computed the total for all of ``arch'', which was 236651 SLOC, and subtracted out linux/arch/i386 code, which totalled to 26178 SLOC; this gave me a total of non-i386 code in linux/arch as 210473 physical SLOC. I then looked through the ``drivers'' directory to see if there were sets of drivers which were non-i386. I identified the following directories, with the SLOC totals as shown:

linux/drivers/sbus/	22354
linux/drivers/macintosh/	6000
linux/drivers/sgi/	4402
linux/drivers/fc4/	3167
linux/drivers/nubus/	421
linux/drivers/acorn/	11850
linux/drivers/s390/	8653

Driver Total:	56847
---------------	-------

Thus, I had a grand total on non-i86 code (including drivers and architecture-specific code) as 267320 physical SLOC. This is, of course, another approximation, since there's certainly other architecture-specific lines, but I believe that is most of it. Running the CodeCount tool on just the C code, once these architectural and driver directories are removed, reveals a logical SLOC of 570,039 of C code.

B.4 Minimum System SLOC

Most of this paper worries about counting an ``entire'' system. However, what's the SLOC size of a ``minimal'' system? Here's an attempt to answer that question.

Red Hat Linux 6.2, CD-ROM #1, file RedHat/base/comps, defines the ``base'' (minimum) Red Hat Linux 6.2 installation as a set of packages. The following are the build directories corresponding to this base (minimum) installation, along with the SLOC counts (as shown above). Note that this creates a text-only system:

Component	SLOC
anacron-2.1	1143
apmd	3012
ash-linux-0.2	9666
at-3.1.7	3084
authconfig-3.0.3	1075
bash-1.14.7	47067
bc-1.05	17682
bdflush-1.5	261
binutils-2.9.5.0.22	467120
bzip2-0.9.5d	4996
chkconfig-1.1.2	717
console-tools-0.3.3	15522
cpio-2.4.2	7617
cracklib, 2.7	1987
dev-2.7.18	0
diffutils-2.7	10914
dump-0.4b15	10187
e2fsprogs-1.18	28169
ed-0.2	7427
egcs-1.1.2	720112
eject-2.0.2	657
file-3.28	2647
fileutils-4.0p	34768
findutils-4.1	11404

Estimating Linux's Size

gawk-3.0.4	26363
gd1.3	20078
gdbm-1.8.0	3315
getty_ps-2.0.7j	2631
glibc-2.1.3	415026
gmp-2.0.2	24583
gnupg-1.0.1	54935
gpm-1.18.1	9725
grep-2.4	10013
groff-1.15	70260
gzip-1.2.4a	6306
hdparm-3.6	1229
initscripts-5.00	3929
isapnptools-1.21	5960
kbdconfig-1.9.2.4	368
kernelcfg-0.5	347
kudzu-0.36	0
ldconfig-1999-02-21	874
ld.so-1.9.5	9731
less-346	14039
lilo	7255
linuxconf-1.17r2	104032
logrotate-3.3.2	1525
mailcap-2.0.6	0
mailx-8.1.1	6968
MAKEDEV-2.5.2	1020
man-1.5hl	9801
mingetty-0.9.4	343
mkbootdisk-1.2.5	314
mkinitrd-2.4.1	288
mktemp-1.5	85
modutils-2.3.9	11775
mouseconfig-4.4	830
mt-st-0.5b	1361
ncompress-4.2.4	1435
ncurses-5.0	61324
net-tools-1.54	11633
newt-0.50.8	7041
pam-0.72	20433
passwd-0.64.1	1194
pciutils-2.1.5	3855
popt-1.4	1099
procmail-3.14	9927
procps-2.0.6	9961
psmisc	1630
pump-0.7.8	1856
pwdb-0.61	9551
quota-2.00-pre3	3804
raidtools-0.90	2424
readline-2.2.1	14941
redhat-logos	0
rootfiles	0
rpm-3.0.4	39861
sash-3.4	6172
sed-3.02	7740
sendmail-8.9.3	42880
setserial-2.15	742
setup-1.2	67
setup-2.1.8	0
shadow-19990827	25236

sh-utils-2.0	17939
slang	28118
slocate-2.1	1883
stat-1.5	280
sysklogd-1.3-31	4038
sysvinit-2.78	6033
tar-1.13.17	14255
termcap-2.0.8	797
texinfo-4.0	28186
textutils-2.0a	36338
time-1.7	1452
timeconfig-3.0.3	346
tmpwatch-2.2	463
utempter-0.5.2	222
util-linux-2.10f	39160
vim-5.6	113241
vixie-cron-3.0.1	2879
which-2.9	2268
zlib-1.1.3	4087

Thus, the contents of the build directories corresponding to the ``base" (minimum) installation totals to 2,819,334 SLOC.

A few notes are in order about this build directory total:

1. Some of the packages listed by a traditional package list aren't shown here because they don't contain any code. Package "basesystem" is a pseudo-package for dependency purposes. Package redhat-release is just a package for keeping track of the base system's version number. Package "filesystem" contains a directory layout.
2. ntsysv's source is in chkconfig-1.1.2; kernel-utils and kernel-pcmcia-cs are part of "linux". Package shadow-utils is in build directory shadow-19990827. Build directory util-linux includes losetup and mount. "dump" is included to include rmt.
3. Sometimes the build directories contain more code than is necessary to create just the parts for the ``base" system; this is a side-effect of how things are packaged. ``info" is included in the base, so we count all of texinfo. The build directory termcap is counted, because libtermcap is in the base. Possibly most important, gcc (egcs) is there because libstdc++ is in the base.
4. Sometimes a large component is included in the base, even though most of the time little of its functionality is used. In particular, the mail transfer agent ``sendmail" is in the base, even though for many users most of sendmail's functionality isn't used. However, for this paper's purposes this isn't a problem. After all, even if sendmail's functionality is often underused, clearly that functionality took time to develop and that functionality is available to those who want it.
5. My tools intentionally eliminated duplicates; it may be that a few files aren't counted here because they're considered duplicates of another build directory not included here. I do not expect this factor to materially change the total.
6. Red Hat Linux is not optimized to be a ``small as possible" distribution; their emphasis is on functionality, not small size. A working Linux distribution could include much less code, depending on its intended application. For example, ``linuxconf" simplifies system configuration, but the system can be configured by editing its system configuration files directly, which would reduce the base system's size. This also includes vim, a full-featured text editor - a simpler editor with fewer functions would be smaller as well.

Many people prefer some sort of graphical interface; here is a minimal configuration of a graphical system, adding the X server, a window manager, and a few tools:

Component	SLOC
XFree86-3.3.6	1291745
Xconfigurator-4.3.5	9741
fvwm-2.2.4	69265
X11R6-contrib-3.3.2	18885

These additional graphical components add 1,389,636 SLOC. Due to oddities of the way the initialization system xinitrc is built, it isn't shown here in the total, but xinitrc has so little code that its omission does not significantly affect the total.

Adding these numbers together, we now have a total of 4,208,970 SLOC for a ``minimal graphical system." Many people would want to add more components. For example, this doesn't include a graphical toolkit (necessary for running most graphical applications). We could add gtk+-1.2.6 (a toolkit needed for running GTK+ based applications), adding 138,118 SLOC. This would now total 4,347,088 for a ``basic graphical system," one able to run basic GTK+ applications.

Let's add a web server to the mix. Adding apache_1.3.12 adds only 77,873 SLOC. We now have 4,424,961 physical SLOC for a basic graphical system plus a web server.

We could then add a graphical desktop environment, but there are so many different options and possibilities that trying to identify a ``minimal" system is hard to do without knowing the specific uses intended for the system. Red Hat defines a standard ``GNOME" and ``KDE" desktop, but these are intended to be highly functional (not ``minimal"). Thus, we'll stop here, with a total of 2.8 million physical SLOC for a minimal text-based system, and total of 4.4 million physical SLOC for a basic graphical system plus a web server.

References

- [Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.
- [Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/develpro.html>.
- [DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the ``Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.
- [FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.
- [Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.
- [Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>
- [Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf
- [Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>
- [Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.
- [Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.
- [NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>
- [OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.
- [Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>
- [Raymond 1999] Raymond, Eric S. January 1999. ``A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.
- [Schneier 2000] Schneier, Bruce. March 15, 2000. ``Software Complexity and Security". *Crypto-Gram*.

<http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...".
<http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet.
<http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*.
http://www.dwheeler.com/oss_fs_refs.html.

[Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

This paper is (C) Copyright 2000 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. When referring to the paper, please refer to it as ``Estimating GNU/Linux's Size'' by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

Estimating Linux's Size

David A. Wheeler (dwheeler@dwheeler.com)

November 6, 2000

Version 1.03

This paper presents size estimates (and their implications) of the source code of a distribution of the Linux operating system (OS), a combination often called GNU/Linux. The distribution used in this paper is Red Hat Linux version 6.2, including the kernel, software development tools, graphics interfaces, client applications, and so on. Other distributions and versions will have different sizes.

In total, this distribution includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars).

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), gcc (a compilation system), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. In this distribution the GPL is the dominant license, and copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses in terms of SLOC. More information is available at <http://www.dwheeler.com/sloc>.

1. Introduction

The Linux operating system (also called GNU/Linux) has gone from an unknown to a powerful market force. One survey found that more Internet servers use Linux than any other operating system [Zoebelein 1999]. IDC found that 25% of all server operating systems purchased in 1999 were Linux, making it second only to Windows NT's 38% [Shankland 2000a].

There appear to be many reasons for this, and not simply because Linux can be obtained at no or low cost. For example, experiments suggest that Linux is highly reliable. A 1995 study of a set of individual components found that the GNU and Linux components had a significantly higher reliability than their proprietary Unix competitors (6% to 9% failure rate with GNU and Linux, versus an average 23% failure rate with the proprietary software using their measurement technique) [Miller 1995]. A ten-month experiment in 1999 by ZDnet found that, while Microsoft's Windows NT crashed every six weeks under a "typical" intranet load, using the same load and request set the Linux systems (from two different distributors) *never* crashed [Vaughan-Nichols 1999].

However, possibly the most important reason for Linux's popularity among many developers and users is that its source code is generally "open source software" and/or "free software" (where the "free" here means "freedom"). A program that is "open source software" or "free software" is essentially a program whose source code can be obtained, viewed, changed, and redistributed without royalties or other limitations of these actions. A more formal definition of "open source software" is available at OSI [1999], a more formal definition of "free software" is available at FSF [2000], and other general information about these topics is available at Wheeler [2000a]. Quantitative rationales for using open source / free software is given in Wheeler [2000b]. The Linux operating system is actually a suite of components, including the Linux kernel on which it is based, and it is packaged, sold, and supported by a variety of distributors. The Linux kernel is "open source software"/"free software", and this is also true for all (or nearly all) other components of a typical Linux distribution. Open source software/free software frees users from being captives of a particular vendor, since it permits users to fix any problems immediately, tailor their system, and analyze their software in arbitrary ways.

Surprisingly, although anyone can analyze Linux for arbitrary properties, I have found little published analysis of the amount of source lines of code (SLOC) contained in a Linux distribution. The only published data I've found was developed by Microsoft in the documents usually called "Halloween I" and "Halloween II". Unfortunately, the meaning, derivation, and assumptions of their numbers is not explained, making the numbers hard to use and truly understand. Even worse, although the two documents were written by essentially the same people at the same time, the numbers in the documents appear (on their surface) to be contradictory. The so-called "Halloween I" document claimed that the Linux kernel (x86 only) was 500,000 lines of code, the Apache web server was 80,000 lines of code, the X-windows server was 1.5 million, and a full Linux distribution was about 10 million lines of code [Halloween I]. The "Halloween II" document seemed to contradict this, saying that "Linux" by 1998 included 1.5 million lines of code. Since "version 2.1.110" is identified as the version number,

presumably this only measures the Linux kernel, and it does note that this measure includes all Linux ports to various architectures [\[Halloween II\]](#). However, this asks as many questions as it answers - what exactly was being measured, and what assumptions were made? You could infer from these documents that the Linux kernel's support for other architectures took one million lines of code - but this appeared unlikely. Another study, [\[Dempsey 1999\]](#), did analyze open source programs, but it primarily focused on statistics about developers and used basic filesize numbers to report about the software.

This paper bridges this gap. In particular, it shows estimates of the size of Linux, and it estimates how much it would cost to rebuild a typical Linux distribution using traditional software development techniques. Various definitions and assumptions are included, so that others can understand exactly what these numbers mean.

For my purposes, I have selected as my "representative" Linux distribution Red Hat Linux version 6.2. I believe this distribution is reasonably representative for several reasons:

1. Red Hat Linux is the most popular Linux distribution sold in 1999 according to IDC [\[Shankland 2000b\]](#). Red Hat sold 48% of all copies in 1999; the next largest distribution in market share sales was SuSE at 15%. Not all Linux copies are "sold" in a way that this study would count, but the study at least shows that Red Hat's distribution is a popular one.
2. Many distributions (such as Mandrake) are based on older versions of Red Hat Linux.
3. All major general-purpose distributions support (at least) the kind of functionality supported by Red Hat Linux, if for no other reason than to compete with Red Hat.
4. All distributors start with the same set of open source software projects from which to choose components to integrate. Therefore, other distributions are likely to choose the same components or similar kinds of components with often similar size.

Different distributions and versions would produce different size figures, but I hope that this paper will be enlightening even though it doesn't try to evaluate "all" distributions. Note that some distributions (such as SuSE) may decide to add many more applications, but also note this would only create larger (not smaller) sizes and estimated levels of effort. At the time that I began this project, version 6.2 was the latest version of Red Hat Linux available, so I selected that version for analysis.

Section 2 briefly describes the approach used to estimate the "size" of this distribution (most of the details are in Appendix A). Section 3 discusses some of the results (with the details in Appendix B). Section 4 presents conclusions, followed by the two appendices.

2. Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; the steps and assumptions made are described in Appendix A.

A few summary points are worth mentioning here, however, for those who don't read appendix A. I included software for all architectures, not just the i386. I did not include "old" versions of software (with the one exception of bash, as discussed in appendix A). I used md5 checksums to identify and ignore duplicate files, so if the same file contents appeared in more than one file, it was only counted once. The code in makefiles and RPM package specifications was not included. Various heuristics were used to detect automatically generated code, and any such code was also excluded from the count. A number of other heuristics were used to determine if a language was a source program file, and if so, what its language was.

The "physical source lines of code" (physical SLOC) measure was used as the primary measure of SLOC in this paper. Less formally, a physical SLOC in this paper is a line with something other than comments and whitespace (tabs and spaces). More specifically, physical SLOC is defined as follows: "a physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character." Comment delimiters (characters other than newlines starting and ending a comment) were considered comment characters. Data lines only including whitespace (e.g., lines with only tabs and spaces in multiline strings) were not included.

Note that the "logical" SLOC is not the primary measure used here; one example of a logical SLOC measure would be the "count of all terminating semicolons in a C file." The "physical" SLOC was chosen instead of the "logical" SLOC because

there were so many different languages that needed to be measured. I had trouble getting freely-available tools to work on this scale, and the non-free tools were too expensive for my budget (nor is it certain that they would have fared any better). Since I had to develop my own tools, I chose a measure that is much easier to implement. [Park \[1992\]](#) actually recommends the use of the physical SLOC measure (as a minimum), for this and other reasons. There are disadvantages to the "physical" SLOC measure. In particular, physical SLOC measures are sensitive to how the code is formatted. However, logical SLOC measures have problems too. First, as noted, implementing tools to measure logical SLOC is more difficult, requiring more sophisticated analysis of the code. Also, there are many different possible logical SLOC measures, requiring even more careful definition. Finally, a logical SLOC measure must be redefined for every language being measured, making inter-language comparisons more difficult. For more information on measuring software size, including the issues and decisions that must be made, see [Kalb \[1990\]](#), [Kalb \[1996\]](#), and [Park \[1992\]](#).

This decision to use physical SLOC also implied that for an effort estimator I needed to use the original COCOMO cost and effort estimation model (see Boehm [1981]), rather than the newer "COCOMO II" model. This is simply because COCOMO II requires logical SLOC as an input instead of physical SLOC.

For programmer salary averages, I used a salary survey from the September 4, 2000 issue of ComputerWorld; their survey claimed that this annual programmer salary averaged \$56,286 in the United States. I was unable to find a publicly-backed average value for overhead, also called the "wrap rate." This value is necessary to estimate the costs of office space, equipment, overhead staff, and so on. I talked to two cost analysts, who suggested that 2.4 would be a reasonable overhead (wrap) rate. Some Defense Systems Management College (DSMC) training material gives examples of 2.3 (125.95%+100%) not including general and administrative (G&A) overhead, and 2.8 when including G&A (125% engineering overhead, plus 25% on top of that amount for G&A) [\[DSMC\]](#). This at least suggests that 2.4 is a plausible estimate. Clearly, these values vary widely by company and region; the information provided in this paper is enough to use different numbers if desired.

3. Results

Given this approach, here are some of the results. Section 3.1 presents the largest components (by SLOC), section 3.2 presents results specifically from the Linux kernel's SLOC, section 3.3 presents total counts by language, section 3.4 presents total counts of files (instead of SLOC), section 3.5 presents total counts grouped by their software licenses, section 3.6 presents total SLOC counts, and section 3.7 presents effort and cost estimates.

3.1 Largest Components by SLOC

Here are the top 25 largest components (as measured by number of source lines of code):

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,csh=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csh=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csh=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csh=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,csh=147,sed=123

206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	c++=180866,ansic=20513,yacc=2284,sh=538,lex=464, perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342, sed=2
199982	gs5.50	ansic=195491,c++=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,c++=9407,perl=3795,pascal=1546, yacc=1507,awk=522,lex=323,sed=297,asm=139,cs=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,c++=1360, cs=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,c++=741,perl=243
138931	kdebase	c++=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,cs=235, sed=35,lisp=12
133193	kaffe-1.0.5	java=65275,ansic=62125,c++=3923,perl=972,sh=814, asm=84
131372	jade-1.2.1	c++=120611,ansic=8228,sh=2150,perl=378,sed=5
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177

Note that the operating system kernel (linux) is the largest single component, at over 1.5 million lines of code (mostly in C). See section 3.2 for a more discussion discussion of the linux kernel.

The next largest component is the X windows server, a critical part of the graphical user interface (GUI). Given the importance of GUIs, the long history of this program (giving it time to accrete functionality), and the many incompatible video displays it must support, this is perhaps not surprising.

Next is the gcc compilation system, including the C and C++ compilers, which is confusingly named ``egcs" instead. The naming conventions of gcc can be confusing, so a little explanation is in order. Officially, the compilation system is called ``gcc". Egcs was a project to experiment with a more open development model for gcc. Red Hat Linux 6.2 used one of the gcc releases from the egcs project, and called the release egcs-1.1.2 to avoid confusion with the official (at that time) gcc releases. The egcs experiment was a success; egcs as a separate project no longer exists, and current gcc development is based on the egcs code and development model. To sum it up, the compilation system is named ``gcc", and the version of gcc used here is a version developed by ``egcs".

Following this is the symbolic debugger and emacs. Emacs is probably not a real surprise; some users use nothing but emacs (e.g., reading their email via emacs), using emacs as a kind of virtual operating system. This is followed by the set of utilities for binary files, and the C library (which is actually used by most other language libraries as well). This is followed by TCL/Tk (a combined language and widget set), PostgreSQL (a relational DBMS), and the GIMP (an excellent client application for editing bitmapped drawings).

Note that language implementations tend to be written in themselves, particularly for their libraries. Thus there is more Perl than any other single language in the Perl implementation, more Python than any other single language in Python, and more Java than any other single language in Kaffe (an implementation of the Java Virtual Machine and library).

3.2 Examination of the Linux Kernel's SLOC

Since the largest single component was the linux kernel (at over 1.5 million SLOC), I examined it further, to learn why it was so large and determine its ramifications.

I found that over 870,000 lines of this code was in the ``drivers" subdirectory, thus, the primary reason the kernel is so large is that it supports so many different kinds of hardware. The linux kernel's design is expressed in its source code directory structure, and no other directory comes close to this size - the second largest is the ``arch" directory (at over 230,000 SLOC), which contains the architecture-unique code for each CPU architecture. Supporting many different filesystems also increases its size, but not as much as expected; the entire filesystem code is not quite 88,000 SLOC. See the appendix for more detail.

Richard Stallman and others have argued that the resulting system often called ``Linux" should instead be called ``GNU/Linux" [Stallman 2000]. In particular, by hiding GNU's contributions (through not including GNU's name), many people are kept unaware of the GNU project and its purpose, which is to encourage a transition to ``free software" (free as in freedom). Certainly, the resulting system was the intentional goal and result of the GNU project's efforts. Another argument used to justify the term ``GNU/Linux" is that it is confusing if both the entire operating system and the operating system kernel are both called ``Linux". Using the term ``Linux" is particularly bizarre for GNU/Hurd, which takes the Debian

GNU/Linux distribution and swaps out one component: the Linux kernel.

The data here can be used to justify calling the system either ``Linux" or ``GNU/Linux." It's clear that the largest single component in the operating system is the Linux kernel, so it's at least understandable how so many people have chosen to name the entire system after its largest single component (``Linux"). It's also clear that there are many contributors, not just the GNU project itself, and some of those contributors do not agree with the GNU project's philosophy. On the other hand, many of the largest components of the system are essentially GNU projects: gcc (packaged under the name ``egcs"), gdb, emacs, binutils (a set of commands for binary files), and glibc (the C library). Other GNU projects in the system include binutils, bash, gawk, make, textutils, sh-utils, gettext, readline, automake, tar, less, findutils, diffutils, and grep. This is not even counting GNOME, a GNU project. In short, the total of the GNU project's code is *much* larger than the Linux kernel's size. Thus, by comparing the total contributed effort, it's certainly justifiable to call the entire system ``GNU/Linux" and not just ``Linux."

These measurements at least debunk one possible explanation of the Halloween documents' measures. Since Halloween I claimed that the x86-only code for the Linux kernel measured 500,000 SLOC, while Halloween II claimed that the kernel (all architectures) was 1.5 million SLOC, one explanation of this difference would be that the code for non-x86 systems was 1 million SLOC. This isn't so; I computed a grand total of 267,320 physical SLOC of non-i86 code (including drivers and architecture-specific code). It seems unlikely that over 700,000 lines of code would have been removed (not added) in the intervening time.

However, other measures (and explanations) are more promising. I also ran the CodeCount tools on the linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for the Linux kernel. When I removed all non-i86 code and re-ran the CodeCount tool on just the C code, a logical SLOC of 570,039 of C code was revealed. Since the Halloween I document reported 500,000 SLOC (when only including x86 code), it appears very likely that the Halloween I paper counted logical SLOC (and only C code) when reporting measurements of the linux kernel. However, the other Halloween I measures appear to be physical SLOC measures: their estimate of 1.5 million SLOC for the X server is closer to the 1.2 million physical SLOC measured here, and their estimate of 80,000 SLOC for Apache is close to the 77,873 SLOC measured here (as shown in Appendix B). These variations in measurements should be expected, since the versions I am measuring are slightly different than the ones they measured, and it is likely that some assumptions are different as well. Meanwhile, Halloween II reported a measure of 1.5 million lines of code for the linux kernel, essentially the same value given here for physical SLOC.

In short, it appears that Halloween I used the ``logical SLOC" measure when measuring the Linux kernel, while all other measures in Halloween I and II used physical SLOC as the measure. I have attempted to contact the Microsoft author to confirm this, but as of yet I have not received such confirmation. In any case, this example clearly demonstrates the need to carefully identify the units of measure and assumptions made in any measurement of SLOC.

3.3 Total Counts by Language

Here are the various programming languages, sorted by the total number of source lines of code:

ansic:	14218806	(80.55%)
cpp:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)
python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Here you can see that C is pre-eminent (with over 80% of the code), followed by C++, LISP, shell, and Perl. Note that the separation of Expect and TCL is somewhat artificial; if combined, they would be next (at 232115), followed by assembly. Following this in order are Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Some of the languages with smaller counts (such as objective-C and Ada) show up primarily as test cases or bindings to support users of those languages. Nevertheless, it's nice to see at least some support for a variety of languages, since each language has some strength for some type of application.

C++ has over a million lines of code, a very respectable showing, and yet at least in this distribution it is far less than C. One could ask why there's so much more C code, particularly against C++. One possible argument is that well-written C++ takes fewer lines of code than does C; while this is often true, that's unlikely to entirely explain this. Another important factor is that many of the larger programs were written before C++ became widely used, and no one wishes to rewrite their C programs into C++. Also, there are a significant number of software developers who prefer C over C++ (e.g., due to simplicity of understanding the entire language), which would certainly affect these numbers. There have been several efforts in the past to switch from C to C++ in the Linux kernel, and they have all failed (for a variety of reasons).

The fact that LISP places so highly (it's in third place) is a little surprising. LISP is used in many components, but its high placement is due to the widespread use of emacs. Emacs itself is written in primarily in its own variant of LISP, and the emacs package itself accounts for 80% (453647/565861) of the total amount of LISP code. In addition, many languages include sophisticated (and large) emacs modes to support development in those languages: Perl includes 5584 lines of LISP, and Python includes another 2333 of LISP that is directly used to support elaborate Emacs modes for program editing. The ``psgml" package is solely an emacs mode for editing SGML documents. The components with the second and third largest amounts of LISP are xlipstat-3-52-17 and scheme-3.2, which are implementations of LISP and Scheme (a LISP dialect) respectively. Other programs (such as the GIMP and Sawmill) also use LISP or one of its variants as a ``control" language to control components built in other languages (in these cases C). LISP has a long history of use in the hacking (computer enthusiast) community, due to powerful influences such as MIT's old ITS community. For more information on the history of hackerdom, including the influence of ITS and LISP, see [\[Raymond 1999\]](#).

3.4 Total Counts of Files

Of course, instead of counting SLOC, you could count just the number of files in various categories, looking for other insights.

Lex/flex and yacc/bison are widely-used program generators. They make respectable showings when counting SLOC, but their widespread use is more obvious when examining the file counts. There are 57 different lex/flex files, and 110 yacc/bison files. Since some build directories use lex/flex or yacc/bison more than once, the count of build directories using these tools is smaller but still respectable: 38 different build directories use lex/flex, and 62 different build directories use yacc/bison.

Other insights can be gained from the file counts shown in appendix B. The number of source code files counted were 72,428. Not included in this count were 5,820 files which contained duplicate contents, and 817 files which were detected as being automatically generated.

These values can be used to compute average SLOC per file across the entire system. For example, for C, there was 14218806 SLOC contained in 52088 files, resulting in an ``average" C file containing 273 (14218806/52088) physical source lines of code.

3.5 Total Counts by License

A software license determines how that software can be used and reused, and open source software licensing has been a subject of great debate. Well-known open source licenses include the GNU General Public License (GPL), the GNU Library/Lesser General Public License (LGPL), the MIT (X) license, the BSD license, and the Artistic license. The GPL and LGPL are termed ``copylefting" licenses, that is, the license is designed to prevent the code from becoming proprietary. See [Perens \[1999\]](#) for more information. Obvious questions include ``what license(s) are developers choosing when they release their software" and ``how much code has been released under the various licenses?"

An approximation of the amount of software using various licenses can be found for this particular distribution. Red Hat Linux 6.2 uses the Red Hat Package Manager (RPM), and RPM supports capturing license data for each package (these are the ``Copyright" and ``License" fields in the specification file). I used this information to determine how much code was covered by each license. Since this field is simply a string of text, there were some variances in the data that I had to clean up, for example, some entries said ``GNU" while most said ``GPL".

This is an imperfect approach. Some packages contain different pieces of code with different licenses. Some packages are "dual licensed", that is, they are released under more than one license. Sometimes these other licenses are noted, while at other times they aren't. There are actually two BSD licenses (the "old" and "new" licenses), but the specification files don't distinguish between them. Also, if the license wasn't one of a small set of licenses, Red Hat tended to assign nondescriptive phrases such as "distributable". Nevertheless, this approach is sufficient to give some insight into the amount of software using various licenses. Future research could examine each license in turn and categorize them; such research might require lawyers to determine when two licenses in certain circumstances are "equal."

Here are the various license types, sorted by the SLOC in the packages with those licenses:

```

9350709 GPL
2865930 Distributable/Freely Distributable/Freeware
1927711 MIT (X)
1087757 LGPL
1060633 BSD
 383922 BSDish/Xish/MITish
 278327 Miscellaneous (QPL, IBM, unknown)
 273882 GPL/BSD
 206237 Artistic or GPL
 104721 LGPL/GPL
   62289 Artistic
   49851 None/Public Domain
     592 Proprietary (Netscape Communicator using Motif)

```

From these numbers, you can determine that:

1. The GPL is far and away the most common license (by lines of code) of any single license. In fact, the category "GPL" all by itself accounts for a simple majority of all code (53%), even when not including packages with multiple licenses (e.g., LGPL/GPL, GPL/BSD, Artistic or GPL, etc.). No matter how you look at it, the GPL is the dominant single license in this distribution.
2. The "distributable" category comes in second. At least some of this code is released under essentially MIT/BSD-style licenses, but more precise information is not included in the RPM specification files.
3. The next most common licenses were the MIT, LGPL, and BSD licenses (in order). This is in line with expectations: the most well-known and well-used open source licenses are the GPL, MIT, LGPL, and BSD licenses. There is some use of the "Artistic" license, but its use is far less; note that papers such as [Perens \[1999\]](#) specifically recommend against using the the Artistic license due to its legal ambiguities.
4. Very little software is released as public domain software ("no copyright"). There may be several factors that account for this. First, if a developer wishes to get credit for their work, this is a poor "license;" by law anyone can claim ownership of "public domain" software. Second, there may be a fear of litigation; both the MIT and BSD licenses permit essentially arbitrary use but forbid lawsuits. While licenses such as MIT's and BSD's are not proof against a lawsuit, they at least provide some legal protection, while releasing software to the public domain provides absolutely no protection. Finally, any software released into the public domain can be re-licensed under any other license, so there's nothing that keeps public domain software in the public domain - any of the other licenses here can "dominate" a public domain license.
5. There is a tiny amount of non-open-source code, which is entirely in one component - Netscape Communicator / Navigator. This component uses the Motif toolkit (which is not open source) and has proprietary code mixed into it. As a result, almost none of the code for this package is included on the CD-ROM - only a small amount of "placeholder" code is there. In the future it is expected that this component will be replaced by the results of the Mozilla project.
6. The packages which are clearly MITish/BSDish licenses (totalling the MIT, BSD, BSDish, and none/public domain entries) total 3,422,117 SLOC, or 19%. It's worth noting that 1,291,745 of these lines (38%) is accounted for by the XFree86 X server. If the XFree86 X server didn't use the MIT license, the total SLOC clearly in this category would go down to 2,130,372 SLOC (12% of the total system).
7. If all "distributable" and Artistic software was also considered MITish/BSDish, the total SLOC would be 6,350,336 (36%). Unfortunately, the information to determine which of these other packages are simply BSDish/Xish licenses is not included in the specification files.
8. The packages which are clearly copylefted (GPL, LGPL, LGPL/GPL) total 10,543,187 (60%).

It is quite clear that in this distribution the GPL is the dominant license and that copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses. This is a simple quantitative explanation why several visible projects

(Mozilla and Troll Tech's Qt) have recently dual-licensed their software with the GPL. When there is so much GPL software, GPL compatibility is critically important to the survival of many open source projects. The most common open source licenses in this distribution are the GPL, MIT, LGPL, and BSD licenses. Note that this is consistent with [Perens \[1999\]](#), who pleads that developers use an existing license instead of developing a new license where possible.

3.6 Total SLOC Counts

Given all of these assumptions, the counting programs compute a total of 17,652,561 physical source lines of code (SLOC); I will simplify this to "over 17 million physical SLOC". This is an astounding amount of code; compare this to reported sizes of other systems:

Product	SLOC
NASA Space Shuttle flight control	420K (shuttle) + 1.4 million (ground)
Sun Solaris (1998-2000)	7-8 million
Microsoft Windows 3.1 (1992)	3 million
Microsoft Windows 95	15 million
Microsoft Windows 98	18 million
Microsoft Windows NT (1992)	4 million
Microsoft Windows NT 5.0 (1998)	20 million

These numbers come from Bruce Schneier's *Crypto-Gram* [\[Schneier 2000\]](#), except for the Space Shuttle numbers which come from a National Academy of Sciences study [\[NAS 1996\]](#). Numbers for later versions of Microsoft products are not shown here because their values have great uncertainty in the published literature. The assumptions of these numbers are unclear (e.g., are these physical or logical lines of code?), but they are likely to be comparable physical SLOC counts.

Schneier also reports that "Linux, even with the addition of X Windows and Apache, is still under 5 million lines of code". At first, this seems to be contradictory, since this paper counts over 17 million SLOC, but Schneier appears to be literally correct in the context of his statement. The phrasing of his sentence suggests that Schneier is considering some sort of "minimal" system, since he considers "even the addition of X Windows" as a significant addition. As shown in appendix section B.4, taking the minimal "base" set of components in Red Hat Linux, and then adding the minimal set of components for graphical interaction (the X Windows's graphical server, library, configuration tool, and a graphics toolkit) and the Apache web server, the total is about 4.4 million physical SLOC - which is less than 5 million. This minimal system doesn't include some useful (but not strictly necessary) components, but a number of useful components could be added while still staying under a total of 5 million SLOC.

However, note the contrast. Many Linux distributions include with their operating systems many applications (e.g., bitmap editors) and development tools (for many different languages). As a result, the entire *delivered* system for such distributions (including Red Hat Linux 6.2) is *much* larger than the 5 million SLOC stated by Schneier. In short, this distribution's size appears similar to the size of Windows 98 and Windows NT 5.0 in 1998.

Microsoft's recent legal battles with the U.S. Department of Justice (DoJ) also involve the bundling of applications with the operating system. However, it's worth noting some differences. First, and most important legally, a judge has ruled that Microsoft is a monopoly, and under U.S. law monopolies aren't allowed to perform certain actions that other organizations may perform. Second, anyone can take Linux, bundle it with an application, and redistribute the resulting product. There is no barrier such as "secret interfaces" or relicensing costs that prevent anyone from making an application work on or integrate with Linux. Third, many Linux distributions include alternatives; users can choose between a number of options, all on the CD-ROM. Thus, while Linux distributions also appear to be going in the direction of adding applications to their system, they do not do so in a way that significantly interferes with a user's ability to select between alternatives.

It's worth noting that SLOC counts do not necessarily measure user functionality very well. For example, smart developers often find creative ways to simplify problems, so programs with small SLOC counts can often provide greater functionality than programs with large SLOC counts. However, there is evidence that SLOC counts correlate to effort (and thus development time), so using SLOC to estimate effort is still valid.

Creating reliable code can require much more effort than creating unreliable code. For example, it's known that the Space Shuttle code underwent rigorous testing and analysis, far more than typical commercial software undergoes, driving up its development costs. However, it cannot be reasonably argued that reliability differences between Linux and either Solaris or Windows NT would necessarily cause Linux to take less effort to develop for a similar size. To see this, let's pretend that Linux had been developed using traditional proprietary means and a similar process to these other products. As noted earlier, experiments suggest that Linux, or at least certain portions of it, is more reliable than either. This would either cost more

money (due to increased testing) or require a substantive change in development process (e.g., through increased peer review). Therefore, Linux's reliability suggests that developing Linux traditionally (at the same level of reliability) would have taken at least the same amount of effort if similar development processes were used as compared to similarly-sized systems.

3.7 Effort and Cost Estimates

Finally, given all the assumptions shown, are the effort values:

```
Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = $ 614421924.71
```

See appendix A for more data on how these effort values were calculated; you can retrieve more information from <http://www.dwheeler.com/sloc>.

4. Conclusions

Red Hat Linux version 6.2 includes well over 17 million lines of physical source lines of code (SLOC). Using the COCOMO cost model, this is estimated to have required over 4,500 person-years of development time. Had this Linux distribution been developed by conventional proprietary means, it's estimated that it would have cost over \$600 million to develop in the U.S. (in year 2000 dollars). Clearly, this demonstrates that it is possible to build large-scale systems using open source approaches.

Many other interesting statistics emerge. The largest components (in order) were the linux kernel (including device drivers), the X-windows server (for the graphical user interface), gcc (a compilation system, with the package name of ``egcs''), and emacs (a text editor and far more). The languages used, sorted by the most lines of code, were C, C++, LISP (including Emacs' LISP and Scheme), shell (including ksh), Perl, Tcl (including expect), assembly (all kinds), Python, yacc/bison, Java, lex/flex, awk, objective-C, C-shell, Ada, Pascal, sed, and Fortran. Here you can see that C is pre-eminent (with over 80% of the code). In this distribution the GPL is the dominant license, and copylefting licenses (the GPL and LGPL) significantly outnumber the BSD/MIT-style licenses in terms of SLOC. The most common open source licenses in this distribution are the GPL, MIT, LGPL, and BSD licenses. More information is available in the appendices and at <http://www.dwheeler.com/sloc>.

It would be interesting to re-run these values on other Linux distributions (such as SuSE and Debian), other open source systems (such as FreeBSD), and other versions of Red Hat (such as Red Hat 7). SuSE and Debian, for example, by policy include many more packages, and would probably produce significantly larger estimates of effort and development cost. It's known that Red Hat 7 includes more source code; Red Hat 7 has had to add another CD-ROM to contain the binary programs, and adds such capabilities as a word processor (abiword) and secure shell (openssh).

Some actions by developers could simplify further similar analyses. The most important would be for programmers to always mark, at the top, any generated files (e.g., with a phrase like ``Automatically generated''). This would do much more than aid counting tools - programmers are likely to accidentally manually edit such files unless the files are *clearly* marked as files that should not be edited. It would be useful if developers would use file extensions consistently and not ``reuse" extension names for other meanings; the suffixes(7) manual page lists a number of already-claimed extensions. This is more difficult for less-used languages; many developers have no idea that ``.m" is a standard extension for objective-C. It would also be nice to have high-quality open source tools for performing logical SLOC counting on all of the languages represented here.

It should be re-emphasized that these are *estimates*; it is very difficult to precisely categorize all files, and some files might confuse the size estimators. Some assumptions had to be made (such as not including makefiles) which, if made differently, would produce different results. Identifying automatically-generated files is very difficult, and it's quite possible that some were miscategorized.

Nevertheless, there are many insights to be gained from the analysis of entire open source systems, and hopefully this paper has provided some of those insights. It is my hope that, since open source systems make it possible for anyone to analyze them, others will pursue many other lines of analysis to gain further insight into these systems.

Appendix A. Details of Approach

My basic approach was to:

1. install the source code files,
2. categorize the files, creating for each package a list of files for each programming language; each file in each list contains source code in that language (excluding duplicate file contents and automatically generated files),
3. count the lines of code for each language for each component, and
4. use the original COCOMO model to estimate the effort to develop each component, and then the cost to develop using traditional methods.

This was not as easy as it sounds; each step is described below. Some steps I describe in some detail, because it's sometimes hard to find the necessary information even when the actual steps are easy. Hopefully, this detail will make it easier for others to do similar activities or to repeat the experiment.

A.1 Installing Source Code

Installing the source code files turned out to be nontrivial. First, I inserted the CD-ROM containing all of the source files (in ``.src.rpm" format) and installed the packages (files) using:

```
mount /mnt/cdrom
cd /mnt/cdrom/SRPMS
rpm -ivh *.src.rpm
```

This installs ``spec" files and compressed source files; another rpm command (``rpm -bp") uses the spec files to uncompress the source files into ``build directories" (as well as apply any necessary patches). Unfortunately, the rpm tool does not enforce any naming consistency between the package names, the spec names, and the build directory names; for consistency this paper will use the names of the build directories, since all later tools based themselves on the build directories.

I decided to (in general) not count ``old" versions of software (usually placed there for compatibility reasons), since that would be counting the same software more than once. Thus, the following components were not included: ``compat-binutils", ``compat-egcs", ``compat-glib", ``compat-libs", ``gtk+10", ``libc-5.3.12" (an old C library), ``libxml10", ``ncurses3", and ``qt1x". I also didn't include egcs64-19980921 and netscape-sparc, which simply repeated something on another architecture that was available on the i386 in a different package. I did make one exception. I kept both bash-1.14.7 and bash2, two versions of the shell command processor, instead of only counting bash2. While bash2 is the later version of the shell available in the package, the main shell actually used by the Red Hat distribution was the older version of bash. The rationale for this decision appears to be backwards compatibility for older shell scripts; this is suggested by the Red Hat package documentation in both bash-1.14.7 and bash2. It seemed wrong to not include one of the most fundamental pieces of the system in the count, so I included it. At 47067 lines of code (ignoring duplicates), bash-1.14.7 is one of the smaller components anyway. Not including this older component would not substantively change the results presented here.

There are two directories, krb4-1.0 and krb5-1.1.1, which appear to violate this rule - but don't. krb5-1.1.1 is the build directory created by krb5.spec, which is in turn installed by the source RPM package krb5-1.1.1-9.src.rpm. This build directory contains Kerberos V5, a trusted-third-party authentication system. The source RPM package krb5-1.1.1-9.src.rpm eventually generates the binary RPM files krb5-configs-1.1.1-9, krb5-libs-1.1.1-9, and krb5-devel-1.1.1-9. You might guess that ``krb4-1.0" is just the older version of Kerberos, but this build directory is created by the spec file krbafs.spec and not just an old version of the code. To quote its description, ``This is the Kerberos to AFS bridging library, built against Kerberos 5. krbafs is a shared library that allows programs to obtain AFS tokens using Kerberos IV credentials, without having to link with official AFS libraries which may not be available for a given platform." For this situation, I simply counted both packages, since their purposes are different.

I was then confronted with a fundamental question: should I count software that only works for another architecture? I was using an i86-type system, but some components are only for Alpha or Sparc systems. I decided that I should count them; even if I didn't use the code today, the ability to use these other architectures in the future was of value and certainly required effort to develop.

This caused complications for creating the build directories. If all installed packages fit the architecture, you can install the uncompressed software by typing:

```
cd /usr/src/redhat/SPECS and typing the command
```

```
rpm -bp *.spec
```

Unfortunately, the rpm tool notes that you're trying to load code for the "wrong" architecture, and (at least at the time) there was no simple "override" flag. Instead, I had to identify each package as belonging to SPARC or ALPHA, and then use the rpm option --target to forcibly load them. For example, I renamed all sparc-specific SPARC file files to end in ".sparc" and could then load them with:

```
rpm -bp --target sparc-redhat-linux *.spec.sparc
```

The following spec files were non-i86: (sparc) audiocntl, elftoaout, ethtool, prtconf, silo, solemul, sparc32; (alpha) aboot, minlabel, quickstrip. In general, these were tools to aid in supporting some part of the boot process or for using system-specific hardware.

Note that not all packages create build directories. For example, "anonftp" is a package that, when installed, sets up an anonymous ftp system. This package doesn't actually install any software; it merely installs a specific configuration of another piece of software (and unsets the configuration when uninstalled). Such packages are not counted at all in this sizing estimate.

Simply loading all the source code requires a fair amount of disk space. Using "du" to measure the disk space requirements (with 1024 byte disk blocks), I obtained the following results:

```
$ du -s /usr/src/redhat/BUILD /usr/src/redhat/SOURCES /usr/src/redhat/SPECS
2375928 /usr/src/redhat/BUILD
592404 /usr/src/redhat/SOURCES
4592 /usr/src/redhat/SPECS
```

Thus, these three directories required 2972924 1K blocks - approximately 3 gigabytes of space. Much more space would be required to compile it all.

A.2 Categorizing Source Code

My next task was to identify all files containing source code (not including any automatically generated source code). This is a non-trivial problem; there are 181,679 ordinary files in the build directory, and I had no interest in doing this identification by hand.

In theory, one could just look at the file extensions (.c for C, .py for python), but this is not enough in practice. Some packages reuse extensions if the package doesn't use that kind of file (e.g., the ".exp" extension of expect was used by some packages as "export" files, and the ".m" of objective-C was used by some packages for module information extracted from C code). Some files don't have extensions, particularly scripts. And finally, files automatically generated by another program should not be counted, since I wished to use the results to estimate effort.

I ended up writing a program of over 600 lines of Perl to perform this identification, which used a number of heuristics to categorize each file into categories. There is a category for each language, plus the categories non-programs, unknown (useful for scanning for problems), automatically generated program files, duplicate files (whose file contents duplicated other files), and zero-length files.

The program first checked for well-known extensions (such as .gif) that cannot be program files, and for a number of common generated filenames. It then peeked at the first line for "#!" followed by a legal script name. If that didn't work, it used the extension to try to determine the category. For a number of languages, the extension was not reliable, so for those languages the categorization program examined the file contents and used a set of heuristics to determine if the file actually belonged that category. If all else failed, the file was placed in the "unknown" category for later analysis. I later looked at the "unknown" items, checking the common extensions to ensure I had not missed any common types of code.

One complicating factor was that I wished to separate C, C++, and objective-C code, but a header file ending with ".h" or ".hpp" file could be any of them. I developed a number of heuristics to determine, for each file, what language it belonged to. For example, if a build directory has exactly one of these languages, determining the correct category for header files is easy. Similarly, if there is exactly one of these in the directory with the header file, it is presumed to be that kind. Finally, a header file with the keyword "class" is almost certainly not a C header file, but a C++ header file.

Detecting automatically generated files was not easy, and it's quite conceivable I missed a number of them. The first 15 lines were examined, to determine if any of them included at the beginning of the line (after spaces and possible comment markers) one of the following phrases: "generated automatically", "automatically generated", "this is a generated file", "generated with the (something) utility", or "do not edit". A number of filename conventions were used, too. For example, any "configure" file is presumed to be automatically generated if there's a "configure.in" file in the same directory.

To eliminate duplicates, the program kept md5 checksums of each program file. Any given md5 checksum would only be counted once. Build directories were processed alphabetically, so this meant that if the same file content was in both directories ``a" and ``b", it would be counted only once as being part of ``a". Thus, some packages with names later in the alphabet may appear smaller than would make sense at first glance. It is very difficult to eliminate ``almost identical" files (e.g., an older and newer version of the same code, included in two separate packages), because it is difficult to determine when ``similar" two files are essentially the ``same" file. Changes such as the use of pretty-printers and massive renaming of variables could make small changes seem large, while the many small files in the system could easily make different files seem the ``same." Thus, I did not try to make such a determination, and just considered files with different contents as different.

It's important to note that different rules could be used to ``count" lines of code. Some kinds of code were intentionally excluded from the count. Many RPM packages include a number of shell commands used to install and uninstall software; the estimate in this paper does not include the code in RPM packages. This estimate also does not include the code in Makefiles (which can be substantive). In both cases, the code in these cases is often cut and pasted from other similar files, so counting such code would probably overstate the actual development effort. In addition, Makefiles are often automatically generated.

On the other hand, this estimate does include some code that others might not count. This estimate includes test code included with the package, which isn't visible directly to users (other than hopefully higher quality of the executable program). It also includes code not used in this particular system, such as code for other architectures and OS's, bindings for languages not compiled into the binaries, and compilation-time options not chosen. I decided to include such code for two reasons. First, this code is validly represents the effort to build each component. Second, it does represent indirect value to the user, because the user can later use those components in other circumstances even if the user doesn't choose to do so by default.

So, after the work of categorizing the files, the following categories of files were created for each build directory (common extensions are shown in parentheses, and the name used in the data tables below are shown in brackets):

1. C (.c) [ansic]
2. C++ (.C, .cpp, .cxx, .cc) [cpp]
3. LISP (.el, .scm, .lsp, .jl) [lisp]
4. shell (.sh) [sh]
5. Perl (.pl, .pm, .perl) [perl]
6. Assembly (.s, .S, .asm) [asm]
7. TCL (.tcl, .tk, .itk) [tcl]
8. Python (.py) [python]
9. Yacc (.y) [yacc]
10. Java (.java) [java]
11. Expect (.exp) [exp]
12. lex (.l) [lex]
13. awk (.awk) [awk]
14. Objective-C (.m) [objc]
15. C shell (.csh) [csh]
16. Ada (.ada, .ads, .adb) [ada]
17. Pascal (.p) [pascal]
18. sed (.sed) [sed]
19. Fortran (.f) [fortran]

Note that we're counting Scheme as a dialect of LISP, and Expect is being counted separately from TCL. The command line shells Bourne shell, the Bourne-again shell (bash), and the K shell are all counted together as ``shell", but the C shell (csh and tcsh) is counted separately.

A.3 Counting Lines of Code

Every language required its own counting scheme. This was more complex than I realized; there were a number of languages involved.

I originally tried to use USC's ``CodeCount" tools to count the code. Unfortunately, this turned out to be buggy and did not handle most of the languages used in the system, so I eventually abandoned it for this task and wrote my own tools. Those who wish to use this tool are welcome to do so; you can learn more from its web site at

<http://sunset.usc.edu/research/CODECOUNT>.

I did manage to use the CodeCount to compute the logical source lines of code for the C portions of the linux kernel. This came out to be 673,627 logical source lines of code, compared to the 1,462,165 lines of physical code (again, this ignores files with duplicate contents).

Since there were a large number of languages to count, I used the "physical lines of code" definition. In this definition, a line of code is a line (ending with newline or end-of-file) with at least one non-comment non-whitespace character. These are known as "non-comment non-blank" lines. If a line only had whitespace (tabs and spaces) it was not counted, even if it was in the middle of a data value (e.g., a multiline string). It is much easier to write programs to measure this value than to measure the "logical" lines of code, and this measure can be easily applied to widely different languages. Since I had to process a large number of different languages, it made sense to choose the measure that is easier to obtain.

[Park \[1992\]](#) presents a framework of issues to be decided when trying to count code. Using Park's framework, here is how code was counted in this paper:

1. Statement Type: I used a physical line-of-code as my basis. I included executable statements, declarations (e.g., data structure definitions), and compiler directives (e.g., preprocessor commands such as #define). I excluded all comments and blank lines.
2. How Produced: I included all programmed code, including any files that had been modified. I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. If a file was in the source package, I included it; if the file had been removed from a source package (including via a patch), I did not include it.
3. Origin: I included all code included in the package.
4. Usage: I included code in or part of the primary product; I did not include code external to the product (i.e., additional applications able to run on the system but not included with the system).
5. Delivery: I counted code delivered as source; not surprisingly, I didn't count code not delivered as source. I also didn't count undelivered code.
6. Functionality: I included both operative and inoperative code. An examples of intentionally "inoperative" code is code turned off by #ifdef commands; since it could be turned on for special purposes, it made sense to count it. An examples of unintentionally "inoperative" code is dead or unused code.
7. Replications: I included master (original) source statements. I also included "physical replicates of master statements stored in the master code". This is simply code cut and pasted from one place to another to reuse code; it's hard to tell where this happens, and since it has to be maintained separately, it's fair to include this in the measure. I excluded copies inserted, instantiated, or expanded when compiling or linking, and I excluded postproduction replicates (e.g., reparameterized systems).
8. Development Status: Since I only measured code included in the packages used to build the delivered system, I declared that all software I was measuring had (by definition) passed whatever "system tests" were required by that component's developers.
9. Languages: I included all languages, as identified earlier in section A.2.
10. Clarifications: I included all statement types. This included nulls, continues, no-ops, lone semicolons, statements that instantiate generics, lone curly braces ({ and }), and labels by themselves.

Park includes in his paper a "basic definition" of physical lines of code, defined using his framework. I adhered to Park's definition unless (1) it was impossible in my technique to do so, or (2) it would appear to make the result inappropriate for use in cost estimation (using COCOMO). COCOMO states that source code:

"includes all program instructions created by project personnel and processed into machine code by some combination of preprocessors, compilers, and assemblers. It excludes comment cards and unmodified utility software. It includes job control language, format statements, and data declarations. Instructions are defined as lines of code."

In summary, though in general I followed Park's definition, I didn't follow Park's "basic definition" in the following ways:

1. How Produced: I excluded code generated with source code generators, converted with automatic translators, and those copied or reused without change. After all, COCOMO states that the only code that should be counted is code "produced by project personnel", whereas these kinds of files are instead the output of "preprocessors and compilers." If code is always maintained as the input to a code generator, and then the code generator is re-run, it's only the code generator input's size that validly measures the size of what is maintained. Note that while I attempted to exclude generated code, this exclusion is based on heuristics which may have missed some cases.
2. Origin: Normally physical SLOC doesn't include an unmodified "vendor-supplied language support library" nor a

``vendor-supplied system or utility". However, in this case this was *exactly* what I was measuring, so I naturally included these as well.

3. Delivery: I didn't count code not delivered as source. After all, since I didn't have it, I couldn't count it.
4. Functionality: I included unintentionally inoperative code (e.g., dead or unused code). There might be such code, but it is very difficult to automatically detect in general for many languages. For example, a program not directly invoked by anything else nor installed by the installer is much more likely to be a test program, which I'm including in the count. Clearly, discerning human ``intent" is hard to automate. Hopefully, unintentionally inoperative code is a small amount of the total delivered code.

Otherwise, I followed Park's ``basic definition" of a physical line of code, even down to Park's language-specific definitions where Park defined them for a language.

One annoying problem was that one file wasn't syntactically correct and it affected the count. File `/usr/src/redhat/BUILD/cdrecord-1.8/mkiso` had an `#ifdef` not taken, and the road not taken had a missing double-quote mark before the word ``cannot":

```
#ifdef  USE_LIBSCHILY
        comerr(Cannot open '%s'.\n", filename);
#endif
        perror ("fopen");
        exit (1);
#endif
```

I solved this by hand-patching the source code (for purposes of counting). There were also some files with intentionally erroneous code (e.g., compiler error tests), but these did not impact the SLOC count.

Several languages turn out to be non-trivial to count:

- In C, C++, and Java, comment markers should be ignored inside strings. Since they have multi-line comment markers this requirement should not be ignored, or a ``/*" inside a string could cause most of the code to be erroneously uncounted.
- Officially, C doesn't have C++'s ``/*" comment marker, but the gcc compiler accepts it and a great deal of C code uses it, so my counters accepted it.
- Perl permits in-line ``perlpod" documents, ``here" documents, and an `__END__` marker that complicate code-counting. Perlpod documents are essentially comments, but a ``here" document may include text to generate them (in which case the perlpod document is data and should be counted). The `__END__` marker indicates the end of the file from Perl's viewpoint, even if there's more text afterwards.
- Python has a convention that, at the beginning of a definition (e.g., of a function, method, or class), an unassigned string can be placed to describe what's being defined. Since this is essentially a comment (though it doesn't syntactically look like one), the counter must avoid counting such strings, which may have multiple lines. To handle this, strings which started the beginning of a line were not counted. Python also has the ``triple quote" operator, permitting multiline strings; these needed to be handled specially. Triple quote strings were normally considered as data, regardless of content, unless they were used as a comment about a definition.
- Assembly languages vary greatly in the comment character they use, so my counter had to handle this variance. I wrote a program which first examined the file to determine if C-style ``/*" comments and C preprocessor commands (e.g., `#include`) were used. If both ``/*" and ``/*" were in the file, it was assumed that C-style comments were used, since it is unlikely that *both* would be used as something else (e.g., as string data) in the same assembly language file. Determining if a file used the C preprocessor was trickier, since many assembly files do use ``#" as a comment character and some preprocessor directives are ordinary words that might be included in a human comment. The heuristic used was: if `#ifdef`, `#endif`, or `#include` are used, the preprocessor is used; if at least three lines have either `#define` or `#else`, then the preprocessor is used. No doubt other heuristics are possible, but this at least seemed to produce reasonable results. The program then determined what the comment character was, by identifying which punctuation mark (from a set of possible marks) was the most common non-space initial character on a line (ignoring ``/" and ``#" if C comments or preprocessor commands, respectively, were used). Once the comment character had been determined, and it had been determined if C-style comments were also allowed, the lines of code could be counted in the file.

Although their values are not used in estimating effort, I also counted the number of files; summaries of these values are included in appendix B.

Since the linux kernel was the largest single component, and I had questions about the various inconsistencies in the ``Halloween" documents, I made additional measures of the Linux kernel.

Some have objected because the counting approach used here includes lines not compiled into code in this Linux distribution. However, the primary objective of these measures was to estimate total effort to develop all of these components. Even if some lines are not normally enabled on Linux, it still required effort to develop that code. Code for other architectures still has value, for example, because it enables users to port to other architectures while using the component. Even if that code is no longer being maintained (e.g., because the architecture has become less popular), nevertheless someone had to invest effort to create it, the results benefitted someone, and if it is needed again it's still there (at least for use as a starting point). Code that is only enabled by compile-time options still has value, because if the options were desired the user could enable them and recompile. Code that is only used for testing still has value, because its use improves the quality of the software directly run by users. It is possible that there is some "dead code" (code that cannot be run under any circumstance), but it is expected that this amount of code is very small and would not significantly affect the results. Andi Kleen (of SuSE) noted that if you wanted to only count compiled and running code, one technique (for some languages) would be to use gcc's `-g` option and use the resulting .stabs debugging information with some filtering (to exclude duplicated inline functions). I determined this to be out-of-scope for this paper, but this approach could be used to make additional measurements of the system.

A.4 Estimating Effort and Costs

For each build directory, I totalled the source lines of code (SLOC) for each language, then totalled those values to determine the SLOC for each directory. I then used the basic Constructive Cost Model (COCOMO) to estimate effort. The basic model is the simplest (and least accurate) model, but I simply did not have the additional information necessary to use the more complex (and more accurate) models. COCOMO is described in depth by Boehm [1981].

Basic COCOMO is designed to estimate the time from product design (after plans and requirements have been developed) through detailed design, code, unit test, and integration testing. Note that plans and requirement development are not included. COCOMO is designed to include management overhead and the creation of documentation (e.g., user manuals) as well as the code itself. Again, see Boehm [1981] for a more detailed description of the model's assumptions.

In the basic COCOMO model, estimated man-months of effort, design through test, equals $2.4 \cdot (\text{KSLOC})^{1.05}$, where KSLOC is the total physical SLOC divided by 1000.

I assumed that each package was built completely independently and that there were no efforts necessary for integration not represented in the code itself. This almost certainly underestimates the true costs, but for most packages it's actually true (many packages don't interact with each other at all). I wished to underestimate (instead of overestimate) the effort and costs, and having no better model, I assumed the simplest possible integration effort. This meant that I applied the model to each component, then summed the results, as opposed to applying the model once to the grand total of all software.

Note that the only input to this model is source lines of code, so some factors simply aren't captured. For example, creating some kinds of data (such as fonts) can be very time-consuming, but this isn't directly captured by this model. Some programs are intentionally designed to be data-driven, that is, they're designed as small programs which are driven by specialized data. Again, this data may be as complex to develop as code, but this is not counted.

Another example of uncaptured factors is the difficulty of writing kernel code. It's generally acknowledged that writing kernel-level code is more difficult than most other kinds of code, because this kind of code is subject to a subtle timing and race conditions, hardware interactions, a small stack, and none of the normal error protections. In this paper I do not attempt to account for this. You could try to use the Intermediate COCOMO model to try to account for this, but again this requires knowledge of other factors that can only be guessed at. Again, the effort estimation probably significantly underestimates the actual effort represented here.

It's worth noting that there is an update to COCOMO, COCOMO II. However, COCOMO II requires as its input logical (not physical) SLOC, and since this measure is much harder to obtain, I did not pursue it for this paper. More information about COCOMO II is available at the web site <http://sunset.usc.edu/research/COCOMOII/index.html>. A nice overview paper where you can learn more about software metrics is Masse [1997].

I assumed that an average U.S. programmer/analyst salary in the year 2000 was \$56,286 per year; this value was from the ComputerWorld, September 4, 2000's Salary Survey. Overhead is much harder to estimate; I did not find a definitive source for information on overheads. After informal discussions with several cost analysts, I determined that an overhead of 2.4 would be representative of the overhead sustained by a typical software development company. Should you disagree with these figures, I've provided all the information necessary to recalculate your own cost figures; just start with the effort estimates and recalculate cost yourself.

Appendix B. More Detailed Results

This appendix provides some more detailed results. B.1 lists the SLOC found in each build directory; B.2 shows counts of files for each category of file; B.3 presents some additional measures about the Linux kernel. B.4 presents some SLOC totals of putatively ``minimal" systems. You can learn more at <http://www.dwheeler.com/sloc>.

B.1 SLOC in Build Directories

The following is a list of all build directories, and the source lines of code (SLOC) found in them, followed by a few statistics counting files (instead of SLOC).

Remember that duplicate files are only counted once, with the build directory ``first in ASCII sort order" receiving any duplicates (to break ties). As a result, some build directories have a smaller number than might at first make sense. For example, the ``kudzu" build directory does contain code, but all of it is also contained in the ``Xconfigurator" build directory.. and since that directory sorts first, the kudzu package is considered to have ``no code".

The columns are SLOC (total physical source lines of code), Directory (the name of the build directory, usually the same or similar to the package name), and SLOC-by-Language (Sorted). This last column lists languages by name and the number of SLOC in that language; zeros are not shown, and the list is sorted from largest to smallest in that build directory. Similarly, the directories are sorted from largest to smallest total SLOC.

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,cs=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,cs=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,cs=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,cs=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,cs=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,sed=2
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,yacc=1507,awk=522,lex=323,sed=297,asm=139,cs=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,cs=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,cs=235,

```

sed=35,lisp=12
133193 kaffe-1.0.5      java=65275,ansic=62125,cpp=3923,perl=972,sh=814,
asm=84
131372 jade-1.2.1      cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672 gnome-libs-1.0.55 ansic=125373,sh=2178,perl=667,awk=277,lisp=177
127536 pine4.21          ansic=126678,sh=766,csch=62,perl=30
121878 ImageMagick-4.2.9  ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
119613 lynx2-8-3           ansic=117385,sh=1860,perl=340,csch=28
116951 mc-4.5.42           ansic=114406,sh=1996,perl=345,awk=148,csch=56
116615 gnumeric-0.48       ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
113272 xlipstat-3-52-17    ansic=91484,lisp=21769,sh=18,csch=1
113241 vim-5.6             ansic=111724,awk=683,sh=469,perl=359,csch=6
109824 php-3.0.15         ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
104032 linuxconf-1.17r2    cpp=93139,perl=4570,sh=2984,java=2741,ansic=598
102674 libgr-2.0.13        ansic=99647,sh=2438,csch=589
100951 lam-6.3.1           ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187,
csch=19
99066 krb4-1.0          ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765,
yacc=1509,lex=236,awk=33
94637 xlockmore-4.15      ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
93940 kdenetwork          cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
92964 samba-2.0.6        ansic=88308,sh=3557,perl=831,awk=158,csch=110
91213 anaconda-6.2.2    ansic=74303,python=13657,sh=1583,yacc=810,lex=732,
perl=128
89959 xscreensaver-3.23 ansic=88488,perl=1070,sh=401
88128 cvs-1.10.7          ansic=68303,sh=17909,perl=902,yacc=826,csch=181,lisp=7
87940 isdn4k-utils        ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
85383 xpdf-0.90           cpp=60427,ansic=21400,sh=3556
81719 inn-2.2.2          ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547,
lex=249,tcl=3
80343 kdelibs           cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116,
sh=35
79997 WindowMaker-0.61.1 ansic=77924,sh=1483,perl=371,lisp=219
78787 extace-1.2.15       ansic=66571,sh=9322,perl=2894
77873 apache_1.3.12       ansic=69191,sh=6781,perl=1846,cpp=55
75257 xpilot-4.1.0        ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68
73817 w3c-libwww-5.2.8    ansic=64754,sh=4678,cpp=3181,perl=1204
72726 ucd-snmp-4.1.1      ansic=64411,perl=5558,sh=2757
72425 gnome-core-1.0.55  ansic=72230,perl=141,sh=54
71810 jikes              cpp=71452,java=358
70260 groff-1.15         cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397,
sh=265,sed=46
69265 fvwm-2.2.4         ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
69246 linux-86           ansic=63328,asm=5276,sh=642
68997 blt2.4g           ansic=58630,tcl=10215,sh=152
68884 squid-2.3.STABLE1  ansic=66305,sh=1570,perl=1009
68560 bash-2.03          ansic=56758,sh=7264,yacc=2808,perl=1730
68453 kdeggraphics       cpp=34208,ansic=29347,sh=4898
65722 xntp3-5.93          ansic=60190,perl=3633,sh=1445,awk=417,asm=37
62922 ppp-2.3.11          ansic=61756,sh=996,exp=82,perl=44,csch=44
62137 sgml-tools-1.0.9    cpp=38543,ansic=19185,perl=2866,lex=560,sh=532,
lisp=309,awk=142
61688 imap-4.7           ansic=61628,sh=60
61324 ncurses-5.0         ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103,
sed=100
60429 kdesupport         ansic=42421,cpp=17810,sh=173,awk=13,csch=12
60302 openldap-1.2.9      ansic=58078,sh=1393,perl=630,python=201
57217 xfig.3.2.3-beta-1   ansic=57212,csch=5
56093 lsof_4.47           ansic=50268,sh=4753,perl=856,awk=214,asm=2
55667 uucp-1.06.1         ansic=52078,sh=3400,perl=189

```


54935	gnupg-1.0.1	ansic=48884,asm=4586,sh=1465
54603	glade-0.5.5	ansic=49545,sh=5058
54431	svglib-1.4.1	ansic=53725,asm=630,perl=54,sh=22
53141	AfterStep-1.8.0	ansic=50898,perl=1168,sh=842,cpp=233
52808	kdeutils	cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
52574	nmh-1.0.3	ansic=50698,sh=1785,awk=74,sed=17
51813	freetype-1.3.1	ansic=48929,sh=2467,cpp=351,cs=53,perl=13
51592	enlightenment-0.15.5	ansic=51569,sh=23
50970	cdrecord-1.8	ansic=48595,sh=2177,perl=194,sed=4
49370	tin-1.4.2	ansic=47763,sh=908,yacc=699
49325	imlib-1.9.7	ansic=49260,sh=65
48223	kdemultimedia	ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
47067	bash-1.14.7	ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
46312	tcsh-6.09.00	ansic=43544,sh=921,lisp=669,perl=593,cs=585
46159	unzip-5.40	ansic=40977,cpp=3778,asm=1271,sh=133
45811	mutt-1.0.1	ansic=45574,sh=237
45589	am-utils-6.0.3	ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
45485	guile-1.3	ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,cs=50
45378	gnuplot-3.7.1	ansic=43276,lisp=661,asm=539,objc=387,cs=297,perl=138,sh=80
44323	mgetty-1.1.21	ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
42880	sendmail-8.9.3	ansic=40364,perl=1737,sh=779
42746	elm2.5.3	ansic=32931,sh=9774,awk=41
41388	p2c-1.22	ansic=38788,pascal=2499,perl=101
41205	gnome-games-1.0.51	ansic=31191,lisp=6966,cpp=3048
39861	rpm-3.0.4	ansic=36994,sh=1505,perl=1355,python=7
39160	util-linux-2.10f	ansic=38627,sh=351,perl=65,cs=62,sed=55
38927	xmms-1.0.1	ansic=38366,asm=398,sh=163
38548	ORBit-0.5.0	ansic=35656,yacc=1750,sh=776,lex=366
38453	zsh-3.0.7	ansic=36208,sh=1763,perl=331,awk=145,sed=6
37515	ircii-4.4	ansic=36647,sh=852,lex=16
37360	tiff-v3.5.4	ansic=32734,sh=4054,cpp=572
36338	textutils-2.0a	ansic=18949,sh=16111,perl=1218,sed=60
36243	exmh-2.1.1	tcl=35844,perl=316,sh=49,exp=34
36239	x11amp-0.9-alpha3	ansic=31686,sh=4200,asm=353
35812	xloadimage.4.1	ansic=35705,sh=107
35554	zip-2.3	ansic=32108,asm=3446
35397	gtk-engines-0.10	ansic=20636,sh=14761
35136	php-2.0.1	ansic=33991,sh=1056,awk=89
34882	pmake	ansic=34599,sh=184,awk=58,sed=41
34772	xpuzzles-5.4.1	ansic=34772
34768	fileutils-4.0p	ansic=31324,sh=2042,yacc=841,perl=561
33203	strace-4.2	ansic=30891,sh=1988,perl=280,lisp=44
32767	trn-3.6	ansic=25264,sh=6843,yacc=660
32277	pilot-link.0.9.3	ansic=26513,java=2162,cpp=1689,perl=971,yacc=660,python=268,tcl=14
31994	korganizer	cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
31174	ncftp-3.0beta21	ansic=30347,cpp=595,sh=232
30438	gnome-pim-1.0.55	ansic=28665,yacc=1773
30122	scheme-3.2	lisp=19483,ansic=10515,sh=124
30061	tcpdump-3.4	ansic=29208,yacc=236,sh=211,lex=206,awk=184,cs=16
29730	screen-3.9.5	ansic=28156,sh=1574
29315	jed	ansic=29315
29091	xchat-1.4.0	ansic=28894,perl=121,python=53,sh=23
28897	ncpfs-2.2.0.17	ansic=28689,sh=182,tcl=26
28449	slrn-0.9.6.2	ansic=28438,sh=11
28261	xfishtank-2.1tp	ansic=28261
28186	texinfo-4.0	ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
28169	e2fsprogs-1.18	ansic=27250,awk=437,sh=339,sed=121,perl=22
28118	slang	ansic=28118

27860	kdegames	cpp=27507,ansic=340,sh=13
27117	librep-0.10	ansic=19381,lisp=5385,sh=2351
27040	mikmod-3.1.6	ansic=26975,sh=55,awk=10
27022	x3270-3.1.1	ansic=26456,sh=478,exp=88
26673	lout-3.17	ansic=26673
26608	Xaw3d-1.3	ansic=26235,yacc=247,lex=126
26363	gawk-3.0.4	ansic=19871,awk=2519,yacc=2046,sh=1927
26146	libxml-1.8.6	ansic=26069,sh=77
25994	xrn-9.02	ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31, csh=13
25915	gv-3.5.8	ansic=25821,sh=94
25479	xpaint	ansic=25456,sh=23
25236	shadow-19990827	ansic=23464,sh=883,yacc=856,perl=33
24910	kdeadmin	cpp=19919,sh=3936,perl=1055
24773	pdksh-5.2.14	ansic=23599,perl=945,sh=189,sed=40
24583	gmp-2.0.2	ansic=17888,asm=5252,sh=1443
24387	mars_nwe	ansic=24158,sh=229
24270	gnome-python-1.0.51	python=14331,ansic=9791,sh=148
23838	kterm-6.2.0	ansic=23838
23666	enscript-1.6.1	ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
22373	sawmill-0.24	ansic=11038,lisp=8172,sh=3163
22279	make-3.78.1	ansic=19287,sh=2029,perl=963
22011	libpng-1.0.5	ansic=22011
21593	xboard-4.0.5	ansic=20640,lex=904,sh=41,csh=5,sed=3
21010	netkit-telnet-0.16	ansic=14796,cpp=6214
20433	pam-0.72	ansic=18936,yacc=634,sh=482,perl=321,lex=60
20125	ical-2.2	cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
20078	gd1.3	ansic=19946,perl=132
19971	wu-ftpd-2.6.0	ansic=17572,yacc=1774,sh=421,perl=204
19500	gnome-utils-1.0.50	ansic=18099,yacc=824,lisp=577
19065	joe	ansic=18841,asm=224
18885	X11R6-contrib-3.3.2	ansic=18616,lex=161,yacc=97,sh=11
18835	glib-1.2.6	ansic=18702,sh=133
18151	git-4.3.19	ansic=16166,sh=1985
18020	xboing	ansic=18006,sh=14
17939	sh-utils-2.0	ansic=13366,sh=3027,yacc=871,perl=675
17765	mtools-3.9.6	ansic=16155,sh=1602,sed=8
17750	gettext-0.10.35	ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9
17682	bc-1.05	ansic=9186,sh=7236,yacc=967,lex=293
17271	fetchmail-5.3.1	ansic=13441,python=1490,sh=1246,yacc=411,perl=321, lex=238,awk=124
17259	sox-12.16	ansic=16659,sh=600
16785	control-center-1.0.51	ansic=16659,sh=126
16266	dhcp-2.0	ansic=15328,sh=938
15967	SVGATextMode-1.9-src	ansic=15079,yacc=340,sh=294,lex=227,sed=15, asm=12
15868	kpilot-3.1b9	cpp=8613,ansic=5640,yacc=1615
15851	taper-6.9a	ansic=15851
15819	mpg123-0.59r	ansic=14900,asm=919
15691	transfig.3.2.1	ansic=15643,sh=38,csh=10
15638	mod_perl-1.21	perl=10278,ansic=5124,sh=236
15522	console-tools-0.3.3	ansic=13335,yacc=986,sh=800,lex=291,perl=110
15456	rpm2html-1.2	ansic=15334,perl=122
15143	gnotepad+-1.1.4	ansic=15143
15108	GXedit1.23	ansic=15019,sh=89
15087	mm2.7	ansic=8044,csh=6924,sh=119
14941	readline-2.2.1	ansic=11375,sh=1890,perl=1676
14912	ispell-3.1	ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385, csh=221,sh=157,perl=85,sed=15
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,csh=29

Estimating Linux's Size

14774	flex-2.5.4	ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12
14587	multimedia	ansic=14577,sh=10
14516	libgtop-1.0.6	ansic=13768,perl=653,sh=64,asm=31
14427	mawk-1.2.2	ansic=12714,yacc=994,awk=629,sh=90
14363	automake-1.4	perl=10622,sh=3337,ansic=404
14350	rsync-2.4.1	ansic=13986,perl=179,sh=126,awk=59
14299	nfs-utils-0.1.6	ansic=14107,sh=165,perl=27
14269	rcs-5.7	ansic=12209,sh=2060
14255	tar-1.13.17	ansic=13014,lisp=592,sh=538,perl=111
14105	wmakerconf-2.1	ansic=13620,perl=348,sh=137
14039	less-346	ansic=14032,awk=7
13779	rxvt-2.6.1	ansic=13779
13586	wget-1.5.3	ansic=13509,perl=54,sh=23
13504	rp3-1.0.7	cpp=10416,ansic=2957,sh=131
13241	iproute2	ansic=12139,sh=1002,perl=100
13100	silo-0.9.8	ansic=10485,asm=2615
12657	macutils	ansic=12657
12639	libungif-4.1.0	ansic=12381,sh=204,perl=54
12633	minicom-1.83.0	ansic=12503,sh=130
12593	audiofile-0.1.9	sh=6440,ansic=6153
12463	gnome-objc-1.0.2	objc=12365,sh=86,ansic=12
12313	jpeg-6a	ansic=12313
12124	ypserv-1.3.9	ansic=11622,sh=460,perl=42
11790	lrzsz-0.12.20	ansic=9512,sh=1263,exp=1015
11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
11633	net-tools-1.54	ansic=11531,sh=102
11404	findutils-4.1	ansic=11160,sh=173,exp=71
11299	xmorphism-1999dec12	ansic=10783,tcl=516
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
10914	diffutils-2.7	ansic=10914
10404	gnorpm-0.9	ansic=10404
10271	gqview-0.7.0	ansic=10271
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
10187	dump-0.4b15	ansic=9422,sh=760,sed=5
10088	piranha	ansic=10048,sh=40
10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
9961	procps-2.0.6	ansic=9959,sh=2
9942	xpat2-1.04	ansic=9942
9927	procmail-3.14	ansic=8090,sh=1837
9873	nss_ldap-105	ansic=9784,perl=89
9801	man-1.5h1	ansic=7377,sh=1802,perl=317,awk=305
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6
9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
9699	bison-1.28	ansic=9650,sh=49
9666	ash-linux-0.2	ansic=9445,sh=221
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
9551	pwdb-0.61	ansic=9488,sh=63
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
9263	ctags-3.4	ansic=9240,sh=23
9138	gftp-2.0.6a	ansic=9138
8939	mkisofs-1.12b5	ansic=8939
8766	pxe-linux	cpp=4463,ansic=3622,asm=681
8572	psgml-1.2.1	lisp=8572
8540	xxgdb-1.12	ansic=8540
8491	gtop-1.0.5	ansic=8151,cpp=340
8356	gedit-0.6.1	ansic=8225,sh=131
8303	dip-3.3.7o	ansic=8207,sh=96
7859	libglade-0.11	ansic=5898,sh=1809,python=152

7826	xpm-3.4k	ansic=7750,sh=39,cpp=37
7740	sed-3.02	ansic=7301,sed=359,sh=80
7617	cpio-2.4.2	ansic=7598,sh=19
7615	esound-0.2.17	ansic=7387,sh=142,csch=86
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
7427	ed-0.2	ansic=7263,sh=164
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,cpp=3
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
7095	xgammon-0.98	ansic=6506,lex=589
7041	newt-0.50.8	ansic=6526,python=515
7030	ee-0.3.11	ansic=7007,sh=23
6976	aboot-0.5	ansic=6680,asm=296
6968	mailx-8.1.1	ansic=6963,sh=5
6877	lpr	ansic=6842,sh=35
6827	gnome-media-1.0.51	ansic=6827
6646	iputils	ansic=6646
6611	patch-2.5	ansic=6561,sed=50
6592	xosview-1.7.1	cpp=6205,ansic=367,awk=20
6550	byacc-1.9	ansic=5520,yacc=1030
6496	pidentd-3.0.10	ansic=6475,sh=21
6391	m4-1.4	ansic=5993,lisp=243,sh=155
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
6234	awesfx-0.4.3a	ansic=6234
6172	sash-3.4	ansic=6172
6116	lsblk	ansic=5325,sh=791
6090	joystick-1.2.15	ansic=6086,sh=4
6072	kdoc	perl=6010,sh=45,cpp=17
6043	irda-utils-0.9.10	ansic=5697,sh=263,perl=83
6033	sysvinit-2.78	ansic=5256,sh=777
6025	pnm2ppa	ansic=5708,sh=317
6021	rpmfind-1.4	ansic=6021
5981	indent-2.2.5	ansic=5958,sh=23
5975	ytalk-3.1	ansic=5975
5960	isapnptools-1.21	ansic=4394,yacc=1383,perl=123,sh=60
5744	gdm-2.0beta2	ansic=5632,sh=112
5594	isdn-config	cpp=3058,sh=2228,perl=308
5526	efax-0.9	ansic=4570,sh=956
5383	acct-6.3.2	ansic=5016,cpp=287,sh=80
5115	libtool-1.3.4	sh=3374,ansic=1741
5111	netkit-ftp-0.16	ansic=5111
4996	bzip2-0.9.5d	ansic=4996
4895	xcpustate-2.5	ansic=4895
4792	libelf-0.6.4	ansic=3310,sh=1482
4780	make-3.78.1_pvm-0.5	ansic=4780
4542	gpgp-0.4	ansic=4441,sh=101
4430	gperf-2.7	cpp=2947,exp=745,ansic=695,sh=43
4367	aumix-1.30.1	ansic=4095,sh=179,sed=93
4087	zlib-1.1.3	ansic=2815,asm=712,cpp=560
4038	sysklogd-1.3-31	ansic=3741,perl=158,sh=139
4024	rep-gtk-0.8	ansic=2905,lisp=971,sh=148
3962	netkit-timed-0.16	ansic=3962
3929	initscripts-5.00	sh=2035,ansic=1866,csch=28
3896	ltracex-0.3.10	ansic=2986,sh=854,awk=56
3885	phhttpd-0.1.0	ansic=3859,sh=26
3860	xdaliclock-2.18	ansic=3837,sh=23
3855	pciutils-2.1.5	ansic=3800,sh=55
3804	quota-2.00-pre3	ansic=3795,sh=9
3675	dosfstools-2.2	ansic=3675
3654	tcp_wrappers_7.6	ansic=3654
3651	ipchains-1.3.9	ansic=2767,sh=884

3625	autofs-3.1.4	ansic=2862,sh=763
3588	netkit-rsh-0.16	ansic=3588
3438	yp-tools-2.4	ansic=3415,sh=23
3433	dialog-0.6	ansic=2834,perl=349,sh=250
3415	ext2ed-0.1	ansic=3415
3315	gdbm-1.8.0	ansic=3290,cpp=25
3245	ypbind-3.3	ansic=1793,sh=1452
3219	playmidi-2.4	ansic=3217,sed=2
3096	xtrojka123	ansic=3087,sh=9
3084	at-3.1.7	ansic=1442,sh=1196,yacc=362,lex=84
3051	dhcpcd-1.3.18-pl3	ansic=2771,sh=280
3012	apmd	ansic=2617,sh=395
2883	netkit-base-0.16	ansic=2883
2879	vixie-cron-3.0.1	ansic=2866,sh=13
2835	gkernit-1.0	ansic=2835
2810	kdetoys	cpp=2618,ansic=192
2791	xjewel-1.6	ansic=2791
2773	mpage-2.4	ansic=2704,sh=69
2758	autoconf-2.13	sh=2226,perl=283,exp=167,ansic=82
2705	autorun-2.61	sh=1985,cpp=720
2661	cdp-0.33	ansic=2661
2647	file-3.28	ansic=2601,perl=46
2645	libghttp-1.0.4	ansic=2645
2631	getty_ps-2.0.7j	ansic=2631
2597	pythonlib-1.23	python=2597
2580	magicdev-0.2.7	ansic=2580
2531	gnome-kberos-0.2	ansic=2531
2490	sndconfig-0.43	ansic=2490
2486	bug-buddy-0.7	ansic=2486
2459	usermode-1.20	ansic=2459
2455	fnlib-0.4	ansic=2432,sh=23
2447	sliplogin-2.1.1	ansic=2256,sh=143,perl=48
2424	raidtools-0.90	ansic=2418,sh=6
2423	netkit-routed-0.16	ansic=2423
2407	nc	ansic=1670,sh=737
2324	up2date-1.13	python=2324
2270	memprof-0.3.0	ansic=2270
2268	which-2.9	ansic=1398,sh=870
2200	printtool	tcl=2200
2163	gnome-linuxconf-0.25	ansic=2163
2141	unarj-2.43	ansic=2141
2065	units-1.55	ansic=1963,perl=102
2048	netkit-ntalk-0.16	ansic=2048
1987	cracklib,2.7	ansic=1919,perl=46,sh=22
1984	cleanfeed-0.95.7b	perl=1984
1977	wmconfig-0.9.8	ansic=1941,sh=36
1941	isicom	ansic=1898,sh=43
1883	slocate-2.1	ansic=1802,sh=81
1857	netkit-rusers-0.16	ansic=1857
1856	pump-0.7.8	ansic=1856
1842	cdecl-2.5	ansic=1002,yacc=765,lex=75
1765	fbset-2.1	ansic=1401,yacc=130,lex=121,perl=113
1653	adjtimex-1.9	ansic=1653
1634	netcfg-2.25	python=1632,sh=2
1630	psmisc	ansic=1624,sh=6
1621	urlview-0.7	ansic=1515,sh=106
1604	fortune-mod-9708	ansic=1604
1531	netkit-tftp-0.16	ansic=1531
1525	logrotate-3.3.2	ansic=1524,sh=1
1473	traceroute-1.4a5	ansic=1436,awk=37

1452	time-1.7	ansic=1395,sh=57
1435	ncompress-4.2.4	ansic=1435
1361	mt-st-0.5b	ansic=1361
1290	cxhextris	ansic=1290
1280	pam_krb5-1	ansic=1280
1272	bsd-finger-0.16	ansic=1272
1229	hdparm-3.6	ansic=1229
1226	procinfo-17	ansic=1145,perl=81
1194	passwd-0.64.1	ansic=1194
1182	auth_ldap-1.4.0	ansic=1182
1146	prtconf-1.3	ansic=1146
1143	anacron-2.1	ansic=1143
1129	xbill-2.0	cpp=1129
1099	popt-1.4	ansic=1039,sh=60
1088	nag	perl=1088
1076	stylesheets-0.13rh	perl=888,sh=188
1075	authconfig-3.0.3	ansic=1075
1049	kpppload-1.04	cpp=1044,sh=5
1020	MAKEDEV-2.5.2	sh=1020
1013	trojka	ansic=1013
987	xmailbox-2.5	ansic=987
967	netkit-rwho-0.16	ansic=967
953	switchdesk-2.1	ansic=314,perl=287,cpp=233,sh=119
897	portmap_4	ansic=897
874	ldconfig-1999-02-21	ansic=874
844	jpeg-6b	sh=844
834	ElectricFence-2.1	ansic=834
830	mouseconfig-4.4	ansic=830
816	rpmlint-0.8	python=813,sh=3
809	kdpms-0.2.8	cpp=809
797	termcap-2.0.8	ansic=797
787	xsysinfo-1.7	ansic=787
770	giftrans-1.12.2	ansic=770
742	setserial-2.15	ansic=742
728	tree-1.2	ansic=728
717	chkconfig-1.1.2	ansic=717
682	lpg	perl=682
657	eject-2.0.2	ansic=657
616	diffstat-1.27	ansic=616
592	netscape-4.72	sh=592
585	usernet-1.0.9	ansic=585
549	genromfs-0.3	ansic=549
548	tksysv-1.1	tcl=526,sh=22
537	minlabel-1.2	ansic=537
506	netkit-bootparamd-0.16	ansic=506
497	locale_config-0.2	ansic=497
491	helptool-2.4	perl=288,tcl=203
480	elftoaout-2.2	ansic=480
463	tmpwatch-2.2	ansic=311,sh=152
445	rhs-printfilters-1.63	sh=443,ansic=2
441	audioctl	ansic=441
404	control-panel-3.13	ansic=319,tcl=85
368	kbdconfig-1.9.2.4	ansic=368
368	vlock-1.3	ansic=368
367	timetool-2.7.3	tcl=367
347	kernelcfg-0.5	python=341,sh=6
346	timeconfig-3.0.3	ansic=318,sh=28
343	mingetty-0.9.4	ansic=343
343	chkfontpath-1.7	ansic=343
332	ethtool-1.0	ansic=332

Estimating Linux's Size

```

314      mkbootdisk-1.2.5 sh=314
302      symlinks-1.2      ansic=302
301      xsri-1.0          ansic=301
294      netkit-rwall-0.16 ansic=294
290      biff+comsat-0.16  ansic=290
288      mkinitrd-2.4.1   sh=288
280      stat-1.5          ansic=280
265      sysreport-1.0     sh=265
261      bdflush-1.5       ansic=202,asm=59
255      ipvsadm-1.1       ansic=255
255      sag-0.6-html      perl=255
245      man-pages-1.28    sh=244,sed=1
240      open-1.4          ansic=240
236      xtoolwait-1.2     ansic=236
222      utempter-0.5.2    ansic=222
222      mkkickstart-2.1   sh=222
221      hellas           sh=179,perl=42
213      rhmask           ansic=213
159      quickstrip-1.1    ansic=159
132      rdate-1.0         ansic=132
131      statserial-1.1    ansic=121,sh=10
107      fwhois-1.00       ansic=107
85       mktemp-1.5        ansic=85
82       modemtool-1.21    python=73,sh=9
67       setup-1.2         ansic=67
56       shaper           ansic=56
52       sparc32-1.1       ansic=52
47       intimed-1.10      ansic=47
23       locale-ja-9       sh=23
16       AnotherLevel-1.0.1 sh=16
11       words-2          sh=11
7        trXFree86-2.1.2   tcl=7
0        install-guide-3.2.html (none)
0        caching-nameserver-6.2 (none)
0        XFree86-ISO8859-2-1.0 (none)
0        rootfiles        (none)
0        ghostscript-fonts-5.50 (none)
0        kudzu-0.36        (none)
0        wvdial-1.41       (none)
0        mailcap-2.0.6     (none)
0        desktop-backgrounds-1.1 (none)
0        redhat-logos      (none)
0        solemul-1.1       (none)
0        dev-2.7.18        (none)
0        urw-fonts-2.0     (none)
0        users-guide-1.0.72 (none)
0        sgml-common-0.1   (none)
0        setup-2.1.8       (none)
0        jadetex           (none)
0        gnome-audio-1.0.0 (none)
0        specs-po-6.2      (none)
0        gimp-data-extras-1.0.0 (none)
0        docbook-3.1       (none)
0        indexhtml-6.2     (none)

```

```

ansic:      14218806 (80.55%)
cpp:        1326212 (7.51%)
lisp:       565861 (3.21%)
sh:         469950 (2.66%)

```

perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)
python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Total Physical Source Lines of Code (SLOC) = 17652561

Total Estimated Person-Years of Development = 4548.36

Average Programmer Annual Salary = 56286

Overhead Multiplier = 2.4

Total Estimated Cost to Develop = \$ 614421924.71

B.2 Counts of Files For Each Category

There were 181,679 ordinary files in the build directory. The following are counts of the number of files (*not* the SLOC) for each language:

ansic:	52088	(71.92%)
cpp:	8092	(11.17%)
sh:	3381	(4.67%)
asm:	1931	(2.67%)
perl:	1387	(1.92%)
lisp:	1168	(1.61%)
java:	1047	(1.45%)
python:	997	(1.38%)
tcl:	798	(1.10%)
exp:	472	(0.65%)
awk:	285	(0.39%)
objc:	260	(0.36%)
sed:	112	(0.15%)
yacc:	110	(0.15%)
csh:	94	(0.13%)
ada:	92	(0.13%)
lex:	57	(0.08%)
fortran:	50	(0.07%)
pascal:	7	(0.01%)

Total Number of Source Code Files = 72428

In addition, when counting the number of files (not SLOC), some files were identified as source code files but nevertheless were not counted for other reasons (and thus not included in the file counts above). Of these source code files, 5,820 files were identified as duplicating the contents of another file, 817 files were identified as files that had been automatically generated, and 65 files were identified as zero-length files.

B.3 Additional Measures of the Linux Kernel

I also made additional measures of the Linux kernel. This kernel is Linux kernel version 2.2.14 as patched by Red Hat. The Linux kernel's design is reflected in its directory structure. Only 8 lines of source code are in its main directory; the rest are in descendent directories. Counting the physical SLOC in each subdirectory (or its descendents) yielded the following:

BUILD/linux/Documentation/	765
BUILD/linux/arch/	236651
BUILD/linux/configs/	0
BUILD/linux/drivers/	876436
BUILD/linux/fs/	88667
BUILD/linux/ibcs/	16619
BUILD/linux/include/	136982
BUILD/linux/init/	1302
BUILD/linux/ipc/	1757
BUILD/linux/kernel/	7436
BUILD/linux/ksymoops-0.7c/	3271
BUILD/linux/lib/	1300
BUILD/linux/mm/	6771
BUILD/linux/net/	105549
BUILD/linux/pcmcia-cs-3.1.8/	34851
BUILD/linux/scripts/	8357

I separately ran the CodeCount tools on the entire linux operating system kernel. Using the CodeCount definition of C logical lines of code, CodeCount determined that this version of the linux kernel included 673,627 logical SLOC in C. This is obviously much smaller than the 1,462,165 of physical SLOC in C, or the 1,526,722 SLOC when all languages are combined for Linux.

However, this included non-i86 code. To make a more reasonable comparison with the Halloween documents, I needed to ignore non-i386 code.

First, I looked at the linux/arch directory, which contained architecture-specific code. This directory had the following subdirectories (architectures): alpha, arm, i386, m68k, mips, ppc, s390, sparc, sparc64. I then computed the total for all of ``arch'', which was 236651 SLOC, and subtracted out linux/arch/i386 code, which totalled to 26178 SLOC; this gave me a total of non-i386 code in linux/arch as 210473 physical SLOC. I then looked through the ``drivers'' directory to see if there were sets of drivers which were non-i386. I identified the following directories, with the SLOC totals as shown:

linux/drivers/sbus/	22354
linux/drivers/macintosh/	6000
linux/drivers/sgi/	4402
linux/drivers/fc4/	3167
linux/drivers/nubus/	421
linux/drivers/acorn/	11850
linux/drivers/s390/	8653

Driver Total: 56847

Thus, I had a grand total on non-i86 code (including drivers and architecture-specific code) as 267320 physical SLOC. This is, of course, another approximation, since there's certainly other architecture-specific lines, but I believe that is most of it. Running the CodeCount tool on just the C code, once these architectural and driver directories are removed, reveals a logical SLOC of 570,039 of C code.

B.4 Minimum System SLOC

Most of this paper worries about counting an ``entire'' system. However, what's the SLOC size of a ``minimal'' system? Here's an attempt to answer that question.

Red Hat Linux 6.2, CD-ROM #1, file RedHat/base/comps, defines the ``base'' (minimum) Red Hat Linux 6.2 installation as a set of packages. The following are the build directories corresponding to this base (minimum)

installation, along with the SLOC counts (as shown above). Note that this creates a text-only system:

Component	SLOC
anacron-2.1	1143
apmd	3012
ash-linux-0.2	9666
at-3.1.7	3084
authconfig-3.0.3	1075
bash-1.14.7	47067
bc-1.05	17682
bdflush-1.5	261
binutils-2.9.5.0.22	467120
bzip2-0.9.5d	4996
chkconfig-1.1.2	717
console-tools-0.3.3	15522
cpio-2.4.2	7617
cracklib, 2.7	1987
dev-2.7.18	0
diffutils-2.7	10914
dump-0.4b15	10187
e2fsprogs-1.18	28169
ed-0.2	7427
egcs-1.1.2	720112
eject-2.0.2	657
file-3.28	2647
fileutils-4.0p	34768
findutils-4.1	11404
gawk-3.0.4	26363
gd1.3	20078
gdbm-1.8.0	3315
getty_ps-2.0.7j	2631
glibc-2.1.3	415026
gmp-2.0.2	24583
gnupg-1.0.1	54935
gpm-1.18.1	9725
grep-2.4	10013
groff-1.15	70260
gzip-1.2.4a	6306
hdparm-3.6	1229
initscripts-5.00	3929
isapnptools-1.21	5960
kbdconfig-1.9.2.4	368
kernelcfg-0.5	347
kudzu-0.36	0
ldconfig-1999-02-21	874
ld.so-1.9.5	9731
less-346	14039
lilo	7255
linuxconf-1.17r2	104032
logrotate-3.3.2	1525
mailcap-2.0.6	0
mailx-8.1.1	6968
MAKEDEV-2.5.2	1020
man-1.5hl	9801
mingetty-0.9.4	343
mkbootdisk-1.2.5	314
mkinitrd-2.4.1	288
mktemp-1.5	85
modutils-2.3.9	11775
mouseconfig-4.4	830

Estimating Linux's Size

mt-st-0.5b	1361
ncompress-4.2.4	1435
ncurses-5.0	61324
net-tools-1.54	11633
newt-0.50.8	7041
pam-0.72	20433
passwd-0.64.1	1194
pciutils-2.1.5	3855
popt-1.4	1099
procmail-3.14	9927
procps-2.0.6	9961
psmisc	1630
pump-0.7.8	1856
pwdb-0.61	9551
quota-2.00-pre3	3804
raidtools-0.90	2424
readline-2.2.1	14941
redhat-logos	0
rootfiles	0
rpm-3.0.4	39861
sash-3.4	6172
sed-3.02	7740
sendmail-8.9.3	42880
setserial-2.15	742
setup-1.2	67
setup-2.1.8	0
shadow-19990827	25236
sh-utils-2.0	17939
slang	28118
slocate-2.1	1883
stat-1.5	280
sysklogd-1.3-31	4038
sysvinit-2.78	6033
tar-1.13.17	14255
termcap-2.0.8	797
texinfo-4.0	28186
textutils-2.0a	36338
time-1.7	1452
timeconfig-3.0.3	346
tmpwatch-2.2	463
utempter-0.5.2	222
util-linux-2.10f	39160
vim-5.6	113241
vixie-cron-3.0.1	2879
which-2.9	2268
zlib-1.1.3	4087

Thus, the contents of the build directories corresponding to the ``base" (minimum) installation totals to 2,819,334 SLOC.

A few notes are in order about this build directory total:

1. Some of the packages listed by a traditional package list aren't shown here because they don't contain any code. Package "basesystem" is a pseudo-package for dependency purposes. Package redhat-release is just a package for keeping track of the base system's version number. Package "filesystem" contains a directory layout.
2. ntsysv's source is in chkconfig-1.1.2; kernel-utils and kernel-pcmcia-cs are part of "linux". Package shadow-utils is in build directory shadow-19990827. Build directory util-linux includes losetup and mount. "dump" is included to include rmt.
3. Sometimes the build directories contain more code than is necessary to create just the parts for the ``base" system; this is a side-effect of how things are packaged. ``info" is included in the base, so we count all of texinfo. The build directory termcap is counted, because libtermcap is in the base. Possibly most important, gcc (egcs) is

there because libstdc++ is in the base.

4. Sometimes a large component is included in the base, even though most of the time little of its functionality is used. In particular, the mail transfer agent ``sendmail" is in the base, even though for many users most of sendmail's functionality isn't used. However, for this paper's purposes this isn't a problem. After all, even if sendmail's functionality is often underused, clearly that functionality took time to develop and that functionality is available to those who want it.
5. My tools intentionally eliminated duplicates; it may be that a few files aren't counted here because they're considered duplicates of another build directory not included here. I do not expect this factor to materially change the total.
6. Red Hat Linux is not optimized to be a ``small as possible" distribution; their emphasis is on functionality, not small size. A working Linux distribution could include much less code, depending on its intended application. For example, ``linuxconf" simplifies system configuration, but the system can be configured by editing its system configuration files directly, which would reduce the base system's size. This also includes vim, a full-featured text editor - a simpler editor with fewer functions would be smaller as well.

Many people prefer some sort of graphical interface; here is a minimal configuration of a graphical system, adding the X server, a window manager, and a few tools:

Component	SLOC
XFree86-3.3.6	1291745
Xconfigurator-4.3.5	9741
fvwm-2.2.4	69265
X11R6-contrib-3.3.2	18885

These additional graphical components add 1,389,636 SLOC. Due to oddities of the way the initialization system xinitrc is built, it isn't shown here in the total, but xinitrc has so little code that its omission does not significantly affect the total.

Adding these numbers together, we now have a total of 4,208,970 SLOC for a ``minimal graphical system." Many people would want to add more components. For example, this doesn't include a graphical toolkit (necessary for running most graphical applications). We could add gtk+-1.2.6 (a toolkit needed for running GTK+ based applications), adding 138,118 SLOC. This would now total 4,347,088 for a ``basic graphical system," one able to run basic GTK+ applications.

Let's add a web server to the mix. Adding apache_1.3.12 adds only 77,873 SLOC. We now have 4,424,961 physical SLOC for a basic graphical system plus a web server.

We could then add a graphical desktop environment, but there are so many different options and possibilities that trying to identify a ``minimal" system is hard to do without knowing the specific uses intended for the system. Red Hat defines a standard ``GNOME" and ``KDE" desktop, but these are intended to be highly functional (not ``minimal"). Thus, we'll stop here, with a total of 2.8 million physical SLOC for a minimal text-based system, and total of 4.4 million physical SLOC for a basic graphical system plus a web server.

References

[Boehm 1981] Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, Inc. ISBN 0-13-822122-7.

[Dempsey 1999] Dempsey, Bert J., Debra Weiss, Paul Jones, and Jane Greenberg. October 6, 1999. UNC Open Source Research Team. Chapel Hill, NC: University of North Carolina at Chapel Hill. <http://www.ibiblio.org/osrt/developpro.html>.

[DSMC] Defense Systems Management College (DSMC). *Indirect Cost Management Guide: Navigating the Sea of Overhead*. Defense Systems Management College Press, Fort Belvoir, VA 22060-5426. Available as part of the ``Defense Acquisition Deskbook." <http://portal.deskbook.osd.mil/reflib/DTNG/009CM/004/009CM004DOC.HTM>.

[FSF 2000] Free Software Foundation (FSF). *What is Free Software?*. <http://www.gnu.org/philosophy/free-sw.html>.

[Halloween I] Valloppillil, Vinod, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Open Source Software: A (New?) Development Methodology" v1.00. <http://www.opensource.org/halloween/halloween1.html>.

[Halloween II] Valloppillil, Vinod and Josh Cohen, with interleaved commentary by Eric S. Raymond. Aug 11, 1998. "Linux OS Competitive Analysis: The Next Java VM?". v1.00. <http://www.opensource.org/halloween/halloween2.html>

[Kalb 1990] Kalb, George E. "Counting Lines of Code, Confusions, Conclusions, and Recommendations". Briefing to the 3rd Annual REVIC User's Group Conference, January 10-12, 1990. http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf

[Kalb 1996] Kalb, George E. October 16, 1996 "Automated Collection of Software Sizing Data" Briefing to the International Society of Parametric Analysts, Southern California Chapter. <http://sunset.usc.edu/research/CODECOUNT/documents/ispa.pdf>

[Masse 1997] Masse, Roger E. July 8, 1997. *Software Metrics: An Analysis of the Evolution of COCOMO and Function Points*. University of Maryland. <http://www.python.org/~rmasse/papers/software-metrics>.

[Miller 1995] Miller, Barton P., David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. 1995. *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. <http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>.

[NAS 1996] National Academy of Sciences (NAS). 1996. *Statistical Software Engineering*. <http://www.nap.edu/html/statsoft/chap2.html>

[OSI 1999]. Open Source Initiative. 1999. *The Open Source Definition*. <http://www.opensource.org/osd.html>.

[Park 1992] Park, R. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/SEI-92-TR-020. <http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html>

[Perens 1999] Perens, Bruce. January 1999. *Open Sources: Voices from the Open Source Revolution*. "The Open Source Definition". ISBN 1-56592-582-3. <http://www.oreilly.com/catalog/opensources/book/perens.html>

[Raymond 1999] Raymond, Eric S. January 1999. "A Brief History of Hackerdom". *Open Sources: Voices from the Open Source Revolution*. <http://www.oreilly.com/catalog/opensources/book/raymond.html>.

[Schneier 2000] Schneier, Bruce. March 15, 2000. "Software Complexity and Security". *Crypto-Gram*. <http://www.counterpane.com/crypto-gram-0003.html>

[Shankland 2000a] Shankland, Stephen. February 14, 2000. "Linux poses increasing threat to Windows 2000". CNET News.com. <http://news.cnet.com/news/0-1003-200-1549312.html>.

[Shankland 2000b] Shankland, Stephen. August 31, 2000. "Red Hat holds huge Linux lead, rivals growing". CNET News.com. <http://news.cnet.com/news/0-1003-200-2662090.html>

[Stallman 2000] Stallman, Richard. October 13, 2000 "By any other name...". <http://www.anchordesk.co.uk/anchordesk/commentary/columns/0,2415,7106622,00.html>.

[Vaughan-Nichols 1999] Vaughan-Nichols, Steven J. Nov. 1, 1999. Can you Trust this Penguin? ZDnet. <http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html>

[Wheeler 2000a] Wheeler, David A. 2000. *Open Source Software / Free Software References*. http://www.dwheeler.com/oss_fs_refs.html.

[Wheeler 2000b] Wheeler, David A. 2000. *Quantitative Measures for Why You Should Consider Open Source / Free Software*. http://www.dwheeler.com/oss_fs_why.html.

[Zoebelein 1999] Zoebelein. April 1999. <http://leb.net/hzo/ioscount>.

This paper is (C) Copyright 2000 David A. Wheeler. All rights reserved. You may download and print it for your own personal use, and of course you may link to it. When referring to the paper, please refer to it as "Estimating GNU/Linux's Size" by David A. Wheeler, located at <http://www.dwheeler.com/sloc>. Please give credit if you refer to any of its techniques or results.

SLOC	Directory	SLOC-by-Language (Sorted)
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,cs=5
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,cs=9,sed=4
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,cs=15
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,cs=5,sed=2
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,cs=147,sed=123
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,sed=2
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
193916	teTeX-1.0	ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,yacc=1507,awk=522,lex=323,sed=297,asm=139,cs=47,lisp=29
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,cs=848,awk=753,lex=222
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,cs=235,sed=35,lisp=12
133193	kaffe-1.0.5	java=65275,ansic=62125,cpp=3923,perl=972,sh=814,asm=84
131372	jade-1.2.1	cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177
127536	pine4.21	ansic=126678,sh=766,cs=62,perl=30
121878	ImageMagick-4.2.9	ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
119613	lynx2-8-3	ansic=117385,sh=1860,perl=340,cs=28
116951	mc-4.5.42	ansic=114406,sh=1996,perl=345,awk=148,cs=56
116615	gnnumeric-0.48	ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
113272	xlispstat-3-52-17	ansic=91484,lisp=21769,sh=18,cs=1
113241	vim-5.6	ansic=111724,awk=683,sh=469,perl=359,cs=6
109824	php-3.0.15	ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
104032	linuxconf-1.17r2	cpp=93139,perl=4570,sh=2984,java=2741,ansic=598
102674	libgr-2.0.13	ansic=99647,sh=2438,cs=589
100951	lam-6.3.1	ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187,cs=19
99066	krb4-1.0	ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765,yacc=1509,lex=236,awk=33
94637	xlockmore-4.15	ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
93940	kdenetwork	cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
92964	samba-2.0.6	ansic=88308,sh=3557,perl=831,awk=158,cs=110
91213	anaconda-6.2.2	ansic=74303,python=13657,sh=1583,yacc=810,lex=732,perl=128
89959	xscreensaver-3.23	ansic=88488,perl=1070,sh=401
88128	cvs-1.10.7	ansic=68303,sh=17909,perl=902,yacc=826,cs=181,lisp=7
87940	isdn4k-utils	ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
85383	xpdf-0.90	cpp=60427,ansic=21400,sh=3556

81719	inn-2.2.2	ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547,lex=249,tcl=3
80343	kdelibs	cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116,sh=35
79997	WindowMaker-0.61.1	ansic=77924,sh=1483,perl=371,lisp=219
78787	extace-1.2.15	ansic=66571,sh=9322,perl=2894
77873	apache_1.3.12	ansic=69191,sh=6781,perl=1846,cpp=55
75257	xpilot-4.1.0	ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68
73817	w3c-libwww-5.2.8	ansic=64754,sh=4678,cpp=3181,perl=1204
72726	ucd-snmp-4.1.1	ansic=64411,perl=5558,sh=2757
72425	gnome-core-1.0.55	ansic=72230,perl=141,sh=54
71810	jikes	cpp=71452,java=358
70260	groff-1.15	cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397,sh=265,sed=46
69265	fvwm-2.2.4	ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
69246	linux-86	ansic=63328,asm=5276,sh=642
68997	blt2.4g	ansic=58630,tcl=10215,sh=152
68884	squid-2.3.STABLE1	ansic=66305,sh=1570,perl=1009
68560	bash-2.03	ansic=56758,sh=7264,yacc=2808,perl=1730
68453	kdegraphics	cpp=34208,ansic=29347,sh=4898
65722	xntp3-5.93	ansic=60190,perl=3633,sh=1445,awk=417,asm=37
62922	ppp-2.3.11	ansic=61756,sh=996,exp=82,perl=44,csch=44
62137	sgml-tools-1.0.9	cpp=38543,ansic=19185,perl=2866,lex=560,sh=532,lisp=309,awk=142
61688	imap-4.7	ansic=61628,sh=60
61324	ncurses-5.0	ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103,sed=100
60429	kdesupport	ansic=42421,cpp=17810,sh=173,awk=13,csch=12
60302	openldap-1.2.9	ansic=58078,sh=1393,perl=630,python=201
57217	xfig.3.2.3-beta-1	ansic=57212,csch=5
56093	lsnf_4.47	ansic=50268,sh=4753,perl=856,awk=214,asm=2
55667	uucp-1.06.1	ansic=52078,sh=3400,perl=189
54935	gnupg-1.0.1	ansic=48884,asm=4586,sh=1465
54603	glade-0.5.5	ansic=49545,sh=5058
54431	svglib-1.4.1	ansic=53725,asm=630,perl=54,sh=22
53141	AfterStep-1.8.0	ansic=50898,perl=1168,sh=842,cpp=233
52808	kdeutils	cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
52574	nmh-1.0.3	ansic=50698,sh=1785,awk=74,sed=17
51813	freetype-1.3.1	ansic=48929,sh=2467,cpp=351,csch=53,perl=13
51592	enlightenment-0.15.5	ansic=51569,sh=23
50970	cdrecord-1.8	ansic=48595,sh=2177,perl=194,sed=4
49370	tin-1.4.2	ansic=47763,sh=908,yacc=699
49325	imlib-1.9.7	ansic=49260,sh=65
48223	kdemultimedia	ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
47067	bash-1.14.7	ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
46312	tcsh-6.09.00	ansic=43544,sh=921,lisp=669,perl=593,csch=585
46159	unzip-5.40	ansic=40977,cpp=3778,asm=1271,sh=133
45811	mutt-1.0.1	ansic=45574,sh=237
45589	am-utils-6.0.3	ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
45485	guile-1.3	ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,csch=50
45378	gnuplot-3.7.1	ansic=43276,lisp=661,asm=539,objc=387,csch=297,perl=138,sh=80
44323	mgetty-1.1.21	ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
42880	sendmail-8.9.3	ansic=40364,perl=1737,sh=779
42746	elm2.5.3	ansic=32931,sh=9774,awk=41
41388	p2c-1.22	ansic=38788,pascal=2499,perl=101
41205	gnome-games-1.0.51	ansic=31191,lisp=6966,cpp=3048
39861	rpm-3.0.4	ansic=36994,sh=1505,perl=1355,python=7
39160	util-linux-2.10f	ansic=38627,sh=351,perl=65,csch=62,sed=55
38927	xmms-1.0.1	ansic=38366,asm=398,sh=163
38548	ORBit-0.5.0	ansic=35656,yacc=1750,sh=776,lex=366
38453	zsh-3.0.7	ansic=36208,sh=1763,perl=331,awk=145,sed=6
37515	ircii-4.4	ansic=36647,sh=852,lex=16
37360	tiff-v3.5.4	ansic=32734,sh=4054,cpp=572
36338	textutils-2.0a	ansic=18949,sh=16111,perl=1218,sed=60

36243	exmh-2.1.1	tcl=35844,perl=316,sh=49,exp=34
36239	xllamp-0.9-alpha3	ansic=31686,sh=4200,asm=353
35812	xloadimage.4.1	ansic=35705,sh=107
35554	zip-2.3	ansic=32108,asm=3446
35397	gtk-engines-0.10	ansic=20636,sh=14761
35136	php-2.0.1	ansic=33991,sh=1056,awk=89
34882	pmake	ansic=34599,sh=184,awk=58,sed=41
34772	xpuzzles-5.4.1	ansic=34772
34768	fileutils-4.0p	ansic=31324,sh=2042,yacc=841,perl=561
33203	strace-4.2	ansic=30891,sh=1988,perl=280,lisp=44
32767	trn-3.6	ansic=25264,sh=6843,yacc=660
32277	pilot-link.0.9.3	ansic=26513,java=2162,cpp=1689,perl=971,yacc=660, python=268,tcl=14
31994	korganizer	cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
31174	ncftp-3.0beta21	ansic=30347,cpp=595,sh=232
30438	gnome-pim-1.0.55	ansic=28665,yacc=1773
30122	scheme-3.2	lisp=19483,ansic=10515,sh=124
30061	tcpdump-3.4	ansic=29208,yacc=236,sh=211,lex=206,awk=184,csch=16
29730	screen-3.9.5	ansic=28156,sh=1574
29315	jed	ansic=29315
29091	xchat-1.4.0	ansic=28894,perl=121,python=53,sh=23
28897	ncpfs-2.2.0.17	ansic=28689,sh=182,tcl=26
28449	slrn-0.9.6.2	ansic=28438,sh=11
28261	xfishtank-2.1tp	ansic=28261
28186	texinfo-4.0	ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
28169	e2fsprogs-1.18	ansic=27250,awk=437,sh=339,sed=121,perl=22
28118	slang	ansic=28118
27860	kdegames	cpp=27507,ansic=340,sh=13
27117	librep-0.10	ansic=19381,lisp=5385,sh=2351
27040	mikmod-3.1.6	ansic=26975,sh=55,awk=10
27022	x3270-3.1.1	ansic=26456,sh=478,exp=88
26673	lout-3.17	ansic=26673
26608	Xaw3d-1.3	ansic=26235,yacc=247,lex=126
26363	gawk-3.0.4	ansic=19871,awk=2519,yacc=2046,sh=1927
26146	libxml-1.8.6	ansic=26069,sh=77
25994	xrn-9.02	ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31, csch=13
25915	gv-3.5.8	ansic=25821,sh=94
25479	xpaint	ansic=25456,sh=23
25236	shadow-19990827	ansic=23464,sh=883,yacc=856,perl=33
24910	kdeadmin	cpp=19919,sh=3936,perl=1055
24773	pdksh-5.2.14	ansic=23599,perl=945,sh=189,sed=40
24583	gmp-2.0.2	ansic=17888,asm=5252,sh=1443
24387	mars_nwe	ansic=24158,sh=229
24270	gnome-python-1.0.51	python=14331,ansic=9791,sh=148
23838	kterm-6.2.0	ansic=23838
23666	enscript-1.6.1	ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
22373	sawmill-0.24	ansic=11038,lisp=8172,sh=3163
22279	make-3.78.1	ansic=19287,sh=2029,perl=963
22011	libpng-1.0.5	ansic=22011
21593	xboard-4.0.5	ansic=20640,lex=904,sh=41,csch=5,sed=3
21010	netkit-telnet-0.16	ansic=14796,cpp=6214
20433	pam-0.72	ansic=18936,yacc=634,sh=482,perl=321,lex=60
20125	ical-2.2	cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
20078	gd1.3	ansic=19946,perl=132
19971	wu-ftpd-2.6.0	ansic=17572,yacc=1774,sh=421,perl=204
19500	gnome-utils-1.0.50	ansic=18099,yacc=824,lisp=577
19065	joe	ansic=18841,asm=224
18885	X11R6-contrib-3.3.2	ansic=18616,lex=161,yacc=97,sh=11
18835	glib-1.2.6	ansic=18702,sh=133
18151	git-4.3.19	ansic=16166,sh=1985
18020	xboing	ansic=18006,sh=14
17939	sh-utils-2.0	ansic=13366,sh=3027,yacc=871,perl=675
17765	mttools-3.9.6	ansic=16155,sh=1602,sed=8
17750	gettext-0.10.35	ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9

17682	bc-1.05	ansic=9186,sh=7236,yacc=967,lex=293
17271	fetchmail-5.3.1	ansic=13441,python=1490,sh=1246,yacc=411,perl=321,lex=238,awk=124
17259	sox-12.16	ansic=16659,sh=600
16785	control-center-1.0.51	ansic=16659,sh=126
16266	dhcp-2.0	ansic=15328,sh=938
15967	SVGATextMode-1.9-src	ansic=15079,yacc=340,sh=294,lex=227,sed=15,asm=12
15868	kpilot-3.1b9	cpp=8613,ansic=5640,yacc=1615
15851	taper-6.9a	ansic=15851
15819	mpg123-0.59r	ansic=14900,asm=919
15691	transfig.3.2.1	ansic=15643,sh=38,csch=10
15638	mod_perl-1.21	perl=10278,ansic=5124,sh=236
15522	console-tools-0.3.3	ansic=13335,yacc=986,sh=800,lex=291,perl=110
15456	rpm2html-1.2	ansic=15334,perl=122
15143	gnotepad+-1.1.4	ansic=15143
15108	GXedit1.23	ansic=15019,sh=89
15087	mm2.7	ansic=8044,csch=6924,sh=119
14941	readline-2.2.1	ansic=11375,sh=1890,perl=1676
14912	ispell-3.1	ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385,csch=221,sh=157,perl=85,sed=15
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,csch=29
14774	flex-2.5.4	ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12
14587	multimedia	ansic=14577,sh=10
14516	libgtop-1.0.6	ansic=13768,perl=653,sh=64,asm=31
14427	mawk-1.2.2	ansic=12714,yacc=994,awk=629,sh=90
14363	automake-1.4	perl=10622,sh=3337,ansic=404
14350	rsync-2.4.1	ansic=13986,perl=179,sh=126,awk=59
14299	nfs-utils-0.1.6	ansic=14107,sh=165,perl=27
14269	rcs-5.7	ansic=12209,sh=2060
14255	tar-1.13.17	ansic=13014,lisp=592,sh=538,perl=111
14105	wmakerconf-2.1	ansic=13620,perl=348,sh=137
14039	less-346	ansic=14032,awk=7
13779	rxvt-2.6.1	ansic=13779
13586	wget-1.5.3	ansic=13509,perl=54,sh=23
13504	rp3-1.0.7	cpp=10416,ansic=2957,sh=131
13241	iproute2	ansic=12139,sh=1002,perl=100
13100	silos-0.9.8	ansic=10485,asm=2615
12657	macutils	ansic=12657
12639	libungif-4.1.0	ansic=12381,sh=204,perl=54
12633	minicom-1.83.0	ansic=12503,sh=130
12593	audiofile-0.1.9	sh=6440,ansic=6153
12463	gnome-objc-1.0.2	objc=12365,sh=86,ansic=12
12313	jpeg-6a	ansic=12313
12124	ypserv-1.3.9	ansic=11622,sh=460,perl=42
11790	lrzsz-0.12.20	ansic=9512,sh=1263,exp=1015
11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
11633	net-tools-1.54	ansic=11531,sh=102
11404	findutils-4.1	ansic=11160,sh=173,exp=71
11299	xmorph-1999dec12	ansic=10783,tcl=516
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
10914	diffutils-2.7	ansic=10914
10404	gnorpm-0.9	ansic=10404
10271	gqview-0.7.0	ansic=10271
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
10187	dump-0.4b15	ansic=9422,sh=760,sed=5
10088	piranha	ansic=10048,sh=40
10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
9961	procps-2.0.6	ansic=9959,sh=2
9942	xpat2-1.04	ansic=9942
9927	procmail-3.14	ansic=8090,sh=1837
9873	nss_ldap-105	ansic=9784,perl=89
9801	man-1.5hl	ansic=7377,sh=1802,perl=317,awk=305
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6

9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
9699	bison-1.28	ansic=9650,sh=49
9666	ash-linux-0.2	ansic=9445,sh=221
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
9551	pwdb-0.61	ansic=9488,sh=63
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
9263	ctags-3.4	ansic=9240,sh=23
9138	gftp-2.0.6a	ansic=9138
8939	mkisofs-1.12b5	ansic=8939
8766	pxe-linux	cpp=4463,ansic=3622,asm=681
8572	psgml-1.2.1	lisp=8572
8540	xxgdb-1.12	ansic=8540
8491	gtop-1.0.5	ansic=8151,cpp=340
8356	gedit-0.6.1	ansic=8225,sh=131
8303	dip-3.3.7o	ansic=8207,sh=96
7859	libglade-0.11	ansic=5898,sh=1809,python=152
7826	xpm-3.4k	ansic=7750,sh=39,cpp=37
7740	sed-3.02	ansic=7301,sed=359,sh=80
7617	cpio-2.4.2	ansic=7598,sh=19
7615	esound-0.2.17	ansic=7387,sh=142,cs=86
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
7427	ed-0.2	ansic=7263,sh=164
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,cpp=3
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
7095	xgammon-0.98	ansic=6506,lex=589
7041	newt-0.50.8	ansic=6526,python=515
7030	ee-0.3.11	ansic=7007,sh=23
6976	aboot-0.5	ansic=6680,asm=296
6968	mailx-8.1.1	ansic=6963,sh=5
6877	lpr	ansic=6842,sh=35
6827	gnome-media-1.0.51	ansic=6827
6646	iputils	ansic=6646
6611	patch-2.5	ansic=6561,sed=50
6592	xosview-1.7.1	cpp=6205,ansic=367,awk=20
6550	byacc-1.9	ansic=5520,yacc=1030
6496	pidentd-3.0.10	ansic=6475,sh=21
6391	m4-1.4	ansic=5993,lisp=243,sh=155
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
6234	awesfx-0.4.3a	ansic=6234
6172	sash-3.4	ansic=6172
6116	ls1k	ansic=5325,sh=791
6090	joystick-1.2.15	ansic=6086,sh=4
6072	kdoc	perl=6010,sh=45,cpp=17
6043	irda-utils-0.9.10	ansic=5697,sh=263,perl=83
6033	sysvinit-2.78	ansic=5256,sh=777
6025	pnm2ppa	ansic=5708,sh=317
6021	rpmfind-1.4	ansic=6021
5981	indent-2.2.5	ansic=5958,sh=23
5975	ytalk-3.1	ansic=5975
5960	isapnptools-1.21	ansic=4394,yacc=1383,perl=123,sh=60
5744	gdm-2.0beta2	ansic=5632,sh=112
5594	isdns-config	cpp=3058,sh=2228,perl=308
5526	efax-0.9	ansic=4570,sh=956
5383	acct-6.3.2	ansic=5016,cpp=287,sh=80
5115	libtool-1.3.4	sh=3374,ansic=1741
5111	netkit-ftp-0.16	ansic=5111
4996	bzip2-0.9.5d	ansic=4996
4895	xcpustate-2.5	ansic=4895
4792	libelf-0.6.4	ansic=3310,sh=1482
4780	make-3.78.1_pvm-0.5	ansic=4780
4542	gpgp-0.4	ansic=4441,sh=101
4430	gperf-2.7	cpp=2947,exp=745,ansic=695,sh=43
4367	aumix-1.30.1	ansic=4095,sh=179,sed=93
4087	zlib-1.1.3	ansic=2815,asm=712,cpp=560

4038	syslogd-1.3-31	ansic=3741,perl=158,sh=139
4024	rep-gtk-0.8	ansic=2905,lisp=971,sh=148
3962	netkit-timed-0.16	ansic=3962
3929	initscripts-5.00	sh=2035,ansic=1866,csch=28
3896	ltraced-0.3.10	ansic=2986,sh=854,awk=56
3885	phhttpd-0.1.0	ansic=3859,sh=26
3860	xdaliclock-2.18	ansic=3837,sh=23
3855	pciutils-2.1.5	ansic=3800,sh=55
3804	quota-2.00-pre3	ansic=3795,sh=9
3675	dosfstools-2.2	ansic=3675
3654	tcp_wrappers_7.6	ansic=3654
3651	ipchains-1.3.9	ansic=2767,sh=884
3625	autofs-3.1.4	ansic=2862,sh=763
3588	netkit-rsh-0.16	ansic=3588
3438	yp-tools-2.4	ansic=3415,sh=23
3433	dialog-0.6	ansic=2834,perl=349,sh=250
3415	ext2ed-0.1	ansic=3415
3315	gdbm-1.8.0	ansic=3290,cpp=25
3245	ypbind-3.3	ansic=1793,sh=1452
3219	playmidi-2.4	ansic=3217,sed=2
3096	xtrojka123	ansic=3087,sh=9
3084	at-3.1.7	ansic=1442,sh=1196,yacc=362,lex=84
3051	dhcpcd-1.3.18-pl3	ansic=2771,sh=280
3012	apmd	ansic=2617,sh=395
2883	netkit-base-0.16	ansic=2883
2879	vixie-cron-3.0.1	ansic=2866,sh=13
2835	gkermid-1.0	ansic=2835
2810	kdetoys	cpp=2618,ansic=192
2791	xjewel-1.6	ansic=2791
2773	mpage-2.4	ansic=2704,sh=69
2758	autoconf-2.13	sh=2226,perl=283,exp=167,ansic=82
2705	autorun-2.61	sh=1985,cpp=720
2661	cdp-0.33	ansic=2661
2647	file-3.28	ansic=2601,perl=46
2645	libghttp-1.0.4	ansic=2645
2631	getty_ps-2.0.7j	ansic=2631
2597	pythonlib-1.23	python=2597
2580	magicdev-0.2.7	ansic=2580
2531	gnome-kermiberos-0.2	ansic=2531
2490	sndconfig-0.43	ansic=2490
2486	bug-buddy-0.7	ansic=2486
2459	usermode-1.20	ansic=2459
2455	fnlib-0.4	ansic=2432,sh=23
2447	sliplogin-2.1.1	ansic=2256,sh=143,perl=48
2424	raidtools-0.90	ansic=2418,sh=6
2423	netkit-routed-0.16	ansic=2423
2407	nc	ansic=1670,sh=737
2324	up2date-1.13	python=2324
2270	memprof-0.3.0	ansic=2270
2268	which-2.9	ansic=1398,sh=870
2200	printtool	tcl=2200
2163	gnome-linuxconf-0.25	ansic=2163
2141	unarj-2.43	ansic=2141
2065	units-1.55	ansic=1963,perl=102
2048	netkit-ntalk-0.16	ansic=2048
1987	cracklib,2.7	ansic=1919,perl=46,sh=22
1984	cleanfeed-0.95.7b	perl=1984
1977	wmconfig-0.9.8	ansic=1941,sh=36
1941	isicom	ansic=1898,sh=43
1883	slocate-2.1	ansic=1802,sh=81
1857	netkit-rusers-0.16	ansic=1857
1856	pump-0.7.8	ansic=1856
1842	cdecl-2.5	ansic=1002,yacc=765,lex=75
1765	fbset-2.1	ansic=1401,yacc=130,lex=121,perl=113
1653	adjtimex-1.9	ansic=1653

1634	netcfg-2.25	python=1632,sh=2
1630	psmisc	ansic=1624,sh=6
1621	urlview-0.7	ansic=1515,sh=106
1604	fortune-mod-9708	ansic=1604
1531	netkit-tftp-0.16	ansic=1531
1525	logrotate-3.3.2	ansic=1524,sh=1
1473	traceroute-1.4a5	ansic=1436,awk=37
1452	time-1.7	ansic=1395,sh=57
1435	ncompress-4.2.4	ansic=1435
1361	mt-st-0.5b	ansic=1361
1290	cxhextris	ansic=1290
1280	pam_krb5-1	ansic=1280
1272	bsd-finger-0.16	ansic=1272
1229	hdparm-3.6	ansic=1229
1226	procinfo-17	ansic=1145,perl=81
1194	passwd-0.64.1	ansic=1194
1182	auth_ldap-1.4.0	ansic=1182
1146	prtconf-1.3	ansic=1146
1143	anacron-2.1	ansic=1143
1129	xbill-2.0	cpp=1129
1099	popt-1.4	ansic=1039,sh=60
1088	nag	perl=1088
1076	stylesheets-0.13rh	perl=888,sh=188
1075	authconfig-3.0.3	ansic=1075
1049	kpppload-1.04	cpp=1044,sh=5
1020	MAKEDEV-2.5.2	sh=1020
1013	trojka	ansic=1013
987	xmailbox-2.5	ansic=987
967	netkit-rwho-0.16	ansic=967
953	switchdesk-2.1	ansic=314,perl=287,cpp=233,sh=119
897	portmap_4	ansic=897
874	ldconfig-1999-02-21	ansic=874
844	jpeg-6b	sh=844
834	ElectricFence-2.1	ansic=834
830	mouseconfig-4.4	ansic=830
816	rpmlint-0.8	python=813,sh=3
809	kdpms-0.2.8	cpp=809
797	termcap-2.0.8	ansic=797
787	xsysinfo-1.7	ansic=787
770	giftrans-1.12.2	ansic=770
742	setserial-2.15	ansic=742
728	tree-1.2	ansic=728
717	chkconfig-1.1.2	ansic=717
682	lpg	perl=682
657	eject-2.0.2	ansic=657
616	diffstat-1.27	ansic=616
592	netscape-4.72	sh=592
585	usernet-1.0.9	ansic=585
549	genromfs-0.3	ansic=549
548	tksysv-1.1	tcl=526,sh=22
537	minlabel-1.2	ansic=537
506	netkit-bootparamd-0.16	ansic=506
497	locale_config-0.2	ansic=497
491	helptool-2.4	perl=288,tcl=203
480	elftoaout-2.2	ansic=480
463	tmpwatch-2.2	ansic=311,sh=152
445	rhs-printfilters-1.63	sh=443,ansic=2
441	audiocntl	ansic=441
404	control-panel-3.13	ansic=319,tcl=85
368	kbdconfig-1.9.2.4	ansic=368
368	vlock-1.3	ansic=368
367	timetool-2.7.3	tcl=367
347	kernelcfg-0.5	python=341,sh=6
346	timeconfig-3.0.3	ansic=318,sh=28
343	mingetty-0.9.4	ansic=343

343	chkfontpath-1.7	ansic=343
332	ethtool-1.0	ansic=332
314	mkbootdisk-1.2.5	sh=314
302	symlinks-1.2	ansic=302
301	xsri-1.0	ansic=301
294	netkit-rwall-0.16	ansic=294
290	biff+comsat-0.16	ansic=290
288	mkinitrd-2.4.1	sh=288
280	stat-1.5	ansic=280
265	sysreport-1.0	sh=265
261	bdflush-1.5	ansic=202,asm=59
255	ipvsadm-1.1	ansic=255
255	sag-0.6-html	perl=255
245	man-pages-1.28	sh=244,sed=1
240	open-1.4	ansic=240
236	xtoolwait-1.2	ansic=236
222	utempter-0.5.2	ansic=222
222	mkkickstart-2.1	sh=222
221	hellas	sh=179,perl=42
213	rhmask	ansic=213
159	quickstrip-1.1	ansic=159
132	rdate-1.0	ansic=132
131	statserial-1.1	ansic=121,sh=10
107	fwhois-1.00	ansic=107
85	mktemp-1.5	ansic=85
82	modemtool-1.21	python=73,sh=9
67	setup-1.2	ansic=67
56	shaper	ansic=56
52	sparc32-1.1	ansic=52
47	intimed-1.10	ansic=47
23	locale-ja-9	sh=23
16	AnotherLevel-1.0.1	sh=16
11	words-2	sh=11
7	trXFree86-2.1.2	tcl=7
0	install-guide-3.2.html	(none)
0	caching-nameserver-6.2	(none)
0	XFree86-ISO8859-2-1.0	(none)
0	rootfiles	(none)
0	ghostscript-fonts-5.50	(none)
0	kudzu-0.36	(none)
0	wvdial-1.41	(none)
0	mailcap-2.0.6	(none)
0	desktop-backgrounds-1.1	(none)
0	redhat-logos	(none)
0	solemul-1.1	(none)
0	dev-2.7.18	(none)
0	urw-fonts-2.0	(none)
0	users-guide-1.0.72	(none)
0	sgml-common-0.1	(none)
0	setup-2.1.8	(none)
0	jadetex	(none)
0	gnome-audio-1.0.0	(none)
0	specspo-6.2	(none)
0	gimp-data-extras-1.0.0	(none)
0	docbook-3.1	(none)
0	indexhtml-6.2	(none)

ansic:	14218806	(80.55%)
cpp:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)

python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = \$ 614421924.71

#Files	Directory	#Files-by-Language (Sorted)
9340	teTeX-1.0	not=5733,unknown=2510,ansic=694,dup=239,sh=65,cpp=40, auto=23,perl=12,awk=7,sed=4,csch=3,yacc=2,pascal=2,zero=2,lex=2,lisp=1,asm=1
8183	XFree86-3.3.6	ansic=4129,unknown=2422,not=1362,sh=76,asm=75,tcl=56, cpp=31,awk=8,yacc=5,perl=5,dup=4,lex=3,sed=3,auto=3,csch=1
6251	glibc-2.1.3	ansic=4396,unknown=762,asm=664,not=223,dup=116,auto=30, sh=25,awk=20,cpp=6,sed=4,perl=4,csch=1
5735	egcs-1.1.2	ansic=2302,cpp=2283,unknown=638,not=196,dup=92,sh=49, fortran=47,exp=47,asm=42,auto=13,lisp=8,objc=7,sed=6,yacc=4,perl=1
5722	linuxconf-1.17r2	not=3793,unknown=1218,cpp=530,sh=94,perl=43,java=19, ansic=15,dup=7,zero=3
5633	linux	ansic=4376,not=528,unknown=300,asm=249,dup=94,auto=33, sh=28,zero=11,perl=6,awk=3,tcl=2,lex=1,yacc=1,sed=1
4854	kdebase	not=3332,unknown=848,cpp=535,ansic=72,sh=32,dup=23, auto=6,perl=6
3769	man-pages-1.28	not=3687,unknown=80,sh=1,sed=1
3727	gdb-19991004	ansic=1553,unknown=757,dup=631,not=356,exp=213,asm=81, auto=61,cpp=40,sh=19,sed=8,yacc=4,awk=2,fortran=1,lisp=1
3167	tcltk-8.0.5	not=970,unknown=968,ansic=561,tcl=445,sh=125,exp=46, dup=24,auto=22,zero=2,awk=2,perl=1,yacc=1
2864	qt-2.1.0-beta1	not=1283,unknown=810,cpp=636,ansic=102,dup=22,auto=3, sh=3,perl=3,yacc=1,lex=1
2593	stylesheets-0.13rh	not=2126,unknown=460,sh=5,perl=2
2557	kdenetwork	not=1321,unknown=553,cpp=519,ansic=41,perl=39,dup=37, auto=35,sh=9,zero=2,tcl=1
2441	binutils-2.9.5.0.22	ansic=931,unknown=612,asm=470,sh=143,not=139, exp=74,dup=18,sed=16,auto=10,yacc=9,cpp=9,lex=6,lisp=1,zero=1,awk=1,perl=1
2215	kdegames	not=1431,unknown=522,cpp=242,dup=15,ansic=3,auto=1, sh=1
2179	postgresql-6.5.3	ansic=839,unknown=662,not=493,sh=59,java=49,cpp=19, tcl=19,python=9,perl=9,lex=5,yacc=4,dup=4,asm=3,auto=2,zero=1,sed=1,csch=1
2160	krb5-1.1.1	ansic=1175,not=419,unknown=383,exp=59,sh=48,auto=31, dup=19,perl=11,yacc=5,sed=4,awk=3,python=1,csch=1,lex=1
2137	emacs-20.5	not=678,lisp=614,ansic=465,unknown=342,dup=18,sh=8, zero=5,auto=2,asm=2,sed=1,csch=1,perl=1
2127	php-3.0.15	not=1389,ansic=356,unknown=346,sh=13,auto=9,yacc=4, perl=4,dup=3,cpp=2,awk=1
1920	AfterStep-APPS-20000124	not=983,ansic=403,unknown=385,dup=67,sh=35, auto=25,cpp=19,perl=3
1894	kdeutils	not=1187,unknown=324,cpp=233,ansic=94,dup=31,auto=15, sh=7,awk=2,sed=1
1735	kaffe-1.0.5	java=918,ansic=412,unknown=203,not=132,sh=32,cpp=21, dup=10,auto=3,perl=3,asm=1
1706	lam-6.3.1	ansic=1016,not=379,unknown=146,cpp=138,sh=16,dup=4, auto=3,fortran=2,perl=1,csch=1
1681	bind-8.2.2_P5	ansic=765,unknown=354,not=322,dup=90,sh=76,perl=34, cpp=16,awk=11,zero=5,auto=3,yacc=2,csch=2,lex=1
1642	Python-1.5.2	python=784,not=290,ansic=247,unknown=185,dup=107, sh=22,lisp=3,auto=2,sed=1,perl=1
1433	kdegraphics	not=730,unknown=242,cpp=236,ansic=187,dup=20,sh=12, auto=6
1383	gnome-libs-1.0.55	not=809,ansic=410,unknown=113,dup=33,sh=6,auto=5, perl=3,lisp=2,awk=2
1369	textutils-2.0a	unknown=1038,not=162,dup=104,ansic=35,sh=15,perl=11, sed=2,auto=2
1323	gnome-core-1.0.55	not=788,ansic=290,unknown=206,dup=33,sh=3,auto=2, perl=1
1314	gnumeric-0.48	not=744,ansic=360,unknown=147,dup=52,auto=4,lisp=2, perl=2,python=1,sh=1,yacc=1
1311	Mesa	ansic=670,unknown=298,not=219,cpp=51,sh=36,asm=25, dup=7,zero=3,auto=2

1307	gs5.50	ansic=886,unknown=203,not=171,sh=26,cpp=10,dup=5,asm=3,perl=1,lisp=1,auto=1
1223	kdemultimedia	not=705,unknown=269,cpp=139,ansic=83,dup=12,auto=5,tcl=4,sh=3,perl=2,awk=1
1211	anaconda-6.2.2	not=691,ansic=325,python=70,unknown=67,dup=33,sh=15,perl=3,yacc=2,lex=2,zero=2,auto=1
1200	gimp-1.0.4	ansic=491,not=369,unknown=237,lisp=90,sh=6,dup=6,auto=1
1134	perl5.005_03	perl=549,unknown=295,ansic=151,sh=88,not=39,dup=9,yacc=2,lisp=1
1105	w3c-libwww-5.2.8	not=537,ansic=364,unknown=124,cpp=44,dup=20,perl=9,sh=5,zero=1,auto=1
1044	squid-2.3.STABLE1	unknown=708,ansic=219,not=88,perl=18,sh=9,auto=1,dup=1
1028	pine4.21	unknown=402,dup=393,ansic=153,not=64,sh=13,csch=2,perl=1
995	krb4-1.0	ansic=532,not=225,unknown=167,cpp=33,perl=18,sh=10,yacc=3,lex=2,asm=2,auto=1,awk=1,dup=1
967	libgr-2.0.13	ansic=410,not=315,unknown=138,dup=86,sh=11,csch=4,auto=3
956	kdelibs	not=361,cpp=349,unknown=156,dup=39,ansic=17,auto=15,perl=15,sh=2,lex=1,yacc=1
872	xfig.3.2.3-beta-1	not=684,ansic=175,unknown=12,csch=1
857	AfterStep-1.8.0	unknown=344,not=329,ansic=158,sh=15,perl=5,auto=4,cpp=2
847	WindowMaker-0.61.1	not=394,ansic=220,unknown=205,sh=15,dup=8,auto=2,perl=2,lisp=1
832	gnome-games-1.0.51	not=447,ansic=150,unknown=132,cpp=41,lisp=33,dup=27,auto=2
824	isd4k-utils	not=297,ansic=272,unknown=127,dup=40,sh=27,auto=25,cpp=20,perl=14,tcl=2
817	mc-4.5.42	ansic=335,not=287,unknown=118,sh=32,dup=28,auto=6,perl=5,zero=3,csch=2,awk=1
811	gtk+-1.2.6	not=372,ansic=316,unknown=98,dup=9,sh=6,perl=4,awk=3,auto=2,lisp=1
806	ncurses-5.0	ansic=289,not=210,unknown=164,ada=92,sh=22,cpp=16,awk=6,auto=3,perl=2,dup=1,sed=1
793	console-tools-0.3.3	unknown=539,not=125,ansic=79,dup=27,sh=14,perl=4,auto=3,lex=1,yacc=1
783	sgml-tools-1.0.9	dup=254,unknown=212,not=131,ansic=78,cpp=73,perl=22,sh=4,auto=4,lex=2,lisp=2,awk=1
780	xntp3-5.93	unknown=365,ansic=225,not=131,sh=19,dup=12,perl=10,awk=9,asm=5,auto=4
744	kdeadmin	not=429,cpp=156,unknown=139,dup=7,auto=5,sh=5,perl=3
729	apache_1.3.12	not=265,ansic=211,unknown=209,sh=27,perl=13,dup=3,cpp=1
715	bash-2.03	unknown=322,ansic=201,sh=78,not=64,dup=43,auto=5,perl=1,yacc=1
712	linux-86	ansic=419,unknown=150,not=84,asm=31,dup=14,sh=12,zero=2
706	ircii-4.4	unknown=588,ansic=97,not=15,sh=4,auto=1,lex=1
682	openldap-1.2.9	ansic=313,not=174,unknown=151,sh=22,perl=13,dup=6,auto=2,python=1
676	xlispstat-3-52-17	ansic=231,not=149,unknown=147,lisp=137,auto=5,dup=4,sh=2,csch=1
657	jade-1.2.1	cpp=401,unknown=133,not=73,ansic=39,dup=4,sh=3,perl=3,sed=1
652	samba-2.0.6	ansic=216,not=173,unknown=133,sh=62,dup=48,perl=13,awk=4,csch=2,auto=1
639	vim-5.6	unknown=332,not=178,ansic=76,dup=32,sh=11,perl=4,awk=3,auto=2,csch=1
631	kdesupport	not=201,ansic=141,unknown=119,cpp=86,dup=75,sh=7,csch=1,awk=1
606	ucd-snmp-4.1.1	ansic=293,not=147,unknown=88,sh=42,perl=33,dup=2,auto=1

605	fvwm-2.2.4	not=321,ansic=189,unknown=53,cpp=15,perl=10,sh=9, dup=5,auto=1,lex=1,yacc=1
604	groff-1.15	unknown=343,cpp=122,not=74,ansic=39,sh=9,sed=4,auto=3, yacc=3,dup=3,perl=2,asm=1,zero=1
565	libxml-1.8.6	unknown=457,not=60,ansic=39,dup=6,sh=2,auto=1
553	cdrecord-1.8	unknown=266,ansic=199,not=65,sh=16,auto=3,dup=2, sed=1,perl=1
534	XFree86-ISO8859-2-1.0	not=477,unknown=57
533	e2fsprogs-1.18	not=184,ansic=182,unknown=136,sh=11,zero=8,awk=4, dup=3,sed=2,auto=2,perl=1
530	ImageMagick-4.2.9	not=205,ansic=141,unknown=95,perl=42,cpp=29,sh=11, dup=4,auto=2,tcl=1
528	libgtop-1.0.6	ansic=284,not=124,dup=60,unknown=46,auto=6,perl=4, sh=3,asm=1
526	sh-utils-2.0	unknown=219,not=147,dup=86,ansic=57,sh=10,perl=4, auto=2,yacc=1
525	install-guide-3.2.html	not=521,unknown=4
520	enlightenment-0.15.5	not=332,ansic=92,unknown=90,dup=2,zero=2,sh=1, auto=1
503	mgetty-1.1.21	ansic=195,unknown=136,not=90,sh=45,perl=30,tcl=3, lisp=3,dup=1
501	imap-4.7	ansic=292,dup=100,unknown=66,not=39,sh=4
496	svglib-1.4.1	not=258,ansic=170,unknown=59,sh=4,asm=2,dup=2,perl=1
493	inn-2.2.2	ansic=197,not=145,unknown=56,sh=50,perl=37,auto=3, yacc=2,lex=1,tcl=1,awk=1
491	xlockmore-4.15	not=220,ansic=179,unknown=53,cpp=18,sh=7,perl=5, tcl=3,auto=2,dup=2,java=2
488	fileutils-4.0p	ansic=179,not=163,unknown=69,sh=50,dup=21,perl=3, auto=2,yacc=1
477	gated-3-5-11	ansic=206,unknown=122,not=121,sh=18,awk=4,csch=2, yacc=1,sed=1,lisp=1,lex=1
474	lout-3.17	unknown=403,ansic=50,not=21
468	xscreensaver-3.23	ansic=211,not=191,unknown=42,dup=17,sh=3,auto=3, perl=1
462	gmp-2.0.2	ansic=249,asm=92,dup=70,not=28,unknown=20,sh=2,auto=1
461	freetype-1.3.1	ansic=181,unknown=162,not=97,cpp=6,auto=5,dup=4, sh=3,csch=2,perl=1
459	ispell-3.1	unknown=259,not=144,ansic=25,sh=7,dup=7,lisp=5,perl=4, sed=3,csch=2,cpp=1,objc=1,yacc=1
447	lynx2-8-3	ansic=222,unknown=128,not=81,sh=8,dup=5,auto=1,csch=1, perl=1
439	xboing	not=225,unknown=141,ansic=72,sh=1
429	gnupg-1.0.1	ansic=171,asm=69,unknown=68,not=62,sh=32,dup=23, auto=4
423	glade-0.5.5	not=200,ansic=164,dup=26,unknown=24,auto=6,sh=3
419	rpm-3.0.4	ansic=133,not=91,dup=75,unknown=54,sh=52,perl=9, auto=4,python=1
416	gawk-3.0.4	unknown=148,awk=113,not=71,ansic=67,sh=10,dup=5, yacc=1,auto=1
413	trXFree86-2.1.2	not=405,unknown=7,tcl=1
410	pam-0.72	not=189,ansic=117,unknown=81,sh=15,yacc=3,perl=3, auto=1,lex=1
407	am-utils-6.0.3	ansic=208,not=135,unknown=25,sh=21,perl=8,dup=3, auto=3,yacc=2,lex=2
407	bash-1.14.7	unknown=152,ansic=135,not=50,sh=48,dup=15,auto=3, asm=2,awk=1,yacc=1
406	control-center-1.0.51	not=261,ansic=71,unknown=45,dup=26,auto=2,sh=1
400	mutt-1.0.1	unknown=212,ansic=122,not=53,dup=6,sh=5,auto=2
395	xpilot-4.1.0	ansic=192,unknown=93,not=78,cpp=28,sh=2,tcl=1,perl=1
393	tcsh-6.09.00	unknown=296,ansic=75,not=13,sh=3,dup=2,lisp=1,perl=1, csch=1,auto=1
389	xpdf-0.90	cpp=107,not=93,unknown=67,ansic=65,dup=51,sh=4,auto=2
387	cvs-1.10.7	ansic=217,unknown=73,not=69,sh=11,perl=8,dup=5,yacc=1, lisp=1,csch=1,auto=1
384	gnuplot-3.7.1	unknown=239,ansic=94,not=28,objc=7,sh=4,asm=3,perl=2,

		csch=2,lisp=2,dup=2,auto=1
382	shadow-19990827	ansic=157,not=91,unknown=87,dup=27,sh=16,auto=2, perl=1,yacc=1
372	extace-1.2.15	ansic=197,not=78,unknown=67,sh=13,dup=8,perl=7,auto=2
369	sendmail-8.9.3	not=162,unknown=140,ansic=55,perl=6,sh=6
362	gnome-utils-1.0.50	not=197,ansic=81,unknown=56,dup=24,auto=2,yacc=1, lisp=1
360	korganizer	not=137,cpp=105,unknown=86,dup=12,ansic=12,yacc=3, auto=1,zero=1,lex=1,sh=1,perl=1
358	util-linux-2.10f	ansic=149,not=118,unknown=69,sh=10,dup=6,csch=3, auto=1,sed=1,perl=1
354	tiff-v3.5.4	not=147,ansic=104,unknown=87,sh=8,dup=5,cpp=3
348	guile-1.3	ansic=197,not=50,unknown=39,lisp=24,asm=19,dup=8, awk=6,sh=3,csch=1,auto=1
348	kdetoys	not=239,unknown=74,cpp=17,dup=12,ansic=5,auto=1
344	nmh-1.0.3	ansic=210,not=84,unknown=39,sh=7,dup=1,awk=1,sed=1, auto=1
338	gtk-engines-0.10	not=222,unknown=37,dup=32,ansic=25,sh=14,auto=8
332	gnome-objc-1.0.2	objc=245,not=57,unknown=17,dup=8,auto=3,sh=1,ansic=1
328	xboard-4.0.5	unknown=273,ansic=27,not=15,dup=5,sh=4,lex=1,sed=1, csch=1,auto=1
320	pmake	ansic=163,unknown=108,not=37,sh=8,awk=2,sed=1,auto=1
318	xmms-1.0.1	ansic=189,not=100,unknown=13,dup=9,asm=3,sh=2,auto=2
316	uucp-1.06.1	ansic=245,unknown=36,not=21,sh=10,perl=2,auto=1, dup=1
316	nag	not=311,unknown=3,perl=2
311	unzip-5.40	unknown=136,ansic=114,not=36,cpp=9,asm=8,sh=4,zero=4
303	sawmill-0.24	not=171,lisp=69,unknown=27,ansic=26,sh=6,dup=3,auto=1
303	tcpdump-3.4	ansic=153,unknown=82,dup=26,not=22,awk=8,sh=5,auto=3, csch=2,lex=1,yacc=1
297	zip-2.3	ansic=140,unknown=109,not=23,asm=18,dup=7
296	fnlib-0.4	not=179,unknown=102,ansic=7,dup=4,auto=3,sh=1
296	automake-1.4	sh=198,not=75,unknown=12,dup=6,perl=3,auto=1,ansic=1
293	enscript-1.6.1	not=115,ansic=56,sh=48,unknown=46,dup=19,perl=4, auto=2,lex=1,lisp=1,yacc=1
291	gnome-pim-1.0.55	not=130,ansic=96,unknown=38,dup=23,yacc=2,auto=2
287	ppp-2.3.11	ansic=136,unknown=104,sh=25,not=18,csch=2,perl=1, exp=1
279	elm2.5.3	ansic=169,unknown=75,not=26,sh=8,awk=1
278	hellas	not=253,unknown=17,perl=4,dup=2,sh=2
277	xpuzzles-5.4.1	ansic=88,not=77,dup=52,unknown=49,auto=11
275	ORBit-0.5.0	ansic=135,not=88,unknown=34,sh=9,dup=4,auto=3,lex=1, yacc=1
274	lsnf_4.47	ansic=193,unknown=24,not=21,sh=20,perl=10,dup=3, awk=2,asm=1
270	rp3-1.0.7	cpp=95,not=80,unknown=37,dup=29,ansic=20,sh=5,auto=4
267	jed	unknown=147,ansic=83,not=34,dup=2,auto=1
260	pilot-link.0.9.3	ansic=96,java=58,not=43,unknown=33,cpp=10,dup=8, perl=6,python=2,tcl=1,yacc=1,auto=1,zero=1
255	make-3.78.1_pvm-0.5	unknown=163,dup=61,not=22,ansic=6,auto=3
254	libungif-4.1.0	not=119,dup=49,ansic=47,unknown=28,sh=6,auto=4,perl=1
252	ed-0.2	unknown=215,ansic=18,not=13,sh=4,auto=1,dup=1
248	tar-1.13.17	dup=75,not=69,unknown=44,ansic=38,sh=18,auto=2,perl=1, lisp=1
248	x11amp-0.9-alpha3	ansic=142,not=70,unknown=14,dup=11,auto=6,sh=3, asm=2
247	wu-ftpd-2.6.0	unknown=152,ansic=58,not=27,sh=4,dup=3,yacc=1,perl=1, auto=1
245	SVGATextMode-1.9-src	ansic=108,unknown=93,not=30,sh=10,lex=1,yacc=1, sed=1,asm=1
243	ncpfs-2.2.0.17	ansic=104,not=81,dup=26,unknown=22,sh=7,auto=2,tcl=1
243	texinfo-4.0	ansic=69,not=59,dup=52,unknown=38,sh=17,awk=3,auto=2, perl=1,sed=1,lisp=1
235	exmh-2.1.1	tcl=123,not=50,unknown=48,perl=4,auto=4,sh=4,exp=2
234	mod_perl-1.21	unknown=113,perl=88,not=20,ansic=11,sh=2

232	php-2.0.1	unknown=92,ansic=72,not=58,sh=7,dup=1,awk=1,auto=1
229	blt2.4g	not=78,ansic=66,tcl=52,unknown=31,auto=1,sh=1
223	xloadimage.4.1	ansic=161,unknown=42,not=14,sh=4,dup=2
220	multimedia	ansic=111,not=96,unknown=11,zero=1,sh=1
219	gettext-0.10.35	ansic=59,not=54,unknown=41,dup=31,sh=27,auto=2,yacc=2,perl=1,sed=1,lisp=1
219	gnome-media-1.0.51	not=106,unknown=48,ansic=36,dup=27,auto=2
218	gnome-python-1.0.51	not=79,python=79,unknown=36,ansic=14,auto=5,dup=4,sh=1
212	gedit-0.6.1	not=90,ansic=57,unknown=36,dup=26,auto=2,sh=1
211	trn-3.6	ansic=104,unknown=53,sh=42,not=8,dup=3,yacc=1
211	kpilot-3.1b9	not=99,cpp=67,unknown=26,dup=10,ansic=6,yacc=2,auto=1
210	tin-1.4.2	ansic=119,unknown=40,not=36,dup=8,sh=5,auto=1,yacc=1
205	ical-2.2	cpp=77,tcl=66,not=39,unknown=9,sh=8,auto=2,ansic=2,perl=1,dup=1
204	ncftp-3.0beta21	ansic=116,unknown=37,not=34,cpp=9,sh=5,dup=2,auto=1
203	AnotherLevel-1.0.1	not=179,unknown=21,sh=3
203	nfs-utils-0.1.6	ansic=99,not=78,unknown=17,sh=5,perl=2,auto=1,dup=1
199	wmakerconf-2.1	not=77,unknown=49,ansic=36,dup=26,perl=7,auto=3,sh=1
197	kpackage-1.3.10	not=107,cpp=53,unknown=22,dup=6,sh=4,ansic=3,auto=1,perl=1
196	make-3.78.1	unknown=113,ansic=49,not=18,dup=8,sh=5,perl=2,auto=1
196	xpat2-1.04	unknown=123,ansic=53,not=19,dup=1
192	mikmod-3.1.6	ansic=88,not=50,unknown=38,dup=12,auto=2,sh=1,awk=1
192	gnorpm-0.9	not=75,ansic=55,unknown=37,dup=23,auto=2
187	gdm-2.0beta2	not=77,unknown=55,dup=27,ansic=19,sh=5,auto=4
186	gv-3.5.8	ansic=126,not=30,unknown=23,sh=6,dup=1
183	lpg	not=177,unknown=4,perl=2
181	sag-0.6-html	not=166,unknown=12,perl=3
179	scheme-3.2	lisp=115,ansic=29,unknown=28,not=6,sh=1
178	xchat-1.4.0	ansic=72,not=66,dup=21,unknown=11,python=3,perl=3,sh=1,auto=1
175	gimp-data-extras-1.0.0	unknown=156,not=15,dup=3,auto=1
172	ee-0.3.11	not=84,ansic=30,unknown=29,dup=26,auto=2,sh=1
170	man-1.5h1	not=91,ansic=34,unknown=30,sh=10,awk=2,perl=2,dup=1
169	fetchmail-5.3.1	ansic=54,unknown=43,not=26,dup=23,sh=13,auto=3,awk=2,perl=2,lex=1,yacc=1,python=1
168	users-guide-1.0.72	not=159,unknown=5,dup=3,auto=1
168	grep-2.4	not=64,unknown=35,dup=27,ansic=26,sh=10,awk=3,auto=2,sed=1
163	gtop-1.0.5	not=70,ansic=39,dup=27,unknown=23,cpp=2,auto=2
160	xpaint	ansic=81,not=62,unknown=16,sh=1
158	x3270-3.1.1	ansic=78,not=43,unknown=28,sh=8,exp=1
157	mttools-3.9.6	ansic=88,not=34,unknown=20,sh=10,sed=2,dup=2,auto=1
157	X11R6-contrib-3.3.2	ansic=86,unknown=52,not=16,yacc=1,lex=1,sh=1
155	ld.so-1.9.5	unknown=73,ansic=42,not=18,asm=16,sh=6
151	file-3.28	unknown=122,ansic=16,not=10,perl=1,auto=1,dup=1
151	m4-1.4	unknown=74,not=43,ansic=17,dup=12,sh=3,auto=1,lisp=1
150	jpeg-6b	dup=94,unknown=34,not=20,auto=1,sh=1
150	enlightenment-conf-0.15	not=84,unknown=28,dup=27,sh=5,ansic=4,auto=2
149	net-tools-1.54	ansic=72,not=64,unknown=12,sh=1
147	rdist-6.1.5	ansic=70,unknown=57,not=12,sh=5,perl=2,yacc=1
145	lrzsz-0.12.20	ansic=37,not=32,dup=28,unknown=20,exp=19,sh=7,auto=2
144	rxvt-2.6.1	unknown=80,not=28,ansic=28,dup=7,auto=1
144	sed-3.02	unknown=64,sed=26,not=25,ansic=16,dup=8,sh=4,auto=1
142	jadetex	unknown=76,not=66
140	xosview-1.7.1	cpp=101,not=21,unknown=10,dup=4,ansic=2,awk=1,auto=1
140	librep-0.10	ansic=43,not=26,lisp=26,dup=24,unknown=14,sh=6,auto=1
140	ypserv-1.3.9	ansic=58,not=55,unknown=13,sh=10,dup=2,auto=1,perl=1
137	findutils-4.1	unknown=55,ansic=42,not=27,dup=9,sh=2,exp=1,auto=1
137	slang	ansic=67,unknown=30,not=22,dup=16,auto=2
136	mawk-1.2.2	ansic=66,unknown=31,awk=25,not=10,sh=2,auto=1,yacc=1
134	nss_ldap-105	unknown=63,ansic=62,not=8,perl=1
133	libglade-0.11	not=73,dup=25,unknown=15,ansic=13,sh=4,auto=2,python=1
133	slrn-0.9.6.2	ansic=65,unknown=47,not=16,auto=2,dup=2,sh=1

132	jpeg-6a	ansic=48,dup=36,unknown=27,not=20,auto=1
132	transfig.3.2.1	not=66,ansic=54,unknown=9,sh=2,csch=1
131	mars_nwe	ansic=65,unknown=56,not=7,sh=3
130	gftp-2.0.6a	not=55,unknown=30,dup=22,ansic=21,auto=2
128	strace-4.2	ansic=67,not=24,unknown=15,sh=15,dup=3,perl=2,auto=1, lisp=1
128	GXedit1.23	ansic=54,not=43,unknown=26,sh=5
128	Xaw3d-1.3	ansic=104,dup=16,unknown=5,yacc=1,not=1,lex=1
125	macutils	ansic=96,not=18,unknown=10,dup=1
125	gnotepad+-1.1.4	not=66,ansic=52,unknown=3,dup=3,auto=1
124	git-4.3.19	ansic=52,unknown=29,not=29,sh=10,dup=3,auto=1
124	glib-1.2.6	ansic=56,not=42,unknown=16,dup=7,sh=2,auto=1
124	wvdial-1.41	dup=83,unknown=27,not=14
124	libtool-1.3.4	not=42,sh=35,unknown=17,ansic=15,dup=10,auto=5
123	bug-buddy-0.7	not=42,unknown=34,dup=26,ansic=19,auto=2
123	xrn-9.02	ansic=89,unknown=17,not=7,csch=3,yacc=2,sh=2,awk=1, perl=1,lex=1
123	pdcksh-5.2.14	ansic=53,unknown=49,not=10,sh=6,dup=2,auto=1,sed=1, perl=1
122	piranha	not=68,unknown=27,ansic=26,sh=1
121	isdn-config	not=73,cpp=28,unknown=8,dup=5,sh=5,perl=1,auto=1
120	modutils-2.3.9	ansic=60,not=33,unknown=13,sh=6,dup=5,lex=1,yacc=1, auto=1
118	sharutils-4.2.1	not=35,unknown=31,ansic=31,dup=11,sh=5,perl=3,auto=2
116	gnuchess-4.0.pl80	unknown=46,ansic=39,not=26,sh=2,auto=1,csch=1,dup=1
116	screen-3.9.5	ansic=46,unknown=41,not=18,sh=9,auto=1,dup=1
115	wget-1.5.3	ansic=42,not=33,unknown=31,dup=6,perl=1,sh=1,auto=1
114	iproute2	ansic=70,unknown=14,sh=14,not=13,perl=3
114	sox-12.16	ansic=74,unknown=19,not=10,sh=7,dup=3,auto=1
114	mm2.7	csch=42,not=28,unknown=21,ansic=18,sh=5
113	bison-1.28	ansic=45,not=28,unknown=26,dup=11,auto=2,sh=1
113	irda-utils-0.9.10	not=50,ansic=33,unknown=22,sh=7,perl=1
112	audiofile-0.1.9	ansic=65,not=21,unknown=16,sh=6,dup=3,auto=1
112	zsh-3.0.7	ansic=43,not=25,unknown=22,sh=14,dup=2,awk=2,perl=2, sed=1,auto=1
111	gperf-2.7	unknown=38,not=31,cpp=26,auto=5,exp=5,ansic=4,sh=1, dup=1
110	minicom-1.83.0	unknown=50,ansic=37,not=21,sh=2
108	pwdb-0.61	ansic=68,not=20,unknown=11,dup=7,sh=2
108	joe	ansic=85,unknown=17,not=5,asm=1
108	jikes	cpp=87,not=16,unknown=3,java=1,auto=1
108	dhcpc-2.0	ansic=66,unknown=21,not=14,sh=7
108	netkit-telnet-0.16	ansic=45,cpp=26,not=25,unknown=9,dup=2,auto=1
107	urw-fonts-2.0	not=70,unknown=37
105	bc-1.05	unknown=34,ansic=33,not=25,sh=6,dup=2,yacc=2,auto=2, lex=1
104	sysvinit-2.78	unknown=30,not=26,ansic=25,sh=23
104	zlib-1.1.3	unknown=45,dup=28,not=11,cpp=8,ansic=8,asm=3,auto=1
103	xbill-2.0	not=65,cpp=17,dup=16,unknown=5
100	mailx-8.1.1	unknown=66,ansic=30,not=3,sh=1
97	mpg123-0.59r	ansic=66,unknown=14,not=13,asm=3,dup=1
96	silo-0.9.8	ansic=47,unknown=16,asm=16,not=9,dup=8
95	gpgp-0.4	not=41,dup=25,unknown=13,ansic=11,sh=3,auto=2
95	yp-tools-2.4	not=39,ansic=21,dup=20,unknown=12,auto=2,sh=1
95	p2c-1.22	ansic=39,unknown=36,not=14,pascal=5,perl=1
93	imlib-1.9.7	ansic=35,not=34,dup=13,unknown=9,auto=1,sh=1
93	readline-2.2.1	ansic=29,dup=23,not=18,unknown=17,sh=4,auto=1,perl=1
93	aumix-1.30.1	not=35,ansic=29,unknown=21,dup=2,sh=2,sed=2,auto=1, zero=1
91	isapnptools-1.21	unknown=52,ansic=24,not=11,perl=2,yacc=1,sh=1
91	rsync-2.4.1	ansic=55,not=14,dup=11,unknown=6,auto=2,perl=1,sh=1, awk=1
89	gzip-1.2.4a	ansic=34,unknown=27,not=23,asm=2,auto=1,sh=1,perl=1
88	procmail-3.14	ansic=50,unknown=16,not=14,sh=8
88	procps-2.0.6	ansic=46,not=22,unknown=19,sh=1

```
87 xdaliclock-2.18 not=59,ansic=17,unknown=4,dup=4,auto=2,sh=1
87 Xconfigurator-4.3.5 not=37,ansic=35,unknown=10,perl=3,python=1,sh=1
87 flex-2.5.4 unknown=32,ansic=30,not=13,lex=4,dup=2,sh=2,yacc=1,
    auto=1,sed=1,awk=1
86 dip-3.3.7o unknown=45,ansic=24,not=14,sh=3
84 magicdev-0.2.7 not=27,unknown=22,dup=22,ansic=11,auto=2
83 fortune-mod-9708 unknown=64,not=13,ansic=6
82 pidentd-3.0.10 ansic=57,unknown=10,not=10,dup=3,auto=1,sh=1
82 pnm2ppa ansic=33,unknown=25,not=20,sh=4
82 cproto-4.6 unknown=33,ansic=23,not=17,sh=5,dup=1,lex=1,yacc=1,
    auto=1
81 sndconfig-0.43 not=35,dup=30,unknown=11,ansic=5
81 taper-6.9a ansic=50,not=17,unknown=13,dup=1
81 pxe-linux ansic=40,cpp=19,unknown=10,not=9,asm=2,dup=1
80 libpng-1.0.5 unknown=37,ansic=32,not=11
79 mouseconfig-4.4 not=37,dup=30,unknown=9,ansic=3
76 lpr not=38,ansic=23,unknown=14,sh=1
76 less-346 ansic=45,unknown=20,not=7,dup=2,awk=1,auto=1
76 phhttpd-0.1.0 not=48,ansic=17,unknown=10,sh=1
76 rpm2html-1.2 ansic=41,unknown=17,not=13,dup=3,perl=1,auto=1
75 lslk ansic=50,not=10,sh=9,unknown=6
75 xtrojka123 unknown=36,ansic=35,not=3,sh=1
74 ash-linux-0.2 ansic=48,unknown=12,not=9,sh=5
73 initscripts-5.00 sh=31,ansic=16,not=15,unknown=10,csch=1
72 memprof-0.3.0 dup=27,not=20,ansic=13,unknown=10,auto=2
71 gdbm-1.8.0 ansic=47,not=8,unknown=7,dup=7,cpp=1,auto=1
70 getty_ps-2.0.7j unknown=41,ansic=18,not=11
70 xmorph-1999dec12 ansic=41,unknown=14,tcl=10,not=5
70 ltrace-0.3.10 ansic=39,not=14,unknown=8,dup=3,sh=3,awk=2,auto=1
70 gpm-1.18.1 ansic=26,not=19,unknown=14,sh=3,lisp=2,awk=2,dup=1,
    yacc=1,sed=1,auto=1
70 esound-0.2.17 ansic=37,not=14,unknown=8,dup=7,sh=2,csch=1,auto=1
69 xpm-3.4k ansic=37,unknown=16,not=14,cpp=1,sh=1
68 rcs-5.7 ansic=29,not=23,unknown=10,sh=3,dup=2,auto=1
68 awesfx-0.4.3a ansic=43,unknown=21,not=4
68 xfishtank-2.1tp ansic=45,unknown=11,not=11,dup=1
68 cpio-2.4.2 ansic=44,unknown=13,not=8,auto=1,dup=1,sh=1
68 aboot-0.5 ansic=41,not=20,asm=4,unknown=3
67 pam_krb5-1 unknown=51,not=10,ansic=4,auto=1,dup=1
67 libelf-0.6.4 ansic=48,not=9,unknown=5,dup=2,sh=2,auto=1
66 netkit-rsh-0.16 not=28,unknown=20,ansic=17,auto=1
66 apmd not=24,unknown=20,sh=11,ansic=11
64 kdpms-0.2.8 not=35,unknown=12,cpp=7,dup=6,auto=4
64 gnome-kberos-0.2 dup=23,not=19,ansic=12,unknown=8,auto=2
61 ghostscript-fonts-5.50 unknown=53,not=8
61 lilo ansic=25,not=18,asm=6,unknown=5,sh=3,perl=2,cpp=1,
    dup=1
61 mkisofs-1.12b5 ansic=23,unknown=14,dup=12,not=11,auto=1
61 dialog-0.6 unknown=33,ansic=13,sh=9,not=5,perl=1
59 switchdesk-2.1 not=33,unknown=8,dup=7,sh=4,cpp=3,ansic=2,auto=1,
    perl=1
59 acct-6.3.2 ansic=28,not=17,unknown=8,dup=2,sh=2,auto=1,cpp=1
59 tcp_wrappers_7.6 ansic=41,unknown=9,not=9
59 autofs-3.1.4 ansic=21,not=17,unknown=16,sh=4,auto=1
58 vixie-cron-3.0.1 unknown=33,ansic=17,not=7,sh=1
58 dump-0.4b15 ansic=28,not=17,unknown=8,dup=2,sh=1,sed=1,auto=1
57 autorun-2.61 unknown=25,not=20,cpp=4,dup=4,sh=3,auto=1
56 kdoc not=15,perl=15,unknown=14,dup=6,sh=4,cpp=1,auto=1
56 indent-2.2.5 ansic=19,unknown=17,not=14,dup=3,auto=1,sh=1,zero=1
56 xjewel-1.6 not=30,ansic=21,unknown=5
55 sliplogin-2.1.1 unknown=31,ansic=13,not=8,sh=2,perl=1
55 netkit-base-0.16 unknown=23,not=15,ansic=15,auto=1,sh=1
54 ctags-3.4 ansic=31,unknown=15,not=6,auto=1,sh=1
54 specsps-6.2 not=50,unknown=4
50 wmconfig-0.9.8 ansic=22,not=16,unknown=5,dup=4,sh=2,auto=1
```



```
50 syslogd-1.3-31 unknown=21,ansic=13,not=8,sh=6,perl=2
50 popt-1.4 not=20,dup=12,unknown=7,ansic=6,auto=3,sh=2
50 diffutils-2.7 ansic=29,unknown=10,not=7,dup=3,auto=1
50 playmidi-2.4 unknown=27,ansic=16,not=4,auto=1,dup=1,sed=1
49 raidtools-0.90 not=19,ansic=17,unknown=10,auto=1,dup=1,sh=1
48 quota-2.00-pre3 not=22,ansic=21,unknown=4,sh=1
47 autoconf-2.13 unknown=17,not=14,sh=8,exp=4,dup=1,perl=1,auto=1,
    ansic=1
47 patch-2.5 ansic=29,not=7,unknown=6,dup=2,sed=2,auto=1
47 kterm-6.2.0 ansic=30,unknown=15,not=2
47 at-3.1.7 ansic=14,unknown=13,not=9,auto=4,sh=4,lex=1,dup=1,
    yacc=1
46 which-2.9 not=20,dup=9,ansic=8,unknown=5,auto=2,sh=2
46 kpppload-1.04 not=23,cpp=11,dup=6,unknown=3,auto=2,sh=1
44 traceroute-1.4a5 unknown=13,dup=11,ansic=9,not=6,auto=3,awk=2
44 cdparanoia-III-alpha9.6 ansic=31,not=7,unknown=2,sh=2,zero=1,auto=1
43 kudzu-0.36 dup=30,unknown=7,not=6
42 gqview-0.7.0 ansic=22,not=19,unknown=1
41 bzip2-0.9.5d unknown=18,ansic=12,not=11
41 libghhttp-1.0.4 ansic=18,not=10,dup=7,unknown=5,auto=1
39 newt-0.50.8 ansic=26,not=7,python=3,unknown=1,auto=1,dup=1
39 netkit-rusers-0.16 not=21,ansic=11,unknown=6,auto=1
39 xxgdb-1.12 ansic=27,unknown=7,not=4,dup=1
39 mpage-2.4 unknown=15,ansic=12,not=10,sh=2
37 dhcpcd-1.3.18-pl3 ansic=16,unknown=8,sh=7,not=6
37 rep-gtk-0.8 unknown=10,not=10,ansic=6,sh=5,dup=3,lisp=2,auto=1
37 netkit-timed-0.16 ansic=20,not=15,unknown=1,auto=1
37 xcpustate-2.5 ansic=28,unknown=7,not=2
37 iputils ansic=15,unknown=13,not=6,dup=3
37 libPropList-0.9.1 ansic=15,not=8,dup=5,unknown=4,sh=2,auto=1,lex=1,
    yacc=1
36 joystick-1.2.15 ansic=24,not=10,unknown=1,sh=1
36 netkit-ntalk-0.16 ansic=21,not=12,unknown=2,auto=1
36 dosfstools-2.2 ansic=18,not=15,unknown=3
35 rpmfind-1.4 ansic=20,not=8,unknown=3,dup=3,auto=1
35 timeconfig-3.0.3 not=31,unknown=2,sh=1,ansic=1
34 netkit-routed-0.16 ansic=18,not=12,unknown=3,auto=1
34 ypbind-3.3 ansic=14,not=12,sh=3,unknown=2,dup=2,auto=1
34 chkconfig-1.1.2 not=28,ansic=4,unknown=2
33 kbdconfig-1.9.2.4 not=30,unknown=2,ansic=1
33 control-panel-3.13 not=29,tcl=2,unknown=1,ansic=1
33 gnome-linuxconf-0.25 not=10,unknown=8,dup=7,ansic=5,auto=3
33 ipchains-1.3.9 not=26,ansic=4,sh=3
32 ext2ed-0.1 ansic=14,unknown=9,not=9
32 nc sh=12,unknown=8,not=7,ansic=5
32 cdp-0.33 ansic=20,unknown=6,not=5,zero=1
32 ytalk-3.1 ansic=16,not=7,unknown=6,dup=2,auto=1
32 prtconf-1.3 unknown=26,not=4,ansic=2
31 authconfig-3.0.3 not=28,unknown=2,ansic=1
31 netscape-4.72 unknown=29,sh=2
30 gnome-audio-1.0.0 unknown=26,not=4
30 cracklib,2.7 unknown=12,ansic=11,not=4,sh=2,perl=1
30 pciutils-2.1.5 ansic=16,not=7,unknown=6,sh=1
29 byacc-1.9 ansic=13,unknown=7,auto=4,not=3,yacc=2
29 time-1.7 ansic=11,unknown=7,not=7,dup=2,auto=1,sh=1
28 netkit-rwho-0.16 not=15,ansic=6,unknown=5,auto=1,dup=1
28 redhat-logos not=26,unknown=2
28 rpmlint-0.8 python=16,unknown=7,not=4,sh=1
26 bsd-finger-0.16 not=12,ansic=10,unknown=3,auto=1
26 xgammon-0.98 ansic=16,unknown=5,not=4,lex=1
26 psgml-1.2.1 lisp=12,unknown=7,not=4,dup=2,auto=1
26 units-1.55 not=9,unknown=7,dup=4,ansic=4,perl=1,auto=1
25 docbook-3.1 unknown=16,not=9
25 xmailbox-2.5 not=13,unknown=7,ansic=5
25 urlview-0.7 not=9,unknown=5,ansic=5,dup=3,sh=2,auto=1
```

25	sash-3.4	ansic=14,unknown=8,not=3
25	netkit-ftp-0.16	ansic=12,not=10,unknown=2,auto=1
24	usermode-1.20	ansic=13,not=9,unknown=2
23	termcap-2.0.8	unknown=15,not=4,ansic=4
22	netkit-rwall-0.16	not=13,ansic=4,unknown=3,auto=1,dup=1
22	gd1.3	ansic=17,not=4,perl=1
22	efax-0.9	ansic=10,not=6,unknown=4,sh=2
21	rhs-printfilters-1.63	not=9,sh=8,unknown=3,ansic=1
20	up2date-1.13	not=9,python=7,unknown=4
19	psmisc	not=8,ansic=6,unknown=4,sh=1
19	netkit-tftp-0.16	not=11,ansic=6,unknown=1,auto=1
19	sgml-common-0.1	unknown=19
19	fbset-2.1	unknown=7,not=5,ansic=4,yacc=1,perl=1,lex=1
19	setup-2.1.8	unknown=18,not=1
18	slocate-2.1	ansic=7,unknown=6,not=4,sh=1
17	pythonlib-1.23	python=14,not=2,unknown=1
17	unarj-2.43	not=7,unknown=6,ansic=4
16	procinfo-17	not=7,unknown=4,ansic=3,perl=2
16	netkit-bootparamd-0.16	not=9,ansic=4,unknown=2,auto=1
16	ncompress-4.2.4	unknown=8,not=6,ansic=2
16	biff+comsat-0.16	not=11,ansic=3,unknown=1,auto=1
16	anacron-2.1	ansic=10,not=5,unknown=1
15	logrotate-3.3.2	ansic=7,not=5,unknown=2,sh=1
15	passwd-0.64.1	ansic=8,not=5,unknown=2
14	open-1.4	unknown=5,not=5,ansic=3,dup=1
14	xsysinfo-1.7	ansic=6,unknown=5,not=3
14	cleanfeed-0.95.7b	unknown=8,not=4,perl=2
14	adjtimex-1.9	not=7,ansic=3,unknown=2,auto=1,dup=1
14	vlock-1.3	ansic=7,not=4,unknown=3
14	ElectricFence-2.1	ansic=6,not=5,unknown=3
13	mt-st-0.5b	unknown=6,not=5,ansic=2
13	pump-0.7.8	not=4,dup=4,ansic=3,unknown=2
13	setup-1.2	not=8,unknown=4,ansic=1
13	gkermit-1.0	ansic=5,unknown=4,not=3,auto=1
13	cxhextris	unknown=5,not=4,ansic=4
13	tksysv-1.1	not=7,unknown=3,sh=2,tcl=1
13	trojka	ansic=9,not=3,unknown=1
12	portmap_4	ansic=8,not=3,unknown=1
12	usernet-1.0.9	not=5,ansic=5,unknown=2
12	setserial-2.15	not=6,unknown=3,ansic=2,auto=1
12	eject-2.0.2	not=6,unknown=5,ansic=1
11	mingetty-0.9.4	unknown=6,not=4,ansic=1
11	diffstat-1.27	not=6,dup=2,ansic=2,auto=1
10	indexhtml-6.2	not=9,unknown=1
10	isicom	unknown=5,ansic=3,not=1,sh=1
10	auth_ldap-1.4.0	not=4,ansic=4,unknown=2
9	printtool	not=5,unknown=3,tcl=1
9	bdf flush-1.5	not=5,asm=2,unknown=1,ansic=1
9	cdecl-2.5	not=4,unknown=2,yacc=1,lex=1,ansic=1
9	helptool-2.4	unknown=3,perl=3,not=2,tcl=1
8	genromfs-0.3	not=5,unknown=2,ansic=1
8	modemtool-1.21	unknown=3,not=3,python=1,sh=1
8	kernelcfg-0.5	unknown=3,not=3,python=1,sh=1
8	statserial-1.1	not=5,unknown=1,sh=1,ansic=1
8	locale_config-0.2	not=3,ansic=3,unknown=2
8	locale-ja-9	unknown=7,sh=1
7	hdparm-3.6	unknown=3,not=3,ansic=1
7	ethtool-1.0	not=4,ansic=2,unknown=1
7	utempter-0.5.2	ansic=4,not=2,unknown=1
7	sysreport-1.0	not=3,sh=3,unknown=1
7	xtoolwait-1.2	unknown=3,not=3,ansic=1
6	tmpwatch-2.2	unknown=2,not=2,sh=1,ansic=1
6	chkfontpath-1.7	not=4,unknown=1,ansic=1
6	rootfiles	unknown=6
6	mailcap-2.0.6	unknown=4,not=2

6	solemul-1.1	unknown=5,not=1
6	netcfg-2.25	unknown=2,not=2,python=1,sh=1
6	tree-1.2	not=4,unknown=1,ansic=1
5	symlinks-1.2	unknown=2,not=2,ansic=1
5	MAKEDEV-2.5.2	not=3,unknown=1,sh=1
5	words-2	unknown=3,sh=2
5	rhmask	unknown=2,not=2,ansic=1
5	elftoaout-2.2	not=3,unknown=1,ansic=1
5	ldconfig-1999-02-21	unknown=2,ansic=2,not=1
5	mkbootdisk-1.2.5	not=2,sh=2,unknown=1
5	fwhois-1.00	unknown=2,not=2,ansic=1
5	timetool-2.7.3	unknown=2,not=2,tcl=1
4	quickstrip-1.1	not=3,ansic=1
4	stat-1.5	not=2,unknown=1,ansic=1
4	mkkickstart-2.1	not=2,unknown=1,sh=1
4	mkinitrd-2.4.1	not=2,unknown=1,sh=1
4	xsri-1.0	not=2,unknown=1,ansic=1
4	sparc32-1.1	not=3,ansic=1
4	ipvsadm-1.1	not=2,unknown=1,ansic=1
4	rdate-1.0	not=3,ansic=1
3	shaper	unknown=1,not=1,ansic=1
3	audioctl	not=2,ansic=1
3	desktop-backgrounds-1.1	unknown=2,not=1
3	mktemp-1.5	not=2,ansic=1
2	intimed-1.10	not=1,ansic=1
2	giftrans-1.12.2	not=1,ansic=1
2	minlabel-1.2	not=1,ansic=1
0	dev-2.7.18	(none)
0	caching-nameserver-6.2	(none)

not:	62241	(34.26%)
ansic:	52088	(28.67%)
unknown:	40308	(22.19%)
cpp:	8092	(4.45%)
dup:	5820	(3.20%)
sh:	3381	(1.86%)
asm:	1931	(1.06%)
perl:	1387	(0.76%)
lisp:	1168	(0.64%)
java:	1047	(0.58%)
python:	997	(0.55%)
auto:	817	(0.45%)
tcl:	798	(0.44%)
exp:	472	(0.26%)
awk:	285	(0.16%)
objc:	260	(0.14%)
sed:	112	(0.06%)
yacc:	110	(0.06%)
csh:	94	(0.05%)
ada:	92	(0.05%)
zero:	65	(0.04%)
lex:	57	(0.03%)
fortran:	50	(0.03%)
pascal:	7	(0.00%)

ansic:	52088	(71.92%)
cpp:	8092	(11.17%)
sh:	3381	(4.67%)
asm:	1931	(2.67%)
perl:	1387	(1.92%)
lisp:	1168	(1.61%)
java:	1047	(1.45%)
python:	997	(1.38%)

tcl:	798	(1.10%)
exp:	472	(0.65%)
awk:	285	(0.39%)
objc:	260	(0.36%)
sed:	112	(0.15%)
yacc:	110	(0.15%)
csh:	94	(0.13%)
ada:	92	(0.13%)
lex:	57	(0.08%)
fortran:	50	(0.07%)
pascal:	7	(0.01%)

Total Number of Files = 181679

Total Number of Source Code Files = 72428

SLOC	Directory	SLOC-by-Language (Sorted)
53141	AfterStep-1.8.0	ansic=50898,perl=1168,sh=842,cpp=233
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
16	AnotherLevel-1.0.1	sh=16
834	ElectricFence-2.1	ansic=834
15108	GXedit1.23	ansic=15019,sh=89
121878	ImageMagick-4.2.9	ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
1020	MAKEDEV-2.5.2	sh=1020
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
38548	ORBit-0.5.0	ansic=35656,yacc=1750,sh=776,lex=366
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342, sed=2
15967	SVGATextMode-1.9-src	ansic=15079,yacc=340,sh=294,lex=227,sed=15, asm=12
79997	WindowMaker-0.61.1	ansic=77924,sh=1483,perl=371,lisp=219
18885	X11R6-contrib-3.3.2	ansic=18616,lex=161,yacc=97,sh=11
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358, yacc=2710,perl=711,awk=393,lex=383,sed=57,csch=5
0	XFree86-ISO8859-2-1.0	(none)
26608	Xaw3d-1.3	ansic=26235,yacc=247,lex=126
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6
6976	aboot-0.5	ansic=6680,asm=296
5383	acct-6.3.2	ansic=5016,cpp=287,sh=80
1653	adjtimex-1.9	ansic=1653
45589	am-utils-6.0.3	ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
91213	anaconda-6.2.2	ansic=74303,python=13657,sh=1583,yacc=810,lex=732, perl=128
1143	anacron-2.1	ansic=1143
77873	apache_1.3.12	ansic=69191,sh=6781,perl=1846,cpp=55
3012	apmd	ansic=2617,sh=395
9666	ash-linux-0.2	ansic=9445,sh=221
3084	at-3.1.7	ansic=1442,sh=1196,yacc=362,lex=84
441	audiocctl	ansic=441
12593	audiofile-0.1.9	sh=6440,ansic=6153
4367	aumix-1.30.1	ansic=4095,sh=179,sed=93
1182	auth_ldap-1.4.0	ansic=1182
1075	authconfig-3.0.3	ansic=1075
2758	autoconf-2.13	sh=2226,perl=283,exp=167,ansic=82
3625	autofs-3.1.4	ansic=2862,sh=763
14363	automake-1.4	perl=10622,sh=3337,ansic=404
2705	autorun-2.61	sh=1985,cpp=720
6234	awesfx-0.4.3a	ansic=6234
47067	bash-1.14.7	ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
68560	bash-2.03	ansic=56758,sh=7264,yacc=2808,perl=1730
17682	bc-1.05	ansic=9186,sh=7236,yacc=967,lex=293
261	bdfush-1.5	ansic=202,asm=59
290	biff+comsat-0.16	ansic=290
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360, csch=848,awk=753,lex=222
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606, cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
9699	bison-1.28	ansic=9650,sh=49
68997	blt2.4g	ansic=58630,tcl=10215,sh=152
1272	bsd-finger-0.16	ansic=1272
2486	bug-buddy-0.7	ansic=2486
6550	byacc-1.9	ansic=5520,yacc=1030
4996	bzip2-0.9.5d	ansic=4996
0	caching-nameserver-6.2	(none)
1842	cdecl-2.5	ansic=1002,yacc=765,lex=75
2661	cdp-0.33	ansic=2661
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
50970	cdrecord-1.8	ansic=48595,sh=2177,perl=194,sed=4
717	chkconfig-1.1.2	ansic=717
343	chkfontpath-1.7	ansic=343
1984	cleanfeed-0.95.7b	perl=1984

15522	console-tools-0.3.3	ansic=13335,yacc=986,sh=800,lex=291,perl=110
16785	control-center-1.0.51	ansic=16659,sh=126
404	control-panel-3.13	ansic=319,tcl=85
7617	cpio-2.4.2	ansic=7598,sh=19
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
1987	cracklib,2.7	ansic=1919,perl=46,sh=22
9263	ctags-3.4	ansic=9240,sh=23
88128	cvs-1.10.7	ansic=68303,sh=17909,perl=902,yacc=826,csch=181,lisp=7
1290	cxhextris	ansic=1290
0	desktop-backgrounds-1.1	(none)
0	dev-2.7.18	(none)
16266	dhcp-2.0	ansic=15328,sh=938
3051	dhcpcd-1.3.18-pl3	ansic=2771,sh=280
3433	dialog-0.6	ansic=2834,perl=349,sh=250
616	diffstat-1.27	ansic=616
10914	diffutils-2.7	ansic=10914
8303	dip-3.3.7o	ansic=8207,sh=96
0	docbook-3.1	(none)
3675	dosfstools-2.2	ansic=3675
10187	dump-0.4b15	ansic=9422,sh=760,sed=5
28169	e2fsprogs-1.18	ansic=27250,awk=437,sh=339,sed=121,perl=22
7427	ed-0.2	ansic=7263,sh=164
7030	ee-0.3.11	ansic=7007,sh=23
5526	efax-0.9	ansic=4570,sh=956
720112	egcs-1.1.2	ansic=598682,cpp=75206,sh=14307,asm=11462,yacc=7988, lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18
657	eject-2.0.2	ansic=657
480	elftoaout-2.2	ansic=480
42746	elm2.5.3	ansic=32931,sh=9774,awk=41
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253, csch=9,sed=4
51592	enlightenment-0.15.5	ansic=51569,sh=23
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
23666	enscript-1.6.1	ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
7615	esound-0.2.17	ansic=7387,sh=142,csch=86
332	ethtool-1.0	ansic=332
36243	exmh-2.1.1	tcl=35844,perl=316,sh=49,exp=34
3415	ext2ed-0.1	ansic=3415
78787	extace-1.2.15	ansic=66571,sh=9322,perl=2894
1765	fbset-2.1	ansic=1401,yacc=130,lex=121,perl=113
17271	fetchmail-5.3.1	ansic=13441,python=1490,sh=1246,yacc=411,perl=321, lex=238,awk=124
2647	file-3.28	ansic=2601,perl=46
34768	fileutils-4.0p	ansic=31324,sh=2042,yacc=841,perl=561
11404	findutils-4.1	ansic=11160,sh=173,exp=71
14774	flex-2.5.4	ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12
2455	fnlib-0.4	ansic=2432,sh=23
1604	fortune-mod-9708	ansic=1604
51813	freetype-1.3.1	ansic=48929,sh=2467,cpp=351,csch=53,perl=13
69265	fvwm-2.2.4	ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
107	fwhois-1.00	ansic=107
138024	gated-3-5-11	ansic=126846,yacc=7799,sh=1554,lex=877,awk=666,csch=235, sed=35,lisp=12
26363	gawk-3.0.4	ansic=19871,awk=2519,yacc=2046,sh=1927
20078	gd1.3	ansic=19946,perl=132
652087	gdb-19991004	ansic=587542,exp=37737,sh=9630,cpp=6735,asm=4139, yacc=4117,lisp=1820,sed=220,awk=142,fortran=5
3315	gdbm-1.8.0	ansic=3290,cpp=25
5744	gdm-2.0beta2	ansic=5632,sh=112
8356	gedit-0.6.1	ansic=8225,sh=131
549	genromfs-0.3	ansic=549
17750	gettext-0.10.35	ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9
2631	getty_ps-2.0.7j	ansic=2631
9138	gftp-2.0.6a	ansic=9138
0	ghostscript-fonts-5.50	(none)

770	giftrans-1.12.2	ansic=770
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
0	gimp-data-extras-1.0.0	(none)
18151	git-4.3.19	ansic=16166,sh=1985
2835	gkermi-1.0	ansic=2835
54603	glade-0.5.5	ansic=49545,sh=5058
18835	glib-1.2.6	ansic=18702,sh=133
415026	glibc-2.1.3	ansic=378753,asm=30644,sh=2520,cpp=1704,awk=910, perl=464,sed=16,cs=15
24583	gmp-2.0.2	ansic=17888,asm=5252,sh=1443
0	gnome-audio-1.0.0	(none)
72425	gnome-core-1.0.55	ansic=72230,perl=141,sh=54
41205	gnome-games-1.0.51	ansic=31191,lisp=6966,cpp=3048
2531	gnome-kerberos-0.2	ansic=2531
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177
2163	gnome-linuxconf-0.25	ansic=2163
6827	gnome-media-1.0.51	ansic=6827
12463	gnome-objc-1.0.2	objc=12365,sh=86,ansic=12
30438	gnome-pim-1.0.55	ansic=28665,yacc=1773
24270	gnome-python-1.0.51	python=14331,ansic=9791,sh=148
19500	gnome-utils-1.0.50	ansic=18099,yacc=824,lisp=577
10404	gnorpm-0.9	ansic=10404
15143	gnotepad+-1.1.4	ansic=15143
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,cs=29
116615	gnnumeric-0.48	ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
54935	gnupg-1.0.1	ansic=48884,asm=4586,sh=1465
45378	gnuplot-3.7.1	ansic=43276,lisp=661,asm=539,objc=387,cs=297,perl=138, sh=80
4430	gperf-2.7	cpp=2947,exp=745,ansic=695,sh=43
4542	gpgp-0.4	ansic=4441,sh=101
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
10271	gqview-0.7.0	ansic=10271
10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
70260	groff-1.15	cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397, sh=265,sed=46
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
35397	gtk-engines-0.10	ansic=20636,sh=14761
8491	gtop-1.0.5	ansic=8151,cpp=340
45485	guile-1.3	ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,cs=50
25915	gv-3.5.8	ansic=25821,sh=94
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
1229	hdparm-3.6	ansic=1229
221	hellas	sh=179,perl=42
491	helptool-2.4	perl=288,tcl=203
20125	ical-2.2	cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
61688	imap-4.7	ansic=61628,sh=60
49325	imlib-1.9.7	ansic=49260,sh=65
5981	indent-2.2.5	ansic=5958,sh=23
0	indexhtml-6.2	(none)
3929	initscripts-5.00	sh=2035,ansic=1866,cs=28
81719	inn-2.2.2	ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547, lex=249,tcl=3
0	install-guide-3.2.html	(none)
47	intimed-1.10	ansic=47
3651	ipchains-1.3.9	ansic=2767,sh=884
13241	iproute2	ansic=12139,sh=1002,perl=100
6646	iputils	ansic=6646
255	ipvsadm-1.1	ansic=255
37515	ircii-4.4	ansic=36647,sh=852,lex=16
6043	irda-utils-0.9.10	ansic=5697,sh=263,perl=83
5960	isapnptools-1.21	ansic=4394,yacc=1383,perl=123,sh=60
5594	isdn-config	cpp=3058,sh=2228,perl=308
87940	isdn4k-utils	ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
1941	isicom	ansic=1898,sh=43

14912	ispell-3.1	ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385, csh=221,sh=157,perl=85,sed=15
131372	jade-1.2.1	cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
0	jadetex	(none)
29315	jed	ansic=29315
71810	jikes	cpp=71452,java=358
19065	joe	ansic=18841,asm=224
6090	joystick-1.2.15	ansic=6086,sh=4
12313	jpeg-6a	ansic=12313
844	jpeg-6b	sh=844
133193	kaffe-1.0.5	java=65275,ansic=62125,cpp=3923,perl=972,sh=814, asm=84
368	kbdconfig-1.9.2.4	ansic=368
24910	kdeadmin	cpp=19919,sh=3936,perl=1055
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
27860	kdegames	cpp=27507,ansic=340,sh=13
68453	kdegraphics	cpp=34208,ansic=29347,sh=4898
80343	kdelibs	cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116, sh=35
48223	kdemultimedia	ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
93940	kdenetwork	cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
60429	kdesupport	ansic=42421,cpp=17810,sh=173,awk=13,csh=12
2810	kdetoys	cpp=2618,ansic=192
52808	kdeutils	cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
6072	kdcc	perl=6010,sh=45,cpp=17
809	kdpms-0.2.8	cpp=809
347	kernelcfg-0.5	python=341,sh=6
31994	korganizer	cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
15868	kpilot-3.1b9	cpp=8613,ansic=5640,yacc=1615
1049	kpppload-1.04	cpp=1044,sh=5
99066	krb4-1.0	ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765, yacc=1509,lex=236,awk=33
222220	krb5-1.1.1	ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528, awk=393,python=348,lex=190,csh=147,sed=123
23838	kterm-6.2.0	ansic=23838
0	kudzu-0.36	(none)
100951	lam-6.3.1	ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187, csh=19
9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
874	ldconfig-1999-02-21	ansic=874
14039	less-346	ansic=14032,awk=7
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
4792	libelf-0.6.4	ansic=3310,sh=1482
2645	libghhttp-1.0.4	ansic=2645
7859	libglade-0.11	ansic=5898,sh=1809,python=152
102674	libgr-2.0.13	ansic=99647,sh=2438,csh=589
14516	libgtop-1.0.6	ansic=13768,perl=653,sh=64,asm=31
22011	libpng-1.0.5	ansic=22011
27117	librep-0.10	ansic=19381,lisp=5385,sh=2351
5115	libtool-1.3.4	sh=3374,ansic=1741
12639	libungif-4.1.0	ansic=12381,sh=204,perl=54
26146	libxml-1.8.6	ansic=26069,sh=77
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,cpp=3
1526722	linux	ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414, yacc=324,lex=230,awk=133,sed=72
69246	linux-86	ansic=63328,asm=5276,sh=642
104032	linuxconf-1.17r2	cpp=93139,perl=4570,sh=2984,java=2741,ansic=598
23	locale-ja-9	sh=23
497	locale_config-0.2	ansic=497
1525	logrotate-3.3.2	ansic=1524,sh=1
26673	lout-3.17	ansic=26673
682	lpg	perl=682
6877	lpr	ansic=6842,sh=35
11790	lrzsz-0.12.20	ansic=9512,sh=1263,exp=1015

6116	lslk	ansic=5325,sh=791
56093	lsuf_4.47	ansic=50268,sh=4753,perl=856,awk=214,asm=2
3896	ltracel-0.3.10	ansic=2986,sh=854,awk=56
119613	lynx2-8-3	ansic=117385,sh=1860,perl=340,csn=28
6391	m4-1.4	ansic=5993,lisp=243,sh=155
12657	macutills	ansic=12657
2580	magicdev-0.2.7	ansic=2580
0	mailcap-2.0.6	(none)
6968	mailx-8.1.1	ansic=6963,sh=5
22279	make-3.78.1	ansic=19287,sh=2029,perl=963
4780	make-3.78.1_pvm-0.5	ansic=4780
9801	man-1.5h1	ansic=7377,sh=1802,perl=317,awk=305
245	man-pages-1.28	sh=244,sed=1
24387	mars_nwe	ansic=24158,sh=229
14427	mawk-1.2.2	ansic=12714,yacc=994,awk=629,sh=90
116951	mc-4.5.42	ansic=114406,sh=1996,perl=345,awk=148,csn=56
2270	memprof-0.3.0	ansic=2270
44323	mgetty-1.1.21	ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
27040	mikmod-3.1.6	ansic=26975,sh=55,awk=10
343	mingetty-0.9.4	ansic=343
12633	minicom-1.83.0	ansic=12503,sh=130
537	minlabel-1.2	ansic=537
314	mkbootdisk-1.2.5	sh=314
288	mkinitrd-2.4.1	sh=288
8939	mkisofs-1.12b5	ansic=8939
222	mkkickstart-2.1	sh=222
85	mktemp-1.5	ansic=85
15087	mm2.7	ansic=8044,csn=6924,sh=119
15638	mod_perl-1.21	perl=10278,ansic=5124,sh=236
82	modemtool-1.21	python=73,sh=9
11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
830	mouseconfig-4.4	ansic=830
2773	mpage-2.4	ansic=2704,sh=69
15819	mpgl23-0.59r	ansic=14900,asm=919
1361	mt-st-0.5b	ansic=1361
17765	mtools-3.9.6	ansic=16155,sh=1602,sed=8
14587	multimedia	ansic=14577,sh=10
45811	mutt-1.0.1	ansic=45574,sh=237
1088	nag	perl=1088
2407	nc	ansic=1670,sh=737
31174	ncftp-3.0beta21	ansic=30347,cpp=595,sh=232
1435	ncompress-4.2.4	ansic=1435
28897	ncpfs-2.2.0.17	ansic=28689,sh=182,tcl=26
61324	ncurses-5.0	ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103, sed=100
11633	net-tools-1.54	ansic=11531,sh=102
1634	netcfg-2.25	python=1632,sh=2
2883	netkit-base-0.16	ansic=2883
506	netkit-bootparamd-0.16	ansic=506
5111	netkit-ftp-0.16	ansic=5111
2048	netkit-ntalk-0.16	ansic=2048
2423	netkit-routed-0.16	ansic=2423
3588	netkit-rsh-0.16	ansic=3588
1857	netkit-rusers-0.16	ansic=1857
294	netkit-rwall-0.16	ansic=294
967	netkit-rwho-0.16	ansic=967
21010	netkit-telnet-0.16	ansic=14796,cpp=6214
1531	netkit-tftp-0.16	ansic=1531
3962	netkit-timed-0.16	ansic=3962
592	netscape-4.72	sh=592
7041	newt-0.50.8	ansic=6526,python=515
14299	nfs-utils-0.1.6	ansic=14107,sh=165,perl=27
52574	nmh-1.0.3	ansic=50698,sh=1785,awk=74,sed=17
9873	nss_ldap-105	ansic=9784,perl=89
240	open-1.4	ansic=240

60302	openldap-1.2.9	ansic=58078,sh=1393,perl=630,python=201
41388	p2c-1.22	ansic=38788,pascal=2499,perl=101
20433	pam-0.72	ansic=18936,yacc=634,sh=482,perl=321,lex=60
1280	pam_krb5-1	ansic=1280
1194	passwd-0.64.1	ansic=1194
6611	patch-2.5	ansic=6561,sed=50
3855	pciutils-2.1.5	ansic=3800,sh=55
24773	pdcksh-5.2.14	ansic=23599,perl=945,sh=189,sed=40
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
3885	phhttpd-0.1.0	ansic=3859,sh=26
35136	php-2.0.1	ansic=33991,sh=1056,awk=89
109824	php-3.0.15	ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
6496	pidentd-3.0.10	ansic=6475,sh=21
32277	pilot-link.0.9.3	ansic=26513,java=2162,cpp=1689,perl=971,yacc=660, python=268,tcl=14
127536	pine4.21	ansic=126678,sh=766,csch=62,perl=30
10088	piranha	ansic=10048,sh=40
3219	playmidi-2.4	ansic=3217,sed=2
34882	pmake	ansic=34599,sh=184,awk=58,sed=41
6025	pnm2ppa	ansic=5708,sh=317
1099	popt-1.4	ansic=1039,sh=60
897	portmap_4	ansic=897
247026	postgresql-6.5.3	ansic=207735,yacc=10718,java=8835,tcl=7709,sh=7399, lex=1642,perl=1206,python=959,cpp=746,asm=70,csch=5,sed=2
62922	ppp-2.3.11	ansic=61756,sh=996,exp=82,perl=44,csch=44
2200	printtool	tcl=2200
1226	procinfo-17	ansic=1145,perl=81
9927	procmail-3.14	ansic=8090,sh=1837
9961	procps-2.0.6	ansic=9959,sh=2
1146	prtconf-1.3	ansic=1146
8572	psgml-1.2.1	lisp=8572
1630	psmisc	ansic=1624,sh=6
1856	pump-0.7.8	ansic=1856
9551	pwdb-0.61	ansic=9488,sh=63
8766	pxe-linux	cpp=4463,ansic=3622,asm=681
2597	pythonlib-1.23	python=2597
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464, perl=417
159	quickstrip-1.1	ansic=159
3804	quota-2.00-pre3	ansic=3795,sh=9
2424	raidtools-0.90	ansic=2418,sh=6
14269	rcc-5.7	ansic=12209,sh=2060
132	rdate-1.0	ansic=132
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
14941	readline-2.2.1	ansic=11375,sh=1890,perl=1676
0	redhat-logos	(none)
4024	rep-gtk-0.8	ansic=2905,lisp=971,sh=148
213	rhmask	ansic=213
445	rhs-printfilters-1.63	sh=443,ansic=2
0	rootfiles	(none)
13504	rp3-1.0.7	cpp=10416,ansic=2957,sh=131
39861	rpm-3.0.4	ansic=36994,sh=1505,perl=1355,python=7
15456	rpm2html-1.2	ansic=15334,perl=122
6021	rpmfind-1.4	ansic=6021
816	rpmlint-0.8	python=813,sh=3
14350	rsync-2.4.1	ansic=13986,perl=179,sh=126,awk=59
13779	rxvt-2.6.1	ansic=13779
255	sag-0.6-html	perl=255
92964	samba-2.0.6	ansic=88308,sh=3557,perl=831,awk=158,csch=110
6172	sash-3.4	ansic=6172
22373	sawmill-0.24	ansic=11038,lisp=8172,sh=3163
30122	scheme-3.2	lisp=19483,ansic=10515,sh=124
29730	screen-3.9.5	ansic=28156,sh=1574
7740	sed-3.02	ansic=7301,sed=359,sh=80
42880	sendmail-8.9.3	ansic=40364,perl=1737,sh=779

742	setserial-2.15	ansic=742
67	setup-1.2	ansic=67
0	setup-2.1.8	(none)
0	sgml-common-0.1	(none)
62137	sgml-tools-1.0.9	cpp=38543,ansic=19185,perl=2866,lex=560,sh=532, lisp=309,awk=142
17939	sh-utils-2.0	ansic=13366,sh=3027,yacc=871,perl=675
25236	shadow-19990827	ansic=23464,sh=883,yacc=856,perl=33
56	shaper	ansic=56
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
13100	silo-0.9.8	ansic=10485,asm=2615
28118	slang	ansic=28118
2447	sliplogin-2.1.1	ansic=2256,sh=143,perl=48
1883	slocate-2.1	ansic=1802,sh=81
28449	slrn-0.9.6.2	ansic=28438,sh=11
2490	sndconfig-0.43	ansic=2490
0	solemul-1.1	(none)
17259	sox-12.16	ansic=16659,sh=600
52	sparc32-1.1	ansic=52
0	specspo-6.2	(none)
68884	squid-2.3.STABLE1	ansic=66305,sh=1570,perl=1009
280	stat-1.5	ansic=280
131	statserial-1.1	ansic=121,sh=10
33203	strace-4.2	ansic=30891,sh=1988,perl=280,lisp=44
1076	stylesheets-0.13rh	perl=888,sh=188
54431	svglib-1.4.1	ansic=53725,asm=630,perl=54,sh=22
953	switchdesk-2.1	ansic=314,perl=287,cpp=233,sh=119
302	symlinks-1.2	ansic=302
4038	syslogd-1.3-31	ansic=3741,perl=158,sh=139
265	sysreport-1.0	sh=265
6033	sysvinit-2.78	ansic=5256,sh=777
15851	taper-6.9a	ansic=15851
14255	tar-1.13.17	ansic=13014,lisp=592,sh=538,perl=111
327021	tcltk-8.0.5	ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762, awk=273,perl=265
3654	tcp_wrappers_7.6	ansic=3654
30061	tcpdump-3.4	ansic=29208,yacc=236,sh=211,lex=206,awk=184,csch=16
46312	tcsh-6.09.00	ansic=43544,sh=921,lisp=669,perl=593,csch=585
193916	teTeX-1.0	ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546, yacc=1507,awk=522,lex=323,sed=297,asm=139,csch=47,lisp=29
797	termcap-2.0.8	ansic=797
28186	texinfo-4.0	ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
36338	textutils-2.0a	ansic=18949,sh=16111,perl=1218,sed=60
37360	tiff-v3.5.4	ansic=32734,sh=4054,cpp=572
1452	time-1.7	ansic=1395,sh=57
346	timeconfig-3.0.3	ansic=318,sh=28
367	timetool-2.7.3	tcl=367
49370	tin-1.4.2	ansic=47763,sh=908,yacc=699
548	tksysv-1.1	tcl=526,sh=22
463	tmpwatch-2.2	ansic=311,sh=152
7	trXFree86-2.1.2	tcl=7
1473	traceroute-1.4a5	ansic=1436,awk=37
15691	transfig.3.2.1	ansic=15643,sh=38,csch=10
728	tree-1.2	ansic=728
32767	trn-3.6	ansic=25264,sh=6843,yacc=660
1013	trojka	ansic=1013
72726	ucd-snmp-4.1.1	ansic=64411,perl=5558,sh=2757
2141	unarj-2.43	ansic=2141
2065	units-1.55	ansic=1963,perl=102
46159	unzip-5.40	ansic=40977,cpp=3778,asm=1271,sh=133
2324	up2date-1.13	python=2324
1621	urlview-0.7	ansic=1515,sh=106
0	urw-fonts-2.0	(none)
2459	usermode-1.20	ansic=2459
585	usernet-1.0.9	ansic=585

0	users-guide-1.0.72	(none)
222	utempter-0.5.2	ansic=222
39160	util-linux-2.10f	ansic=38627,sh=351,perl=65,csch=62,sed=55
55667	uucp-1.06.1	ansic=52078,sh=3400,perl=189
113241	vim-5.6	ansic=111724,awk=683,sh=469,perl=359,csch=6
2879	vixie-cron-3.0.1	ansic=2866,sh=13
368	vlock-1.3	ansic=368
73817	w3c-libwww-5.2.8	ansic=64754,sh=4678,cpp=3181,perl=1204
13586	wget-1.5.3	ansic=13509,perl=54,sh=23
2268	which-2.9	ansic=1398,sh=870
14105	wmakerconf-2.1	ansic=13620,perl=348,sh=137
1977	wmconfig-0.9.8	ansic=1941,sh=36
11	words-2	sh=11
19971	wu-ftpd-2.6.0	ansic=17572,yacc=1774,sh=421,perl=204
0	wvdial-1.41	(none)
36239	x11amp-0.9-alpha3	ansic=31686,sh=4200,asm=353
27022	x3270-3.1.1	ansic=26456,sh=478,exp=88
1129	xbill-2.0	cpp=1129
21593	xboard-4.0.5	ansic=20640,lex=904,sh=41,csch=5,sed=3
18020	xboing	ansic=18006,sh=14
29091	xchat-1.4.0	ansic=28894,perl=121,python=53,sh=23
4895	xcpustate-2.5	ansic=4895
3860	xdaliclock-2.18	ansic=3837,sh=23
57217	xfig.3.2.3-beta-1	ansic=57212,csch=5
28261	xfishtank-2.1tp	ansic=28261
7095	xgammon-0.98	ansic=6506,lex=589
2791	xjewel-1.6	ansic=2791
113272	xlispstat-3-52-17	ansic=91484,lisp=21769,sh=18,csch=1
35812	xloadimage.4.1	ansic=35705,sh=107
94637	xlockmore-4.15	ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
987	xmailbox-2.5	ansic=987
38927	xmms-1.0.1	ansic=38366,asm=398,sh=163
11299	xmorph-1999dec12	ansic=10783,tcl=516
65722	xntp3-5.93	ansic=60190,perl=3633,sh=1445,awk=417,asm=37
6592	xosview-1.7.1	cpp=6205,ansic=367,awk=20
25479	xpaint	ansic=25456,sh=23
9942	xpat2-1.04	ansic=9942
85383	xpdf-0.90	cpp=60427,ansic=21400,sh=3556
75257	xpilot-4.1.0	ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68
7826	xpm-3.4k	ansic=7750,sh=39,cpp=37
34772	xpuzzles-5.4.1	ansic=34772
25994	xrn-9.02	ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31,csch=13
89959	xscreensaver-3.23	ansic=88488,perl=1070,sh=401
301	xsri-1.0	ansic=301
787	xsystinfo-1.7	ansic=787
236	xtoolwait-1.2	ansic=236
3096	xtrojka123	ansic=3087,sh=9
8540	xxgdb-1.12	ansic=8540
3438	yp-tools-2.4	ansic=3415,sh=23
3245	ypbind-3.3	ansic=1793,sh=1452
12124	ypserv-1.3.9	ansic=11622,sh=460,perl=42
5975	ytalk-3.1	ansic=5975
35554	zip-2.3	ansic=32108,asm=3446
4087	zlib-1.1.3	ansic=2815,asm=712,cpp=560
38453	zsh-3.0.7	ansic=36208,sh=1763,perl=331,awk=145,sed=6

ansic:	14218806	(80.55%)
cpp:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)

python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = \$ 614421924.71

SLOC	Directory	SLOC-by-Language (Sorted)
53141	AfterStep-1.8.0	ansic=50898,perl=1168,sh=842,cpp=233
140130	AfterStep-APPS-20000124	ansic=135806,sh=3340,cpp=741,perl=243
16	AnotherLevel-1.0.1	sh=16
834	ElectricFence-2.1	ansic=834
15108	GXedit1.23	ansic=15019,sh=89
121878	ImageMagick-4.2.9	ansic=99383,sh=11143,cpp=8870,perl=2024,tcl=458
1020	MAKEDEV-2.5.2	sh=1020
231072	Mesa	ansic=195796,cpp=17717,asm=13467,sh=4092
38548	ORBit-0.5.0	ansic=35656,yacc=1750,sh=776,lex=366
200628	Python-1.5.2	python=100935,ansic=96323,lisp=2353,sh=673,perl=342,sed=2
15967	SVGATextMode-1.9-src	ansic=15079,yacc=340,sh=294,lex=227,sed=15,asm=12
79997	WindowMaker-0.61.1	ansic=77924,sh=1483,perl=371,lisp=219
18885	X11R6-contrib-3.3.2	ansic=18616,lex=161,yacc=97,sh=11
1291745	XFree86-3.3.6	ansic=1246420,asm=14913,sh=13433,tcl=8362,cpp=4358,yacc=2710,perl=711,awk=393,lex=383,sed=57,csch=5
0	XFree86-ISO8859-2-1.0 (none)	
26608	Xaw3d-1.3	ansic=26235,yacc=247,lex=126
9741	Xconfigurator-4.3.5	ansic=9578,perl=125,sh=32,python=6
6976	aboot-0.5	ansic=6680,asm=296
5383	acct-6.3.2	ansic=5016,cpp=287,sh=80
1653	adjtimex-1.9	ansic=1653
45589	am-utils-6.0.3	ansic=33389,sh=8950,perl=2421,lex=454,yacc=375
91213	anaconda-6.2.2	ansic=74303,python=13657,sh=1583,yacc=810,lex=732,perl=128
1143	anacron-2.1	ansic=1143
77873	apache_1.3.12	ansic=69191,sh=6781,perl=1846,cpp=55
3012	apmd	ansic=2617,sh=395
9666	ash-linux-0.2	ansic=9445,sh=221
3084	at-3.1.7	ansic=1442,sh=1196,yacc=362,lex=84
441	audioctl	ansic=441
12593	audiofile-0.1.9	sh=6440,ansic=6153
4367	aumix-1.30.1	ansic=4095,sh=179,sed=93
1182	auth_ldap-1.4.0	ansic=1182
1075	authconfig-3.0.3	ansic=1075
2758	autoconf-2.13	sh=2226,perl=283,exp=167,ansic=82
3625	autofs-3.1.4	ansic=2862,sh=763
14363	automake-1.4	perl=10622,sh=3337,ansic=404
2705	autorun-2.61	sh=1985,cpp=720
6234	awesfx-0.4.3a	ansic=6234
47067	bash-1.14.7	ansic=41654,sh=3140,yacc=2197,asm=48,awk=28
68560	bash-2.03	ansic=56758,sh=7264,yacc=2808,perl=1730
17682	bc-1.05	ansic=9186,sh=7236,yacc=967,lex=293
261	bdflush-1.5	ansic=202,asm=59
290	biff+comsat-0.16	ansic=290
155035	bind-8.2.2_P5	ansic=131946,sh=10068,perl=7607,yacc=2231,cpp=1360,csch=848,awk=753,lex=222
467120	binutils-2.9.5.0.22	ansic=407352,asm=27575,exp=12265,sh=7398,yacc=5606,cpp=4454,lex=1479,sed=557,lisp=394,awk=24,perl=16
9699	bison-1.28	ansic=9650,sh=49
68997	blt2.4g	ansic=58630,tcl=10215,sh=152
1272	bsd-finger-0.16	ansic=1272
2486	bug-buddy-0.7	ansic=2486
6550	byacc-1.9	ansic=5520,yacc=1030
4996	bzip2-0.9.5d	ansic=4996
0	caching-nameserver-6.2 (none)	
1842	cdecl-2.5	ansic=1002,yacc=765,lex=75
2661	cdp-0.33	ansic=2661
7227	cdparanoia-III-alpha9.6	ansic=6006,sh=1221
50970	cdrecord-1.8	ansic=48595,sh=2177,perl=194,sed=4
717	chkconfig-1.1.2	ansic=717
343	chkfontpath-1.7	ansic=343
1984	cleanfeed-0.95.7b	perl=1984
15522	console-tools-0.3.3	ansic=13335,yacc=986,sh=800,lex=291,perl=110
16785	control-center-1.0.51	ansic=16659,sh=126
404	control-panel-3.13	ansic=319,tcl=85
7617	cpio-2.4.2	ansic=7598,sh=19
9607	cproto-4.6	ansic=7600,lex=985,yacc=761,sh=261
1987	cracklib,2.7	ansic=1919,perl=46,sh=22
9263	ctags-3.4	ansic=9240,sh=23
88128	cvs-1.10.7	ansic=68303,sh=17909,perl=902,yacc=826,csch=181,lisp=7
1290	cxhextris	ansic=1290
0	desktop-backgrounds-1.1 (none)	
0	dev-2.7.18 (none)	
16266	dhcp-2.0	ansic=15328,sh=938
3051	dhcpcd-1.3.18-pl3	ansic=2771,sh=280
3433	dialog-0.6	ansic=2834,perl=349,sh=250
616	diffstat-1.27	ansic=616
10914	diffutils-2.7	ansic=10914
8303	dip-3.3.7o	ansic=8207,sh=96
0	docbook-3.1 (none)	
3675	dosfstools-2.2	ansic=3675

10187	dump-0.4b15	ansic=9422,sh=760,sed=5
28169	e2fsprogs-1.18	ansic=27250,awk=437,sh=339,sed=121,perl=22
7427	ed-0.2	ansic=7263,sh=164
7030	ee-0.3.11	ansic=7007,sh=23
5526	efax-0.9	ansic=4570,sh=956
720112	egcs-1.1.2	
ansic=598682	cpp=75206,sh=14307,asm=11462,yacc=7988,lisp=7252,exp=2887,fortran=1515,objc=482,sed=313,perl=18	
657	eject-2.0.2	ansic=657
480	elftoaout-2.2	ansic=480
42746	elm2.5.3	ansic=32931,sh=9774,awk=41
625073	emacs-20.5	lisp=453647,ansic=169624,perl=884,sh=652,asm=253,csch=9,sed=4
51592	enlightenment-0.15.5	ansic=51569,sh=23
11721	enlightenment-conf-0.15	ansic=6232,sh=5489
23666	enscript-1.6.1	ansic=22365,lex=429,perl=308,sh=291,yacc=164,lisp=109
7615	esound-0.2.17	ansic=7387,sh=142,csch=86
332	ethtool-1.0	ansic=332
36243	exmh-2.1.1	tcl=35844,perl=316,sh=49,exp=34
3415	ext2ed-0.1	ansic=3415
78787	extace-1.2.15	ansic=66571,sh=9322,perl=2894
1765	fbset-2.1	ansic=1401,yacc=130,lex=121,perl=113
17271	fetchmail-5.3.1	
ansic=13441	python=1490,sh=1246,yacc=411,perl=321,lex=238,awk=124	
2647	file-3.28	ansic=2601,perl=46
34768	fileutils-4.0p	ansic=31324,sh=2042,yacc=841,perl=561
11404	findutils-4.1	ansic=11160,sh=173,exp=71
14774	flex-2.5.4	ansic=13011,lex=1045,yacc=605,awk=72,sh=29,sed=12
2455	fnlib-0.4	ansic=2432,sh=23
1604	fortune-mod-9708	ansic=1604
51813	freetype-1.3.1	ansic=48929,sh=2467,cpp=351,csch=53,perl=13
69265	fvwm-2.2.4	ansic=63496,cpp=2463,perl=1835,sh=723,yacc=596,lex=152
107	fwhois-1.00	ansic=107
138024	gated-3-5-11	
ansic=126846	yacc=7799,sh=1554,lex=877,awk=666,csch=235,sed=35,lisp=12	
26363	gawk-3.0.4	ansic=19871,awk=2519,yacc=2046,sh=1927
20078	gd1.3	ansic=19946,perl=132
652087	gdb-19991004	
ansic=587542	exp=37737,sh=9630,cpp=6735,asm=4139,yacc=4117,lisp=1820,sed=220,awk=142,fortran=5	
3315	gdbm-1.8.0	ansic=3290,cpp=25
5744	gdm-2.0beta2	ansic=5632,sh=112
8356	gedit-0.6.1	ansic=8225,sh=131
549	genromfs-0.3	ansic=549
17750	gettext-0.10.35	ansic=13414,lisp=2030,sh=1983,yacc=261,perl=53,sed=9
2631	getty_ps-2.0.7j	ansic=2631
9138	gftp-2.0.6a	ansic=9138
0	ghostscript-fonts-5.50	(none)
770	giftrans-1.12.2	ansic=770
235702	gimp-1.0.4	ansic=225211,lisp=8497,sh=1994
0	gimp-data-extras-1.0.0	(none)
18151	git-4.3.19	ansic=16166,sh=1985
2835	gkermi-1.0	ansic=2835
54603	glade-0.5.5	ansic=49545,sh=5058
18835	glib-1.2.6	ansic=18702,sh=133
415026	glibc-2.1.3	
ansic=378753	asm=30644,sh=2520,cpp=1704,awk=910,perl=464,sed=16,csch=15	
24583	gmp-2.0.2	ansic=17888,asm=5252,sh=1443
0	gnome-audio-1.0.0	(none)
72425	gnome-core-1.0.55	ansic=72230,perl=141,sh=54
41205	gnome-games-1.0.51	ansic=31191,lisp=6966,cpp=3048
2531	gnome-kerberos-0.2	ansic=2531
128672	gnome-libs-1.0.55	ansic=125373,sh=2178,perl=667,awk=277,lisp=177
2163	gnome-linuxconf-0.25	ansic=2163
6827	gnome-media-1.0.51	ansic=6827
12463	gnome-objc-1.0.2	objc=12365,sh=86,ansic=12
30438	gnome-pim-1.0.55	ansic=28665,yacc=1773
24270	gnome-python-1.0.51	python=14331,ansic=9791,sh=148
19500	gnome-utils-1.0.50	ansic=18099,yacc=824,lisp=577
10404	gnorpm-0.9	ansic=10404
15143	gnotepad+-1.1.4	ansic=15143
14871	gnuchess-4.0.pl80	ansic=14584,sh=258,csch=29
116615	gnnumeric-0.48	ansic=115592,yacc=600,lisp=191,sh=142,perl=67,python=23
54935	gnupg-1.0.1	ansic=48884,asm=4586,sh=1465
45378	gnuplot-3.7.1	ansic=43276,lisp=661,asm=539,objc=387,csch=297,perl=138,sh=80
4430	gperf-2.7	cpp=2947,exp=745,ansic=695,sh=43
4542	gpgp-0.4	ansic=4441,sh=101
9725	gpm-1.18.1	ansic=8107,yacc=1108,lisp=221,sh=209,awk=74,sed=6
10271	gqview-0.7.0	ansic=10271
10013	grep-2.4	ansic=9852,sh=103,awk=49,sed=9
70260	groff-1.15	cpp=59453,ansic=5276,yacc=2957,asm=1866,perl=397,sh=265,sed=46
199982	gs5.50	ansic=195491,cpp=2266,asm=968,sh=751,lisp=405,perl=101
138118	gtk+-1.2.6	ansic=137006,perl=479,sh=352,awk=274,lisp=7
35397	gtk-engines-0.10	ansic=20636,sh=14761

8491	gtop-1.0.5	ansic=8151,cpp=340
45485	guile-1.3	ansic=38823,lisp=4626,asm=1514,sh=310,awk=162,csch=50
25915	gv-3.5.8	ansic=25821,sh=94
6306	gzip-1.2.4a	ansic=5813,asm=458,sh=24,perl=11
1229	hdparm-3.6	ansic=1229
221	hellas	sh=179,perl=42
491	helptool-2.4	perl=288,tcl=203
20125	ical-2.2	cpp=12651,tcl=6763,sh=624,perl=60,ansic=27
61688	imap-4.7	ansic=61628,sh=60
49325	imlib-1.9.7	ansic=49260,sh=65
5981	indent-2.2.5	ansic=5958,sh=23
0	indexhtml-6.2	(none)
3929	initscripts-5.00	sh=2035,ansic=1866,csch=28
81719	inn-2.2.2	ansic=62403,perl=10485,sh=5465,awk=1567,yacc=1547,lex=249,tcl=3
0	install-guide-3.2.html	(none)
47	intimed-1.10	ansic=47
3651	ipchains-1.3.9	ansic=2767,sh=884
13241	iproute2	ansic=12139,sh=1002,perl=100
6646	iputils	ansic=6646
255	ipvsadm-1.1	ansic=255
37515	ircii-4.4	ansic=36647,sh=852,lex=16
6043	irda-utils-0.9.10	ansic=5697,sh=263,perl=83
5960	isapnptools-1.21	ansic=4394,yacc=1383,perl=123,sh=60
5594	isdn-config	cpp=3058,sh=2228,perl=308
87940	isdn4k-utils	ansic=78752,perl=3369,sh=3089,cpp=2708,tcl=22
1941	isicom	ansic=1898,sh=43
14912	ispell-3.1	
ansic=8380,lisp=3372,yacc=1712,cpp=585,objc=385,csch=221,sh=157,perl=85,sed=15		
131372	jade-1.2.1	cpp=120611,ansic=8228,sh=2150,perl=378,sed=5
0	jadetex	(none)
29315	jed	ansic=29315
71810	jikes	cpp=71452,java=358
19065	joe	ansic=18841,asm=224
6090	joystick-1.2.15	ansic=6086,sh=4
12313	jpeg-6a	ansic=12313
844	jpeg-6b	sh=844
133193	kaffe-1.0.5	java=65275,ansic=62125,cpp=3923,perl=972,sh=814,asm=84
368	kbdconfig-1.9.2.4	ansic=368
24910	kdeadmin	cpp=19919,sh=3936,perl=1055
138931	kdebase	cpp=113971,ansic=23016,perl=1326,sh=618
27860	kdegames	cpp=27507,ansic=340,sh=13
68453	kdegraphics	cpp=34208,ansic=29347,sh=4898
80343	kdelibs	cpp=71217,perl=5075,ansic=3660,yacc=240,lex=116,sh=35
48223	kdemultimedia	ansic=24248,cpp=22275,tcl=1004,sh=621,perl=73,awk=2
93940	kdenetwork	cpp=80075,ansic=7422,perl=6260,sh=134,tcl=49
60429	kdesupport	ansic=42421,cpp=17810,sh=173,awk=13,csch=12
2810	kdetoys	cpp=2618,ansic=192
52808	kdeutils	cpp=41365,ansic=9693,sh=1434,awk=311,sed=5
6072	kdoc	perl=6010,sh=45,cpp=17
809	kdpms-0.2.8	cpp=809
347	kernelcfg-0.5	python=341,sh=6
31994	korganizer	cpp=23402,ansic=5884,yacc=2271,perl=375,lex=61,sh=1
10958	kpackage-1.3.10	cpp=8863,sh=1852,ansic=124,perl=119
15868	kpilot-3.1b9	cpp=8613,ansic=5640,yacc=1615
1049	kpppload-1.04	cpp=1044,sh=5
99066	krb4-1.0	
ansic=84077,asm=5163,cpp=3775,perl=2508,sh=1765,yacc=1509,lex=236,awk=33		
222220	krb5-1.1.1	
ansic=192822,exp=19364,sh=4829,yacc=2476,perl=1528,awk=393,python=348,lex=190,csch=147,sed=123		
23838	kterm-6.2.0	ansic=23838
0	kudzu-0.36	(none)
100951	lam-6.3.1	ansic=86177,cpp=10569,sh=3677,perl=322,fortran=187,csch=19
9731	ld.so-1.9.5	ansic=6960,asm=2401,sh=370
874	ldconfig-1999-02-21	ansic=874
14039	less-346	ansic=14032,awk=7
10267	libPropList-0.9.1	sh=5974,ansic=3982,lex=172,yacc=139
4792	libelf-0.6.4	ansic=3310,sh=1482
2645	libghttp-1.0.4	ansic=2645
7859	libglade-0.11	ansic=5898,sh=1809,python=152
102674	libgr-2.0.13	ansic=99647,sh=2438,csch=589
14516	libgtop-1.0.6	ansic=13768,perl=653,sh=64,asm=31
22011	libpng-1.0.5	ansic=22011
27117	librep-0.10	ansic=19381,lisp=5385,sh=2351
5115	libtool-1.3.4	sh=3374,ansic=1741
12639	libungif-4.1.0	ansic=12381,sh=204,perl=54
26146	libxml-1.8.6	ansic=26069,sh=77
7255	lilo	ansic=3522,asm=2557,sh=740,perl=433,cpp=3
1526722	linux	
ansic=1462165,asm=59574,sh=2860,perl=950,tcl=414,yacc=324,lex=230,awk=133,sed=72		
69246	linux-86	ansic=63328,asm=5276,sh=642
104032	linuxconf-1.17r2	cpp=93139,perl=4570,sh=2984,java=2741,ansic=598

23	locale-ja-9	sh=23
497	locale_config-0.2	ansic=497
1525	logrotate-3.3.2	ansic=1524,sh=1
26673	lout-3.17	ansic=26673
682	lpg	perl=682
6877	lpr	ansic=6842,sh=35
11790	lrzsz-0.12.20	ansic=9512,sh=1263,exp=1015
6116	lsk	ansic=5325,sh=791
56093	lsof_4.47	ansic=50268,sh=4753,perl=856,awk=214,asm=2
3896	ltrace-0.3.10	ansic=2986,sh=854,awk=56
119613	lynx2-8-3	ansic=117385,sh=1860,perl=340,csch=28
6391	m4-1.4	ansic=5993,lisp=243,sh=155
12657	macutils	ansic=12657
2580	magicdev-0.2.7	ansic=2580
0	mailcap-2.0.6	(none)
6968	mailx-8.1.1	ansic=6963,sh=5
22279	make-3.78.1	ansic=19287,sh=2029,perl=963
4780	make-3.78.1_pvm-0.5	ansic=4780
9801	man-1.5hl	ansic=7377,sh=1802,perl=317,awk=305
245	man-pages-1.28	sh=244,sed=1
24387	mars_nwe	ansic=24158,sh=229
14427	mawk-1.2.2	ansic=12714,yacc=994,awk=629,sh=90
116951	mc-4.5.42	ansic=114406,sh=1996,perl=345,awk=148,csch=56
2270	memprof-0.3.0	ansic=2270
44323	mgetty-1.1.21	ansic=33757,perl=5889,sh=3638,tcl=756,lisp=283
27040	mikmod-3.1.6	ansic=26975,sh=55,awk=10
343	mingetty-0.9.4	ansic=343
12633	minicom-1.83.0	ansic=12503,sh=130
537	minlabel-1.2	ansic=537
314	mkbootdisk-1.2.5	sh=314
288	mkinitrd-2.4.1	sh=288
8939	mkisofs-1.12b5	ansic=8939
222	mkkickstart-2.1	sh=222
85	mktemp-1.5	ansic=85
15087	mm2.7	ansic=8044,csch=6924,sh=119
15638	mod_perl-1.21	perl=10278,ansic=5124,sh=236
82	modemtool-1.21	python=73,sh=9
11775	modutils-2.3.9	ansic=9309,sh=1620,lex=484,yacc=362
830	mouseconfig-4.4	ansic=830
2773	mpage-2.4	ansic=2704,sh=69
15819	mpg123-0.59r	ansic=14900,asm=919
1361	mt-st-0.5b	ansic=1361
17765	mtools-3.9.6	ansic=16155,sh=1602,sed=8
14587	multimedia	ansic=14577,sh=10
45811	mutt-1.0.1	ansic=45574,sh=237
1088	nag	perl=1088
2407	nc	ansic=1670,sh=737
31174	ncftp-3.0beta21	ansic=30347,cpp=595,sh=232
1435	ncompress-4.2.4	ansic=1435
28897	ncpfs-2.2.0.17	ansic=28689,sh=182,tcl=26
61324	ncurses-5.0	ansic=45856,ada=8217,cpp=3720,sh=2822,awk=506,perl=103,sed=100
11633	net-tools-1.54	ansic=11531,sh=102
1634	netcfg-2.25	python=1632,sh=2
2883	netkit-base-0.16	ansic=2883
506	netkit-bootparamd-0.16	ansic=506
5111	netkit-ftp-0.16	ansic=5111
2048	netkit-ntalk-0.16	ansic=2048
2423	netkit-routed-0.16	ansic=2423
3588	netkit-rsh-0.16	ansic=3588
1857	netkit-rusers-0.16	ansic=1857
294	netkit-rwall-0.16	ansic=294
967	netkit-rwho-0.16	ansic=967
21010	netkit-telnet-0.16	ansic=14796,cpp=6214
1531	netkit-tftp-0.16	ansic=1531
3962	netkit-timed-0.16	ansic=3962
592	netscape-4.72	sh=592
7041	newt-0.50.8	ansic=6526,python=515
14299	nfs-utils-0.1.6	ansic=14107,sh=165,perl=27
52574	nmh-1.0.3	ansic=50698,sh=1785,awk=74,sed=17
9873	nss_ldap-105	ansic=9784,perl=89
240	open-1.4	ansic=240
60302	openldap-1.2.9	ansic=58078,sh=1393,perl=630,python=201
41388	p2c-1.22	ansic=38788,pascal=2499,perl=101
20433	pam-0.72	ansic=18936,yacc=634,sh=482,perl=321,lex=60
1280	pam_krb5-1	ansic=1280
1194	passwd-0.64.1	ansic=1194
6611	patch-2.5	ansic=6561,sed=50
3855	pciutils-2.1.5	ansic=3800,sh=55
24773	pdsh-5.2.14	ansic=23599,perl=945,sh=189,sed=40
206237	perl5.005_03	perl=94712,ansic=89366,sh=15654,lisp=5584,yacc=921
3885	phhttpd-0.1.0	ansic=3859,sh=26

35136	php-2.0.1	ansic=33991,sh=1056,awk=89
109824	php-3.0.15	ansic=105901,yacc=1887,sh=1381,perl=537,awk=90,cpp=28
6496	pidentd-3.0.10	ansic=6475,sh=21
32277	pilot-link-0.9.3	
ansic=26513,	java=2162,cpp=1689,perl=971,yacc=660,python=268,tcl=14	
127536	pine4.21	ansic=126678,sh=766,csch=62,perl=30
10088	piranha	ansic=10048,sh=40
3219	playmidi-2.4	ansic=3217,sed=2
34882	pmake	ansic=34599,sh=184,awk=58,sed=41
6025	pnm2ppa	ansic=5708,sh=317
1099	popt-1.4	ansic=1039,sh=60
897	portmap_4	ansic=897
247026	postgresql-6.5.3	
ansic=207735,	yacc=10718,java=8835,tcl=7709,sh=7399,lex=1642,perl=1206,python=959,cpp=746,asm=70,csch=5,sed=2	
62922	ppp-2.3.11	ansic=61756,sh=996,exp=82,perl=44,csch=44
2200	printtool	tcl=2200
1226	procinfo-17	ansic=1145,perl=81
9927	procmail-3.14	ansic=8090,sh=1837
9961	procps-2.0.6	ansic=9959,sh=2
1146	prtconf-1.3	ansic=1146
8572	psgml-1.2.1	lisp=8572
1630	psmisc	ansic=1624,sh=6
1856	pump-0.7.8	ansic=1856
9551	pwdb-0.61	ansic=9488,sh=63
8766	pxe-linux	cpp=4463,ansic=3622,asm=681
2597	pythonlib-1.23	python=2597
205082	qt-2.1.0-beta1	cpp=180866,ansic=20513,yacc=2284,sh=538,lex=464,perl=417
159	quickstrip-1.1	ansic=159
3804	quota-2.00-pre3	ansic=3795,sh=9
2424	raidtools-0.90	ansic=2418,sh=6
14269	rcs-5.7	ansic=12209,sh=2060
132	rdate-1.0	ansic=132
9465	rdist-6.1.5	ansic=8306,sh=553,yacc=489,perl=117
14941	readline-2.2.1	ansic=11375,sh=1890,perl=1676
0	redhat-logos	(none)
4024	rep-gtk-0.8	ansic=2905,lisp=971,sh=148
213	rhmask	ansic=213
445	rhs-printfilters-1.63	sh=443,ansic=2
0	rootfiles	(none)
13504	rp3-1.0.7	cpp=10416,ansic=2957,sh=131
39861	rpm-3.0.4	ansic=36994,sh=1505,perl=1355,python=7
15456	rpm2html-1.2	ansic=15334,perl=122
6021	rpmfind-1.4	ansic=6021
816	rpmlint-0.8	python=813,sh=3
14350	rsync-2.4.1	ansic=13986,perl=179,sh=126,awk=59
13779	rxvt-2.6.1	ansic=13779
255	sag-0.6-html	perl=255
92964	samba-2.0.6	ansic=88308,sh=3557,perl=831,awk=158,csch=110
6172	sash-3.4	ansic=6172
22373	sawmill-0.24	ansic=11038,lisp=8172,sh=3163
30122	scheme-3.2	lisp=19483,ansic=10515,sh=124
29730	screen-3.9.5	ansic=28156,sh=1574
7740	sed-3.02	ansic=7301,sed=359,sh=80
42880	sendmail-8.9.3	ansic=40364,perl=1737,sh=779
742	setserial-2.15	ansic=742
67	setup-1.2	ansic=67
0	setup-2.1.8	(none)
0	sgml-common-0.1	(none)
62137	sgml-tools-1.0.9	
cpp=38543,	ansic=19185,perl=2866,lex=560,sh=532,lisp=309,awk=142	
17939	sh-utils-2.0	ansic=13366,sh=3027,yacc=871,perl=675
25236	shadow-19990827	ansic=23464,sh=883,yacc=856,perl=33
56	shaper	ansic=56
7570	sharutils-4.2.1	ansic=5511,perl=1741,sh=318
13100	silo-0.9.8	ansic=10485,asm=2615
28118	slang	ansic=28118
2447	sliplogin-2.1.1	ansic=2256,sh=143,perl=48
1883	slocate-2.1	ansic=1802,sh=81
28449	slrn-0.9.6.2	ansic=28438,sh=11
2490	sndconfig-0.43	ansic=2490
0	solemul-1.1	(none)
17259	sox-12.16	ansic=16659,sh=600
52	sparc32-1.1	ansic=52
0	specspo-6.2	(none)
68884	squid-2.3.STABLE1	ansic=66305,sh=1570,perl=1009
280	stat-1.5	ansic=280
131	statserial-1.1	ansic=121,sh=10
33203	strace-4.2	ansic=30891,sh=1988,perl=280,lisp=44
1076	stylesheets-0.13rh	perl=888,sh=188
54431	svglib-1.4.1	ansic=53725,asm=630,perl=54,sh=22
953	switchdesk-2.1	ansic=314,perl=287,cpp=233,sh=119

302	symlinks-1.2	ansic=302
4038	sysklogd-1.3-31	ansic=3741,perl=158,sh=139
265	sysreport-1.0	sh=265
6033	sysvinit-2.78	ansic=5256,sh=777
15851	taper-6.9a	ansic=15851
14255	tar-1.13.17	ansic=13014,lisp=592,sh=538,perl=111
327021	tcltk-8.0.5	
ansic=240093,tcl=71947,sh=8531,exp=5150,yacc=762,awk=273,perl=265		
3654	tcp_wrappers_7.6	ansic=3654
30061	tcpdump-3.4	ansic=29208,yacc=236,sh=211,lex=206,awk=184,csch=16
46312	tcsh-6.09.00	ansic=43544,sh=921,lisp=669,perl=593,csch=585
193916	TeX-1.0	
ansic=166041,sh=10263,cpp=9407,perl=3795,pascal=1546,yacc=1507,awk=522,lex=323,sed=297,asm=139,csch=47,lisp=29		
797	termcap-2.0.8	ansic=797
28186	texinfo-4.0	ansic=26404,sh=841,awk=451,perl=256,lisp=213,sed=21
36338	textutils-2.0a	ansic=18949,sh=16111,perl=1218,sed=60
37360	tiff-v3.5.4	ansic=32734,sh=4054,cpp=572
1452	time-1.7	ansic=1395,sh=57
346	timeconfig-3.0.3	ansic=318,sh=28
367	timetool-2.7.3	tcl=367
49370	tin-1.4.2	ansic=47763,sh=908,yacc=699
548	tksysv-1.1	tcl=526,sh=22
463	tmpwatch-2.2	ansic=311,sh=152
7	trXFree86-2.1.2	tcl=7
1473	traceroute-1.4a5	ansic=1436,awk=37
15691	transfig.3.2.1	ansic=15643,sh=38,csch=10
728	tree-1.2	ansic=728
32767	trn-3.6	ansic=25264,sh=6843,yacc=660
1013	trojka	ansic=1013
72726	ucd-snmp-4.1.1	ansic=64411,perl=5558,sh=2757
2141	unarcj-2.43	ansic=2141
2065	units-1.55	ansic=1963,perl=102
46159	unzip-5.40	ansic=40977,cpp=3778,asm=1271,sh=133
2324	up2date-1.13	python=2324
1621	urlview-0.7	ansic=1515,sh=106
0	urw-fonts-2.0	(none)
2459	usermode-1.20	ansic=2459
585	usenet-1.0.9	ansic=585
0	users-guide-1.0.72	(none)
222	utempter-0.5.2	ansic=222
39160	util-linux-2.10f	ansic=38627,sh=351,perl=65,csch=62,sed=55
55667	uucp-1.06.1	ansic=52078,sh=3400,perl=189
113241	vim-5.6	ansic=111724,awk=683,sh=469,perl=359,csch=6
2879	vixie-cron-3.0.1	ansic=2866,sh=13
368	vlock-1.3	ansic=368
73817	w3c-libwww-5.2.8	ansic=64754,sh=4678,cpp=3181,perl=1204
13586	wget-1.5.3	ansic=13509,perl=54,sh=23
2268	which-2.9	ansic=1398,sh=870
14105	wmakerconf-2.1	ansic=13620,perl=348,sh=137
1977	wmconfig-0.9.8	ansic=1941,sh=36
11	words-2	sh=11
19971	wu-ftpd-2.6.0	ansic=17572,yacc=1774,sh=421,perl=204
0	wvdial-1.41	(none)
36239	x11amp-0.9-alpha3	ansic=31686,sh=4200,asm=353
27022	x3270-3.1.1	ansic=26456,sh=478,exp=88
1129	xbill-2.0	cpp=1129
21593	xboard-4.0.5	ansic=20640,lex=904,sh=41,csch=5,sed=3
18020	xboing	ansic=18006,sh=14
29091	xchat-1.4.0	ansic=28894,perl=121,python=53,sh=23
4895	xcpustate-2.5	ansic=4895
3860	xdaliclock-2.18	ansic=3837,sh=23
57217	xfig.3.2.3-beta-1	ansic=57212,csch=5
28261	xfishtank-2.1tp	ansic=28261
7095	xgammon-0.98	ansic=6506,lex=589
2791	xjewel-1.6	ansic=2791
113272	xlispstat-3-52-17	ansic=91484,lisp=21769,sh=18,csch=1
35812	xloadimage.4.1	ansic=35705,sh=107
94637	xlockmore-4.15	ansic=89816,cpp=1987,tcl=1541,sh=859,java=285,perl=149
987	xmailbox-2.5	ansic=987
38927	xmms-1.0.1	ansic=38366,asm=398,sh=163
11299	xmorph-1999dec12	ansic=10783,tcl=516
65722	xntp3-5.93	ansic=60190,perl=3633,sh=1445,awk=417,asm=37
6592	xosview-1.7.1	cpp=6205,ansic=367,awk=20
25479	xpaint	ansic=25456,sh=23
9942	xpat2-1.04	ansic=9942
85383	xpdf-0.90	cpp=60427,ansic=21400,sh=3556
75257	xpilot-4.1.0	ansic=68669,tcl=3479,cpp=1896,sh=1145,perl=68
7826	xpm-3.4k	ansic=7750,sh=39,cpp=37
34772	xpuzzles-5.4.1	ansic=34772
25994	xrn-9.02	ansic=24686,yacc=888,sh=249,lex=92,perl=35,awk=31,csch=13
89959	xscreensaver-3.23	ansic=88488,perl=1070,sh=401

301	xsri-1.0	ansic=301
787	xsysinfo-1.7	ansic=787
236	xtoolwait-1.2	ansic=236
3096	xtrojka123	ansic=3087,sh=9
8540	xxgdb-1.12	ansic=8540
3438	yp-tools-2.4	ansic=3415,sh=23
3245	ypbind-3.3	ansic=1793,sh=1452
12124	ypserv-1.3.9	ansic=11622,sh=460,perl=42
5975	ytalk-3.1	ansic=5975
35554	zip-2.3	ansic=32108,asm=3446
4087	zlib-1.1.3	ansic=2815,asm=712,cpp=560
38453	zsh-3.0.7	ansic=36208,sh=1763,perl=331,awk=145,sed=6

ansic:	14218806	(80.55%)
cpp:	1326212	(7.51%)
lisp:	565861	(3.21%)
sh:	469950	(2.66%)
perl:	245860	(1.39%)
asm:	204634	(1.16%)
tcl:	152510	(0.86%)
python:	140725	(0.80%)
yacc:	97506	(0.55%)
java:	79656	(0.45%)
exp:	79605	(0.45%)
lex:	15334	(0.09%)
awk:	14705	(0.08%)
objc:	13619	(0.08%)
csh:	10803	(0.06%)
ada:	8217	(0.05%)
pascal:	4045	(0.02%)
sed:	2806	(0.02%)
fortran:	1707	(0.01%)

Total Physical Source Lines of Code (SLOC) = 17652561
Total Estimated Person-Years of Development = 4548.36
Average Programmer Annual Salary = 56286
Overhead Multiplier = 2.4
Total Estimated Cost to Develop = \$ 614421924.71

SLOC	Dir	tcl	exp	ansic	cpp	objc	python	asm	sh	cs
java	lisp			perl	awk	sed	ada	fortran	pascal	yacc
lex										
53141	AfterStep-1.8.0			50898	233	0	0	0	842	0
0	0	0	0	1168	0	0	0	0	0	0
0										
140130	AfterStep-APPS-20000124			135806	741	0	0	0	3340	0
0	0	0	0	243	0	0	0	0	0	0
0										
16	AnotherLevel-1.0.1			0	0	0	0	0	16	0
0	0	0	0	0	0	0	0	0	0	0
0										
834	ElectricFence-2.1			834	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
15108	GXedit1.23			15019	0	0	0	0	89	0
0	0	0	0	0	0	0	0	0	0	0
0										
121878	ImageMagick-4.2.9			99383	8870	0	0	0	11143	0
0	0	458	0	2024	0	0	0	0	0	0
0										
1020	MAKEDEV-2.5.2			0	0	0	0	0	1020	0
0	0	0	0	0	0	0	0	0	0	0
0										
231072	Mesa			195796	17717	0	0	13467	4092	0
0	0	0	0	0	0	0	0	0	0	0
0										
38548	ORBit-0.5.0			35656	0	0	0	0	776	0
0	0	0	0	0	0	0	0	0	0	1750
366										
200628	Python-1.5.2			96323	0	0	100935	0	673	0
0	2353	0	0	342	0	2	0	0	0	0
0										
15967	SVGATextMode-1.9-src			15079	0	0	0	12	294	0
0	0	0	0	0	0	15	0	0	0	340
227										
79997	WindowMaker-0.61.1			77924	0	0	0	0	1483	0
0	219	0	0	371	0	0	0	0	0	0
0										
18885	X11R6-contrib-3.3.2			18616	0	0	0	0	11	0
0	0	0	0	0	0	0	0	0	0	97
161										
1291745	XFree86-3.3.6			1246420	4358	0	0	14913	13433	5
0	0	8362	0	711	393	57	0	0	0	2710
383										
0	XFree86-ISO8859-2-1.0			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
26608	Xaw3d-1.3			26235	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	247
126										
9741	Xconfigurator-4.3.5			9578	0	0	6	0	32	0
0	0	0	0	125	0	0	0	0	0	0
0										
6976	aboot-0.5			6680	0	0	0	296	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
5383	acct-6.3.2			5016	287	0	0	0	80	0
0	0	0	0	0	0	0	0	0	0	0
0										
1653	adjtimex-1.9			1653	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
45589	am-utils-6.0.3			33389	0	0	0	0	8950	0
0	0	0	0	2421	0	0	0	0	0	375

454										
91213	anaconda-6.2.2			74303	0	0	13657	0	1583	0
0	0	0	0	128	0	0	0	0	0	810
732										
1143	anacron-2.1			1143	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
77873	apache_1.3.12			69191	55	0	0	0	6781	0
0	0	0	0	1846	0	0	0	0	0	0
0										
3012	apmd			2617	0	0	0	0	395	0
0	0	0	0	0	0	0	0	0	0	0
0										
9666	ash-linux-0.2			9445	0	0	0	0	221	0
0	0	0	0	0	0	0	0	0	0	0
0										
3084	at-3.1.7			1442	0	0	0	0	1196	0
0	0	0	0	0	0	0	0	0	0	362
84										
441	audioctl			441	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
12593	audiofile-0.1.9			6153	0	0	0	0	6440	0
0	0	0	0	0	0	0	0	0	0	0
0										
4367	aumix-1.30.1			4095	0	0	0	0	179	0
0	0	0	0	0	0	93	0	0	0	0
0										
1182	auth_ldap-1.4.0			1182	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
1075	authconfig-3.0.3			1075	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
2758	autoconf-2.13			82	0	0	0	0	2226	0
0	0	0	167	283	0	0	0	0	0	0
0										
3625	autofs-3.1.4			2862	0	0	0	0	763	0
0	0	0	0	0	0	0	0	0	0	0
0										
14363	automake-1.4			404	0	0	0	0	3337	0
0	0	0	0	10622	0	0	0	0	0	0
0										
2705	autorun-2.61			0	720	0	0	0	1985	0
0	0	0	0	0	0	0	0	0	0	0
0										
6234	awesfx-0.4.3a			6234	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
47067	bash-1.14.7			41654	0	0	0	48	3140	0
0	0	0	0	0	28	0	0	0	0	2197
0										
68560	bash-2.03			56758	0	0	0	0	7264	0
0	0	0	0	1730	0	0	0	0	0	2808
0										
17682	bc-1.05			9186	0	0	0	0	7236	0
0	0	0	0	0	0	0	0	0	0	967
293										
261	bdf flush-1.5			202	0	0	0	59	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
290	biff+comsat-0.16			290	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
155035	bind-8.2.2_P5			131946	1360	0	0	0	10068	848

0	0	0	0	7607	753	0	0	0	0	2231
222										
467120	binutils-2.9.5.0.22			407352	4454	0	0	27575	7398	0
0	394	0	12265	16	24	557	0	0	0	5606
1479										
9699	bison-1.28			9650	0	0	0	0	49	0
0	0	0	0	0	0	0	0	0	0	0
0										
68997	blt2.4g			58630	0	0	0	0	152	0
0	0	10215	0	0	0	0	0	0	0	0
0										
1272	bsd-finger-0.16			1272	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
2486	bug-buddy-0.7			2486	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
6550	byacc-1.9			5520	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1030
0										
4996	bzip2-0.9.5d			4996	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	caching-nameserver-6.2			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
1842	cdecl-2.5			1002	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	765
75										
2661	cdp-0.33			2661	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
7227	cdparanoia-III-alpha9.6			6006	0	0	0	0	1221	0
0	0	0	0	0	0	0	0	0	0	0
0										
50970	cdrecord-1.8			48595	0	0	0	0	2177	0
0	0	0	0	194	0	4	0	0	0	0
0										
717	chkconfig-1.1.2			717	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
343	chkfontpath-1.7			343	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
1984	cleanfeed-0.95.7b			0	0	0	0	0	0	0
0	0	0	0	1984	0	0	0	0	0	0
0										
15522	console-tools-0.3.3			13335	0	0	0	0	800	0
0	0	0	0	110	0	0	0	0	0	986
291										
16785	control-center-1.0.51			16659	0	0	0	0	126	0
0	0	0	0	0	0	0	0	0	0	0
0										
404	control-panel-3.13			319	0	0	0	0	0	0
0	0	85	0	0	0	0	0	0	0	0
0										
7617	cpio-2.4.2			7598	0	0	0	0	19	0
0	0	0	0	0	0	0	0	0	0	0
0										
9607	cproto-4.6			7600	0	0	0	0	261	0
0	0	0	0	0	0	0	0	0	0	761
985										
1987	cracklib,2.7			1919	0	0	0	0	22	0
0	0	0	0	46	0	0	0	0	0	0
0										

9263	ctags-3.4			9240	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
88128	cvs-1.10.7			68303	0	0	0	0	17909	181
0	7	0	0	902	0	0	0	0	0	826
0										
1290	cxhextris			1290	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	desktop-backgrounds-1.1			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	dev-2.7.18			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
16266	dhcp-2.0			15328	0	0	0	0	938	0
0	0	0	0	0	0	0	0	0	0	0
0										
3051	dhcpcd-1.3.18-pl3			2771	0	0	0	0	280	0
0	0	0	0	0	0	0	0	0	0	0
0										
3433	dialog-0.6			2834	0	0	0	0	250	0
0	0	0	0	349	0	0	0	0	0	0
0										
616	diffstat-1.27			616	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
10914	diffutils-2.7			10914	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
8303	dip-3.3.7o			8207	0	0	0	0	96	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	docbook-3.1			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
3675	dosfstools-2.2			3675	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
10187	dump-0.4b15			9422	0	0	0	0	760	0
0	0	0	0	0	0	5	0	0	0	0
0										
28169	e2fsprogs-1.18			27250	0	0	0	0	339	0
0	0	0	0	22	437	121	0	0	0	0
0										
7427	ed-0.2			7263	0	0	0	0	164	0
0	0	0	0	0	0	0	0	0	0	0
0										
7030	ee-0.3.11			7007	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
5526	efax-0.9			4570	0	0	0	0	956	0
0	0	0	0	0	0	0	0	0	0	0
0										
720112	egcs-1.1.2			598682	75206	482	0	11462	14307	0
0	7252	0	2887	18	0	313	0	1515	0	7988
0										
657	eject-2.0.2			657	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
480	elftoaout-2.2			480	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
42746	elm2.5.3			32931	0	0	0	0	9774	0
0	0	0	0	0	41	0	0	0	0	0

0										
625073	emacs-20.5			169624	0	0	0	253	652	9
0	453647	0	0	884	0	4	0	0	0	0
0										
51592	enlightenment-0.15.5			51569	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
11721	enlightenment-conf-0.15			6232	0	0	0	0	5489	0
0	0	0	0	0	0	0	0	0	0	0
0										
23666	enscript-1.6.1			22365	0	0	0	0	291	0
0	109	0	0	308	0	0	0	0	0	164
429										
7615	esound-0.2.17			7387	0	0	0	0	142	86
0	0	0	0	0	0	0	0	0	0	0
0										
332	ethtool-1.0			332	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
36243	exmh-2.1.1			0	0	0	0	0	49	0
0	0	35844	34	316	0	0	0	0	0	0
0										
3415	ext2ed-0.1			3415	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
78787	extace-1.2.15			66571	0	0	0	0	9322	0
0	0	0	0	2894	0	0	0	0	0	0
0										
1765	fbset-2.1			1401	0	0	0	0	0	0
0	0	0	0	113	0	0	0	0	0	130
121										
17271	fetchmail-5.3.1			13441	0	0	1490	0	1246	0
0	0	0	0	321	124	0	0	0	0	411
238										
2647	file-3.28			2601	0	0	0	0	0	0
0	0	0	0	46	0	0	0	0	0	0
0										
34768	fileutils-4.0p			31324	0	0	0	0	2042	0
0	0	0	0	561	0	0	0	0	0	841
0										
11404	findutils-4.1			11160	0	0	0	0	173	0
0	0	0	71	0	0	0	0	0	0	0
0										
14774	flex-2.5.4			13011	0	0	0	0	29	0
0	0	0	0	0	72	12	0	0	0	605
1045										
2455	fnlib-0.4			2432	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
1604	fortune-mod-9708			1604	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
51813	freetype-1.3.1			48929	351	0	0	0	2467	53
0	0	0	0	13	0	0	0	0	0	0
0										
69265	fvwm-2.2.4			63496	2463	0	0	0	723	0
0	0	0	0	1835	0	0	0	0	0	596
152										
107	fwhois-1.00			107	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
138024	gated-3-5-11			126846	0	0	0	0	1554	235
0	12	0	0	0	666	35	0	0	0	7799
877										
26363	gawk-3.0.4			19871	0	0	0	0	1927	0

0	0	0	0	0	2519	0	0	0	0	2046
0										
20078	gd1.3			19946	0	0	0	0	0	0
0	0	0	0	132	0	0	0	0	0	0
0										
652087	gdb-19991004			587542	6735	0	0	4139	9630	0
0	1820	0	37737	0	142	220	0	5	0	4117
0										
3315	gdbm-1.8.0			3290	25	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
5744	gdm-2.0beta2			5632	0	0	0	0	112	0
0	0	0	0	0	0	0	0	0	0	0
0										
8356	gedit-0.6.1			8225	0	0	0	0	131	0
0	0	0	0	0	0	0	0	0	0	0
0										
549	genromfs-0.3			549	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
17750	gettext-0.10.35			13414	0	0	0	0	1983	0
0	2030	0	0	53	0	9	0	0	0	261
0										
2631	getty_ps-2.0.7j			2631	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
9138	gftp-2.0.6a			9138	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	ghostscript-fonts-5.50			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
770	giftrans-1.12.2			770	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
235702	gimp-1.0.4			225211	0	0	0	0	1994	0
0	8497	0	0	0	0	0	0	0	0	0
0										
0	gimp-data-extras-1.0.0			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
18151	git-4.3.19			16166	0	0	0	0	1985	0
0	0	0	0	0	0	0	0	0	0	0
0										
2835	gkermit-1.0			2835	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
54603	glade-0.5.5			49545	0	0	0	0	5058	0
0	0	0	0	0	0	0	0	0	0	0
0										
18835	glib-1.2.6			18702	0	0	0	0	133	0
0	0	0	0	0	0	0	0	0	0	0
0										
415026	glibc-2.1.3			378753	1704	0	0	30644	2520	15
0	0	0	0	464	910	16	0	0	0	0
0										
24583	gmp-2.0.2			17888	0	0	0	5252	1443	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	gnome-audio-1.0.0			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
72425	gnome-core-1.0.55			72230	0	0	0	0	54	0
0	0	0	0	141	0	0	0	0	0	0
0										

41205	gnome-games-1.0.51	31191	3048	0	0	0	0	0
0	6966	0	0	0	0	0	0	0
0								
2531	gnome-kerberos-0.2	2531	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
128672	gnome-libs-1.0.55	125373	0	0	0	0	2178	0
0	177	0	0	667	277	0	0	0
0								
2163	gnome-linuxconf-0.25	2163	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
6827	gnome-media-1.0.51	6827	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
12463	gnome-objc-1.0.2	12	0	12365	0	0	86	0
0	0	0	0	0	0	0	0	0
0								
30438	gnome-pim-1.0.55	28665	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1773
0								
24270	gnome-python-1.0.51	9791	0	0	14331	0	148	0
0	0	0	0	0	0	0	0	0
0								
19500	gnome-utils-1.0.50	18099	0	0	0	0	0	0
0	577	0	0	0	0	0	0	824
0								
10404	gnorpm-0.9	10404	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
15143	gnotepad+-1.1.4	15143	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
14871	gnuchess-4.0.pl80	14584	0	0	0	0	258	29
0	0	0	0	0	0	0	0	0
0								
116615	gnumeric-0.48	115592	0	0	23	0	142	0
0	191	0	0	67	0	0	0	600
0								
54935	gnupg-1.0.1	48884	0	0	0	4586	1465	0
0	0	0	0	0	0	0	0	0
0								
45378	gnuplot-3.7.1	43276	0	387	0	539	80	297
0	661	0	0	0	0	0	0	0
0								
4430	gperf-2.7	695	2947	0	0	0	43	0
0	0	0	0	0	0	0	0	0
0								
4542	gpgp-0.4	4441	0	0	0	0	101	0
0	0	0	0	0	0	0	0	0
0								
9725	gpm-1.18.1	8107	0	0	0	0	209	0
0	221	0	0	0	74	6	0	1108
0								
10271	gqview-0.7.0	10271	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
10013	grep-2.4	9852	0	0	0	0	103	0
0	0	0	0	0	49	9	0	0
0								
70260	groff-1.15	5276	59453	0	0	1866	265	0
0	0	0	0	397	0	46	0	2957
0								
199982	gs5.50	195491	2266	0	0	968	751	0
0	405	0	0	101	0	0	0	0

0										
138118	gtk+-1.2.6			137006	0	0	0	0	352	0
0	7	0	0	479	274	0	0	0	0	0
0										
35397	gtk-engines-0.10			20636	0	0	0	0	14761	0
0	0	0	0	0	0	0	0	0	0	0
0										
8491	gtop-1.0.5			8151	340	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
45485	guile-1.3			38823	0	0	0	1514	310	50
0	4626	0	0	0	162	0	0	0	0	0
0										
25915	gv-3.5.8			25821	0	0	0	0	94	0
0	0	0	0	0	0	0	0	0	0	0
0										
6306	gzip-1.2.4a			5813	0	0	0	458	24	0
0	0	0	0	11	0	0	0	0	0	0
0										
1229	hdparm-3.6			1229	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
221	hellas			0	0	0	0	0	179	0
0	0	0	0	42	0	0	0	0	0	0
0										
491	helptool-2.4			0	0	0	0	0	0	0
0	0	203	0	288	0	0	0	0	0	0
0										
20125	ical-2.2			27	12651	0	0	0	624	0
0	0	6763	0	60	0	0	0	0	0	0
0										
61688	imap-4.7			61628	0	0	0	0	60	0
0	0	0	0	0	0	0	0	0	0	0
0										
49325	imlib-1.9.7			49260	0	0	0	0	65	0
0	0	0	0	0	0	0	0	0	0	0
0										
5981	indent-2.2.5			5958	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	indexhtml-6.2			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
3929	initscripts-5.00			1866	0	0	0	0	2035	28
0	0	0	0	0	0	0	0	0	0	0
0										
81719	inn-2.2.2			62403	0	0	0	0	5465	0
0	0	3	0	10485	1567	0	0	0	0	1547
249										
0	install-guide-3.2.html			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
47	intimed-1.10			47	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
3651	ipchains-1.3.9			2767	0	0	0	0	884	0
0	0	0	0	0	0	0	0	0	0	0
0										
13241	iproute2			12139	0	0	0	0	1002	0
0	0	0	0	100	0	0	0	0	0	0
0										
6646	iputils			6646	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
255	ipvsadm-1.1			255	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0
0										
37515	ircii-4.4			36647	0	0	0	0	852	0
0	0	0	0	0	0	0	0	0	0	0
16										
6043	irda-utils-0.9.10			5697	0	0	0	0	263	0
0	0	0	0	83	0	0	0	0	0	0
0										
5960	isapnptools-1.21			4394	0	0	0	0	60	0
0	0	0	0	123	0	0	0	0	0	1383
0										
5594	isdn-config			0	3058	0	0	0	2228	0
0	0	0	0	308	0	0	0	0	0	0
0										
87940	isdn4k-utils			78752	2708	0	0	0	3089	0
0	0	22	0	3369	0	0	0	0	0	0
0										
1941	isicom			1898	0	0	0	0	43	0
0	0	0	0	0	0	0	0	0	0	0
0										
14912	ispell-3.1			8380	585	385	0	0	157	221
0	3372	0	0	85	0	15	0	0	0	1712
0										
131372	jade-1.2.1			8228	120611	0	0	0	2150	0
0	0	0	0	378	0	5	0	0	0	0
0										
0	jadetex			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
29315	jed			29315	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
71810	jikes			0	71452	0	0	0	0	0
358	0	0	0	0	0	0	0	0	0	0
0										
19065	joe			18841	0	0	0	224	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
6090	joystick-1.2.15			6086	0	0	0	0	4	0
0	0	0	0	0	0	0	0	0	0	0
0										
12313	jpeg-6a			12313	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
844	jpeg-6b			0	0	0	0	0	844	0
0	0	0	0	0	0	0	0	0	0	0
0										
133193	kaffe-1.0.5			62125	3923	0	0	84	814	0
65275	0	0	0	972	0	0	0	0	0	0
0										
368	kbdconfig-1.9.2.4			368	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
24910	kdeadmin			0	19919	0	0	0	3936	0
0	0	0	0	1055	0	0	0	0	0	0
0										
138931	kdebase			23016	113971	0	0	0	618	0
0	0	0	0	1326	0	0	0	0	0	0
0										
27860	kdegames			340	27507	0	0	0	13	0
0	0	0	0	0	0	0	0	0	0	0
0										
68453	kdegraphics			29347	34208	0	0	0	4898	0
0	0	0	0	0	0	0	0	0	0	0
0										

80343	kdelibs			3660	71217	0	0	0	35	0
0	0	0	0	5075	0	0	0	0	0	240
116										
48223	kdemultimedia			24248	22275	0	0	0	621	0
0	0	1004	0	73	2	0	0	0	0	0
0										
93940	kdenetwork			7422	80075	0	0	0	134	0
0	0	49	0	6260	0	0	0	0	0	0
0										
60429	kdesupport			42421	17810	0	0	0	173	12
0	0	0	0	0	13	0	0	0	0	0
0										
2810	kdetoys			192	2618	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
52808	kdeutils			9693	41365	0	0	0	1434	0
0	0	0	0	0	311	5	0	0	0	0
0										
6072	kdoc			0	17	0	0	0	45	0
0	0	0	0	6010	0	0	0	0	0	0
0										
809	kdpms-0.2.8			0	809	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
347	kernelcfg-0.5			0	0	0	341	0	6	0
0	0	0	0	0	0	0	0	0	0	0
0										
31994	korganizer			5884	23402	0	0	0	1	0
0	0	0	0	375	0	0	0	0	0	2271
61										
10958	kpackage-1.3.10			124	8863	0	0	0	1852	0
0	0	0	0	119	0	0	0	0	0	0
0										
15868	kpilot-3.1b9			5640	8613	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1615
0										
1049	kpppload-1.04			0	1044	0	0	0	5	0
0	0	0	0	0	0	0	0	0	0	0
0										
99066	krb4-1.0			84077	3775	0	0	5163	1765	0
0	0	0	0	2508	33	0	0	0	0	1509
236										
222220	krb5-1.1.1			192822	0	0	348	0	4829	147
0	0	0	19364	1528	393	123	0	0	0	2476
190										
23838	kterm-6.2.0			23838	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	kudzu-0.36			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
100951	lam-6.3.1			86177	10569	0	0	0	3677	19
0	0	0	0	322	0	0	0	187	0	0
0										
9731	ld.so-1.9.5			6960	0	0	0	2401	370	0
0	0	0	0	0	0	0	0	0	0	0
0										
874	ldconfig-1999-02-21			874	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
14039	less-346			14032	0	0	0	0	0	0
0	0	0	0	0	7	0	0	0	0	0
0										
10267	libPropList-0.9.1			3982	0	0	0	0	5974	0
0	0	0	0	0	0	0	0	0	0	139

172										
4792	libelf-0.6.4			3310	0	0	0	0	1482	0
0	0	0	0	0	0	0	0	0	0	0
0										
2645	libhttp-1.0.4			2645	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
7859	libglade-0.11			5898	0	0	152	0	1809	0
0	0	0	0	0	0	0	0	0	0	0
0										
102674	libgr-2.0.13			99647	0	0	0	0	2438	589
0	0	0	0	0	0	0	0	0	0	0
0										
14516	libgtop-1.0.6			13768	0	0	0	31	64	0
0	0	0	0	653	0	0	0	0	0	0
0										
22011	libpng-1.0.5			22011	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
27117	librep-0.10			19381	0	0	0	0	2351	0
0	5385	0	0	0	0	0	0	0	0	0
0										
5115	libtool-1.3.4			1741	0	0	0	0	3374	0
0	0	0	0	0	0	0	0	0	0	0
0										
12639	libungif-4.1.0			12381	0	0	0	0	204	0
0	0	0	0	54	0	0	0	0	0	0
0										
26146	libxml-1.8.6			26069	0	0	0	0	77	0
0	0	0	0	0	0	0	0	0	0	0
0										
7255	lilo			3522	3	0	0	2557	740	0
0	0	0	0	433	0	0	0	0	0	0
0										
1526722	linux			1462165	0	0	0	59574	2860	0
0	0	414	0	950	133	72	0	0	0	324
230										
69246	linux-86			63328	0	0	0	5276	642	0
0	0	0	0	0	0	0	0	0	0	0
0										
104032	linuxconf-1.17r2			598	93139	0	0	0	2984	0
2741	0	0	0	4570	0	0	0	0	0	0
0										
23	locale-ja-9			0	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
497	locale_config-0.2			497	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
1525	logrotate-3.3.2			1524	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0
0										
26673	lout-3.17			26673	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
682	lpg			0	0	0	0	0	0	0
0	0	0	0	682	0	0	0	0	0	0
0										
6877	lpr			6842	0	0	0	0	35	0
0	0	0	0	0	0	0	0	0	0	0
0										
11790	lrzsz-0.12.20			9512	0	0	0	0	1263	0
0	0	0	1015	0	0	0	0	0	0	0
0										
6116	lslk			5325	0	0	0	0	791	0

0	0	0	0	0	0	0	0	0	0	0
0										
56093	lsof_4.47			50268	0	0	0	2	4753	0
0	0	0	0	856	214	0	0	0	0	0
0										
3896	ltrace-0.3.10			2986	0	0	0	0	854	0
0	0	0	0	0	56	0	0	0	0	0
0										
119613	lynx2-8-3			117385	0	0	0	0	1860	28
0	0	0	0	340	0	0	0	0	0	0
0										
6391	m4-1.4			5993	0	0	0	0	155	0
0	243	0	0	0	0	0	0	0	0	0
0										
12657	macutils			12657	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
2580	magicdev-0.2.7			2580	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	mailcap-2.0.6			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
6968	mailx-8.1.1			6963	0	0	0	0	5	0
0	0	0	0	0	0	0	0	0	0	0
0										
22279	make-3.78.1			19287	0	0	0	0	2029	0
0	0	0	0	963	0	0	0	0	0	0
0										
4780	make-3.78.1_pvm-0.5			4780	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
9801	man-1.5h1			7377	0	0	0	0	1802	0
0	0	0	0	317	305	0	0	0	0	0
0										
245	man-pages-1.28			0	0	0	0	0	244	0
0	0	0	0	0	0	1	0	0	0	0
0										
24387	mars_nwe			24158	0	0	0	0	229	0
0	0	0	0	0	0	0	0	0	0	0
0										
14427	mawk-1.2.2			12714	0	0	0	0	90	0
0	0	0	0	0	629	0	0	0	0	994
0										
116951	mc-4.5.42			114406	0	0	0	0	1996	56
0	0	0	0	345	148	0	0	0	0	0
0										
2270	memprof-0.3.0			2270	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
44323	mgetty-1.1.21			33757	0	0	0	0	3638	0
0	283	756	0	5889	0	0	0	0	0	0
0										
27040	mikmod-3.1.6			26975	0	0	0	0	55	0
0	0	0	0	0	10	0	0	0	0	0
0										
343	mingetty-0.9.4			343	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
12633	minicom-1.83.0			12503	0	0	0	0	130	0
0	0	0	0	0	0	0	0	0	0	0
0										
537	minlabel-1.2			537	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										

314	mkbootdisk-1.2.5	0	0	0	0	0	314	0
0	0	0	0	0	0	0	0	0
0								
288	mkinitrd-2.4.1	0	0	0	0	0	288	0
0	0	0	0	0	0	0	0	0
0								
8939	mkisofs-1.12b5	8939	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
222	mkkickstart-2.1	0	0	0	0	0	222	0
0	0	0	0	0	0	0	0	0
0								
85	mktemp-1.5	85	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
15087	mm2.7	8044	0	0	0	0	119	6924
0	0	0	0	0	0	0	0	0
0								
15638	mod_perl-1.21	5124	0	0	0	0	236	0
0	0	0	0	10278	0	0	0	0
0								
82	modemtool-1.21	0	0	0	73	0	9	0
0	0	0	0	0	0	0	0	0
0								
11775	modutils-2.3.9	9309	0	0	0	0	1620	0
0	0	0	0	0	0	0	0	362
484								
830	mouseconfig-4.4	830	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
2773	mpage-2.4	2704	0	0	0	0	69	0
0	0	0	0	0	0	0	0	0
0								
15819	mpg123-0.59r	14900	0	0	0	919	0	0
0	0	0	0	0	0	0	0	0
0								
1361	mt-st-0.5b	1361	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
17765	mtools-3.9.6	16155	0	0	0	0	1602	0
0	0	0	0	8	0	0	0	0
0								
14587	multimedia	14577	0	0	0	0	10	0
0	0	0	0	0	0	0	0	0
0								
45811	mutt-1.0.1	45574	0	0	0	0	237	0
0	0	0	0	0	0	0	0	0
0								
1088	nag	0	0	0	0	0	0	0
0	0	0	0	1088	0	0	0	0
0								
2407	nc	1670	0	0	0	0	737	0
0	0	0	0	0	0	0	0	0
0								
31174	ncftp-3.0beta21	30347	595	0	0	0	232	0
0	0	0	0	0	0	0	0	0
0								
1435	ncompress-4.2.4	1435	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0								
28897	ncpfs-2.2.0.17	28689	0	0	0	0	182	0
0	0	26	0	0	0	0	0	0
0								
61324	ncurses-5.0	45856	3720	0	0	0	2822	0
0	0	0	0	103	506	100	8217	0

0									
11633	net-tools-1.54		11531	0	0	0	0	102	0
0	0	0	0	0	0	0	0	0	0
0									
1634	netcfg-2.25		0	0	0	1632	0	2	0
0	0	0	0	0	0	0	0	0	0
0									
2883	netkit-base-0.16		2883	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
506	netkit-bootparamd-0.16		506	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
5111	netkit-ftp-0.16		5111	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
2048	netkit-ntalk-0.16		2048	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
2423	netkit-routed-0.16		2423	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
3588	netkit-rsh-0.16		3588	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
1857	netkit-rusers-0.16		1857	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
294	netkit-rwall-0.16		294	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
967	netkit-rwho-0.16		967	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
21010	netkit-telnet-0.16		14796	6214	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
1531	netkit-tftp-0.16		1531	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
3962	netkit-timed-0.16		3962	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
592	netscape-4.72		0	0	0	0	0	592	0
0	0	0	0	0	0	0	0	0	0
0									
7041	newt-0.50.8		6526	0	0	515	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
14299	nfs-utils-0.1.6		14107	0	0	0	0	165	0
0	0	0	27	0	0	0	0	0	0
0									
52574	nmh-1.0.3		50698	0	0	0	0	1785	0
0	0	0	0	74	17	0	0	0	0
0									
9873	nss_ldap-105		9784	0	0	0	0	0	0
0	0	0	89	0	0	0	0	0	0
0									
240	open-1.4		240	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0									
60302	openldap-1.2.9		58078	0	0	201	0	1393	0
0	0	0	630	0	0	0	0	0	0
0									
41388	p2c-1.22		38788	0	0	0	0	0	0

0	0	0	0	101	0	0	0	0	2499	0
0										
20433	pam-0.72			18936	0	0	0	0	482	0
0	0	0	0	321	0	0	0	0	0	634
60										
1280	pam_krb5-1			1280	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
1194	passwd-0.64.1			1194	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
6611	patch-2.5			6561	0	0	0	0	0	0
0	0	0	0	0	0	50	0	0	0	0
0										
3855	pciutils-2.1.5			3800	0	0	0	0	55	0
0	0	0	0	0	0	0	0	0	0	0
0										
24773	pdksh-5.2.14			23599	0	0	0	0	189	0
0	0	0	0	945	0	40	0	0	0	0
0										
206237	perl5.005_03			89366	0	0	0	0	15654	0
0	5584	0	0	94712	0	0	0	0	0	921
0										
3885	phhttpd-0.1.0			3859	0	0	0	0	26	0
0	0	0	0	0	0	0	0	0	0	0
0										
35136	php-2.0.1			33991	0	0	0	0	1056	0
0	0	0	0	0	89	0	0	0	0	0
0										
109824	php-3.0.15			105901	28	0	0	0	1381	0
0	0	0	0	537	90	0	0	0	0	1887
0										
6496	pidentd-3.0.10			6475	0	0	0	0	21	0
0	0	0	0	0	0	0	0	0	0	0
0										
32277	pilot-link.0.9.3			26513	1689	0	268	0	0	0
2162	0	14	0	971	0	0	0	0	0	660
0										
127536	pine4.21			126678	0	0	0	0	766	62
0	0	0	0	30	0	0	0	0	0	0
0										
10088	piranha			10048	0	0	0	0	40	0
0	0	0	0	0	0	0	0	0	0	0
0										
3219	playmidi-2.4			3217	0	0	0	0	0	0
0	0	0	0	0	0	2	0	0	0	0
0										
34882	pmake			34599	0	0	0	0	184	0
0	0	0	0	0	58	41	0	0	0	0
0										
6025	pnm2ppa			5708	0	0	0	0	317	0
0	0	0	0	0	0	0	0	0	0	0
0										
1099	popt-1.4			1039	0	0	0	0	60	0
0	0	0	0	0	0	0	0	0	0	0
0										
897	portmap_4			897	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
247026	postgresql-6.5.3			207735	746	0	959	70	7399	5
8835	0	7709	0	1206	0	2	0	0	0	10718
1642										
62922	ppp-2.3.11			61756	0	0	0	0	996	44
0	0	0	82	44	0	0	0	0	0	0
0										

2200	printtool			0	0	0	0	0	0	0
0	0	2200	0	0	0	0	0	0	0	0
0										
1226	procinfo-17			1145	0	0	0	0	0	0
0	0	0	0	81	0	0	0	0	0	0
0										
9927	procmail-3.14			8090	0	0	0	0	1837	0
0	0	0	0	0	0	0	0	0	0	0
0										
9961	procps-2.0.6			9959	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	0	0
0										
1146	prtconf-1.3			1146	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
8572	psgml-1.2.1			0	0	0	0	0	0	0
0	8572	0	0	0	0	0	0	0	0	0
0										
1630	psmisc			1624	0	0	0	0	6	0
0	0	0	0	0	0	0	0	0	0	0
0										
1856	pump-0.7.8			1856	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
9551	pwdb-0.61			9488	0	0	0	0	63	0
0	0	0	0	0	0	0	0	0	0	0
0										
8766	pxe-linux			3622	4463	0	0	681	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
2597	pythonlib-1.23			0	0	0	2597	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
205082	qt-2.1.0-beta1			20513	180866	0	0	0	538	0
0	0	0	0	417	0	0	0	0	0	2284
464										
159	quickstrip-1.1			159	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
3804	quota-2.00-pre3			3795	0	0	0	0	9	0
0	0	0	0	0	0	0	0	0	0	0
0										
2424	raidtools-0.90			2418	0	0	0	0	6	0
0	0	0	0	0	0	0	0	0	0	0
0										
14269	rcs-5.7			12209	0	0	0	0	2060	0
0	0	0	0	0	0	0	0	0	0	0
0										
132	rdate-1.0			132	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
9465	rdist-6.1.5			8306	0	0	0	0	553	0
0	0	0	0	117	0	0	0	0	0	489
0										
14941	readline-2.2.1			11375	0	0	0	0	1890	0
0	0	0	0	1676	0	0	0	0	0	0
0										
0	redhat-logos			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
4024	rep-gtk-0.8			2905	0	0	0	0	148	0
0	971	0	0	0	0	0	0	0	0	0
0										
213	rhmask			213	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

0										
445	rhs-printfilters-1.63			2	0	0	0	0	443	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	rootfiles			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
13504	rp3-1.0.7			2957	10416	0	0	0	131	0
0	0	0	0	0	0	0	0	0	0	0
0										
39861	rpm-3.0.4			36994	0	0	7	0	1505	0
0	0	0	0	1355	0	0	0	0	0	0
0										
15456	rpm2html-1.2			15334	0	0	0	0	0	0
0	0	0	0	122	0	0	0	0	0	0
0										
6021	rpmfind-1.4			6021	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
816	rpmlint-0.8			0	0	0	813	0	3	0
0	0	0	0	0	0	0	0	0	0	0
0										
14350	rsync-2.4.1			13986	0	0	0	0	126	0
0	0	0	0	179	59	0	0	0	0	0
0										
13779	rxvt-2.6.1			13779	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
255	sag-0.6-html			0	0	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0	0
0										
92964	samba-2.0.6			88308	0	0	0	0	3557	110
0	0	0	0	831	158	0	0	0	0	0
0										
6172	sash-3.4			6172	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
22373	sawmill-0.24			11038	0	0	0	0	3163	0
0	8172	0	0	0	0	0	0	0	0	0
0										
30122	scheme-3.2			10515	0	0	0	0	124	0
0	19483	0	0	0	0	0	0	0	0	0
0										
29730	screen-3.9.5			28156	0	0	0	0	1574	0
0	0	0	0	0	0	0	0	0	0	0
0										
7740	sed-3.02			7301	0	0	0	0	80	0
0	0	0	0	0	0	359	0	0	0	0
0										
42880	sendmail-8.9.3			40364	0	0	0	0	779	0
0	0	0	0	1737	0	0	0	0	0	0
0										
742	setserial-2.15			742	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
67	setup-1.2			67	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	setup-2.1.8			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	sgml-common-0.1			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
62137	sgml-tools-1.0.9			19185	38543	0	0	0	532	0

0	309	0	0	2866	142	0	0	0	0	0
560										
17939	sh-utils-2.0			13366	0	0	0	0	3027	0
0	0	0	0	675	0	0	0	0	0	871
0										
25236	shadow-19990827			23464	0	0	0	0	883	0
0	0	0	0	33	0	0	0	0	0	856
0										
56	shaper			56	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
7570	sharutils-4.2.1			5511	0	0	0	0	318	0
0	0	0	0	1741	0	0	0	0	0	0
0										
13100	silos-0.9.8			10485	0	0	0	2615	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
28118	slang			28118	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
2447	sliplogin-2.1.1			2256	0	0	0	0	143	0
0	0	0	0	48	0	0	0	0	0	0
0										
1883	slocate-2.1			1802	0	0	0	0	81	0
0	0	0	0	0	0	0	0	0	0	0
0										
28449	slrn-0.9.6.2			28438	0	0	0	0	11	0
0	0	0	0	0	0	0	0	0	0	0
0										
2490	sndconfig-0.43			2490	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	solemul-1.1			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
17259	sox-12.16			16659	0	0	0	0	600	0
0	0	0	0	0	0	0	0	0	0	0
0										
52	sparc32-1.1			52	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	specspo-6.2			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
68884	squid-2.3.STABLE1			66305	0	0	0	0	1570	0
0	0	0	0	1009	0	0	0	0	0	0
0										
280	stat-1.5			280	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
131	statserial-1.1			121	0	0	0	0	10	0
0	0	0	0	0	0	0	0	0	0	0
0										
33203	strace-4.2			30891	0	0	0	0	1988	0
0	44	0	0	280	0	0	0	0	0	0
0										
1076	stylesheets-0.13rh			0	0	0	0	0	188	0
0	0	0	0	888	0	0	0	0	0	0
0										
54431	svgalib-1.4.1			53725	0	0	0	630	22	0
0	0	0	0	54	0	0	0	0	0	0
0										
953	switchdesk-2.1			314	233	0	0	0	119	0
0	0	0	0	287	0	0	0	0	0	0
0										

302	symlinks-1.2			302	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
4038	syslogd-1.3-31			3741	0	0	0	0	139	0
0	0	0	0	158	0	0	0	0	0	0
0										
265	sysreport-1.0			0	0	0	0	0	265	0
0	0	0	0	0	0	0	0	0	0	0
0										
6033	sysvinit-2.78			5256	0	0	0	0	777	0
0	0	0	0	0	0	0	0	0	0	0
0										
15851	taper-6.9a			15851	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
14255	tar-1.13.17			13014	0	0	0	0	538	0
0	592	0	0	111	0	0	0	0	0	0
0										
327021	tcltk-8.0.5			240093	0	0	0	0	8531	0
0	0	71947	5150	265	273	0	0	0	0	762
0										
3654	tcp_wrappers_7.6			3654	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
30061	tcpdump-3.4			29208	0	0	0	0	211	16
0	0	0	0	0	184	0	0	0	0	236
206										
46312	tcsh-6.09.00			43544	0	0	0	0	921	585
0	669	0	0	593	0	0	0	0	0	0
0										
193916	teTeX-1.0			166041	9407	0	0	139	10263	47
0	29	0	0	3795	522	297	0	0	1546	1507
323										
797	termcap-2.0.8			797	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
28186	texinfo-4.0			26404	0	0	0	0	841	0
0	213	0	0	256	451	21	0	0	0	0
0										
36338	textutils-2.0a			18949	0	0	0	0	16111	0
0	0	0	0	1218	0	60	0	0	0	0
0										
37360	tiff-v3.5.4			32734	572	0	0	0	4054	0
0	0	0	0	0	0	0	0	0	0	0
0										
1452	time-1.7			1395	0	0	0	0	57	0
0	0	0	0	0	0	0	0	0	0	0
0										
346	timeconfig-3.0.3			318	0	0	0	0	28	0
0	0	0	0	0	0	0	0	0	0	0
0										
367	timetool-2.7.3			0	0	0	0	0	0	0
0	0	367	0	0	0	0	0	0	0	0
0										
49370	tin-1.4.2			47763	0	0	0	0	908	0
0	0	0	0	0	0	0	0	0	0	699
0										
548	tksysv-1.1			0	0	0	0	0	22	0
0	0	526	0	0	0	0	0	0	0	0
0										
463	tmpwatch-2.2			311	0	0	0	0	152	0
0	0	0	0	0	0	0	0	0	0	0
0										
7	trXFree86-2.1.2			0	0	0	0	0	0	0
0	0	7	0	0	0	0	0	0	0	0

0										
1473	traceroute-1.4a5			1436	0	0	0	0	0	0
0	0	0	0	0	37	0	0	0	0	0
0										
15691	transfig.3.2.1			15643	0	0	0	0	38	10
0	0	0	0	0	0	0	0	0	0	0
0										
728	tree-1.2			728	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
32767	trn-3.6			25264	0	0	0	0	6843	0
0	0	0	0	0	0	0	0	0	0	660
0										
1013	trojka			1013	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
72726	ucd-snmp-4.1.1			64411	0	0	0	0	2757	0
0	0	0	0	5558	0	0	0	0	0	0
0										
2141	unarj-2.43			2141	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
2065	units-1.55			1963	0	0	0	0	0	0
0	0	0	0	102	0	0	0	0	0	0
0										
46159	unzip-5.40			40977	3778	0	0	1271	133	0
0	0	0	0	0	0	0	0	0	0	0
0										
2324	up2date-1.13			0	0	0	2324	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
1621	urlview-0.7			1515	0	0	0	0	106	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	urw-fonts-2.0			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
2459	usermode-1.20			2459	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
585	usernet-1.0.9			585	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
0	users-guide-1.0.72			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
222	utempter-0.5.2			222	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
39160	util-linux-2.10f			38627	0	0	0	0	351	62
0	0	0	0	65	0	55	0	0	0	0
0										
55667	uucp-1.06.1			52078	0	0	0	0	3400	0
0	0	0	0	189	0	0	0	0	0	0
0										
113241	vim-5.6			111724	0	0	0	0	469	6
0	0	0	0	359	683	0	0	0	0	0
0										
2879	vixie-cron-3.0.1			2866	0	0	0	0	13	0
0	0	0	0	0	0	0	0	0	0	0
0										
368	vlock-1.3			368	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
73817	w3c-libwww-5.2.8			64754	3181	0	0	0	4678	0

0	0	0	0	1204	0	0	0	0	0	0
0										
13586	wget-1.5.3			13509	0	0	0	0	23	0
0	0	0	0	54	0	0	0	0	0	0
0										
2268	which-2.9			1398	0	0	0	0	870	0
0	0	0	0	0	0	0	0	0	0	0
0										
14105	wmakerconf-2.1			13620	0	0	0	0	137	0
0	0	0	0	348	0	0	0	0	0	0
0										
1977	wmconfig-0.9.8			1941	0	0	0	0	36	0
0	0	0	0	0	0	0	0	0	0	0
0										
11	words-2			0	0	0	0	0	11	0
0	0	0	0	0	0	0	0	0	0	0
0										
19971	wu-ftpd-2.6.0			17572	0	0	0	0	421	0
0	0	0	0	204	0	0	0	0	0	1774
0										
0	wvdial-1.41			0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
36239	x11amp-0.9-alpha3			31686	0	0	0	353	4200	0
0	0	0	0	0	0	0	0	0	0	0
0										
27022	x3270-3.1.1			26456	0	0	0	0	478	0
0	0	0	88	0	0	0	0	0	0	0
0										
1129	xbill-2.0			0	1129	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
21593	xboard-4.0.5			20640	0	0	0	0	41	5
0	0	0	0	0	0	3	0	0	0	0
904										
18020	xboing			18006	0	0	0	0	14	0
0	0	0	0	0	0	0	0	0	0	0
0										
29091	xchat-1.4.0			28894	0	0	53	0	23	0
0	0	0	0	121	0	0	0	0	0	0
0										
4895	xcpustate-2.5			4895	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
3860	xdaliclock-2.18			3837	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
57217	xfig.3.2.3-beta-1			57212	0	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0	0
0										
28261	xfishtank-2.1tp			28261	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
7095	xgammon-0.98			6506	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
589										
2791	xjewel-1.6			2791	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
113272	xlispstat-3-52-17			91484	0	0	0	0	18	1
0	21769	0	0	0	0	0	0	0	0	0
0										
35812	xloadimage.4.1			35705	0	0	0	0	107	0
0	0	0	0	0	0	0	0	0	0	0
0										

94637	xlockmore-4.15			89816	1987	0	0	0	859	0
285	0	1541	0	149	0	0	0	0	0	0
0										
987	xmailbox-2.5			987	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
38927	xmms-1.0.1			38366	0	0	0	398	163	0
0	0	0	0	0	0	0	0	0	0	0
0										
11299	xmorph-1999dec12			10783	0	0	0	0	0	0
0	0	516	0	0	0	0	0	0	0	0
0										
65722	xntp3-5.93			60190	0	0	0	37	1445	0
0	0	0	0	3633	417	0	0	0	0	0
0										
6592	xosview-1.7.1			367	6205	0	0	0	0	0
0	0	0	0	0	20	0	0	0	0	0
0										
25479	xpaint			25456	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
9942	xpat2-1.04			9942	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
85383	xpdf-0.90			21400	60427	0	0	0	3556	0
0	0	0	0	0	0	0	0	0	0	0
0										
75257	xpilot-4.1.0			68669	1896	0	0	0	1145	0
0	0	3479	0	68	0	0	0	0	0	0
0										
7826	xpm-3.4k			7750	37	0	0	0	39	0
0	0	0	0	0	0	0	0	0	0	0
0										
34772	xpuzzles-5.4.1			34772	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
25994	xrn-9.02			24686	0	0	0	0	249	13
0	0	0	0	35	31	0	0	0	0	888
92										
89959	xscreensaver-3.23			88488	0	0	0	0	401	0
0	0	0	0	1070	0	0	0	0	0	0
0										
301	xsri-1.0			301	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
787	xsyinfo-1.7			787	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
236	xtoolwait-1.2			236	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
3096	xtrojka123			3087	0	0	0	0	9	0
0	0	0	0	0	0	0	0	0	0	0
0										
8540	xxgdb-1.12			8540	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
3438	yp-tools-2.4			3415	0	0	0	0	23	0
0	0	0	0	0	0	0	0	0	0	0
0										
3245	ypbind-3.3			1793	0	0	0	0	1452	0
0	0	0	0	0	0	0	0	0	0	0
0										
12124	ypserv-1.3.9			11622	0	0	0	0	460	0
0	0	0	0	42	0	0	0	0	0	0

0										
5975	ytalk-3.1			5975	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
35554	zip-2.3			32108	0	0	0	3446	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
4087	zlib-1.1.3			2815	560	0	0	712	0	0
0	0	0	0	0	0	0	0	0	0	0
0										
38453	zsh-3.0.7			36208	0	0	0	0	1763	0
0	0	0	0	331	145	6	0	0	0	0
0										
17652561		Totals			14218806		1326212	13619	140725	204634
469950	10803	79656	565861	152510	79605	245860	14705	2806	8217	1707
4045	97506	15334	Percentages				80.55	7.51	0.08	0.80
1.16	2.66	0.06	0.45	3.21	0.86	0.45	1.39	0.08	0.02	0.05
0.01	0.02	0.55	0.09							
	Code Percentages				80.55	7.51	0.08	0.80	1.16	2.66
0.06	0.45	3.21	0.86	0.45	1.39	0.08	0.02	0.05	0.01	0.02
0.55	0.09									

Total Physical Source Lines of Code (SLOC) = 17652561
 Total Estimated Person-Years of Development = 4548.36
 Average Programmer Annual Salary = 56286
 Overhead Multiplier = 2.4
 Total Estimated Cost to Develop = \$ 614421924.71

60302 openldap-1.2.9 openldap.spec Artistic
1987 cracklib,2.7 cracklib.spec artistic

206237 perl5.005_03 perl.spec Artistic or GPL

327021 tcltk-8.0.5 tcltk.spec BSD
247026 postgresql-6.5.3 postgresql.spec BSD
89959 xscreensaver-3.23 xscreensaver.spec BSD
61688 imap-4.7 imap.spec BSD
45589 am-utils-6.0.3 am-utils.spec BSD
42880 sendmail-8.9.3 sendmail.spec BSD
34882 pmake pmake.spec BSD
30061 tcpdump-3.4 tcpdump.spec BSD
25236 shadow-19990827 shadow-utils.spec BSD
21010 netkit-telnet-0.16 telnet.spec BSD
19971 wu-ftpd-2.6.0 wu-ftpd.spec BSD
10187 dump-0.4b15 dump.spec BSD
9731 ld.so-1.9.5 ld.so.spec BSD
9666 ash-linux-0.2 ash-0.2.spec BSD
9465 rdist-6.1.5 rdist.spec BSD
8766 pxe-linux pxe.spec BSD
6968 mailx-8.1.1 mailx.spec BSD
6646 iputils iputils.spec BSD
5975 ytalk-3.1 ytalk-3.1.spec BSD
5111 netkit-ftp-0.16 ftp.spec BSD
4087 zlib-1.1.3 zlib.spec BSD
3962 netkit-timed-0.16 timed.spec BSD
3804 quota-2.00-pre3 quota.spec BSD
3588 netkit-rsh-0.16 rsh.spec BSD
2883 netkit-base-0.16 inetd.spec BSD
2773 mpage-2.4 mpage.spec BSD
2447 sliplogin-2.1.1 sliplogin.spec BSD
2423 netkit-routed-0.16 routed.spec BSD
2048 netkit-ntalk-0.16 talk.spec BSD
1857 netkit-rusers-0.16 rusers.spec BSD
1604 fortune-mod-9708 fortune-mod.spec BSD
1531 netkit-tftp-0.16 tftp.spec BSD
1473 traceroute-1.4a5 traceroute.spec BSD
1361 mt-st-0.5b mt-st.spec BSD
1272 bsd-finger-0.16 finger.spec BSD
1194 passwd-0.64.1 passwd.spec BSD
967 netkit-rwho-0.16 rwho.spec BSD
897 portmap_4 portmap.spec BSD
770 giftrans-1.12.2 giftrans.spec BSD
506 netkit-bootparamd-0.16 bootparamd.spec BSD
441 audiocctl audiocctl.spec.sparc BSD
294 netkit-rwall-0.16 rwall.spec BSD
290 biff+comsat-0.16 comsat.spec BSD
131 statserial-1.1 statserial.spec BSD
107 fwhois-1.00 fwhois.spec BSD
85 mktemp-1.5 mktemp.spec BSD

131372 jade-1.2.1 jade.spec Copyright 1997 James Clark (BSDish)
72726 ucd-snmp-4.1.1 ucd-snmp.spec BSDish
73817 w3c-libwww-5.2.8 W3C (BSDish)
51813 freetype-1.3.1 freetype-1.3.1.spec BSD-like
20078 gd1.3 gd.spec BSD-style
15456 rpm2html-1.2 rpm2html.spec W3C Copyright (BSD like).
12639 libungif-4.1.0 libungif.spec X Consortium-like
6021 rpmfind-1.4 rpmfind.spec W3C Copyright (BSD like).

592 netscape-4.72 netscape.spec Commercial

12657 macutils macutils.spec distributable
200628 Python-1.5.2 python.spec distributable

193916 teTeX-1.0 tetex.spec distributable
155035 bind-8.2.2_P5 bind.spec distributable
138024 gated-3-5-11 gated.spec distributable
127536 pine4.21 pine.spec distributable
113272 xispstat-3-52-17 xispstat.spec Distributable
65722 xntp3-5.93 xntp3.spec distributable
62922 ppp-2.3.11 ppp.spec distributable
61324 ncurses-5.0 ncurses.spec distributable
54431 svgalib-1.4.1 svgalib.spec distributable
49370 tin-1.4.2 tin.spec distributable
46312 tcsh-6.09.00 tcsh.spec distributable
46159 unzip-5.40 unzip.spec distributable
45378 gnuplot-3.7.1 gnuplot.spec distributable
44323 mgetty-1.1.21 mgetty.spec distributable
42746 elm2.5.3 elm.spec distributable
41388 p2c-1.22 p2c.spec distributable
39160 util-linux-2.10f util-linux.spec distributable
37515 ircii-4.4 ircii.spec distributable
37360 tiff-v3.5.4 libtiff.spec distributable
35554 zip-2.3 zip.spec distributable
33203 strace-4.2 strace.spec distributable
32767 trn-3.6 trn.spec distributable
31174 ncftp-3.0beta21 ncftp.spec Distributable
25994 xrn-9.02 xrn.spec Distributable
23838 kterm-6.2.0 kterm-6.2.0.spec distributable
22011 libpng-1.0.5 libpng.spec distributable
20125 ical-2.2 ical.spec distributable
17259 sox-12.16 sox.spec distributable
16266 dhcp-2.0 dhcp.spec distributable
15819 mpg123-0.59r mpg123.spec distributable
15691 transfig.3.2.1 transfig-3.2.1.spec distributable
15087 mm2.7 metamail.spec Distributable
13779 rxvt-2.6.1 rxvt.spec distributable
12313 jpeg-6a libjpeg-6a.spec distributable
9942 xpat2-1.04 xpat.spec distributable
9927 procmail-3.14 procmail.spec distributable
9741 Xconfigurator-4.3.5 Xconfigurator.spec distributable
6976 aboot-0.5 aboot.spec.alpha distributable
6877 lpr lpr.spec distributable
4792 libelf-0.6.4 libelf-0.6.4.spec distributable
3654 tcp_wrappers_7.6 tcp_wrappers.spec Distributable
3096 xtrojka123 xtrojka.spec distributable
2879 vixie-cron-3.0.1 vixie-cron.spec distributable
2647 file-3.28 file.spec distributable
2631 getty_ps-2.0.7j getty_ps.spec Distributable
2141 unarj-2.43 unarj.spec distributable
1984 cleanfeed-0.95.7b cleanfeed.spec distributable
1842 cdecl-2.5 cdecl-2.5.spec distributable
1653 adjtimex-1.9 adjtimex.spec distributable
1630 psmisc psmisc.spec distributable
1290 cxhextris cxhextris.spec distributable
1229 hdparm-3.6 hdparm.spec distributable
1088 nag nag.spec distributable
1076 stylesheets-0.13rh stylesheets.spec distributable
1013 trojka trojka.spec distributable
844 jpeg-6b libjpeg.spec distributable
830 mouseconfig-4.4 mouseconfig.spec distributable
682 lpg lpg.spec distributable
616 diffstat-1.27 diffstat.spec distributable
302 symlinks-1.2 symlinks-1.2.spec distributable
255 sag-0.6-html sag.spec distributable
245 man-pages-1.28 man-pages.spec distributable
221 hellas XFree86-ISO8859-7.spec distributable
159 quickstrip-1.1 quickstrip.spec.alpha distributable
16 AnotherLevel-1.0.1 AnotherLevel.spec distributable

7 trXFree86-2.1.2 XFree86-ISO8859-9.spec distributable
0 docbook-3.1 docbook.spec distributable
0 indexhtml-6.2 indexhtml.spec distributable
0 install-guide-3.2.html install-guide.spec distributable
0 jadetex jadetex.spec distributable
0 XFree86-ISO8859-2-1.0 XFree86-ISO8859-2.spec distributable
56093 lsof_4.47 lsof.spec Free
6116 lslk lslk.spec Free
99066 krb4-1.0 krbafs.spec Freely Distributable
15143 gnotepad+-1.1.4 gnotepad+.spec Freely distributable
15108 GXedit1.23 gxedit.spec Freely distributable
0 sgml-common-0.1 sgml-common.spec freely distributable
77873 apache_1.3.12 apache.spec Freely distributable and usable
17271 fetchmail-5.3.1 fetchmail.spec freely redistributable
4895 xcpustate-2.5 xcpustate.spec Freely redistributable
121878 ImageMagick-4.2.9 ImageMagick.spec freeware
113241 vim-5.6 vim.spec freeware
102674 libgr-2.0.13 libgr.spec freeware
57217 xfig.3.2.3-beta-1 xfig.spec Freeware
52574 nmh-1.0.3 nmh.spec freeware
36243 exmh-2.1.1 exmh.spec freeware
47 intimed-1.10 intimed-1.10.spec freeware
11 words-2 words-2.spec freeware
62137 sgml-tools-1.0.9 sgml-tools.spec freeware. See COPYRIGHT file.

1526722 linux kernel-2.2.14.spec GPL
720112 egcs-1.1.2 egcs.spec GPL
652087 gdb-19991004 gdb.spec GPL
625073 emacs-20.5 emacs.spec GPL
467120 binutils-2.9.5.0.22 binutils.spec GPL
235702 gimp-1.0.4 gimp.spec GPL
199982 gs5.50 ghostscript-5.50.spec GPL
140130 AfterStep-APPS-20000124 AfterStep-APPS.spec GPL
138931 kdatabase kdatabase-1.1.2.spec GPL
133193 kaffe-1.0.5 kaffe.spec GPL
119613 lynx2-8-3 lynx.spec GPL
116951 mc-4.5.42 mc.spec GPL
116615 gnumeric-0.48 gnumeric.spec GPL
109824 php-3.0.15 php.spec GPL
104032 linuxconf-1.17r2 linuxconf.spec GPL
100951 lam-6.3.1 lam.spec GPL
93940 kdenetwork kdenetwork.spec GPL
92964 samba-2.0.6 samba.spec GPL
91213 anaconda-6.2.2 anaconda.spec GPL
88128 cvs-1.10.7 cvs.spec GPL
87940 isdn4k-utils isdn4k-utils.spec GPL
85383 xpdf-0.90 xpdf.spec GPL
81719 inn-2.2.2 inn.spec GPL
79997 WindowMaker-0.61.1 WindowMaker.spec GPL
78787 extace-1.2.15 extace.spec GPL
75257 xpilot-4.1.0 xpilot.spec GPL
72425 gnome-core-1.0.55 gnome-core.spec GPL
70260 groff-1.15 groff.spec GPL
69265 fvwm-2.2.4 fvwm2.spec GPL
69246 linux-86 dev86.spec GPL
68884 squid-2.3.STABLE1 squid.spec GPL
68560 bash-2.03 bash2.spec GPL
68453 kdegraphics kdegraphics-1.1.2.spec GPL
55667 uucp-1.06.1 uucp.spec GPL
54935 gnupg-1.0.1 gnupg.spec GPL
54603 glade-0.5.5 glade.spec GPL
53141 AfterStep-1.8.0 AfterStep.spec GPL
52808 kdeutils kdeutils-1.1.2.spec GPL
51592 enlightenment-0.15.5 enlightenment.spec GPL
50970 cdrecord-1.8 cdrecord.spec GPL

48223 kdemultimedia kdemultimedia-1.1.2.spec GPL
47067 bash-1.14.7 bash-1.14.7.spec GPL
45811 mutt-1.0.1 mutt-i.spec GPL
45485 guile-1.3 guile.spec GPL
39861 rpm-3.0.4 rpm.spec GPL
38927 xmms-1.0.1 xmms.spec GPL
38453 zsh-3.0.7 zsh.spec GPL
36338 textutils-2.0a textutils.spec GPL
36239 xllamp-0.9-alpha3 xllamp.spec GPL
35397 gtk-engines-0.10 gtk-engines.spec GPL
35136 php-2.0.1 php.spec GPL
34768 fileutils-4.0p fileutils.spec GPL
32277 pilot-link.0.9.3 pilot-link.spec GPL
31994 korganizer korganizer-1.1.2.spec GPL
30438 gnome-pim-1.0.55 gnome-pim.spec GPL
30122 scheme-3.2 umb-scheme-3.2.spec GPL
29730 screen-3.9.5 screen.spec GPL
29315 jed jed.spec GPL
29091 xchat-1.4.0 xchat.spec GPL
28897 ncpfs-2.2.0.17 ncpfs.spec GPL
28449 slrn-0.9.6.2 slrn.spec GPL
28186 texinfo-4.0 texinfo.spec GPL
28169 e2fsprogs-1.18 e2fsprogs.spec GPL
28118 slang slang.spec GPL
27860 kdegames kdegames-1.1.2.spec GPL
27117 librep-0.10 librep.spec GPL
26673 lout-3.17 lout.spec GPL
26363 gawk-3.0.4 gawk.spec GPL
25915 gv-3.5.8 gv.spec GPL
24910 kdedadmin kdedadmin-1.1.2.spec GPL
24387 mars_nwe mars-nwe.spec GPL
23666 enscript-1.6.1 enscript.spec GPL
22373 sawmill-0.24 sawmill.spec GPL
22279 make-3.78.1 make.spec GPL
21593 xboard-4.0.5 xboard.spec GPL
19500 gnome-utils-1.0.50 gnome-utils.spec GPL
19065 joe joe.spec GPL
18151 git-4.3.19 git.spec GPL
17939 sh-utils-2.0 sh-utils.spec GPL
17765 mtools-3.9.6 mtools.spec GPL
17750 gettext-0.10.35 gettext.spec GPL
17682 bc-1.05 bc.spec GPL
15967 SVGATextMode-1.9-src SVGATextMode.spec GPL
15868 kpilot-3.1b9 kpilot-3.1.spec GPL
15851 taper-6.9a taper.spec GPL
15638 mod_perl-1.21 mod_perl.spec GPL
15522 console-tools-0.3.3 console-tools.spec GPL
14941 readline-2.2.1 readline.spec GPL
14912 ispell-3.1 ispell.spec GPL
14871 gnuchess-4.0.pl80 gnuchess.spec GPL
14774 flex-2.5.4 flex.spec GPL
14587 multimedia multimedia.spec GPL
14427 mawk-1.2.2 mawk-1.2.0.spec GPL
14363 automake-1.4 automake.spec GPL
14350 rsync-2.4.1 rsync.spec GPL
14299 nfs-utils-0.1.6 nfs-utils.spec GPL
14269 rcs-5.7 rcs.spec GPL
14255 tar-1.13.17 tar.spec GPL
14105 wmakerconf-2.1 wmakerconf.spec GPL
14039 less-346 less.spec GPL
13586 wget-1.5.3 wget.spec GPL
13504 rp3-1.0.7 rp3.spec GPL
13241 iproute2 iproute.spec GPL
13100 silo-0.9.8 silo.spec.sparc GPL
12633 minicom-1.83.0 minicom.spec GPL

12124 ypserv-1.3.9 ypserv.spec GPL
11790 lrzsz-0.12.20 lrzsz.spec GPL
11775 modutils-2.3.9 modutils.spec GPL
11721 enlightenment-conf-0.15 e-conf.spec GPL
11633 net-tools-1.54 net-tools.spec GPL
11404 findutils-4.1 findutils.spec GPL
11299 xmorph-1999dec12 xmorph.spec GPL
10958 kpackage-1.3.10 kpackage.spec GPL
10914 diffutils-2.7 diffutils.spec GPL
10404 gnorpm-0.9 gnorpm.spec GPL
10271 gqview-0.7.0 gqview.spec GPL
10088 piranha piranha.spec GPL
10013 grep-2.4 grep.spec GPL
9961 procps-2.0.6 procps.spec GPL
9801 man-1.5h1 man.spec GPL
9725 gpm-1.18.1 gpm.spec GPL
9699 bison-1.28 bison.spec GPL
9263 ctags-3.4 ctags.spec GPL
9138 gftp-2.0.6a gftp.spec GPL
8939 mkisofs-1.12b5 mkisofs.spec GPL
8572 psgml-1.2.1 psgml.spec GPL
8356 gedit-0.6.1 gedit.spec GPL
8303 dip-3.3.7o dip.spec GPL
7740 sed-3.02 sed.spec GPL
7617 cpio-2.4.2 cpio.spec GPL
7615 esound-0.2.17 esound.spec GPL
7570 sharutils-4.2.1 sharutils.spec GPL
7427 ed-0.2 ed.spec GPL
7227 cdparanoia-III-alpha9.6 cdparanoia.spec GPL
7095 xgammon-0.98 xgammon.spec GPL
7030 ee-0.3.11 ee.spec GPL
6611 patch-2.5 patch.spec GPL
6391 m4-1.4 m4-1.4.spec GPL
6306 gzip-1.2.4a gzip.spec GPL
6172 sash-3.4 sash.spec GPL
6090 joystick-1.2.15 joystick.spec GPL
6072 kdoc kdoc-2.0.spec GPL
6043 irda-utils-0.9.10 irda-utils.spec GPL
6033 sysvinit-2.78 SysVinit.spec GPL
6025 pnm2ppa pnm2ppa.spec GPL
5981 indent-2.2.5 indent.spec GPL
5960 isapnptools-1.21 isapnptools.spec GPL
5594 isdn-config isdn-config.spec GPL
5526 efax-0.9 efax.spec GPL
5383 acct-6.3.2 psacct-6.3.spec GPL
5115 libtool-1.3.4 libtool.spec GPL
4996 bzip2-0.9.5d bzip2.spec GPL
4780 make-3.78.1_pvm-0.5 make_pvm.spec GPL
4542 gpgp-0.4 gpgp.spec GPL
4430 gperf-2.7 gperf.spec GPL
4367 aumix-1.30.1 aumix.spec GPL
4038 sysklogd-1.3-31 sysklogd.spec GPL
4024 rep-gtk-0.8 rep-gtk.spec GPL
3929 initscripts-5.00 initscripts.spec GPL
3896 ltrace-0.3.10 ltrace.spec GPL
3885 phhttpd-0.1.0 phhttpd.spec GPL
3855 pciutils-2.1.5 pciutils.spec GPL
3675 dosfstools-2.2 dosfstools.spec GPL
3651 ipchains-1.3.9 ipchains.spec GPL
3625 autofs-3.1.4 autofs.spec GPL
3438 yp-tools-2.4 yp-tools.spec GPL
3433 dialog-0.6 dialog-0.6.spec GPL
3415 ext2ed-0.1 ext2ed.spec GPL
3315 gdbm-1.8.0 gdbm.spec GPL
3245 ypbind-3.3 ypbind.spec GPL

3219 playmidi-2.4 playmidi.spec GPL
3084 at-3.1.7 at.spec GPL
3051 dhcpcd-1.3.18-pl3 dhcpcd.spec GPL
3012 apmd apmd.spec GPL
2835 gkermi-1.0 gkermi.spec GPL
2810 kdetoys kdetoys-1.1.2.spec GPL
2758 autoconf-2.13 autoconf.spec GPL
2705 autorun-2.61 autorun.spec GPL
2661 cdp-0.33 cdp.spec GPL
2597 pythonlib-1.23 pythonlib.spec GPL
2580 magicdev-0.2.7 magicdev.spec GPL
2531 gnome-kberos-0.2 gnome-kberos.spec GPL
2490 sndconfig-0.43 sndconfig.spec GPL
2486 bug-buddy-0.7 bug-buddy.spec GPL
2459 usermode-1.20 usermode.spec GPL
2424 raidtools-0.90 raidtools.spec GPL
2407 nc nc.spec GPL
2324 up2date-1.13 up2date.spec GPL
2270 memprof-0.3.0 memprof.spec GPL
2268 which-2.9 which-2.spec GPL
2200 printtool printtool.spec GPL
2163 gnome-linuxconf-0.25 gnome-linuxconf.spec GPL
2065 units-1.55 units.spec GPL
1977 wmconfig-0.9.8 wmconfig.spec GPL
1883 slocate-2.1 slocate.spec GPL
1765 fbset-2.1 fbset.spec GPL
1634 netcfg-2.25 netcfg.spec GPL
1621 urlview-0.7 urlview.spec GPL
1525 logrotate-3.3.2 logrotate.spec GPL
1452 time-1.7 time.spec GPL
1226 procinfo-17 procinfo.spec GPL
1182 auth_ldap-1.4.0 auth_ldap.spec GPL
1146 prtconf-1.3 prtconf.spec.sparc GPL
1143 anacron-2.1 anacron.spec GPL
1075 authconfig-3.0.3 authconfig.spec GPL
1049 kpppload-1.04 kpppload.spec GPL
953 switchdesk-2.1 switchdesk.spec GPL
874 ldconfig-1999-02-21 ldconfig.spec GPL
834 ElectricFence-2.1 ElectricFence.spec GPL
816 rpmlint-0.8 rpmlint.spec GPL
809 kdpms-0.2.8 kdpms.spec GPL
742 setserial-2.15 setserial.spec GPL
728 tree-1.2 tree.spec GPL
717 chkconfig-1.1.2 chkconfig.spec GPL
657 eject-2.0.2 eject.spec GPL
585 usernet-1.0.9 usernet.spec GPL
549 genromfs-0.3 genromfs.spec GPL
548 tksysv-1.1 tksysv.spec GPL
537 minlabel-1.2 minlabel.spec.alpha GPL
497 locale_config-0.2 locale_config.spec GPL
491 helptool-2.4 helptool-2.4.spec GPL
480 elftoaout-2.2 elftoaout.spec.sparc GPL
463 tmpwatch-2.2 tmpwatch.spec GPL
445 rhs-printfilters-1.63 rhs-printfilters.spec GPL
404 control-panel-3.13 control-panel.spec GPL
368 kbdconfig-1.9.2.4 kbdconfig.spec GPL
368 vlock-1.3 vlock.spec GPL
367 timetool-2.7.3 timetool.spec GPL
347 kernelcfg-0.5 kernelcfg.spec GPL
346 timeconfig-3.0.3 timeconfig.spec GPL
343 chkfontpath-1.7 chkfontpath.spec GPL
343 mingetty-0.9.4 mingetty-0.9.4.spec GPL
332 ethtool-1.0 ethtool.spec.sparc GPL
314 mkbootdisk-1.2.5 mkbootdisk.spec GPL
301 xsri-1.0 xsri.spec GPL

288 mkinitrd-2.4.1 mkinitrd.spec GPL
265 sysreport-1.0 sysreport.spec GPL
255 ipvsadm-1.1 ipvsadm.spec GPL
240 open-1.4 open-1.4.spec GPL
236 xtoolwait-1.2 xtoolwait-1.2.spec GPL
222 mkkickstart-2.1 mkkickstart.spec GPL
213 rhmask rhmask.spec GPL
132 rdate-1.0 rdate.spec GPL
82 modemtool-1.21 modemtool-1.21.spec GPL
56 shaper shapercfg.spec GPL
52 spar32-1.1 spar32.spec.sparc GPL
23 locale-ja-9 locale-ja.spec GPL
0 ghostscript-fonts-5.50 ghostscript-fonts.spec GPL
0 gimp-data-extras-1.0.0 gimp-data-extras.spec GPL
0 kudzu-0.36 kudzu.spec GPL
0 solemul-1.1 solemul.spec.sparc GPL
0 specs30-6.2 specs30.spec GPL
0 users-guide-1.0.72 gnome-users-guide.spec GPL
1941 isicom isicom.spec GPL (not Firmware)
0 urw-fonts-2.0 urw-fonts.spec GPL, URW holds copyright

6234 awesfx-0.4.3a awesfx.spec GPL/distributable
231072 Mesa Mesa.spec GPL/XFree86
20433 pam-0.72 pam.spec GPL or BSD
9551 pwdb-0.61 pwdb.spec GPL or BSD
6592 xosview-1.7.1 xosview.spec GPL with some BSD

415026 glibc-2.1.3 glibc-2.1.spec LGPL
138118 gtk+-1.2.6 gtk+.spec LGPL
128672 gnome-libs-1.0.55 gnome-libs.spec LGPL
80343 kdelibs kdelibs-1.1.2.spec LGPL
49325 imlib-1.9.7 imlib.spec LGPL
41205 gnome-games-1.0.51 gnome-games.spec LGPL
27040 mikmod-3.1.6 mikmod.spec LGPL
26146 libxml-1.8.6 libxml.spec LGPL
24583 gmp-2.0.2 gmp.spec LGPL
24270 gnome-python-1.0.51 gnome-python.spec LGPL
18835 glib-1.2.6 glib.spec LGPL
16785 control-center-1.0.51 control-center.spec LGPL
14516 libgtop-1.0.6 libgtop.spec LGPL
12593 audiofile-0.1.9 audiofile.spec LGPL
12463 gnome-objc-1.0.2 gnome-objc.spec LGPL
10267 libPropList-0.9.1 libPropList.spec LGPL
9873 nss_ldap-105 nss_ldap.spec LGPL
8491 gtop-1.0.5 gtop.spec LGPL
7859 libglade-0.11 libglade.spec LGPL
7041 newt-0.50.8 newt.spec LGPL
6827 gnome-media-1.0.51 gnome-media.spec LGPL
2645 libghttp-1.0.4 libghttp.spec LGPL
2455 fnlib-0.4 fnlib.spec LGPL
1280 pam_krb5-1 pam_krb5.spec LGPL
1099 popt-1.4 popt.spec LGPL
0 desktop-backgrounds-1.1 desktop-backgrounds.spec LGPL
0 gnome-audio-1.0.0 gnome-audio.spec LGPL
0 wvdial-1.41 wvdial.spec LGPL

60429 kdesupport kdesupport-1.1.2.spec LGPL/GPL
38548 ORBit-0.5.0 ORBit.spec LGPL/GPL
5744 gdm-2.0beta2 gdm.spec LGPL/GPL

1291745 XFree86-3.3.6 XFree86.spec MIT
94637 xlockmore-4.15 xlockmore.spec MIT
68997 blt2.4g blt.spec MIT
35812 xloadimage-4.1 xloadimage-4.1.spec MIT
34772 xpuzzles-5.4.1 xpuzzles.spec MIT

28261 xfishtank-2.1tp xfishtank.spec MIT
27022 x3270-3.1.1 x3270.spec MIT
26608 Xaw3d-1.3 Xaw3d-1.3.spec MIT
25479 xpaint xpaint.spec MIT
18885 Xl1R6-contrib-3.3.2 Xl1R6-contrib-3.3.2.spec MIT
18020 xboing xboing.spec MIT
8540 xxgdb-1.12 xxgdb.spec MIT
7826 xpm-3.4k xpm.spec MIT
7255 lilo lilo-0.21.spec MIT
3860 xdaliclock-2.18 xdaliclock.spec MIT
2791 xjewel-1.6 xjewel.spec MIT
1856 pump-0.7.8 pump.spec MIT
1129 xbill-2.0 xbill.spec MIT
987 xmailbox-2.5 xmailbox-2.5.spec MIT
787 xsysinfo-1.7 xsysinfo.spec MIT
222 utempter-0.5.2 utempter.spec MIT
222220 krb5-1.1.1 krb5.spec MIT, freely distributable.

1020 MAKEDEV-2.5.2 MAKEDEV.spec none
797 termcap-2.0.8 termcap.spec none
280 stat-1.5 stat.spec none
261 bdf flush-1.5 bdf flush-1.5.spec None
24773 pdksh-5.2.14 pdksh.spec Public Domain
9607 cproto-4.6 cproto.spec Public Domain
6550 byacc-1.9 byacc-1.9.spec public domain
6496 pidentd-3.0.10 pidentd.spec Public domain
67 setup-1.2 setup.spec public domain
0 caching-nameserver-6.2 caching-nameserver.spec Public Domain
0 dev-2.7.18 dev.spec public domain
0 mailcap-2.0.6 mailcap.spec public domain
0 rootfiles rootfiles.spec public domain
0 setup-2.1.8 setup.spec public domain

205082 qt-2.1.0-beta1 qt-2.1.0.spec QPL
71810 jikes jikes.spec IBM Public License Version 1.0
0 redhat-logos redhat-logos.spec Copyright 1999 Red Hat, Inc.
1435 ncompress-4.2.4 ncompress-4.2.4.spec unknown

AfterStep-APPS.spec:Summary: Various applets for use with AfterStep and compatible window managers.

AfterStep.spec:Summary: An X window manager which emulates the look and feel of NEXTSTEP(R).

AnotherLevel.spec:Summary: A customized configuration of the fvwm2 window manager.

ElectricFence.spec:Summary: A debugger which detects memory allocation violations.

ImageMagick.spec:Summary: An X application for displaying and manipulating images.

ImageMagick.spec:%package devel

ImageMagick.spec:Summary: Static libraries and header files for ImageMagick app development.

MAKEDEV.spec:Summary: A script for creating the device files in /dev.

Mesa.spec:Summary: A 3-D graphics library similar to OpenGL.

Mesa.spec:%package devel

Mesa.spec:Summary: Development files for the Mesa 3-D graphics library.

ORBit.spec:Summary: A high-performance CORBA Object Request Broker.

ORBit.spec:%package devel

ORBit.spec:Summary: Development libraries, header files and utilities for ORBit.

SVGATextMode.spec:Summary: A utility for improving the appearance of text consoles.

SysVinit.spec:Summary: Programs which control basic system processes.

WindowMaker.spec:Summary: A window manager for the X Window System.

X11R6-contrib-3.3.2.spec:Summary: A collection of user-contributed X Window System programs.

XFree86-ISO8859-2.spec:Summary: Central European language fonts for the X Window System.

XFree86-ISO8859-2.spec:%package 75dpi-fonts

XFree86-ISO8859-2.spec:Summary: A set of 75 dpi Central European language fonts for X.

XFree86-ISO8859-2.spec:%package 100dpi-fonts

XFree86-ISO8859-2.spec:Summary: ISO 8859-2 fonts in 100 dpi resolution for the X Window System.

XFree86-ISO8859-2.spec:%package Typel-fonts

XFree86-ISO8859-2.spec:Summary: Type 1 scalable Central European language (ISO8859-2) fonts for X.

XFree86-ISO8859-7.spec:Summary: Greek language fonts for the X Window System.

XFree86-ISO8859-7.spec:%package 75dpi-fonts

XFree86-ISO8859-7.spec:Summary: ISO 8859-7 fonts in 75 dpi resolution for the X Window System.

XFree86-ISO8859-7.spec:%package 100dpi-fonts

XFree86-ISO8859-7.spec:Summary: ISO 8859-7 fonts in 100 dpi resolution for the X Window System.

XFree86-ISO8859-7.spec:%package Typel-fonts

XFree86-ISO8859-7.spec:Summary: Type 1 scalable Greek (ISO 8859-7) fonts

XFree86-ISO8859-9.spec:Summary: Turkish language fonts and modmaps for X.

XFree86-ISO8859-9.spec:%package 75dpi-fonts

XFree86-ISO8859-9.spec:Summary: 75 dpi Turkish (ISO8859-9) fonts for X.

XFree86-ISO8859-9.spec:%package 100dpi-fonts

XFree86-ISO8859-9.spec:Summary: 100 dpi Turkish (ISO8859-9) fonts for X.

XFree86.spec:Summary: The basic fonts, programs and docs for an X workstation.

XFree86.spec:%package 75dpi-fonts

XFree86.spec:Summary: A set of 75 dpi resolution fonts for the X Window System.

XFree86.spec:%package 100dpi-fonts

XFree86.spec:Summary: X Window System 100dpi fonts.

XFree86.spec:%package cyrillic-fonts

XFree86.spec:Summary: Cyrillic fonts for X.

XFree86.spec:%package libs

XFree86.spec:Summary: Shared libraries needed by the X Window System version 11 release 6.

XFree86.spec:%package devel

XFree86.spec:Summary: X11R6 static libraries, headers and programming man pages.

XFree86.spec:%package doc

XFree86.spec:Summary: Documentation on various X11 programming interfaces.

XFree86.spec:%package S3

XFree86.spec:Summary: The XFree86 server for video cards based on the S3 chip.

XFree86.spec:%package I128

XFree86.spec:Summary: The XFree86 server for Number Nine Imagine 128 video cards.

XFree86.spec:%package S3V

XFree86.spec:Summary: The XFree86 server for video cards based on the S3 Virge chip.
XFree86.spec:%package Mach64
XFree86.spec:Summary: The XFree86 server for Mach64 based video cards.
XFree86.spec:%package Sun
XFree86.spec:Summary: X server for Suns with monochrome and 8-bit color SBUS framebuffers.
XFree86.spec:%package SunMono
XFree86.spec:Summary: X server for Sun computers with monochrome SBUS framebuffers only.
XFree86.spec:%package Sun24
XFree86.spec:Summary: The X server for Suns with all supported SBUS framebuffers.
XFree86.spec:%package Xvfb
XFree86.spec:Summary: A virtual framebuffer X Windows System server for XFree86.
XFree86.spec:%package Xnest
XFree86.spec:Summary: A nested XFree86 server.
XFree86.spec:%package 8514
XFree86.spec:Summary: The XFree86 server program for older IBM 8514 or compatible video cards.
XFree86.spec:%package AGX
XFree86.spec:Summary: The XFree86 server for AGX-based video cards.
XFree86.spec:%package Mach32
XFree86.spec:Summary: The XFree86 server for Mach32 based video cards.
XFree86.spec:%package Mach8
XFree86.spec:Summary: The XFree86 server for Mach8 video cards.
XFree86.spec:%package Mono
XFree86.spec:Summary: A generic XFree86 monochrome server for VGA cards.
XFree86.spec:%package P9000
XFree86.spec:Summary: The XFree86 server for P9000 cards.
XFree86.spec:%package SVGA
XFree86.spec:Summary: An XFree86 server for most simple framebuffer SVGA devices.
XFree86.spec:%package VGA16
XFree86.spec:Summary: A generic XFree86 server for VGA16 boards.
XFree86.spec:%package W32
XFree86.spec:Summary: The XFree86 server for video cards based on ET4000/W32 chips.
XFree86.spec:%package 3DLabs
XFree86.spec:Summary: The XFree86 server for 3DLabs video cards.
XFree86.spec:%package TGA
XFree86.spec:Summary: X server for systems with Digital TGA boards based on DC21040 chips.
XFree86.spec:%package FBDev
XFree86.spec:Summary: The X server for the generic frame buffer device on some machines.
XFree86.spec:%package XF86Setup
XFree86.spec:Summary: A graphical user interface configuration tool for the X Window System.
XFree86.spec:%package xfs
XFree86.spec:Summary: A font server for the X Window System.
Xaw3d-1.3.spec:Summary: A version of the MIT Athena widget set for X.
Xaw3d-1.3.spec:%package devel
Xaw3d-1.3.spec:Summary: Header files and static libraries for development using Xaw3d.
Xconfigurator.spec:Summary: The Red Hat Linux configuration tool for the X Window System.
aboot.spec.alpha:Summary: A bootloader which can be started from the SRM console.
adjtimex.spec:Summary: A utility for adjusting kernel time variables.
am-utils.spec:Summary: Automount utilities including an updated version of Amd.
anaconda.spec:Summary: The Red Hat Linux installer.
anacron.spec:Summary: A cron-like program that can run jobs lost during downtime.
anonftp.spec:Summary: A fast, read-only, anonymous FTP server.
apache.spec:Summary: The most widely used Web server on the Internet.
apache.spec:%package devel
apache.spec:Summary: Development tools for the Apache Web server.
apache.spec:%package manual
apache.spec:Summary: Documentation for the Apache Web server.
apmd.spec:Summary: Advanced Power Management (APM) BIOS utilities for laptops.
ash-0.2.spec:Summary: A smaller version of the Bourne shell (sh).

at.spec:Summary: Job spooling tools.

audioctl.spec.sparc:Summary: An audio control application for Linux/SPARC systems.

audiofile.spec:Summary: A library for accessing various audio file formats.

audiofile.spec:%package devel

audiofile.spec:Summary: Libraries, includes and other files to develop audiofile applications.

aumix.spec:Summary: An ncurses-based audio mixer.

auth_ldap.spec:Summary: This is a LDAP authentication module for Apache

authconfig.spec:Summary: Text-mode tool for setting up NIS and shadow passwords.

autoconf.spec:Summary: A GNU tool for automatically configuring source code.

autofs.spec:Summary: A tool for automatically mounting and unmounting filesystems.

automake.spec:Summary: A GNU tool for automatically creating Makefiles.

autorun.spec:Summary: A CD-ROM mounting utility.

awesfx.spec:Summary: Utility programs for the AWE32 sound driver.

basesystem.spec:Summary: The skeleton package which defines a simple Red Hat Linux system.

bash-1.14.7.spec:Summary: The GNU Bourne Again shell (bash) version 1.14.

bash2.spec:Summary: The GNU Bourne Again shell (bash) version 2.03.

bash2.spec:%package doc

bash2.spec:Summary: Documentation for the GNU Bourne Again shell (bash) version 2.03.

bc.spec:Summary: GNU's bc (a numeric processing language) and dc (a calculator).

bdflush-1.5.spec:Summary: The daemon which starts the flushing of dirty buffers back to disk.

bind.spec:Summary: A DNS (Domain Name System) server.

bind.spec:%package utils

bind.spec:Summary: Utilities for querying DNS name servers.

bind.spec:%package devel

bind.spec:Summary: Include files and libraries needed for bind DNS development.

binutils.spec:Summary: A GNU collection of binary utilities.

bison.spec:Summary: A GNU general-purpose parser generator.

blt.spec:Summary: A Tk toolkit extension, including widgets, geometry managers, etc.

bootparamd.spec:Summary: A server process which provides boot information to diskless clients.

bug-buddy.spec:Summary: A GUI bug reporting tool for the GNOME GUI desktop.

byacc-1.9.spec:Summary: A public domain Yacc parser generator.

bzip2.spec:Summary: A file compression utility.

caching-nameserver.spec:Summary: The configuration files for setting up a caching name server.

cdecl-2.5.spec:Summary: Programs for encoding and decoding C and C++ function declarations.

cdp.spec:Summary: An interactive text-mode program for playing audio CD-ROMs.

cdparanoia.spec:Summary: A Compact Disc Digital Audio (CDDA) extraction tool.

cdrecord.spec:Summary: A command line CD/DVD recording program.

cdrecord.spec:%package devel

cdrecord.spec:Summary: The libschily SCSI user level transport library.

cdrecord.spec:%package -n mkisofs

cdrecord.spec:Summary: Creates an image of an ISO9660 filesystem.

cdrecord.spec:%package -n cdda2wav

cdrecord.spec:Summary: A utility for sampling/copying .wav files from digital audio CDs.

chkconfig.spec:Summary: A system tool for maintaining the /etc/rc.d hierarchy.

chkconfig.spec:%package -n ntsysv

chkconfig.spec:Summary: A tool to set the stop/start of system services in a runlevel.

chkfontpath.spec:Summary: Simple interface for editing the font path for the X font server.

cleanfeed.spec:Summary: A spam filter for Usenet news servers.

compat-binutils.spec.old:Summary: A GNU collection of binary utilities.

compat-egcs.spec.old:Summary: Experimental GNU Compiler System for Red Hat 5.2 backwards compatibility

compat-egcs.spec.old:%package c++

compat-egcs.spec.old:Summary: C++ support for Red Hat 5.2 backwards compatibility C compiler

compat-egcs.spec.old:%package objc

compat-egcs.spec.old:Summary: Objective C support for Red Hat 5.2 backwards compatibility C compiler

compat-egcs.spec.old:%package g77
compat-egcs.spec.old:Summary: Fortran 77 support for Red Hat 5.2 backwards compatibility C compiler
compat-glibc.spec.old:Summary: GNU libc for Red Hat Linux 5.2 backwards compatibility
compat-libs.spec.old:Summary: Runtime and developemnt libraries for Red Hat Linux 5.2 backwards compatibility
comsat.spec:Summary: A mail checker client and the comsat mail checking server.
console-tools.spec:Summary: Tools for configuring the console.
control-center.spec:Summary: The GNOME Control Center.
control-center.spec:%package devel
control-center.spec:Summary: The GNOME Control Center development environment.
control-panel.spec:Summary: A Red Hat sysadmin utility program launcher for X.
cpio.spec:Summary: A GNU archiving program.
cproto.spec:Summary: Generates function prototypes and variable declarations from C code.
cracklib.spec:Summary: A password-checking library.
cracklib.spec:%package dicts
cracklib.spec:Summary: The standard CrackLib dictionaries.
crontabs.spec:Summary: Root crontab files used to schedule the execution of programs.
ctags.spec:Summary: A C programming language indexing and/or cross-reference tool.
cvs.spec:Summary: A version control system.
cxhextris.spec:Summary: An X Window System color version of xhextris.
desktop-backgrounds.spec:Summary: Desktop background images.
dev.spec:Summary: The most commonly-used entries in the /dev directory.
dev86.spec:Summary: A real mode 80x86 assembler and linker.
dhcp.spec:Summary: A DHCP (Dynamic Host Configuration Protocol) server and relay agent.
dhcpcd.spec:Summary: A DHCP (Dynamic Host Configuration Protocol) client.
dialog-0.6.spec:Summary: A utility for creating TTY dialog boxes.
diffstat.spec:Summary: A utility which provides statistics based on the output of diff.
diffutils.spec:Summary: A GNU collection of diff utilities.
dip.spec:Summary: Handles the connections needed for dialup IP links.
docbook.spec:Summary: An SGML DTD for technical documentation.
dosfstools.spec:Summary: Utilities for making and checking MS-DOS FAT filesystems on Linux.
dump.spec:Summary: Programs for backing up and restoring filesystems.
dump.spec:%package -n rmt
dump.spec:Summary: Provides certain programs with access to remote tape devices.
dump.spec:%package -n dump-static
dump.spec:Summary: Programs for backing up and restoring filesystems.
e-conf.spec:Summary: The Enlightenment window manager configuration tool.
e2fsprogs.spec:Summary: Utilities for managing the second extended (ext2) filesystem.
e2fsprogs.spec:%package devel
e2fsprogs.spec:Summary: Ext2 filesystem-specific static libraries and headers.
ed.spec:Summary: The GNU line editor.
ee.spec:Summary: The Electric Eyes image viewer application.
efax.spec:Summary: A program for faxing using a Class 1, 2 or 2.0 fax modem.
egcs.spec:Summary: The GNU Compiler Collection.
egcs.spec:%package c++
egcs.spec:Summary: C++ support for the gcc compiler.
egcs.spec:%package objc
egcs.spec:Summary: Objective C support for the gcc compiler.
egcs.spec:%package g77
egcs.spec:Summary: Fortran 77 support for the gcc compiler.
egcs.spec:%package -n libstdc++
egcs.spec:Summary: The EGCS Standard C++ library v3.
egcs.spec:%package -n cpp
egcs.spec:Summary: The GNU C-Compatible Compiler Preprocessor.
egcs64-19980921.spec.sparc.not:Summary: Experimental GNU Compiler System for sparc64
eject.spec:Summary: A program that ejects removable media using software control.
elftoaout.spec.sparc:Summary: A utility for converting ELF binaries to a.out binaries.
elm.spec:Summary: The elm mail user agent.
emacs.spec:Summary: The libraries needed to run the GNU Emacs text editor.
emacs.spec:%package el
emacs.spec:Summary: The sources for elisp programs included with Emacs.
emacs.spec:%package leim

emacs.spec:Summary: Emacs Lisp code for input methods for international characters.
emacs.spec:%package nox
emacs.spec:Summary: The Emacs text editor without support for the X Window System.
emacs.spec:%package X11
emacs.spec:Summary: The Emacs text editor for the X Window System.
enlightenment.spec:Summary: The Enlightenment window manager.
enscript.spec:Summary: A plain ASCII to PostScript converter.
esound.spec:Summary: Allows several audio streams to play on a single audio device.
esound.spec:%package devel
esound.spec:Summary: Development files for Esound applications.
etcskel.spec:Summary: Red Hat Linux default files for new users' home directories.
ethtool.spec.sparc:Summary: Ethernet settings tool for SPARC HME cards
exmh.spec:Summary: The exmh mail handling system.
ext2ed.spec:Summary: An ext2 filesystem editor.
extace.spec:Summary: A GNOME sound displayer.
fbset.spec:Summary: Tools for managing a frame buffer's video mode properties.
fetchmail.spec:Summary: A remote mail retrieval and forwarding utility.
fetchmail.spec:%package -n fetchmailconf
fetchmail.spec:Summary: A GUI utility for configuring your fetchmail preferences.
file.spec:Summary: A utility for determining file types.
filesystem.spec:Summary: The basic directory layout for a Linux system.
fileutils.spec:Summary: The GNU versions of common file management utilities.
findutils.spec:Summary: The GNU versions of find utilities (find and xargs).
finger.spec:Summary: The finger client.
finger.spec:%package server
finger.spec:Summary: The finger daemon.
flex.spec:Summary: A tool for creating scanners (text pattern recognizers).
fnlib.spec:Summary: A color font rendering library for X11R6.
fnlib.spec:%package devel
fnlib.spec:Summary: Headers, static libraries and documentation for Fnlib.
fortune-mod.spec:Summary: A program which will display a fortune.
freetype-1.3.1.spec:Summary: A free and portable TrueType font rendering engine.
freetype-1.3.1.spec:%package utils
freetype-1.3.1.spec:Summary: Several utilities to manipulate and examine TrueType fonts.
freetype-1.3.1.spec:%package devel
freetype-1.3.1.spec:Summary: Header files and static library for development with FreeType.
ftp.spec:Summary: The standard UNIX FTP (File Transfer Protocol) client.
fvwm2.spec:Summary: An improved version of the FVWM window manager for X.
fvwm2.spec:%package icons
fvwm2.spec:Summary: Graphics used by the FVWM and FVWM2 window managers.
fwhois.spec:Summary: A finger-style whois program.
gated.spec:Summary: The public release version of the Gated routing daemon.
gawk.spec:Summary: The GNU version of the awk text processing utility.
gd.spec:Summary: A graphics library for drawing .gif files.
gd.spec:%package devel
gd.spec:Summary: The development libraries and header files for gd.
gdb.spec:Summary: A GNU source-level debugger for C, C++ and Fortran.
gdbm.spec:Summary: A GNU set of database routines which use extensible hashing.
gdbm.spec:%package devel
gdbm.spec:Summary: Development libraries and header files for the gdbm library.
gdm.spec:Summary: The GNOME Display Manager.
gedit.spec:Summary: gEdit is a small but powerful text editor for GNOME.
gedit.spec:%package devel
gedit.spec:Summary: The files needed for developing plug-ins for the gEdit editor.
genromfs.spec:Summary: Utility for creating romfs filesystems.
gettext.spec:Summary: GNU libraries and utilities for producing multi-lingual messages.
getty_ps.spec:Summary: The getty and ugetty programs.
gftp.spec:Summary: A multi-threaded FTP client for the X Window System.
ghostscript-5.50.spec:Summary: A PostScript(TM) interpreter and renderer.
ghostscript-fonts.spec:Summary: Fonts for the Ghostscript PostScript(TM) interpreter.
giftrans.spec:Summary: A program for making transparent GIFs from non-transparent GIFs.
gimp-data-extras.spec:Summary: The GNU Image Manipulation Program
gimp.spec:Summary: The GNU Image Manipulation Program.

gimp.spec:%package devel
gimp.spec:Summary: The GIMP plug-in and extension development kit.
gimp.spec:%package libgimp
gimp.spec:Summary: Libraries for the GIMP (GNU Image Manipulation Program).
git.spec:Summary: A set of GNU Interactive Tools.
gkermit.spec:Summary: A utility for transferring files using the Kermit protocol.
glade.spec:Summary: A GTK+ GUI builder.
glib.spec:Summary: A library of handy utility functions.
glib.spec:%package devel
glib.spec:Summary: The GIMP ToolKit (GTK+) and GIMP Drawing Kit (GDK) support library.
glibc-2.1.spec:Summary: The GNU libc libraries.
glibc-2.1.spec:%package devel
glibc-2.1.spec:Summary: Header and object files for development using standard C libraries.
glibc-2.1.spec:%package profile
glibc-2.1.spec:Summary: The GNU libc libraries, including support for gprof profiling.
glibc-2.1.spec:%package -n nscd
glibc-2.1.spec:Summary: A Name Service Caching Daemon (nscd).
gmp.spec:Summary: A GNU arbitrary precision library.
gmp.spec:%package devel
gmp.spec:Summary: Development tools for the GNU MP arbitrary precision library.
gnome-audio.spec:Summary: Sounds for GNOME events.
gnome-audio.spec:%package extra
gnome-audio.spec:Summary: Files needed for customizing GNOME event sounds.
gnome-core.spec:Summary: The core programs for the GNOME GUI desktop environment.
gnome-core.spec:%package devel
gnome-core.spec:Summary: The core libraries and include files for GNOME panel development.
gnome-games.spec:Summary: GNOME games.
gnome-games.spec:%package devel
gnome-games.spec:Summary: GNOME games development libraries.
gnome-kerberos.spec:Summary: Kerberos 5 tools for GNOME.
gnome-libs.spec:Summary: The libraries needed to run the GNOME GUI desktop environment.
gnome-libs.spec:%package devel
gnome-libs.spec:Summary: Libraries and include files for developing GNOME applications.
gnome-linuxconf.spec:Summary: The GNOME front-end for linuxconf.
gnome-media.spec:Summary: GNOME media programs.
gnome-objc.spec:Summary: Objective C libraries for the GNOME desktop environment.
gnome-objc.spec:%package devel
gnome-objc.spec:Summary: Files needed to develop Objective C GNOME applications.
gnome-pim.spec:Summary: The GNOME Personal Information Manager.
gnome-pim.spec:%package devel
gnome-pim.spec:Summary: GNOME PIM development files
gnome-python.spec:Summary: The sources for the PyGTK and PyGNOME Python extension modules.
gnome-python.spec:%package -n pygtk
gnome-python.spec:Summary: Python bindings for the GTK+ widget set.
gnome-python.spec:%package -n pygtk-libglade
gnome-python.spec:Summary: A wrapper for the libglade library for use with PyGTK.
gnome-python.spec:%package -n pygnome-libglade
gnome-python.spec:Summary: GNOME support for the libglade python wrapper
gnome-python.spec:%package -n pygnome
gnome-python.spec:Summary: Python bindings for the GNOME libraries.
gnome-python.spec:%package -n pygnome-applet
gnome-python.spec:Summary: Python bindings for GNOME Panel applets.
gnome-python.spec:%package -n pygnome-caplet
gnome-python.spec:Summary: Python bindings for GNOME Panel applets.
gnome-users-guide.spec:Summary: The GNOME Users' Guide.
gnome-utils.spec:Summary: GNOME utility programs.
gnorpm.spec:Summary: A graphical front-end to RPM for GNOME.
gnotepad+.spec:Summary: A small and simple but versatile text editor for X.
gnuchess.spec:Summary: The GNU chess program.
gnnumeric.spec:Summary: A full-featured spreadsheet for GNOME.
gnupg.spec:Summary: A GNU utility for secure communication and data storage.
gnuplot.spec:Summary: A program for plotting mathematical expressions and data.

gperf.spec:Summary: A perfect hash function generator.
gpgp.spec:Summary: Gnome frontend for GNU Privacy Guard
gpm.spec:Summary: A mouse server for the Linux console.
gpm.spec:%package devel
gpm.spec:Summary: Libraries and header files for developing mouse driven programs.
gqview.spec:Summary: An image viewer.
grep.spec:Summary: The GNU versions of grep pattern matching utilities.
groff.spec:Summary: A document formatting system.
groff.spec:%package perl
groff.spec:Summary: Parts of the groff formatting system that require Perl.
groff.spec:%package gxditview
groff.spec:Summary: An X previewer for groff text processor output.
gtk+.spec:Summary: The GIMP ToolKit (GTK+), a library for creating GUIs for X.
gtk+.spec:%package devel
gtk+.spec:Summary: Development tools for GTK+ (GIMP ToolKit) applications.
gtk+10.spec.old:Summary: The GIMP ToolKit (GTK+), a library for creating GUIs for X.
gtk+10.spec.old:%package -n glib10
gtk+10.spec.old:Summary: A library of handy utility functions.
gtk-engines.spec:Summary: Theme engines for GTK+.
gtop.spec:Summary: The GNOME system monitor.
guile.spec:Summary: A GNU implementation of Scheme for application extensibility.
guile.spec:%package devel
guile.spec:Summary: Libraries and header files for the GUILE extensibility library.
gv.spec:Summary: A X front-end for the Ghostscript PostScript(TM) interpreter.
gxedit.spec:Summary: A multi-function text editor which uses GTK+.
gzip.spec:Summary: The GNU data compression program.
hdparm.spec:Summary: A utility for displaying and/or setting hard disk parameters.
helptool-2.4.spec:Summary: A graphical user interface tool which searches for help files.
ical.spec:Summary: An X Window System-based calendar program.
imap.spec:Summary: Server daemons for IMAP and POP network mail protocols.
imap.spec:%package devel
imap.spec:Summary: Development tools for programs which will use the IMAP library.
imlib.spec:Summary: An image loading and rendering library for X11R6.
imlib.spec:%package devel
imlib.spec:Summary: Development tools for Imlib applications.
imlib.spec:%package cfgeditor
imlib.spec:Summary: A configuration editor for the Imlib library.
indent.spec:Summary: A GNU program for formatting C code.
indexhtml.spec:Summary: The Web page you'll see after installing Red Hat Linux.
inetd.spec:Summary: The inetd networking program.
initscripts.spec:Summary: The inittab file and the /etc/rc.d scripts.
inn.spec:Summary: The InterNetNews (INN) system, an Usenet news server.
inn.spec:%package devel
inn.spec:Summary: The INN (InterNetNews) library.
inn.spec:%package -n inews
inn.spec:Summary: Sends Usenet articles to a local news server for distribution.
install-guide.spec:Summary: The Linux Documentation Project Getting Started Guide in HTML format.
intimed-1.10.spec:Summary: A time server for synchronizing networked machines' clocks.
ipchains.spec:Summary: Tools for managing Linux kernel packet filtering capabilities.
iproute.spec:Summary: Enhanced IP routing and network devices configuration tools
iputils.spec:Summary: The ping program for checking to see if network hosts are alive.
ipvsadm.spec:Summary: administration tool for Virtual Server
ircii.spec:Summary: An Internet Relay Chat (IRC) client.
irda-utils.spec:Summary: Utilities for infrared communication between devices.
isapnptools.spec:Summary: Utilities for configuring ISA Plug-and-Play (PnP) devices.
isdn-config.spec:Summary: A tool for configuring ISDN dialup connections.
isdn4k-utils.spec:Summary: Utilities for configuring an ISDN subsystem.
isdn4k-utils.spec:%package -n xisdnload
isdn4k-utils.spec:Summary: An ISDN connection load average display for the X Window System.
isicom.spec:Summary: Multitech IntelligentSerialInternal (ISI) Support Tools
ispell.spec:Summary: An interactive spelling checker program.
ispell.spec:%package catalan

ispell.spec:Summary: Ispell spelling checker files for checking Catalan words.
ispell.spec:%package czech
ispell.spec:Summary: Ispell spelling checker files for Czech words.
ispell.spec:%package danish
ispell.spec:Summary: Ispell spelling checker files for Danish words.
ispell.spec:%package dutch
ispell.spec:Summary: Files needed to create an Ispell dictionary of Dutch words.
ispell.spec:%package esperanto
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Esperanto words.
ispell.spec:%package french
ispell.spec:Summary: Files needed to generate an Ispell dictionary of French words.
ispell.spec:%package german
ispell.spec:Summary: Files needed to generate an Ispell dictionary of German words.
ispell.spec:%package greek
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Greek words.
ispell.spec:%package italian
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Italian words.
ispell.spec:%package norwegian
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Norwegian words.
ispell.spec:%package polish
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Polish words.
ispell.spec:%package portuguese
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Portuguese words.
ispell.spec:%package russian
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Russian words.
ispell.spec:%package spanish
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Spanish words.
ispell.spec:%package swedish
ispell.spec:Summary: Files needed to generate an Ispell dictionary of Swedish words.
ispell.spec:%package dicts
ispell.spec:Summary: Files for generating non-English Ispell dictionaries.
jade.spec:Summary: A parser and tools for SGML plus DSSSL.
jadetex.spec:Summary: LaTeX macros for converting Jade TeX output into DVI/PS/PDF.
jed.spec:Summary: A fast, compact editor based on the S-Lang screen library.
jed.spec:%package common
jed.spec:Summary: Files needed by any Jed text editor.
jed.spec:%package xjed
jed.spec:Summary: The X Window System version of the Jed text editor.
jed.spec:%package -n rgrep
jed.spec:Summary: A grep utility which can recursively descend through directories.
jikes.spec:Summary: A Java source file to bytecode compiler.
joe.spec:Summary: An easy to use, modeless text editor.
joystick.spec:Summary: Utilities for configuring most popular joysticks.
kaffe.spec:Summary: A free virtual machine for running Java(TM) code.
kbdconfig.spec:Summary: A text-based interface for setting and loading a keyboard map.
kdeadmin-1.1.2.spec:Summary: K Desktop Environment - System Administration Tools
kdebase-1.1.2.spec:Summary: Core files for the K Desktop Environment (KDE).
kdebase-1.1.2.spec:%package lowcolor-icons
kdebase-1.1.2.spec:Summary: Low color (8-bit display friendly) icons for KDE.
kdebase-1.1.2.spec:%package 3d-screensavers
kdebase-1.1.2.spec:Summary: 3-D (OpenGL) screensavers for the K Desktop Environment.
kdegames-1.1.2.spec:Summary: K Desktop Environment - Games
kdegraphics-1.1.2.spec:Summary: K Desktop Environment - Graphics Applications
kdelibs-1.1.2.spec:Summary: Libraries for the K Desktop Environment (KDE).
kdelibs-1.1.2.spec:%package devel
kdelibs-1.1.2.spec:Summary: Header files and documentation for compiling KDE applications.
kdemultimedia-1.1.2.spec:Summary: Multimedia applications for the K Desktop Environment (KDE).
kdenetwork.spec:Summary: Network applications for the K Desktop Environment (KDE).
kdesupport-1.1.2.spec:Summary: Support libraries for the K Desktop Environment (KDE).
kdesupport-1.1.2.spec:%package devel
kdesupport-1.1.2.spec:Summary: Header files and documentation for the KDE support libraries.
kdetoys-1.1.2.spec:Summary: K Desktop Environment - Toys and Amusements

kdeutils-1.1.2.spec:Summary: K Desktop Environment - Utilities

kdcc-2.0.spec:Summary: Documentation for the K Desktop Environment (KDE) 2.0.

kdpms.spec:Summary: kdpms - Configures the power management functions of your monitor.

kernel-2.2.14.spec:Summary: The Linux kernel (the core of the Linux operating system).

kernel-2.2.14.spec:%package source

kernel-2.2.14.spec:Summary: The source code for the Linux kernel.

kernel-2.2.14.spec:%package headers

kernel-2.2.14.spec:Summary: Header files for the Linux kernel.

kernel-2.2.14.spec:%package doc

kernel-2.2.14.spec:Summary: Various documentation bits found in the kernel source.

kernel-2.2.14.spec:%package pcmcia-cs

kernel-2.2.14.spec:Summary: The daemon and device drivers for using PCMCIA adapters.

kernel-2.2.14.spec:%package utils

kernel-2.2.14.spec:Summary: Kernel related utilities.

kernel-2.2.14.spec:%package ibcs

kernel-2.2.14.spec:Summary: Files which allow iBCS2 programs to run.

kernel-2.2.14.spec:%package smp

kernel-2.2.14.spec:Summary: The Linux kernel compiled for SMP machines.

kernel-2.2.14.spec:%package BOOT

kernel-2.2.14.spec:Summary: The version of the Linux kernel used on installation boot disks.

kernel-2.2.14.spec:%package BOOTsmp

kernel-2.2.14.spec:Summary: The Linux kernel used on installation boot disks for SMP machines.

kernelcfg.spec:Summary: A Red Hat utility for configuring the kernel daemon.

korganizer-1.1.2.spec:Summary: A calendar and scheduling program for KDE.

kpackage.spec:Summary: Kpackage is a graphical RPM package manager for KDE.

kpilot-3.1.spec:Summary: KPilot - pilot synchronization tools for KDE

kpppload.spec:Summary: A PPP connection load monitor for KDE.

krb5.spec:Summary: Kerberos Authentication System

krb5.spec:%package configs

krb5.spec:Summary: Kerberos 5 sample configuration file(s).

krb5.spec:%package workstation

krb5.spec:Summary: Kerberos 5 programs for use on workstations.

krb5.spec:%package server

krb5.spec:Summary: Kerberos 5 server programs.

krb5.spec:%package libs

krb5.spec:Summary: Kerberos 5 shared libraries.

krb5.spec:%package devel

krb5.spec:Summary: Header files and libraries necessary to compile Kerberos 5 programs.

krbafs.spec:Summary: Kerberos to AFS bridging library, built against Kerberos 5.

krbafs.spec:%package utils

krbafs.spec:Summary: Kerberos / AFS utility binaries.

kterm-6.2.0.spec:Summary: A Kanji (Japanese character set) terminal emulator for X.

kudzu.spec:Summary: The Red Hat Linux hardware probing tool.

kudzu.spec:%package devel

kudzu.spec:Summary: The development library for hardware probing.

lam.spec:Summary: LAM (Local Area Multicomputer) programming environment

ld.so.spec:Summary: A dynamic loader for a.out format files.

ldconfig.spec:Summary: Creates a shared library cache and maintains symlinks for ld.so.

less.spec:Summary: A text file browser similar to more, but better.

libPropList.spec:Summary: Ensures program compatibility with GNUstep/OPENSTEP.

libc-5.3.12.spec.old:Summary: The compatibility libraries needed by old libc.so.5 applications.

libelf-0.6.4.spec:Summary: An ELF object file access library.

libghttp.spec:Summary: GNOME http client library.

libghttp.spec:%package devel

libghttp.spec:Summary: GNOME http client development

libglade.spec:Summary: The libglade library for loading user interfaces.

libglade.spec:%package devel

libglade.spec:Summary: The files needed for libglade application development.

libgr.spec:Summary: A library for handling different graphics file formats.

libgr.spec:%package devel

libgr.spec:Summary: Development tools for programs which will use the libgr library.

libgr.spec:%package progs

libgr.spec:Summary: Tools for manipulating graphics files in libgr supported formats.
libgtop.spec:Summary: The LibGTop library
libgtop.spec:%package devel
libgtop.spec:Summary: Libraries, includes and other files to develop LibGTop applications.
libgtop.spec:%package examples
libgtop.spec:Summary: These are examples for LibGTop, a library which retrieves information about your system, such as CPU and memory usage.
libjpeg-6a.spec:Summary: A backwards-compatible JPEG image manipulation library.
libjpeg.spec:Summary: A library for manipulating JPEG image format files.
libjpeg.spec:%package devel
libjpeg.spec:Summary: Development tools for programs which will use the libjpeg library.
libpng.spec:Summary: A library of functions for manipulating PNG image format files.
libpng.spec:%package devel
libpng.spec:Summary: Development tools for programs to manipulate PNG image format files.
librep.spec:Summary: An embeddable LISP environment.
librep.spec:%package devel
librep.spec:Summary: Include files and link libraries for librep development.
libtermcap.spec:Summary: A basic system library for accessing the termcap database.
libtermcap.spec:%package devel
libtermcap.spec:Summary: Development tools for programs which will access the termcap database.
libtiff.spec:Summary: A library of functions for manipulating TIFF format image files.
libtiff.spec:%package devel
libtiff.spec:Summary: Development tools for programs which will use the libtiff library.
libtool.spec:Summary: The GNU libtool, which simplifies the use of shared libraries.
libungif.spec:Summary: A library for manipulating GIF format image files.
libungif.spec:%package devel
libungif.spec:Summary: Development tools for programs which will use the libungif library.
libungif.spec:%package progs
libungif.spec:Summary: Programs for manipulating GIF format image files.
libxml.spec:Summary: An XML library.
libxml.spec:%package devel
libxml.spec:Summary: Libraries, includes and other files to develop libxml applications.
libxml10.spec.old:Summary: Backward compatibility XML library.
lilo-0.21.spec:Summary: The boot loader for Linux and other operating systems.
linuxconf.spec:Summary: An extremely capable system configuration tool.
linuxconf.spec:%package devel
linuxconf.spec:Summary: The tools needed for developing linuxconf modules.
locale-ja.spec:Summary: Locale definition files for Japanese.
locale_config.spec:Summary: Locale configuration tool.
logrotate.spec:Summary: Rotates, compresses, removes and mails system log files.
lout.spec:Summary: The Lout document formatting language.
lout.spec:%package doc
lout.spec:Summary: The documentation for the Lout document formatting language.
lpg.spec:Summary: The LDP's Linux programming guide in HTML format.
lpr.spec:Summary: A utility that manages print jobs.
lrzsz.spec:Summary: The lrz and lsz modem communications programs.
lslk.spec:Summary: A lock file lister.
lsof.spec:Summary: A utility which lists open files on a Linux/UNIX system.
ltrace.spec:Summary: Tracks runtime library calls from dynamically linked executables.
lynx.spec:Summary: A text-based Web browser.
m4-1.4.spec:Summary: The GNU macro processor.
macutils.spec:Summary: Utilities for manipulating Macintosh file formats.
magicdev.spec:Summary: A GNOME daemon for automatically mounting/playing CDs.
mailcap.spec:Summary: Associates helper applications with particular file types.
mailx.spec:Summary: The /bin/mail program, which is used to send mail via shell scripts.
make.spec:Summary: A GNU tool which simplifies the build process for users.
make_pvm.spec:Summary: PVM enhanced version of GNU make

man-pages.spec:Summary: Man (manual) pages from the Linux Documentation Project.

man.spec:Summary: A set of documentation tools: man, apropos and whatis.

mars-nwe.spec:Summary: NetWare file and print servers which run on Linux systems.

mawk-1.2.0.spec:Summary: An interpreter for the awk programming language.

mc.spec:Summary: A user-friendly file manager and visual shell.

mc.spec:%package -n gmc

mc.spec:Summary: The GNOME version of the Midnight Commander file manager.

mc.spec:%package -n mcserve

mc.spec:Summary: Server for the Midnight Commander network file management system.

memprof.spec:Summary: A tool for memory profiling and leak detection.

metamail.spec:Summary: A program for handling multimedia mail using the mailcap file.

mgetty.spec:Summary: A getty replacement for use with data and fax modems.

mgetty.spec:%package sendfax

mgetty.spec:Summary: Provides support for sending faxes over a modem.

mgetty.spec:%package voice

mgetty.spec:Summary: A program for using your modem and mgetty as an answering machine.

mgetty.spec:%package viewfax

mgetty.spec:Summary: An X Window System fax viewer.

mikmod.spec:Summary: A MOD music file player.

mingetty-0.9.4.spec:Summary: A compact getty program for virtual consoles only.

minicom.spec:Summary: A text-based modem control and terminal emulation program.

minlabel.spec.alpha:Summary: Creates, views and edits BSD UNIX style disk labels on Alphas.

mkbootdisk.spec:Summary: Creates an initial ramdisk image for preloading modules.

mkinitrd.spec:Summary: Creates an initial ramdisk image for preloading modules.

mkisofs.spec:Summary: Creates an image of an ISO9660 filesystem.

mkkickstart.spec:Summary: Writes a kickstart description of the current machine.

mktemp.spec:Summary: A small utility for safely making /tmp files.

mkxauth.spec:Summary: A utility for managing .Xauthority files.

mod_perl.spec:Summary: A Perl interpreter for the Apache Web server.

modemtool-1.21.spec:Summary: A tool for selecting the serial port your modem is connected to.

modutils.spec:Summary: The kernel daemon (kernelld) and kernel module utilities.

mount.spec:Summary: Programs for mounting and unmounting filesystems.

mount.spec:%package -n losetup

mount.spec:Summary: Programs for setting up and configuring loopback devices.

mouseconfig.spec:Summary: The Red Hat Linux mouse configuration tool.

mpage.spec:Summary: A tool for printing multiple pages of text on each printed page.

mpg123.spec:Summary: An MPEG audio player.

mt-st.spec:Summary: Programs to control tape device operations.

mttools.spec:Summary: Programs for accessing MS-DOS disks without mounting the disks.

multimedia.spec:Summary: Several X utilities mainly for use with multimedia files.

mutt-i.spec:Summary: A text mode mail user agent.

nag.spec:Summary: The Linux Documentation Project's Network Administrators' Guide.

nc.spec:Summary: Reads and writes data across network connections using TCP or UDP.

ncftp.spec:Summary: An improved FTP client.

ncompress-4.2.4.spec:Summary: Fast compression and decompression utilities.

ncpfs.spec:Summary: Utilities for the ncpfs filesystem, a NetWare client for Linux.

ncpfs.spec:%package -n ipxutils

ncpfs.spec:Summary: Tools for configuring and debugging IPX interfaces and networks.

ncurses.spec:Summary: A CRT screen handling and optimization package.

ncurses.spec:%package devel

ncurses.spec:Summary: The development files for applications which use ncurses.

ncurses3.spec.old:Summary: An older version (1.9.9e) of the ncurses terminal control library.

net-tools.spec:Summary: The basic tools for setting up networking.

netcfg.spec:Summary: A network configuration tool.

netscape-sparc.spec.sparc.not:Summary: A Web browser, news reader and e-mail client.

netscape-sparc.spec.sparc.not:%package -n netscape-common

netscape-sparc.spec.sparc.not:Summary: Files shared by the Netscape Navigator and Communicator.

netscape-sparc.spec.sparc.not:%package -n netscape-communicator

netscape-sparc.spec.sparc.not:Summary: A Web browser, news reader and e-mail client.

netscape-sparc.spec.sparc.not:%package -n netscape-navigator

netscape-sparc.spec.sparc.not:Summary: The Netscape Navigator Web browser.

netscape.spec:Summary: A Web browser, news reader and e-mail client.
netscape.spec:%package -n netscape-common
netscape.spec:Summary: Files shared by the Netscape Navigator and Communicator.
netscape.spec:%package -n netscape-communicator
netscape.spec:Summary: A Web browser, news reader and e-mail client.
netscape.spec:%package -n netscape-navigator
netscape.spec:Summary: The Netscape Navigator Web browser.
newt.spec:Summary: A development library for text mode user interfaces.
newt.spec:%package devel
newt.spec:Summary: Newt windowing toolkit development files.
nfs-utils.spec:Summary: NFS utilities and supporting daemons for the kernel NFS server.
nmh.spec:Summary: A capable mail handling system with a command line interface.
nss_ldap.spec:Summary: NSS library and PAM module for LDAP.
open-1.4.spec:Summary: A tool which will start a program on a virtual console.
openldap.spec:Summary: LDAP servers, libraries, utilities, tools and sample clients.
openldap.spec:%package devel
openldap.spec:Summary: OpenLDAP development libraries and header files.
p2c.spec:Summary: A Pascal to C translator.
p2c.spec:%package devel
p2c.spec:Summary: Files for p2c Pascal to C translator development.
pam.spec:Summary: A security tool which provides authentication for applications.
pam_krb5.spec:Summary: Kerberos 5 Pluggable Authentication Module
passwd.spec:Summary: The passwd utility for setting/changing passwords using PAM.
patch.spec:Summary: The GNU patch command, for modifying/upgrading files.
pciutils.spec:Summary: Linux PCI utilities.
pciutils.spec:%package devel
pciutils.spec:Summary: Linux PCI development library.
pdksh.spec:Summary: A public domain clone of the Korn shell (ksh).
perl.spec:Summary: The Perl programming language.
phhttpd.spec:Summary: An HTTP accelerator.
php.spec:Summary: The PHP HTML-embedded scripting language for use with Apache.
php.spec:%package manual
php.spec:Summary: The manual for PHP3, in HTML format.
php.spec:%package pgsql
php.spec:Summary: A PostgreSQL database module for PHP3.
php.spec:%package imap
php.spec:Summary: An IMAP module for PHP3.
php.spec:%package ldap
php.spec:Summary: An Apache Web server module for PHP3 applications that use LDAP.
phpfi.spec:Summary: The PHP/FI PHP language module for the Apache Web server.
pidentd.spec:Summary: An implementation of the RFC1413 identification server.
pilot-link.spec:Summary: File transfer utilities between Linux and PalmPilots.
pilot-link.spec:%package devel
pilot-link.spec:Summary: PalmPilot development header files.
pine.spec:Summary: A commonly used, MIME compliant mail and news reader.
piranha.spec:Summary: Cluster administration tools
piranha.spec:%package gui
piranha.spec:Summary: The cute little graphical configuration tool for lvs
piranha.spec:%package docs
piranha.spec:Summary: Documentation for the Red Hat clustering environment
playmidi.spec:Summary: A MIDI sound file player.
playmidi.spec:%package X11
playmidi.spec:Summary: An X Window System based MIDI sound file player.
pmake.spec:Summary: The BSD 4.4 version of make.
pmake.spec:%package customs
pmake.spec:Summary: A remote execution facility for pmake.
pnm2ppa.spec:Summary: Drivers for printing to HP PPA printers
popt.spec:Summary: A C library for parsing command line parameters.
portmap.spec:Summary: A program which manages RPC connections.
postgresql.spec:Summary: PostgreSQL client programs and libraries.
postgresql.spec:%package server
postgresql.spec:Summary: The programs needed to create and run a PostgreSQL server.
postgresql.spec:%package devel
postgresql.spec:Summary: PostgreSQL development header files and libraries.

postgresql.spec:%package tcl
postgresql.spec:Summary: A Tcl-based GUI front end for psql and related PostgreSQL clients.
postgresql.spec:%package odbc
postgresql.spec:Summary: The ODBC driver needed for accessing a PostgreSQL DB using ODBC.
postgresql.spec:%package perl
postgresql.spec:Summary: Development module needed for Perl code to access a PostgreSQL DB.
postgresql.spec:%package python
postgresql.spec:Summary: Development module for Python code to access a PostgreSQL DB.
postgresql.spec:%package jdbc
postgresql.spec:Summary: Files needed for Java programs to access a PostgreSQL database.
postgresql.spec:%package test
postgresql.spec:Summary: The test suite distributed with PostgreSQL.
ppp.spec:Summary: The PPP (Point-to-Point Protocol) daemon.
printtool.spec:Summary: A printer configuration tool with a graphical user interface.
procinfo.spec:Summary: A tool for gathering and displaying system information.
procmail.spec:Summary: The procmail mail processing program.
procps.spec:Summary: Utilities for monitoring your system and processes on your system.
procps.spec:%package X11
procps.spec:Summary: An X based system message monitoring utility.
prtconf.spec.sparc:Summary: SPARC OpenPROM dump utility
psacct-6.3.spec:Summary: Utilities for monitoring process activities.
psgml.spec:Summary: A GNU emacs major mode for editing SGML documents.
psmisc.spec:Summary: Utilities for managing processes on your system.
pump.spec:Summary: A Bootp and DHCP client for automatic IP configuration.
pvm.spec:Summary: Libraries for distributed computing.
pvm.spec:%package gui
pvm.spec:Summary: TCL/TK graphical frontend to monitor and manage a PVM cluster.
pwdb.spec:Summary: The password database library.
pxe.spec:Summary: A Linux PXE (Preboot eXecution Environment) server.
python.spec:Summary: An interpreted, interactive object-oriented programming language.
python.spec:%package devel
python.spec:Summary: The libraries and header files needed for Python development.
python.spec:%package tools
python.spec:Summary: A collection of development tools included with Python.
python.spec:%package docs
python.spec:Summary: Documentation for the Python programming language.
python.spec:%package -n tkinter
python.spec:Summary: A graphical user interface for the Python scripting language.
pythonlib.spec:Summary: A library of Python code used by various Red Hat Linux programs.
qt-2.1.0.spec:Summary: The shared library for the Qt GUI toolkit.
qt-2.1.0.spec:%package devel
qt-2.1.0.spec:Summary: Development files and documentation for the Qt GUI toolkit.
qt-2.1.0.spec:%package GL
qt-2.1.0.spec:Summary: An OpenGL add-on for the Qt software toolkit.
qt-2.1.0.spec:%package NSPlugin
qt-2.1.0.spec:Summary: An add-on for creating Netscape plug-ins using the Qt GUI toolkit.
qt-2.1.0.spec:%package Xt
qt-2.1.0.spec:Summary: An Xt (X Toolkit) compatibility add-on for the Qt GUI toolkit.
qtlx.spec.old:Summary: A backward compatible library for apps linked to Qt 1.x.
qtlx.spec.old:%package devel
qtlx.spec.old:Summary: Qt 1.x development files for legacy applications.
qtlx.spec.old:%package GL
qtlx.spec.old:Summary: An OpenGL (3-D graphics) add-on for the Qt GUI toolkit.
quickstrip.spec.alpha:Summary: An Alpha tool for removing debugging information from kernel images.
quota.spec:Summary: System administration tools for monitoring users' disk usage.
raidtools.spec:Summary: Tools for creating and maintaining software RAID devices.
rcs.spec:Summary: Revision Control System (RCS) file version management tools.
rdate.spec:Summary: Tool for retrieving the date/time from another machine on your

network.

rdist.spec:Summary: Maintains identical copies of files on multiple machines.

readline.spec:Summary: A library for reading and returning lines from a terminal.

readline.spec:%package devel

readline.spec:Summary: Development files for programs which will use the readline library.

redhat-logos.spec:Summary: Red Hat-related icons and pictures.

redhat-release.spec.in:Summary: Red Hat Linux release file

rep-gtk.spec:Summary: GTK+ binding for librep Lisp environment

rep-gtk.spec:%package libglade

rep-gtk.spec:Summary: librep binding for the libglade library for loading user interfaces.

rhmask.spec:Summary: Generates and restores mask files based on original and update files.

rhs-printfilters.spec:Summary: Red Hat print filters, for use with the printtool.

rootfiles.spec:Summary: The basic required files for the root user's directory.

routed.spec:Summary: The routing daemon which maintains routing tables.

rp3.spec:Summary: The Red Hat graphical PPP management tool.

rpm.spec:Summary: The Red Hat package management system.

rpm.spec:%package devel

rpm.spec:Summary: Development files for applications which will manipulate RPM packages.

rpm.spec:%package build

rpm.spec:Summary: Scripts and executable programs used to build packages.

rpm.spec:%package python

rpm.spec:Summary: Python bindings for apps which will manipulate RPM packages.

rpm.spec:%package -n popt

rpm.spec:Summary: A C library for parsing command line parameters.

rpm2html.spec:Summary: Translates an RPM database and dependency information into HTML.

rpmdb-redhat.spec.in:Summary: The entire rpm database for the %{rpmdbdistribution} distribution.

rpmfind.spec:Summary: Finds and transfers RPM files for a specified program.

rpmlint.spec:Summary: A development tool for checking the correctness of RPM packages.

rsh.spec:Summary: Clients for remote access commands (rsh, rlogin, rcp).

rsh.spec:%package server

rsh.spec:Summary: Servers for remote access commands (rsh, rlogin, rcp).

rsync.spec:Summary: A program for synchronizing files over a network.

rusers.spec:Summary: Displays the users logged into machines on the local network.

rusers.spec:%package server

rusers.spec:Summary: Server for the rusers protocol.

rwall.spec:Summary: Client for sending messages to a host's logged in users.

rwall.spec:%package server

rwall.spec:Summary: Server for sending messages to a host's logged in users.

rwho.spec:Summary: Displays who is logged in to local network machines.

rxvt.spec:Summary: A color VT102 terminal emulator for the X Window System.

sag.spec:Summary: The LDP's System Administrators' Guide in HTML format.

samba.spec:Summary: Samba SMB server.

samba.spec:%package client

samba.spec:Summary: Samba (SMB) client programs.

samba.spec:%package common

samba.spec:Summary: Files used by both Samba servers and clients.

sash.spec:Summary: A statically linked shell, including some built-in basic commands.

sawmill.spec:Summary: An extensible window manager for the X Window System.

sawmill.spec:%package gnome

sawmill.spec:Summary: GNOME support for the sawmill window manager.

sawmill.spec:%package themer

sawmill.spec:Summary: A GUI for creating sawmill window manager themes.

screen.spec:Summary: A screen manager that supports multiple logins on one terminal.

sed.spec:Summary: A GNU stream text editor.

sendmail.spec:Summary: A widely used Mail Transport Agent (MTA).

sendmail.spec:%package doc

sendmail.spec:Summary: Documentation about the Sendmail Mail Transport Agent program.

sendmail.spec:%package cf

sendmail.spec:Summary: The files needed to reconfigure Sendmail.

setserial.spec:Summary: A utility for configuring serial ports.

setup.spec:Summary: A set of system configuration and setup files.

setuptool.spec:Summary: A text mode system configuration tool.

sgml-common.spec:Summary: Common SGML catalog and DTD files.

sgml-tools.spec:Summary: A text formatting package based on SGML.

sh-utils.spec:Summary: A set of GNU utilities commonly used in shell scripts.

shadow-utils.spec:Summary: Utilities for managing shadow password files and user/group accounts.

shapecfg.spec:Summary: A configuration tool for setting traffic bandwidth parameters.

sharutils.spec:Summary: The GNU shar utilities for packaging and unpacking shell archives.

silo.spec.sparc:Summary: The SILO boot loader for SPARCs.

slang.spec:Summary: The shared library for the S-Lang extension language.

slang.spec:%package devel

slang.spec:Summary: The static library and header files for development using S-Lang.

sliplogin.spec:Summary: A login program for SLIP connections.

slocate.spec:Summary: Finds files on a system via a central database.

slrn.spec:Summary: A powerful, easy to use, threaded Internet news reader.

slrn.spec:%package pull

slrn.spec:Summary: Offline news reading support for the SLRN news reader.

sndconfig.spec:Summary: The Red Hat Linux sound configuration tool.

solemul.spec.sparc:Summary: Directory tree required by Solaris emulation module.

sox.spec:Summary: A general purpose sound file conversion tool.

sox.spec:%package -n sox-devel

sox.spec:Summary: The SoX sound file format converter libraries.

sparc32.spec.sparc:Summary: A SPARC32 compilation environment.

specspo.spec:Summary: Red Hat package descriptions, summaries, and groups.

squid.spec:Summary: The Squid proxy caching server.

stat.spec:Summary: A tool for finding out information about a specified file.

statserial.spec:Summary: A tool which displays the status of serial port modem lines.

strace.spec:Summary: Tracks and displays system calls associated with a running process.

stylesheets.spec:Summary: The stylesheets used at Cygnus for SGML conversion.

svgalib.spec:Summary: A low-level fullscreen SVGA graphics library.

svgalib.spec:%package devel

svgalib.spec:Summary: Development tools for programs using the SVGAlib graphics library.

switchdesk.spec:Summary: A desktop environment switcher for GNOME, KDE and AnotherLevel.

switchdesk.spec:%package kde

switchdesk.spec:Summary: A KDE interface for the Desktop Switcher.

switchdesk.spec:%package gnome

switchdesk.spec:Summary: A GNOME interface for the Desktop Switcher.

symlinks-1.2.spec:Summary: A utility which maintains a system's symbolic links.

sysklogd.spec:Summary: System logging and kernel message trapping daemons.

sysreport.spec:Summary: Gathers system hardware and configuration information.

talk.spec:Summary: Talk client for one-on-one Internet chatting.

talk.spec:%package server

talk.spec:Summary: The talk server for one-on-one Internet chatting.

taper.spec:Summary: A menu-driven file backup system.

tar.spec:Summary: A GNU file archiving program.

tcltk.spec:Summary: A Tcl/Tk development environment: tcl, tk, tix, tclX, expect, and itcl.

tcltk.spec:%package -n tcl

tcltk.spec:Summary: An embeddable scripting language.

tcltk.spec:%package -n tk

tcltk.spec:Summary: The Tk GUI toolkit for Tcl, with shared libraries.

tcltk.spec:%package -n expect

tcltk.spec:Summary: A tcl extension for simplifying program-script interaction.

tcltk.spec:%package -n tclx

tcltk.spec:Summary: Tcl/Tk extensions for POSIX systems.

tcltk.spec:%package -n tix

tcltk.spec:Summary: A set of capable widgets for Tk.

tcltk.spec:%package -n itcl

tcltk.spec:Summary: Object-oriented mega-widgets for Tcl.

tcp_wrappers.spec:Summary: A security tool which acts as a wrapper for TCP daemons.

tcpdump.spec:Summary: A network traffic monitoring tool.
tcpdump.spec:%package -n libpcap
tcpdump.spec:Summary: A system-independent interface for user-level packet capture.
tcpdump.spec:%package -n arpwatrch
tcpdump.spec:Summary: Network monitoring tools for tracking IP addresses on a network.
tcsh.spec:Summary: An enhanced version of csh, the C shell.
telnet.spec:Summary: The client program for the telnet remote login protocol.
telnet.spec:%package server
telnet.spec:Summary: The server program for the telnet remote login protocol.
termcap.spec:Summary: The terminal feature database used by certain applications.
tetex.spec:Summary: The TeX text formatting system.
tetex.spec:%package latex
tetex.spec:Summary: The LaTeX front end for the TeX text formatting system.
tetex.spec:%package xdvi
tetex.spec:Summary: An X viewer for DVI files.
tetex.spec:%package dvips
tetex.spec:Summary: A DVI to PostScript converter for the TeX text formatting system.
tetex.spec:%package dvilj
tetex.spec:Summary: A DVI to HP PCL (Printer Control Language) converter.
tetex.spec:%package afm
tetex.spec:Summary: A converter for PostScript(TM) font metric files, for use with TeX.
tetex.spec:%package fonts
tetex.spec:Summary: The font files for the TeX text formatting system.
tetex.spec:%package doc
tetex.spec:Summary: The documentation files for the TeX text formatting system.
texinfo.spec:Summary: Tools needed to create Texinfo format documentation files.
texinfo.spec:%package -n info
texinfo.spec:Summary: A stand-alone TTY-based reader for GNU texinfo documentation.
textutils.spec:Summary: A set of GNU text file modifying utilities.
tftp.spec:Summary: The client for the Trivial File Transfer Protocol (TFTP).
tftp.spec:%package server
tftp.spec:Summary: The server for the Trivial File Transfer Protocol (TFTP).
time.spec:Summary: A GNU utility for monitoring a program's use of system resources.
timeconfig.spec:Summary: Text mode tools for setting system time parameters.
timed.spec:Summary: Programs for maintaining networked machines' time synchronization.
timetool.spec:Summary: A utility for setting the system's date and time.
tin.spec:Summary: A basic Internet news reader.
tksysv.spec:Summary: An X editor for editing runlevel services.
tmpwatch.spec:Summary: A utility for removing files based on when they were last accessed.
traceroute.spec:Summary: Traces the route taken by packets over a TCP/IP network.
transfig-3.2.1.spec:Summary: A utility for converting FIG files (made by xfig) to other formats.
tree.spec:Summary: A utility which displays a tree view of the contents of directories.
trn.spec:Summary: A news reader that displays postings in threaded format.
trojka.spec:Summary: A non-X game of falling blocks.
ucd-snmp.spec:Summary: A collection of SNMP protocol tools from UC-Davis.
ucd-snmp.spec:%package utils
ucd-snmp.spec:Summary: Network management utilities using SNMP, from the UCD-SNMP project.
ucd-snmp.spec:%package devel
ucd-snmp.spec:Summary: The development environment for the UCD-SNMP project.
umb-scheme-3.2.spec:Summary: An implementation of the Scheme programming language.
unarj.spec:Summary: An uncompressor for .arj format archive files.
units.spec:Summary: A utility for converting amounts from one unit to another.
unzip.spec:Summary: A utility for unpacking zip files.
up2date.spec:Summary: Red Hat Linux Update Agent
urlview.spec:Summary: An URL extractor/viewer for use with Mutt.
urw-fonts.spec:Summary: Free versions of the 35 standard PostScript fonts.
usermode.spec:Summary: Graphical tools for certain user account management tasks.
usernet.spec:Summary: A graphical utility for controlling network interfaces.
utempter.spec:Summary: A privileged helper for utmp/wtmp updates.
util-linux.spec:Summary: A collection of basic system utilities.
uucp.spec:Summary: The uucp utility for copying files between systems.
vim.spec:Summary: The VIM editor.

vim.spec:%package common
vim.spec:Summary: The common files needed by any version of the VIM editor.
vim.spec:%package minimal
vim.spec:Summary: A minimal version of the VIM editor.
vim.spec:%package enhanced
vim.spec:Summary: A version of the VIM editor which includes recent enhancements.
vim.spec:%package X11
vim.spec:Summary: The VIM version of the vi editor for the X Window System.
vixie-cron.spec:Summary: The Vixie cron daemon for executing specified programs at set times.
vlock.spec:Summary: A program which locks one or more virtual consoles.
w3c-libwww.spec:Summary: HTTP library of common code
w3c-libwww.spec:%package devel
w3c-libwww.spec:Summary: Libraries and header files for programs that use libwww.
w3c-libwww.spec:%package apps
w3c-libwww.spec:Summary: Applications built using Libwww web library: e.g. Robot, command line tool, etc.
wget.spec:Summary: A utility for retrieving files using the HTTP or FTP protocols.
which-2.spec:Summary: Displays where a particular program in your path is located.
wmakerconf.spec:Summary: A configuration tool for the Window Maker window manager for X.
wmconfig.spec:Summary: A helper application for configuring X window managers.
words-2.spec:Summary: A dictionary of English words for the /usr/dict directory.
wu-ftpd.spec:Summary: An FTP daemon provided by Washington University.
wvdial.spec:Summary: A heuristic autodialer for PPP connections.
x11amp.spec:Summary: X11 mp3 player with features not unlike WinAMP.
x11amp.spec:%package devel
x11amp.spec:Summary: Static libraries and header files for x11amp.
x3270.spec:Summary: An X Window System based IBM 3278/3279 terminal emulator.
xbill.spec:Summary: Stop Bill from loading his OS into all the computers.
xboard.spec:Summary: An X Window System graphical chessboard.
xboing.spec:Summary: A Breakout style X Window System based game.
xchat.spec:Summary: A GTK+ IRC (chat) client.
xcpustate.spec:Summary: An X Window System based CPU state monitor.
xdaliclock.spec:Summary: A clock for the X Window System.
xfig.spec:Summary: An X Window System tool for drawing basic vector graphics.
xfishtank.spec:Summary: An X Window System graphic display of an animated aquarium.
xgammon.spec:Summary: An X Window System based backgammon game.
xinitrc.spec:Summary: The default startup script for the X Window System.
xjewel.spec:Summary: An X Window System game of falling jewel blocks.
xlispstat.spec:Summary: An implementation of the Lisp language with statistics extensions.
xloadimage-4.1.spec:Summary: An X Window System based image viewer.
xlockmore.spec:Summary: An X terminal locking program.
xmailbox-2.5.spec:Summary: An X Window System utility which notifies you of new mail.
xmms.spec:Summary: An MP3 player for X which resembles Winamp.
xmms.spec:%package devel
xmms.spec:Summary: Static libraries and header files for Xmms plug-in development.
xmms.spec:%package gnome
xmms.spec:Summary: A GNOME panel applet for the Xmms multimedia player.
xmorph.spec:Summary: An X Window System tool for creating morphed images.
xntp3.spec:Summary: Synchronizes system time using the Network Time Protocol (NTP).
xosview.spec:Summary: An X Window System utility for monitoring system resources.
xpaint.spec:Summary: An X Window System image editing or paint program.
xpat.spec:Summary: A set of Solitaire type games for the X Window System.
xpdf.spec:Summary: A PDF file viewer for the X Window System.
xpilot.spec:Summary: An X Window System based multiplayer aerial combat game.
xpm.spec:Summary: A pixmap library for the X Window System.
xpm.spec:%package devel
xpm.spec:Summary: Tools for developing apps which will use the XPM pixmap library.
xpuzzles.spec:Summary: Geometric puzzles and toys for the X Window System.
xrn.spec:Summary: An X Window System based news reader.
xscreensaver.spec:Summary: A set of X Window System screensavers.
xsri.spec:Summary: A program for displaying images on the background for X.
xsysinfo.spec:Summary: An X Window System kernel parameter monitoring tool.

xtoolwait-1.2.spec:Summary: A utility which aims to decrease X session startup time.
xtrojka.spec:Summary: An X Window System falling blocks game.
xxgdb.spec:Summary: An X Window System graphical interface for the GNU gdb debugger.
yp-tools.spec:Summary: NIS (or YP) client programs.
ypbind.spec:Summary: The NIS daemon which binds NIS clients to an NIS domain.
ypserv.spec:Summary: The NIS (Network Information Service) server.
ytalk-3.1.spec:Summary: A chat program for multiple users.
zip.spec:Summary: A file compression and packaging utility compatible with PKZIP.
zlib.spec:Summary: The zlib compression and decompression library.
zlib.spec:%package devel
zlib.spec:Summary: Header files and libraries for developing apps which will use zlib.
zsh.spec:Summary: A shell similar to ksh, but with improvements.

SLOCCount

This is the home page of ``SLOCCount'', a set of tools for counting physical Source Lines of Code (SLOC) in a large number of languages of a potentially large set of programs. This suite of tools was used in my paper [Estimating Linux's Size](#) to measure the SLOC of an entire Linux distribution. It runs on Linux, Windows, and hopefully on other systems too. To run on Windows, you have to install [Cygwin](#) first.

SLOCCount has a number of ease-of-use features. You can easily install it, particularly on RPM-based Linux systems, and for most situations, all you need to do is type:

```
sloccount directoryname
```

SLOCCount is released under the General Public License (GPL), and can measure the following languages (common extensions for the language are listed in parentheses):

1. Ada (.ada, .ads, .adb)
2. Assembly (.s, .S, .asm)
3. awk (.awk)
4. Bourne shell and variants (.sh)
5. C (.c)
6. C++ (.C, .cpp, .cxx, .cc)
7. C shell (.csh)
8. Expect (.exp)
9. Fortran (.f)
10. Java (.java)
11. lex (.l)
12. LISP/Scheme (.el, .scm, .lsp, .jl)
13. Makefile (makefile) - not normally shown.
14. Objective-C (.m)
15. Pascal (.p)
16. Perl (.pl, .pm, .perl)
17. Python (.py)
18. sed (.sed)
19. SQL (.sql) - not normally shown.
20. TCL (.tcl, .tk, .itk)
21. Yacc (.y)

SLOCCount includes a number of heuristics, so it can automatically detect file types, even those that don't use the ``standard'' extensions, and conversely, it can detect many kinds of files that have a standard extension but aren't really of that type. The SLOC counters have enough smarts to handle oddities of several languages. For example, SLOCCount examines assembly language files, determines the comment scheme, and then correctly counts the lines automatically. It also correctly handles language constructs that are often mishandled by other tools, such as Python's constant strings

when used as comments and Perl's "perlpod" documentation.

SLOCCount comes with extensive documentation - you should be able to just pick it up and use it.

You can:

- [View the SLOCCount documentation](#)
 - [View the ChangeLog](#)
 - Download version 2.0 (the latest version). You can get it in the following formats:
 1. [tar.gz](#),
 2. [i386 binary RPM v4](#) (suitable for Red Hat Linux 7 on Intel),
 3. [Source RPM v4](#) (suitable for non-Intel, and should be recompilable on any RPM-based system using RPM version 4 or later).
 - Download an older version. Versions available:
 1. [version 1.0](#).
 2. [version 1.2](#).
 3. [version 1.4](#).
 4. [version 1.6](#).
 5. [version 1.8](#).
 6. version 1.9 as [tar.gz](#), [i386 binary RPM v4](#) (suitable for Red Hat Linux 7 on Intel), or [Source RPM v4](#) (suitable for non-Intel, and should be recompilable on any RPM-based system using RPM version 4 or later).
-

You can also view [my home page](#).

Secure Programming for Linux and Unix HOWTO

This is the main web site for the *Secure Programming for Linux and Unix HOWTO*. This document provides a set of design and implementation guidelines for writing secure programs for Linux and Unix systems. Such programs include application programs used as viewers of remote data, web applications (including CGI scripts), network servers, and setuid/setgid programs. This document includes specific guidance for a number of languages, including C, C++, Java, Perl, Python, and Ada95.

This document is part of the [Linux Documentation Project](http://www.linuxdoc.org/) (LDP), and hence is also distributed in various Linux distributions. However, note that the LDP's version or the version in a CD-ROM distribution may not be as current as the main (master) web site at "<http://www.dwheeler.com/secure-programs/>".

If you want to just view the HOWTO online, you can jump immediately to the [online HTML version](#). You can get download the HOWTO (say for printing) in lots of formats: pick from [PDF](#), [Postscript](#), [RTF](#), [tarball of all HTML files](#), [ASCII text](#) (not recommended), [gzipped SGML and related files \(DocBook DTD\)](#), and [uncompressed SGML \(DocBook DTD\)](#) formats. The uncompressed SGML version is the ``master" version I edit; all the other formats are generated from this master version. You can also see the [ChangeLog](#), and users of the SGML format may find the [makefile](#) useful.

If you have comments, proposed improvements, or intend to translate it to another human language, please send email to dwheeler@dwheeler.com. Notice that as of version 2.00, the document is in SGML using the DocBook DTD; previous versions up through version 1.60 were in SGML but they used the Linuxdoc DTD. I've kept [old versions of this document](#) to help translators deal with this transition in format. Originally this document was the ``Secure Programming for Linux HOWTO", but I've expanded it to cover Unix systems too. This document keeps getting longer than the typical HOWTO; I may eventually split this into a ``short form" and a ``longer form". I'm also thinking how to handle publication for the ``long form", since I think many people will want a nice bound version of it so they can read it easily.

Regarding translations: A Japanese translation is available in [HTML](#) and [SGML](#) formats. A Chinese translation is available - you can get it as [Chinese Big5](#) or [Chinese GB](#). A Spanish translation is in progress. I hope to mention or link to additional translations as I learn about them. Please contact me before translating, so that duplicate work can be avoided (for example, perhaps multiple translators could divide the work). Also, I'd like to link to them so that people who prefer languages other than English can quickly find a translation if available. I *cannot* guarantee that the translations accurately reflect the original English work; I'm sorry, but I'm simply not qualified to judge that. If you find an error in translation, please contact the translators directly. I am very grateful to these people who have taken the time to translate this fairly lengthy work.

I have a few little scripts and programs here that are related to the book's material, for example, [url.pl](#) is a short script I use for testing the complex URI validation patterns.

Feel free to see my main web site at <http://www.dwheeler.com/>.

CERT Coordination Center

[Home](#)[Site Index](#)[Search](#)[Contact](#)[Feed](#)[Vulnerabilities,
Incidents & Fixes](#)[Security
Practices and
Evaluations](#)[Survivability
Research and
Analysis](#)[Training
and
Education](#)**Options**[Advisories](#)[Vulnerability
Notes
Database](#)[Incident
Notes](#)[Current
Activity](#)**Related**[Summaries](#)[Tech Tips](#)[AirCERT](#)[Employment
Opportunities](#)**more links**[CERT Statistics](#)[Vulnerability
Disclosure Policy](#)[CERT
Knowledgebase](#)[System
Administrator
courses](#)[CSIRT courses](#)[Other Sources of
Security
Information](#)[Channels](#)**Message**

Welcome to the new Incidents, Quick Fixes, and Vulnerabilities area of the CERT/CC web site.

Related Sites[Internet Security Alliance](#)

CERT/CC Current Activity

The CERT/CC Current Activity web page is a regularly updated summary of the most frequent, high-impact types of security incidents and vulnerabilities currently being reported to the CERT/CC.

Last reviewed: Monday, September 24 2001 @ 12:51:02 EDT (-0400 UTC)

	Date Added	Last Updated
• W32/Nimda	September 18	September 20
• W32/SirCam Malicious Code	July 23	September 20
• Cache Corruption on Microsoft DNS Servers	August 31	August 31
• "Code Red" Related Activity	August 02	September 20
• Exploitation of a buffer overflow in telnetd	July 30	August 17
• Scans and Probes	-	August 17

W32/Nimda

The CERT/CC continues to receive a steady stream of reports of W32/Nimda although the volume of reports has dropped significantly since it first appeared on Tuesday. Sites are strongly encouraged to read [CERT Advisory CA-2001-26](#) for detailed information on W32/Nimda.

W32/SirCam Virus

The CERT/CC continues to receive reports of a piece of malicious code known as W32/SirCam. W32/SirCam arrives in email with the following characteristics:

Subject: (same as the name of the attached file)

Body of English version:
(first line) Hi! How are you?
(last line) See you later. Thanks

Body of Spanish version:
(first line) Hola como estas?
(last line) Nos vemos pronto, gracias.

Attachment: A random filename with a double extension (like example.doc.bat)

Detailed information about W32/SirCam can be found in [CERT Advisory CA-2001-22](#) or by visiting the sites listed on our [Computer Virus Resources](#) page. Users are strongly encouraged to visit their anti-virus vendor's website for information on how to properly remove W32/SirCam from an infected computer.

Cache Corruption on Microsoft DNS Servers

The CERT/CC has received reports from sites experiencing DNS cache corruption on systems running Microsoft DNS Server. The default configuration of this software allows for data from malicious or incorrectly configured DNS servers to be cached and supplied to client computers using that cache to resolve DNS information.

Please see:

- [CERT Incident Note IN-2001-11](#)

"Code Red" Related Activity

- **"Code Green" Worm**

The CERT/CC has received a few reports of a new worm known as "Code Green". The "Code Green"

worm is designed to patch systems vulnerable to the "Code Red" worms and attempt to remove backdoors left by "Code Red II".

"Code Green" will leave the following signature in web server logs (the presence of this log entry does not necessarily indicate an infection):

```
"GET /default.ida?Code_Green_<I_like_the_colour
_-_><AntiCodeRed-CodeRedIII-IDQ_Patcher>_V1.0_b
eta_written_by_'Der_HexXer'-Wuerzburg_Germany-_
is_dedicated_to_my_sisterli_'Doro'.Save_Whale_a
nd_visit_<www.buhaboard.de>_and_<www.buha-secur
ity.de>%u9090%u6858%ucbd3%u7801%u9090%u6858%ucb
d3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8
190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u
00=a HTTP/1.0"
```

• "Code Red II" Worm

The CERT/CC continues to receive reports of a worm commonly referred to as "Code Red II". The widespread, automated attack and propagation characteristics of "Code Red II" have caused bandwidth denial-of-service conditions in isolated portions of the Internet, particularly near groups of compromised hosts where "Code Red II" is running. Detailed information about the "Code Red II" worm can be found in:

- [CERT Incident Note IN-2001-09](#)

• "Code Red" Worm

We also continue to receive reports of the original "Code Red" worm. Machines infected by this worm started scanning the Internet for vulnerable servers again on September 1st 2001.

Please see these documents for more information:

- [A Very Real and Present Threat to the Internet: Resurgence in Code Red Scanning Activity](#)
- [CERT Advisory CA-2001-13](#)
- [CERT Advisory CA-2001-19](#)
- [CERT Advisory CA-2001-23](#)

• "Code Red" Worm Crashes IIS 4.0 Servers with URL Redirection Enabled

Along with the large number of "Code Red" and "Code Red II" reports indicating that systems are compromised, the CERT/CC has received a smaller yet still significant number of reports where Windows NT 4.0 IIS 4.0 systems have been adversely affected by the high volume of "Code Red" scanning activity. A recently discovered vulnerability can cause an IIS 4.0 server (patched against "Code Red" according to [Microsoft Security Bulletin MS01-033](#)) with URL redirection enabled to crash when scanned by the "Code Red" worms.

Please see:

- ["Code Red" Worm Crashes IIS 4.0 Servers with URL Redirection Enabled](#)

The CERT/CC is interested in receiving reports of "Code Red" activity. If machines under your administrative control are compromised, please send mail to cert@cert.org.

Exploitation of a buffer overflow in telnetd

The CERT/CC has received reports of exploitation of the buffer overflow in the telnetd program discussed in [CERT Advisory CA-2001-21](#). This vulnerability can crash the telnetd server, or be leveraged to gain root access to the host. Sites are encouraged to read the [advisory](#) and take appropriate steps to protect any machines running the telnetd service.

The CERT/CC is interested in receiving reports of this activity. If machines under your administrative control are compromised, please send mail to cert@cert.org.

Scans and Probes

We receive many daily reports of scanning and probing activity. The most frequent reports tend to involve services that have well-known vulnerabilities. Internet hosts continue to be affected by exploitation of well-known vulnerabilities in many of these services.

Service Name	Port/Protocol	Related Information
--------------	---------------	---------------------

ftp	21/tcp	IN-2001-01 , Widespread Compromises via "ramen" Toolkit IN-2000-10 , Widespread Exploitation of rcp.statd and wu-ftpd Vulnerabilities CA-2000-13 , Two Input Validation Problems In FTPD AA-2000-02 , wu-ftpd "site exec" Vulnerability CA-1999-13 , Multiple Vulnerabilities in WU-FTPD CA-1997-27 , FTP Bounce
ssh	22/tcp	CA-1999-15 , Buffer Overflows in SSH Daemon and RSAREF2 Library
telnet	23/tcp	IN-2000-09 , Systems Compromised Through a Vulnerability in the IRIX telnet daemon CA-2001-21 , Buffer Overflow in telnetd
domain	53/tcp 53/udp	CA-2001-02 , Multiple Vulnerabilities in BIND CA-2000-20 , Multiple Denial-of-Service Problems in ISC BIND IN-2000-04 , Denial of Service Attacks using Nameservers CA-2000-03 , Continuing Compromises of Nameservers CA-1999-14 , Multiple Vulnerabilities in BIND CA-1998-05 , Multiple Vulnerabilities in BIND
http	80/tcp	CA-2001-11 , sadmind/IIS Worm CA-2001-23 , Continued Threat of the "Code Red" Worm
"linuxconf" on some Linux distributions	98/tcp	Some Linux distributions ship with linuxconf, a program which listens on TCP port 98. While we do not have any reports of intruders actively exploiting vulnerabilities in linuxconf, these probes may be used to identify linux machines that have other vulnerabilities.
pop2	109/tcp	ipop2d buffer overflow
pop3	110/tcp	Qpopper buffer overflow CA-1997-09 , Vulnerability in IMAP and POP
sunrpc	111/tcp 111/udp	CA-2001-05 , Exploitation of snmpXdmid IN-2001-01 , Widespread Compromises via "ramen" Toolkit IN-2000-10 , Widespread Exploitation of rcp.statd and wu-ftpd Vulnerabilities CA-2000-17 , Input Validation Problem in rpc.statd CA-1999-16 , Buffer Overflow in Sun Solstice AdminSuite Daemon sadmind CA-1999-12 , Buffer overflow in amd CA-1999-08 , Buffer overflow in rpc.cmsd CA-1999-05 , Vulnerability in statd exposes vulnerability in automountd CA-1998-12 , Remotely Exploitable Buffer Overflow Vulnerability in mountd CA-1998-11 , Vulnerability in ToolTalk RPC service CA-2001-11 , sadmind/IIS Worm
netbios-ns netbios-dgm netbios-ssn	137/udp 138/udp 139/tcp	IN-2000-03 , 911 Worm IN-2000-02 , Exploitation of Unprotected Windows Networking Shares CA-2001-23 , Continued Threat of the "Code Red" Worm
imap	143/tcp	CA-1998-09 , Buffer Overflow in Some Implementations of IMAP Servers CA-1997-09 , Vulnerability in IMAP and POP
printer	515/tcp	IN-2001-01 , Widespread Compromises via "ramen" Toolkit Vulnerability Note VU#382365 , LPRng can pass user-supplied input as a format string parameter to syslog() calls

klogind	543/tcp	CA-2000-06 , Multiple Buffer Overflows in Kerberos Authenticated Services
socks	1080/tcp	VN-1998-03 , WinGate IP Laundering
SGL objectserver	5135/tcp	20000303-01-PX , Vulnerability in IRIX 5.3 and 6.2 objectserver
SubSeven	27374/tcp	IN-2001-07 , W32/Leaves: Exploitation of previously installed SubSeven Trojan Horses
ICMP echo ICMP echo reply	ICMP type 8 ICMP type 0	CA-1998-01 , "smurf" IP Denial-of-Service Attacks

For an overview of incident and vulnerability activity during the last quarter, see the most recent [CERT Summary](#).

Copyright 1999, 2000, 2001 Carnegie Mellon University.

[See the conditions for use, disclaimers, and copyright information.](#)

CERT® and CERT Coordination Center® are registered in the U.S. Patent and Trademark office.

Is Open Source Good for Security?

There's been a lot of debate by security practitioners about the impact of open source approaches on security. One of the key issues is that open source exposes the source code to examination by everyone, both the attackers and defenders, and reasonable people disagree about the ultimate impact of this situation.

Here are a few quotes from people who've examined the topic. Bruce Schneier argues that smart engineers should ``demand open source code for anything related to security" [Schneier 1999], and he also discusses some of the preconditions which must be met to make open source software secure. Vincent Rijmen, a developer of the winning Advanced Encryption Standard (AES) encryption algorithm, believes that the open source nature of Linux provides a superior vehicle to making security vulnerabilities easier to spot and fix, ``Not only because more people can look at it, but, more importantly, because the model forces people to write more clear code, and to adhere to standards. This in turn facilitates security review" [Rijmen 2000]. Elias Levy (Aleph1) discusses some of the problems in making open source software secure in his article ["Is Open Source Really More Secure than Closed?"](#). His summary is:

So does all this mean Open Source Software is no better than closed source software when it comes to security vulnerabilities? No. Open Source Software certainly does have the potential to be more secure than its closed source counterpart. But make no mistake, simply being open source is no guarantee of security.

John Viega's article ["The Myth of Open Source Security"](#) also discusses issues, and summarizes things this way:

Open source software projects can be more secure than closed source projects. However, the very things that can make open source programs secure -- the availability of the source code, and the fact that large numbers of users are available to look for and fix security holes -- can also lull people into a false sense of security.

[Michael H. Warfield's "Musings on open source security"](#) is much more positive about the impact of open source software on security. Fred Schneider doesn't believe that open source helps security, saying ``there is no reason to believe that the many eyes inspecting (open) source code would be successful in identifying bugs that allow system security to be compromised" and claiming that ``bugs in the code are not the dominant means of attack" [Schneider 2000]. He also claims that open source rules out control of the construction process, though in practice there is such control - all major open source programs have one or a few official versions with ``owners" with reputations at stake. Peter G. Neumann discusses ``open-box" software (in which source code is available, possibly only under certain conditions), saying ``Will open-box software really improve system security? My answer is not by itself, although the potential is considerable" [Neumann 2000]. [Natalie Walker Whitlock's IBM DeveloperWorks article](#) discusses the pros and cons as well.

Sometimes it's noted that a vulnerability that exists but is unknown can't be exploited, so the system ``practically secure." In theory this is true, but the problem is that once someone finds the vulnerability, the finder may just exploit the vulnerability instead of helping to fix it. Having unknown

vulnerabilities doesn't really make the vulnerabilities go away; it simply means that the vulnerabilities are a time bomb, with no way to know when they'll be exploited. Fundamentally, the problem of someone exploiting a vulnerability they discover is a problem for both open and closed source systems. It's been argued that a system without source code is more secure in this sense because, since there's less information available for an attacker, it would be harder for an attacker to find the vulnerabilities. A counter-argument is that attackers generally don't need source code, and if they want to use source code they can use disassemblers to re-create the source code of the product. See Flake [2001] for one discussion of how closed code can still be examined for security vulnerabilities (e.g., using disassemblers). In contrast, defenders won't usually look for problems if they don't have the source code, so not having the source code puts defenders at a disadvantage compared to attackers.

It's sometimes argued that open source programs, because there's no enforced control by a single company, permit people to insert Trojan Horses and other malicious code. Trojan horses can be inserted into open source code, true, but they can also be inserted into proprietary code. A disgruntled or bribed employee can insert malicious code, and in many organizations it's much less likely to be found than in an open source program. After all, no one outside the organization can review the source code, and few companies review their code internally (or, even if they do, few can be assured that the reviewed code is actually what is used). And the notion that a closed-source company can be sued later has little evidence; nearly all licenses disclaim all warranties, and courts have generally not held software development companies liable.

Borland's Interbase server is an interesting case in point. Some time between 1992 and 1994, Borland inserted an intentional ``back door" into their database server, ``Interbase". This back door allowed any local or remote user to manipulate any database object and install arbitrary programs, and in some cases could lead to controlling the machine as ``root". This vulnerability stayed in the product for at least 6 years - no one else could review the product, and Borland had no incentive to remove the vulnerability. Then Borland released its source code on July 2000. The "Firebird" project began working with the source code, and uncovered this serious security problem with InterBase in December 2000. By January 2001 the CERT announced the existence of this back door as [CERT advisory CA-2001-01](#). What's discouraging is that the backdoor can be easily found simply by looking at an ASCII dump of the program (a common cracker trick). Once this problem was found by open source developers reviewing the code, it was patched quickly. You could argue that, by keeping the password unknown, the program stayed safe, and that opening the source made the program less secure. I think this is nonsense, since ASCII dumps are trivial to do and well-known as a standard attack technique, and not all attackers have sudden urges to announce vulnerabilities - in fact, there's no way to be certain that this vulnerability has not been exploited many times. It's clear that after the source was opened, the source code was reviewed over time, and the vulnerabilities found and fixed. One way to characterize this is to say that the original code was vulnerable, its vulnerabilities became easier to exploit when it was first made open source, and then finally these vulnerabilities were fixed.

The advantages of having source code open extends not just to software that is being attacked, but also extends to vulnerability assessment scanners. Vulnerability assessment scanners intentionally look for vulnerabilities in configured systems. A recent Network Computing evaluation found that the best scanner (which, among other things, found the most legitimate vulnerabilities) was Nessus, an open source scanner [Forristal 2001].

So, what's the bottom line? I personally believe that when a program is first made open source, it often starts less secure for any users (through exposure of vulnerabilities), and over time (say a few years) it has the potential to be much more secure than a closed program. Just making a program open source doesn't suddenly make a program secure, and making an open source program secure is not

guaranteed:

- First, people have to actually review the code. This is one of the key points of debate - will people really review code in an open source project? All sorts of factors can reduce the amount of review: being a niche or rarely-used product (where there are few potential reviewers), having few developers, and use of a rarely-used computer language.

One factor that can particularly reduce review likelihood is not actually being open source. Some vendors like to posture their ``disclosed source" (also called ``source available") programs as being open source, but since the program owner has extensive exclusive rights, others will have far less incentive to work ``for free" for the owner on the code. Even open source licenses which have unusually asymmetric rights (such as the MPL) have this problem. After all, people are less likely to voluntarily participate if someone else will have rights to their results that they don't have (as Bruce Perens says, ``who wants to be someone else's unpaid employee?"). In particular, since the most incentivized reviewers tend to be people trying to modify the program, this disincentive to participate reduces the number of ``eyeballs". Elias Levy made this mistake in his article about open source security; his examples of software that had been broken into (e.g., TIS's Gauntlet) were not, at the time, open source.

- Second, the people developing and reviewing the code must know how to write secure programs. Hopefully the existence of this book will help. Clearly, it doesn't matter if there are ``many eyeballs" if none of the eyeballs know what to look for.
- Third, once found, these problems need to be fixed quickly and their fixes distributed. Open source systems tend to fix the problems quickly, but the distribution is not always smooth. For example, the OpenBSD developers do an excellent job of reviewing code for security flaws - but they don't always report the identified problems back to the original developer. Thus, it's quite possible for there to be a fixed version in one system, but for the flaw to remain in another.

Another advantage of open source is that, if you find a problem, you can fix it immediately.

In short, the effect on security of open source software is still a major debate in the security community, though a large number of prominent experts believe that it has great potential to be more secure.

[Prev](#)

Security Principles

[Home](#)

[Up](#)

[Next](#)

Types of Secure Programs

Security Principles

There are many general security principles which you should be familiar with; one good place for general information on information security is the Information Assurance Technical Framework (IATF) [NSA 2000]. NIST has identified high-level ``generally accepted principles and practices" [Swanson 1996]. You could also look at a general textbook on computer security, such as [Pfleeger 1997]. A few security principles are summarized here.

Often computer security goals are described in terms of three overall goals:

- *Confidentiality* (also known as secrecy), meaning that the computing system's assets are accessible only by authorized parties.
- *Integrity*, meaning that the assets can only be modified by authorized parties in authorized ways.
- *Availability*, meaning that the assets are accessible to the authorized parties in a timely manner (as determined by the systems requirements). The failure to meet this goal is called a denial of service.

Some people define additional security goals, while others lump those additional goals as special cases of these three goals. For example, some separately identify non-repudiation as a goal; this is the ability to ``prove" that a sender sent or receiver received a message, even if the sender or receiver wishes to deny it later. Privacy is sometimes addressed separately from confidentiality; some define this as protecting the confidentiality of a *user* (e.g., their identity) instead of the data. Most goals require identification and authentication, which is sometimes listed as a separate goal. Often auditing (also called accountability) is identified as a desirable security goal. Sometimes ``access control" and ``authenticity" are listed separately as well. In any case, it is important to identify your program's overall security goals, no matter how you group those goals together, so that you'll know when you've met them.

Sometimes these goals are a response to a known set of threats, and sometimes some of these goals are required by law. For example, for U.S. banks and other financial institutions, there's a new privacy law called the ``Gramm-Leach-Bliley" (GLB) Act. This law mandates disclosure of personal information shared and means of securing that data, requires disclosure of personal information that will be shared with third parties, and directs institutions to give customers a chance to opt out of data sharing. [Jones 2000]

There is sometimes conflict between security and some other general system/software engineering principles. Security can sometimes interfere with ``ease of use", for example, installing a secure configuration may take more effort than a ``trivial" installation that works but is insecure. Often, this apparant conflict can be resolved, for example, by re-thinking a problem it's often possible to make a secure system also easy to use. There's also sometimes a conflict between security and abstraction (information hiding); for example, some high-level library routines may be implemented securely or not, but their specifications won't tell you. In the end, if your application must be secure, you must do things yourself if you can't be sure otherwise - yes, the library should be fixed, but it's your users who will be hurt by your poor choice of library routines.

A good general security principle is ``defense in depth"; you should have numerous defense mechanisms (``layers") in place, designed so that an attacker has to defeat multiple mechanisms to perform a successful attack.

[Prev](#)

Background

[Home](#)

[Up](#)

[Next](#)

Is Open Source Good for
Security?

Types of Secure Programs

Many different types of programs may need to be secure programs (as the term is defined in this book). Some common types are:

- Application programs used as viewers of remote data. Programs used as viewers (such as word processors or file format viewers) are often asked to view data sent remotely by an untrusted user (this request may be automatically invoked by a web browser). Clearly, the untrusted user's input should not be allowed to cause the application to run arbitrary programs. It's usually unwise to support initialization macros (run when the data is displayed); if you must, then you must create a secure sandbox (a complex and error-prone task). Be careful of issues such as buffer overflow, discussed in [Chapter 5](#), which might allow an untrusted user to force the viewer to run an arbitrary program.
- Application programs used by the administrator (root). Such programs shouldn't trust information that can be controlled by non-administrators.
- Local servers (also called daemons).
- Network-accessible servers (sometimes called network daemons).
- Web-based applications (including CGI scripts). These are a special case of network-accessible servers, but they're so common they deserve their own category. Such programs are invoked indirectly via a web server, which filters out some attacks but nevertheless leaves many attacks that must be withstood.
- Applets (i.e., programs downloaded to the client for automatic execution). This is something Java is especially famous for, though other languages (such as Python) support mobile code as well. There are several security viewpoints here; the implementor of the applet infrastructure on the client side has to make sure that the only operations allowed are ``safe" ones, and the writer of an applet has to deal with the problem of hostile hosts (in other words, you can't normally trust the client). There is some research attempting to deal with running applets on hostile hosts, but frankly I'm sceptical of the value of these approaches and this subject is exotic enough that I don't cover it further here.
- setuid/setgid programs. These programs are invoked by a local user and, when executed, are immediately granted the privileges of the program's owner and/or owner's group. In many ways these are the hardest programs to secure, because so many of their inputs are under the control of the untrusted user and some of those inputs are not obvious.

This book merges the issues of these different types of program into a single set. The disadvantage of this approach is that some of the issues identified here don't apply to all types of programs. In particular, setuid/setgid programs have many surprising inputs and several of the guidelines here only apply to them. However, things are not so clear-cut, because a particular program may cut across these boundaries (e.g., a CGI script may be setuid or setgid, or be configured in a way that has the same effect), and some programs are divided into several executables each of which can be considered

a different ``type" of program. The advantage of considering all of these program types together is that we can consider all issues without trying to apply an inappropriate category to a program. As will be seen, many of the principles apply to all programs that need to be secured.

There is a slight bias in this book towards programs written in C, with some notes on other languages such as C++, Perl, Python, Ada95, and Java. This is because C is the most common language for implementing secure programs on Unix-like systems (other than CGI scripts, which tend to use Perl), and most other languages' implementations call the C library. This is not to imply that C is somehow the ``best" language for this purpose, and most of the principles described here apply regardless of the programming language used.

[Prev](#)

Is Open Source Good for
Security?

[Home](#)
[Up](#)

[Next](#)

Paranoia is a Virtue

Secure Programming for Linux and Unix HOWTO

David A. Wheeler

[Copyright](#) © 1999, 2000, 2001 by David A. Wheeler

This book provides a set of design and implementation guidelines for writing secure programs for Linux and Unix systems. Such programs include application programs used as viewers of remote data, web applications (including CGI scripts), network servers, and setuid/setgid programs. Specific guidelines for C, C++, Java, Perl, Python, TCL, and Ada95 are included.

Table of Contents

1. [Introduction](#)

2. [Background](#)

[History of Unix, Linux, and Open Source / Free Software](#)

[Unix](#)

[Free Software Foundation](#)

[Linux](#)

[Open Source / Free Software](#)

[Comparing Linux and Unix](#)

[Security Principles](#)

[Is Open Source Good for Security?](#)

[Types of Secure Programs](#)

[Paranoia is a Virtue](#)

[Why Did I Write This Document?](#)

[Sources of Design and Implementation Guidelines](#)

[Other Sources of Security Information](#)

[Document Conventions](#)

3. [Summary of Linux and Unix Security Features](#)

[Processes](#)

[Process Attributes](#)

[POSIX Capabilities](#)

[Process Creation and Manipulation](#)

[Files](#)

[Filesystem Object Attributes](#)

[Creation Time Initial Values](#)

[Changing Access Control Attributes](#)

[Using Access Control Attributes](#)

[Filesystem Hierarchy](#)

[System V IPC](#)

[Sockets and Network Connections](#)

[Signals](#)

[Quotas and Limits](#)

[Dynamically Linked Libraries](#)

[Audit](#)

[PAM](#)

[Specialized Security Extensions for Unix-like Systems](#)

4. [Validate All Input](#)

[Command line](#)

[Environment Variables](#)

[Some Environment Variables are Dangerous](#)

[Environment Variable Storage Format is Dangerous](#)

[The Solution - Extract and Erase](#)

[File Descriptors](#)

[File Contents](#)

[Web-Based Application Inputs \(Especially CGI Scripts\)](#)

[Other Inputs](#)

[Human Language \(Locale\) Selection](#)

[How Locales are Selected](#)

[Locale Support Mechanisms](#)

[Legal Values](#)

[Bottom Line](#)

[Character Encoding](#)

[Introduction to Character Encoding](#)

[Introduction to UTF-8](#)

[UTF-8 Security Issues](#)

[UTF-8 Legal Values](#)

[UTF-8 Illegal Values](#)

[UTF-8 Related Issues](#)

[Prevent Cross-site Malicious Content on Input](#)

[Filter HTML/URIs That May Be Re-presented](#)

[Remove or Forbid Some HTML Data](#)

[Encoding HTML Data](#)

[Validating HTML Data](#)

[Validating Hypertext Links \(URIs/URLs\)](#)

[Other HTML tags](#)

[Related Issues](#)

[Forbid HTTP GET To Perform Non-Queries](#)

[Limit Valid Input Time and Load Level](#)

5. [Avoid Buffer Overflow](#)

[Dangers in C/C++](#)

[Library Solutions in C/C++](#)

[Standard C Library Solution](#)

[Static and Dynamically Allocated Buffers](#)

[strcpy and strcat](#)

[libmib](#)

[Libsafe](#)

[Other Libraries](#)

[Compilation Solutions in C/C++](#)

[Other Languages](#)

6. [Structure Program Internals and Approach](#)

[Follow Good Software Engineering Principles for Secure Programs](#)

[Secure the Interface](#)

[Minimize Privileges](#)

[Minimize the Privileges Granted](#)

[Minimize the Time the Privilege Can Be Used](#)

[Minimize the Time the Privilege is Active](#)

[Minimize the Modules Granted the Privilege](#)

[Consider Using FSUID To Limit Privileges](#)

[Consider Using Chroot to Minimize Available Files](#)

[Consider Minimizing the Accessible Data](#)

[Consider Minimizing the Resources Available](#)

[Avoid Creating Setuid/Setgid Scripts](#)

[Configure Safely and Use Safe Defaults](#)

[Load Initialization Values Safely](#)

[Fail Safe](#)

[Avoid Race Conditions](#)

[Sequencing \(Non-Atomic\) Problems](#)

[Locking](#)

[Trust Only Trustworthy Channels](#)

[Set up a Trusted Path](#)

[Use Internal Consistency-Checking Code](#)

[Self-limit Resources](#)

[Prevent Cross-Site Malicious Content](#)

[Explanation of the Problem](#)

[Solutions to Cross-Site Malicious Content](#)

[Be Careful with Data Types](#)

7. [Carefully Call Out to Other Resources](#)

[Call Only Safe Library Routines](#)

[Limit Call-outs to Valid Values](#)

[Call Only Interfaces Intended for Programmers](#)

[Check All System Call Returns](#)

[Avoid Using vfork\(2\)](#)

[Counter Web Bugs When Retrieving Embedded Content](#)

[Hide Sensitive Information](#)

8. [Send Information Back Judiciously](#)

[Minimize Feedback](#)

[Don't Include Comments](#)

[Handle Full/Unresponsive Output](#)

[Control Data Formatting \("Format Strings"\)](#)

[Control Character Encoding in Output](#)

[Prevent Include/Configuration File Access](#)

9. [Language-Specific Issues](#)

[C/C++](#)

[Perl](#)

[Python](#)

[Shell Scripting Languages \(sh and csh Derivatives\)](#)

[Ada](#)

[Java](#)

[TCL](#)

10. [Special Topics](#)

[Passwords](#)

[Random Numbers](#)

[Specially Protect Secrets \(Passwords and Keys\) in User Memory](#)

[Cryptographic Algorithms and Protocols](#)

[Using PAM](#)

[Tools](#)

[Windows CE](#)

[Write Audit Records](#)

[Miscellaneous](#)

11. [Conclusion](#)

12. [Bibliography](#)

A. [History](#)

B. [Acknowledgements](#)

C. [About the Documentation License](#)

D. [GNU Free Documentation License](#)

E. [Endorsements](#)

F. [About the Author](#)

List of Tables

4-1. [Legal UTF-8 Sequences](#)

4-2. [Illegal UTF-8 initial sequences](#)

List of Figures

1-1. [Abstract View of a Program](#)

[Next](#)
Introduction

Chapter 2. Background

I issued an order and a search was made, and it was found that this city has a long history of revolt against kings and has been a place of rebellion and sedition.

Ezra 4:19 (NIV)

Table of Contents

[History of Unix, Linux, and Open Source / Free Software](#)

[Security Principles](#)

[Is Open Source Good for Security?](#)

[Types of Secure Programs](#)

[Paranoia is a Virtue](#)

[Why Did I Write This Document?](#)

[Sources of Design and Implementation Guidelines](#)

[Other Sources of Security Information](#)

[Document Conventions](#)

History of Unix, Linux, and Open Source / Free Software

Unix

In 1969-1970, Kenneth Thompson, Dennis Ritchie, and others at AT&T Bell Labs began developing a small operating system on a little-used PDP-7. The operating system was soon christened Unix, a pun on an earlier operating system project called MULTICS. In 1972-1973 the system was rewritten in the programming language C, an unusual step that was visionary: due to this decision, Unix was the first widely-used operating system that could switch from and outlive its original hardware. Other innovations were added to Unix as well, in part due to synergies between Bell Labs and the academic community. In 1979, the ``seventh edition" (V7) version of Unix was released, the grandfather of all extant Unix systems.

After this point, the history of Unix becomes somewhat convoluted. The academic community, led by Berkeley, developed a variant called the Berkeley Software Distribution (BSD), while AT&T continued developing Unix under the names ``System III" and later ``System V". In the late 1980's through early 1990's the ``wars" between these two major strains raged. After many years each variant adopted many of the key features of the other. Commercially, System V won the ``standards wars"

(getting most of its interfaces into the formal standards), and most hardware vendors switched to AT&T's System V. However, System V ended up incorporating many BSD innovations, so the resulting system was more a merger of the two branches. The BSD branch did not die, but instead became widely used for research, for PC hardware, and for single-purpose servers (e.g., many web sites use a BSD derivative).

The result was many different versions of Unix, all based on the original seventh edition. Most versions of Unix were proprietary and maintained by their respective hardware vendor, for example, Sun Solaris is a variant of System V. Three versions of the BSD branch of Unix ended up as open source: FreeBSD (concentrating on ease-of-installation for PC-type hardware), NetBSD (concentrating on many different CPU architectures), and a variant of NetBSD, OpenBSD (concentrating on security). More general information about Unix history can be found at <http://www.datametrics.com/tech/unix/uxhistry/brf-hist.htm> and <http://perso.wanadoo.fr/levenez/unix>. Much more information about the BSD history can be found in [McKusick 1999] and <ftp://ftp.freebsd.org/pub/FreeBSD/FreeBSD-current/src/share/misc/bsd-family-tree>.

Those interested in reading an advocacy piece that presents arguments for using Unix-like systems should see <http://www.unix-vs-nt.org>.

Free Software Foundation

In 1984 Richard Stallman's Free Software Foundation (FSF) began the GNU project, a project to create a free version of the Unix operating system. By free, Stallman meant software that could be freely used, read, modified, and redistributed. The FSF successfully built a vast number of useful components, including a C compiler (gcc), an impressive text editor (emacs), and a host of fundamental tools. However, in the 1990's the FSF was having trouble developing the operating system kernel [FSF 1998]; without a kernel the rest of their software would not work.

Linux

In 1991 Linus Torvalds began developing an operating system kernel, which he named ``Linux" [Torvalds 1999]. This kernel could be combined with the FSF material and other components (in particular some of the BSD components and MIT's X-windows software) to produce a freely-modifiable and very useful operating system. This book will term the kernel itself the ``Linux kernel" and an entire combination as ``Linux". Note that many use the term ``GNU/Linux" instead for this combination.

In the Linux community, different organizations have combined the available components differently. Each combination is called a ``distribution", and the organizations that develop distributions are called ``distributors". Common distributions include Red Hat, Mandrake, SuSE, Caldera, Corel, and Debian. There are differences between the various distributions, but all distributions are based on the same foundation: the Linux kernel and the GNU glibc libraries. Since both are covered by ``copyleft" style licenses, changes to these foundations generally must be made available to all, a unifying force between the Linux distributions at their foundation that does not exist between the BSD and AT&T-derived Unix systems. This book is not specific to any Linux distribution; when it discusses Linux it presumes Linux kernel version 2.2 or greater and the C library glibc 2.1 or greater, valid assumptions for essentially all current major Linux distributions.

Open Source / Free Software

Increased interest in software that is freely shared has made it increasingly necessary to define and explain it. A widely used term is "open source software", which is further defined in [OSI 1999]. Eric Raymond [1997, 1998] wrote several seminal articles examining its various development processes. Another widely-used term is "free software", where the "free" is short for "freedom": the usual explanation is "free speech, not free beer." Neither phrase is perfect. The term "free software" is often confused with programs whose executables are given away at no charge, but whose source code cannot be viewed, modified, or redistributed. Conversely, the term "open source" is sometime (ab)used to mean software whose source code is visible, but for which there are limitations on use, modification, or redistribution. This book uses the term "open source" for its usual meaning, that is, software which has its source code freely available for use, viewing, modification, and redistribution; a more detailed definition is contained in the [Open Source Definition](#). In some cases, a difference in motive is suggested; those preferring the term "free software" wish to strongly emphasize the need for freedom, while those using the term may have other motives (e.g., higher reliability) or simply wish to appear less strident. For information on this definition of free software, and the motivations behind it, can be found at <http://www.fsf.org>.

Those interested in reading advocacy pieces for open source software and free software should see <http://www.opensource.org> and <http://www.fsf.org>. There are other documents which examine such software, for example, Miller [1995] found that the open source software were noticeably more reliable than proprietary software (using their measurement technique, which measured resistance to crashing due to random input).

Comparing Linux and Unix

This book uses the term "Unix-like" to describe systems intentionally like Unix. In particular, the term "Unix-like" includes all major Unix variants and Linux distributions. Note that many people simply use the term "Unix" to describe these systems instead.

Linux is not derived from Unix source code, but its interfaces are intentionally like Unix. Therefore, Unix lessons learned generally apply to both, including information on security. Most of the information in this book applies to any Unix-like system. Linux-specific information has been intentionally added to enable those using Linux to take advantage of Linux's capabilities.

Unix-like systems share a number of security mechanisms, though there are subtle differences and not all systems have all mechanisms available. All include user and group ids (uids and gids) for each process and a filesystem with read, write, and execute permissions (for user, group, and other). See Thompson [1974] and Bach [1986] for general information on Unix systems, including their basic security mechanisms. [Chapter 3](#) summarizes key security features of Unix and Linux.

[Prev](#)[Home](#)[Next](#)

Introduction

Security Principles

Flawfinder

This is the main web site for *flawfinder*, a program that examines source code looking for security weaknesses ("flaws"). Unlike ITS4, flawfinder is completely open source (as [defined by the Open Source Definition](#)) and it's fully free software (as [defined by the Free Software Foundation's GNU project](#)). Flawfinder is released under the General Public License (GPL).

Flawfinder works on Unix-like systems today (it's been tested on GNU/Linux), and it should be easy to port to Windows systems. It requires Python to run.

Downloading

Just select this to get flawfinder:

- [Download the current version of flawfinder \("tarball" format\)](#).

The current version of flawfinder is 0.15. Flawfinder is reliable; I've assigned it a low version number because its vulnerability database is small and needs to grow. If you want to see how it's changed, view its [ChangeLog](#).

If you're not sure you want to take the plunge to install the program, you can just look at the documentation in [PDF](#) or [Postscript](#) format. You can even go look at the [flawfinder source code](#).

Installation

On Unix-like systems, you can uncompress and install it in the usual manner:

```
gunzip  flawfinder-*.tar.gz
tar xvf flawfinder-*.tar
cd flawfinder-*
su
make install
```

Simple end-user installation processes, etc., are to come.

Speed

flawfinder is written in Python, to simplify the task of writing and extending it. Python code is not as fast as C code, but for the task I believe it's just fine. Flawfinder version 0.12 on a 400Mhz Pentium II system analyzed 51055 lines in 39.7 seconds, resulting in an average of 1285 analyzed lines/second.

RATS

Unbenowst to me, while I was developing flawfinder, Secure Software Solutions simultaneously developed [RATS](#), which is also a source code scanner. We agreed to release our programs simultaneously (on May 21, 2001), and we agreed to mention each other's programs in our announcements (you can even see the original [flawfinder announcement](#)). Now that we've both released our code, we plan to coordinate so that there will be a single ``best of breed" source code scanner that is open source / free software. Exactly how this will happen is not yet clear, so be prepared for future announcements.

You might want to look at my [Secure Programming HOWTO web page](#).

You can also view [my home page](#).

"Browse" Home Page

"browse" is a small program that will display an arbitrary reference (URL or file) on a Unix-like system, trying a number of different browsers (controllable by the user) until it finds one that works.

It implements the [Secure BROWSER convention](#), a modification of Eric Raymond's [BROWSER convention](#) that's been modified so it's not a security hazard. Just set the BROWSER environment variable to select your browser(s). Indeed, "browse" can also be used as a reference implementation if you want to implement the convention yourself.

Using browse is easy, just give it "--" and a URL. Here's an example:

```
$ browse -- 'http://www.dwheeler.com'
```

The Secure BROWSER convention is secure, while the original BROWSER convention isn't. In the original BROWSER convention, a URL of ";rm -fr /;" would increase your free disk space. The secure BROWSER convention is still simple to implement, too. The secure BROWSER convention is still being discussed by security folks; very soon I hope to have the "final" version. see the specification for details.

Here are some relevant pages:

- [Secure BROWSER specification](#)
- browse man page (in [man](#), [Postscript](#), and [PDF](#) formats)
- [Current version of browse program](#)
- [Test script, which tests browse](#)
- [Security analysis of original BROWSER specification](#)
- [Helper program that invokes Netscape](#), useful as an entry in BROWSER. this version abandons Netscape more quickly if there's no graphical display, and it correctly handles the special characters ",)#%".
- [The license for the browse code](#) (it's the MIT/X license)

If you find a security hole, please contact David A. Wheeler at dwheeler@dwheeler.com (no spam please). Note that originally, "browse" was named "show_url".

You can [go up to see David A. Wheeler's site](#).

David A. Wheeler's Java Security Tutorial

David A. Wheeler's Java Security Tutorial (a briefing) is available here. Giving the entire tutorial takes about 2.5 hours. The tutorial is not public domain, but it is open source and licensed under the GPL.

The tutorial is available in [PowerPoint \(ppt\)](#) and [Portable Document Format \(PDF\)](#) formats.

You can contact David A. Wheeler at dwheeler@dwheeler.com.

You can see the rest of my website at <http://www.dwheeler.com>

Java Implementations

by David K. Friedman and David A. Wheeler

February 5, 2001

Abstract

This article briefly introduces Java, provides a bit of Java history, and describes various Java implementations.

Introduction: What's Java?

[Sun Microsystems](#) has developed a set of technologies called [Java](#) (TM). Using Java you can create various kinds of programs including traditional applications and "applets" that automatically run when a user views a web page. Many users are very attracted to Java's goal of "write once, run anywhere" (WORA), that is, using this technology the same program can be highly portable and run on many different platforms (e.g. Windows, Mac, Unix, and Linux). The Java technologies can be divided into four components:

1. *The Java Virtual Machine (JVM) which runs class files.* The JVM is an abstract computer that executes programs stored in "class" files. The JVM can be implemented on real computers in many different ways, and that's the point: as long as your computer faithfully recreates this abstract computer, it can run programs stored in class files. For example, the JVM might be implemented as an interpreter built into a web browser, or as a separate program that interprets the class files. Your computer could implement the JVM by transforming the class files into an executable program specific for that machine just before running them (this is called a "just-in-time" compiler). In fact, your computer hardware might implement the JVM directly. As long as you have an implementation of the JVM, you can run Java programs, compiled into class files. Class files are also called [bytecode](#) files.
2. *The Java language.* The Java language is an object oriented computer programming language that resembles C++ and Objective-C in syntax, with its semantics being more like a cross between Ada (for typesafety) and Smalltalk. It supports automatic garbage collection and single inheritance (with multiple inheritance of interfaces) but lacks enumerations and templates.
3. *A compiler that generates class files.* The JVM runs class files, so you need a way to create them. Sun has developed a compiler that takes programs written in the Java language and generates Java class files. Other vendors have also developed compilers that generate class files. Note that you don't need to use the Java language to generate Java class files.
4. *The Java library.* The Java technology includes a set of components for a platform-independent graphical user interface (GUI) as well as other useful components.

Sometimes the terminology can rather confusing. Many people, including Sun, use the term "Java" for each of these different components and for the technology as a whole. You'll need to determine what they mean by its context. The key point is that when people "run a Java program", they're actually running a set of class files on their version of the JVM.

Note: the term "Java" is trademarked by Sun. For purposes of this page, "Java" will also include anything implementing (or trying to implement) Sun's specifications of Java, but this is not intended to infringe on the status of Sun's trademark.

Java History

Java is an outgrowth of the Green Project which was started in 1990 by Patrick Naughton, Mike Sheridan, and James Gosling. The idea was simply to plan and anticipate the "next wave" in computing. Initially, it was called Oak after a tree outside of James Gosling's office. However, that name was in use by another product from a different company so the name Java was used. The technology quickly caught on and Java was christened on May 23 of 1993 when Netscape Navigator agreed to make Java a part of its browser software. Today, Java is bigger than ever. It is integral part of the leading web browsers (Internet Explorer and Netscape Navigator) and can run applications on nearly every modern operating system.

However, one thing that we would like to make clear, is that Java is not something which is terribly new. Contrary to what Sun says, virtual machines existed in the 1980s in UCSD p-code, and researchers have been discussing mobile code for years. Still, Java productized these well.

Why *not* Sun?

Of course, you can always download the current Java Development Kit (JDK) from Sun's [Javasoft](#) and if you are just learning Java that may be the simplest thing to do. However, there may be several reasons why you would want software from someone else:

1. **You want to use [open source](#) / [free software](#) technology:** When you download from Sun you must agree to a [license](#) which states that you are not allowed to modify or redistribute the software except under special conditions. If you are a developer, you may want something that you can modify, port, and so on. You may also not want some other organization to have that much control over your infrastructure. In our list, we have specified which packages are open source, including those that are [GPL](#) (GNU Public License) or [LGPL](#) (Lesser GNU Public License). For more information, you can see the [Open Source Initiative web site](#), the [Free Software Foundation](#), [Open Source Software / Free Software References](#), and [Quantitative Measures](#).
2. **You would prefer to write in another language:** As stated before you do not need to write Java source code to be able to create bytecode. We list here several compilers which take source code from a different language and compile it to bytecode.
3. **You would like increased performance:** One of the key problems with the Java virtual machine is that performance (i.e., speed) may suffer. For many applications, perhaps performance is not important, but for other applications it's quite important. Sun's original Java implementation performed security checks on code every time it was loaded, and then it interpreted byte codes one-by-one. More recently, some implementations include a ``just-in-time" compiler which quickly compiles at least some of the code before it's executing (various heuristics may be used to try to determine what should be compiled). Some Java implementations take a more traditional (``ahead-of-time") approach: they take the Java source code or byte code and compile it to machine-specific native code. This code will work faster (especially since the compiler can spend more time optimizing its results); note that the native code will only work on that kind of machine. Some systems support a hybrid approach: you can compile most code ahead-of-time, but dynamically load code and have them call each other.

Note that some required portions of the Java language and byte code (e.g., garbage collection) may make it very difficult for a Java program to be equal in performance to a well-tuned C/C++ program; the extent of this difference is often debated.

4. **Sun doesn't have a Java Development Kit (JDK) for your platform** Clearly you can't just download from Sun if you are in this boat. Sun's Java implementation is available on [Windows 95/98/NT](#), [Solaris](#), [Linux](#), and several other platforms. However, your platform may not be supported - embedded developers often fit in this category.

Table

Here is our table of current Java compilers, virtual machines, interpreters, translators and other assorted Java related tools. We have tried to include only those product that we think are current and may be useful to you. Others who appear to be currently inactive are listed below in our [Apparently Dead Projects](#) section. Hopefully, this list will be continually updated. [Let us know](#) if you think a new product should be added.

You should also take a look at [Marco Schmidt's List of Java compilers and virtual machines](#), which also tries to capture this sort of information.

Product	Company/Organization	Cost/License	Components	Description
AppletMagic (more info)	AverStar	\$/Proprietary	Ada95 Compiler	Compiles Ada95 to bytecode
BulletTrain (more info)	NaturalBridge	\$/Proprietary	Bytecode compiler, library	Java bytecode compiler to native code, library, and runtime for Windows
Dinkum® Jcore Library (more info)	Dinkumware, Ltd	\$/Proprietary	Library	Reimplemented Java library; used by JFE .
GCJ (GNU compiler for Java) (more info)	Red Hat	Free/ LGPL *	Compiler	Compiles Java code to native code, Java code to bytecode, or bytecode to machine code
GJ (Generic Java) (more info)	Pizza Group	Free/Unknown	Compiles Extended Java to bytecode	Adds support for generics to Java language
GNU Classpath (more info)	GNU /Open Source	Free/ LGPL *	Library	Library of essential Java packages.

GNAT (more info)	Ada Core Technologies	Free/ GPL * (except for the runtime code and library which is like LGPL)	Ada95 Compiler	GNAT is able to compile Ada 95 code to bytecode.
Jacks (more info)	IBM Research	Free/GPL*	Test Suite	Regression testing suite for Java source code compilers
Japhar (more info)	Hungry Programmers	Free/ LGPL *	Virtual Machine	Open source Java VM.
Jikes (more info)	IBM Research	Free/ IBM Public License *	Compiler	Compiles Java source code to bytecode
JDK (Java Development Kit) (more info)	SMI Sun Microsystems Incorporated	Free/Proprietary	Complete Implementation. Compiler , Library , Virtual Machine	The Java development kit.
JFE (more info)	Edison Design Group (EDG)	\$/Proprietary	Compiler/analysis front end	Generates intermediate form and C; uses Dinkum Jcore
JOVE (more info)	Instantiations	\$/Proprietary	Compiler	Compiles bytecode to native code.
JPython (more info)	Python Software Activity	Free/ Open Source * (mostly, see below for details)	Python Compiler	Compiles Python to bytecode.
Kada (more info)	Kada Systems (formerly Emwerks)	\$/Proprietary	Compiler , code minimizer, Library , Virtual Machine	Java implementation for mobile/handheld devices
Kaffe (more info)	TransVirtual/Kaffe.org	Free/ GPL * --or-- Custom Edition for commercial developers	Compiler , Library , Virtual Machine	A complete Java implementation which supports both Sun's Java and Microsoft's.
Kawa (more info)	Per Bothner	Free/ GPL * (if you don't modify then you can use it in proprietary projects ; see license or download for more details)	Scheme Compiler	Compiles Scheme to Java bytecode.

LaTTe (more info)	Seoul National University IBM, T.J Watson Research Center	Free/ BSD license * (with slight modification from the original)	Virtual Machine with JIT compiler, and incomplete Library .	Virtual Machine which has its own JIT compiler.
Mauve (more info)	Red Hat	Free/ GPL *	Java Test Suite	A free Java Test Suite.
Microsoft SDK (Software Development Kit) for Java (more info)	Microsoft	Free/Proprietary	Compiler , Library	The Microsoft SDK contains tools, samples, documentation and the Microsoft Java Compiler.
NetRexx (more info)	IBM	Free/Closed Source	RexxLibrary , RexxCompiler	Compiles a dialect of Rexx to Java bytecode.
PERC (more info)	NewMonics	\$/Proprietary	Java Virtual Machine, ahead-of-time compiler, class libraries.	Java virtual machine especially for development of network element management and control applications.
Pizza (more info)	University of South Australia	Free/ Public Domain *	Extended Java Compiler	Compiles Java code. Makes several extensions including generic types.
SableVM (more info)	Sable Research Group McGill University	Free/ GPL *	Virtual Machine , Library	Java Virtual Machine. Comes with SablePath , a library derived from GNU Classpath .
TurboJ (more info)	Groupe Silicomp	\$/Proprietary	Compiler	Compiles Java bytecode to native machine code; sold to OEMs.
TowerJ (more info)	TowerJ	\$/Proprietary	Compiler	Compiles Java bytecode to native machine code.

*Open Source Software (OSS)

One tool which we have decided to not to include are Compiler Compilers. We feel they are too specialized a thing which only a few people in reality are generally interested in. However, we still list them here in case that may be what you are looking for: [SableCC](#), [JavaCC](#), and [ANTLR](#).

We've not included technologies that are similar to Java but don't appear to be Java. For example, JayaCard's documentation (<http://perso.wanadoo.fr/dgil/jaya/index.htm> or <http://www.jayacard.com>) doesn't clearly say that it implements Java (though its ideas appear similar). That doesn't make the technology less valuable, but we have to draw the line somewhere.

Further Descriptions

[AppletMagic](#), [AverStar](#) -- A commercial Ada 95 compiler from AverStar. Compiles Ada 95 to bytecode. Ada developers should also look at [GNAT](#) from [Ada Core Technologies](#).

[BulletTrain](#), [NaturalBridge](#) -- BulletTrain is a system for statically compiling and linking JVM bytecode applications. It includes an optimizing ahead-of-time compiler, a linker and recompilation manager, a runtime, and a core set of optimized libraries compatible with Sun(TM)'s Java 2 release. It is specifically designed for Windows platforms (2000, Windows NT, Windows 98, and Windows 95). It supports over 32,000 threads.

[Dinkum® Jcore Library](#), [Dinkumware](#) -- The Dinkum Jcore Library implements essential Java classes. It can be used as the runtime library for [JFE](#) (the Java cross-compiler front end from Edison Design Group), or as a replacement core library for use with a Java Virtual Machine (JVM).

[GCJ](#), [Red Hat](#) -- Version 2.95.1. GNU Compiler for Java (GCJ) compiles Java code to native code, Java code to bytecode, or bytecode to machine code. Supports everything within the [Java language Specification v1.0](#). The library GCJ uses is called libgcj, so far no support for [RMI](#), [AWT](#), or related [GUI](#) libraries, and apparently it has little to no security features.

[GJ \(Generic Java\)](#), [Pizza Group](#) -- Compiles an extended form of Java into standard Java bytecode. This extension supports generics. This is follow-on work from the work on [Pizza](#). The executable (class files) can be freely downloaded, but the downloaded files I obtained do not contain the source code for the compiler itself. A website problem prevented accesses to the licensing information.

[GNU Classpath](#) -- Intended as an alternative to the proprietary library offered by Sun. No public releases yet, but the interim version can be downloaded. The web site was last updated in late May 2000. The "target" compiler for this library will be [Japhar](#) which is also licensed under the [LGPL](#). The people behind [SableVM](#) adapted [GNU Classpath](#) for their use and called the result [SablePath](#). You can check on the current [status](#).

[GNAT](#), [Ada Core Technologies](#) -- Ada 95 compiler. Originally started at [NYU](#) this software is now in use by major companies including Boeing, Nortel, Lockheed and [others](#). With regards to the license, the runtime environment and library are licensed under a modified GPL permitting their use in proprietary programs. However, the compiler is covered under the standard GPL, so the compiler itself can be modified but may not be made proprietary. You may purchase technical support. See

[GNATInfo](#) for more information regarding the license.

[Jacks](#), [IBM Research](#) -- This is a regression testing suite for Java source code compilers, to see if the compiler meets the requirements of the Java Language Specification Version 2. Jacks is licensed under the GPL (note that it can be used to test non-GPL compilers). Jacks is implemented in Tcl, and works on both Unix-like and Windows systems. [Maya Stodte's developerWorks article](#) discusses Jacks. Note that Jacks does not test a Java runtime (JVM) or Java class library (other test suites, such as [Mauve](#), exist to do that).

[Japhar](#), [Hungry Programmers](#) -- Java virtual machine currently in the development stage, version 0.08. It is not clear if work on the project is still active. The web site was last modified in March 2000. Japhar is open source and is released under the [GPL](#). Japhar was checked against Mauve and you can view the [results](#).

[JDK \(Java Development Kit\)](#), [SMI Sun Microsystems Incorporated\(JavaSoft Division\)](#) -- Complete Java implementation. Probably the best for a beginner or someone who is just learning, and in our estimation the one currently used by the majority of all users. You will get a complete implementation, including a compiler, virtual machine, and library. Many of the binaries are available at no charge but you are not allowed to make copies or redistribute them. They do offer some of the source code for reference purposes only. You might think twice about looking at the source code though if you want to be considered a "clean" developer of another implementation, i.e., someone who hasn't seen Sun's implementation and can create their own without being accused of copyright infringement.

[Jikes](#), [IBM Research](#) -- Compiles Java source code to bytecode. IBM claims it has an extremely fast compile speed, and that it strictly adheres to the [Java Language Specification](#). Last updated November 3, 1999.

[JFE](#), [Edison Design Group \(EDG\)](#)-- Compiles Java source code into an intermediate format suitable for analysis, and can generate C code from the intermediate form (which can then be compiled into an executable that doesn't require a JVM). See [Dinkum](#).

[JOVE](#), [Instantiations](#) -- Compiles Java to native code. The pricing information for this software can be found [here](#). You might want this software if you feel your Java applications need to run faster, and that you can substantially improve performance by compiling to native code. It's compatible with JDK 1.3. Note that [GCJ](#), performs the same function; we don't have data comparing their relative strengths.

[JPython](#), [Python Software Activity](#) -- Compiles Python to bytecode. With regards to the license, as of now JPython is released under two different licenses:

1. The first one claims to be compliant with the [Open Source Definition](#) but has not yet sought certification from the [Open Source Organization](#), (according to their web site see [license](#)) But, the general Python license has been [certified](#). This is the license associated with the [standard python software](#). You can see the source and modify it if you wish. This first version does *not* contain the regular expression checker [OROMatcher](#).
2. The second version does contain [OROMatcher](#). Now the version of [OROMatcher](#) which comes with JPython is available at no charge but is not open source. Hence the separate license. However, [OROMatcher](#) is expected to be open source in the next version.

[Kada](#), [Kada Systems](#) -- Full-function Java implementations for handheld/mobile devices. The Kada toolset includes compilers, a code minimizer (to reduce application size), and a source-level debugger. Their Java-compatible virtual machine emphasizes full functionality, speed, and small size. [Kada Systems](#) was formerly named [Emwerks](#).

[Kaffe](#), [TransVirtual/Kaffe.org](#) -- Complete implementation of Java. Kaffe is useful if you want to support both Microsoft's Java implementation and Sun's. It is designed to the [Personal Java 1.1.1 specification](#) so it is more oriented toward embedded systems (VCRs, Phones, Cars, Microwaves etc.). However, you can still use it to write regular applications and applets.

TransVirtual claims its Virtual Machine is several times faster than Sun's and that it is only 30% slower than vanilla C. They offer AWT and Swing support. There is corporate interest in Kaffe: [Compaq](#), [Schlumberger](#) and [Whistle](#) have used it for various applications. However, there are also a number of drawbacks; Kaffe does not have many security features that Sun's implementation does, including a bytecode verifier and honoring public/private access modifiers. It does not support the Java 2 security model including stack inspection and code signing.

Kaffe was written by [Tim Wilkinson](#) and [Peter Mehltz](#). Be aware that there are two distinct versions of Kaffe. The desktop edition is licensed under the GPL, so you can't use it in a proprietary project. However, TransVirtual also offers a custom edition which can be licensed to you for use in a proprietary project. This is mirrored by the two different web sites. From the company website, ([TransVirtual](#)) you may either obtain the Desktop edition or the Custom edition. From the [Kaffe.org](#) site you may only obtain the Open Desktop edition. In general, the .org site is more oriented toward other developers, and interested parties who might feel like contributing to the project, and the .com site is more oriented towards companies who might use Kaffe for their projects. Kaffe.org has made it clear that your contributions to the GPL version will not be used in the custom edition without your permission.

[Kawa](#), [Per Bothner](#) -- Compiles Scheme to Java bytecode. Scheme is a programming language similar to LISP and Algol and is often used in introductory Computer Science classes. With regards to the license it is [GPLed](#) but you can use it in a proprietary program as long as you don't modify it. See [license](#). Still active. It was last updated June 26, 2000.

[LaTTe](#), [Seoul National University](#) -- Virtual machine with its own JIT compiler. Developed at Seoul National University under the sponsorship of the [IBM, T.J Watson Research Center](#), this software is still in the research stage. The goal is to improve the performance of the [JIT](#) compiler component of the Java virtual machine. The class library is incomplete and there is no GUI support, i.e., [AWT](#) or Swing.

[Mauve](#), [Red Hat](#) -- A test suite for Java. This piece of software tests a JVM and library for compliance against Sun's Java specification. This software is open source and licensed under the [GPL](#).

[Microsoft SDK \(Software Development Kit\) for Java](#), [Microsoft](#) -- Microsoft has its own Java implementation. This package comes with Microsoft's Java compiler and library. You can also download Microsoft's Java Virtual Machine separately. This was part of a [lawsuit](#) between Microsoft and Sun. After 3 years, the lawsuit was finally resolved in January 2001. The resolution was not really a surprise, according to InfoWorld (Januar 29, 2001), since it didn't appear that Microsoft was in a position to win the case. Basically, Microsoft blatantly violated Sun's trademark by repeatedly placing Sun's Java logo on software that failed to meet Sun's compatibility testing, in violations of their contract. Many believe Microsoft didn't want to meet the compatibility tests since a compatible Java

implementation would enable the same Java software to run on both Microsoft and non-Microsoft platforms. As a result of the court decision, Microsoft has to pay \$20 million to Sun, Microsoft will not receive future Java licenses, Microsoft is under a permanent injunction against the use of the Java-compatible logo, and only the existing inventory of Java 1.1.14 can be distributed (under a limited license that expires in seven years). In response, Microsoft is trying to get developers to switch to Microsoft's proprietary ".NET" program and Microsoft's own "C#" language; how successful this will be is yet to be seen.

[NetRexx](#), [IBM](#) -- This software allows you to compile a dialect/variant of [Rexx](#) (Rexx is a pure procedural language developed by [Mike Cowlishaw](#) in the 1970's for [IBM](#)). It will generate Java source code, and now interpret code directly, with no compile step at all. The software is free but not open source.

[PERC](#), [NewMonics](#) -- PERC is a Java virtual machine for development of network element management and control applications. PERC supports incremental and de-fragmenting real-time garbage collection and deterministic real-time tasking. Also available is an ahead-of-time compiler. Developers can use Windows NT or Linux. Executables can run on Linux, Wind River's VxWorks[tm], ISI's pSOSystem[tm], Windows NT, Phar Lap's ETS, and VenturCom's RTX.

[Pizza](#), [University of South Australia](#) -- A compiler for an extended Java language that generates bytecode. It includes support for generic types (i.e. parametric polymorphism, often seen in the form of C++ templates), first class functions, and pattern matching. It is intended to be a superset of Java, but still compiles to regular Java bytecode and you can still use all the old Java libraries. See [GJ](#).

[SableVM](#), [Sable Research Group](#), [McGill University](#) -- Developed by a group of students and one professor at McGill University, this is similar to [LaTTe](#) in its goal to create a better, faster, Java Virtual Machine. It derives its [SablePath](#) library from the [GNU Classpath](#). Still alpha version software. It is open source under the [GPL](#).

[TurboJ](#), [Groupe Silicomp](#) -- Compiles Java bytecodes to native executable code. This is OEM'ed by a number of vendors: Hewlett Packard (as TurboChai), Wind River systems, and Fujitsu/Siemens. It supports a mix of both interpreted code and compiled code, and works as an adjunct to a native existing Java run-time system.

[TowerJ](#), [TowerJ](#) -- Compiles Java bytecode to native machine code. Allows developers to specify which portions of a system can be dynamically modified (opened) and which portions can not (closed).

Apparently Dead Projects

Here we list some projects that we found which appear to have little activity going on right now. They may still have useful nuggets so we've listed them here.

Product	Company/Organization	Cost/License	Components	Description
Harissa (more info)	Irisa	Free/ GPL *	Compiler , Interpreter	Compiles bytecode to C . Also has an Java interpreter
Toba (more info)	University of Arizona	Free/License??	Translator	Translates Java class files into C source code.

*Open Source Software (OSS)

[Harissa](#), [Irisa](#) -- Compiles bytecode to C and includes a small support library. Developed at a French University this research prototype is at version alpha3.1c. The last change occurred in January 1999. Harissa only supports JDK 1.0.2.

[Toba](#), [University of Arizona](#) -- Translates Java class files into C source code. The current version of Toba is 1.1c which was released in mid April of 1999. The link to license info about the software appears to be dead, also current development on this project has ceased. It may still may have some useful nuggets for other projects though.

Conclusion

If you know of other Java tools or alternate implementations that you think ought to be listed, please contact us at dwheeler@dwheeler.com. We don't believe there is one clear software package that will satisfy all objectives. However, we encourage you to investigate the packages listed here. You might find what you are looking for.

Links

Some other sources of Java information:

1. [Marco Schmidt's List of Java compilers and virtual machines](#), which also tries to capture this sort of information.
2. [Programming Languages for the Java Virtual Machine](#), which specializes in listing different compilers for different languages that generate Java source code or Java bytecode.
3. [Yahoo! Java](#)
4. [Gamelan](#)
5. [Javalobby](#)
6. [Java Repository](#)
7. [TransVirtual's List of Resources](#)
8. [The Java Spec Report \(Specification Errata\)](#)

AdaCGI Home Page

AdaCGI is an Ada 95 interface to the ["Common Gateway Interface" \(CGI\)](#). AdaCGI makes it easier to create Ada programs that can be invoked by World Wide Web (WWW) HTTP servers using the standard CGI interface. Using it, you can create Ada programs that perform queries or other processing by request from a WWW user. AdaCGI was formerly named "Package CGI".

AdaCGI is open source/free software, and is released using the LGPL ("Lesser General Public License") license. See the documentation for more information. You can use this library in proprietary programs but any changes to the library must stay as open source. No payment is required, but please provide credit if you use this package.

[Version 1.6](#) is the latest stable version; it was released 2000-November-1. The obsolete versions [1.4](#) and [1.5](#) are still available.

If you're interested in checking it out, you can:

1. [See a "screenshot"](#)
2. [View the documentation](#)
3. [View the spec](#)
4. [View the body](#)

I gave a presentation on AdaCGI to [DC SIGAda](#) in Rockville, MD on February 10, 2000. I updated them on September 25, 2000 for the SIGAda 2000 conference. You can get the latest version of this presentation in [Microsoft PowerPoint](#) and [PDF](#) formats.

If you're going to take the plunge and use it, you can **download** the stable version of AdaCGI in the following formats:

1. [zip format](#)
2. [tarball format](#)
3. [RPM format for Linux i386](#)
4. [SRPM format \(source installation\)](#)
5. I don't have a Debian (.deb) format version; until someone creates one, use the zip or tarball format if you use a Debian-based system. You can also use ``alien" to convert it; I'm told alien does a pretty good job with this package.

If you would like to join the AdaCGI mailing list, send an email to adacgi-list-request@adapower.com with phrase "subscribe" in the message body. You can then send email to adacgi-list@adapower.com to actually send messages to the mailing list subscribers.

Testimonial:

I can't thank you enough for your great creation! I have been extensively using the CGI package (version 1.5) in the last couple of weeks and have totally fallen in love with it! It is simple and does everything one could ask for! So thank you, I don't know how I could put together my project without it!

-- Alex Gertsen (AlexGertsen@libertybay.com)

I rarely hear about AdaCGI use; it seems to "just work". Also, AdaCGI sites doesn't look any different than other web sites unless they choose to identify themselves. A few sites that I know use AdaCGI include:

- An open dictionary review system to support [Den store danske ordliste](#) (a dictionary of the Danish language); this is at [Skåne Sjælland Linux User Group \(SSLUG\)](#), possibly the largest Linux user group in the world.
- A [voting system](#).

Mortality note: I don't intend to die anytime soon :-). However, should I decease or become permanently incapacitated, Clyde Roby (croby@ida.org) will receive the copyright of AdaCGI and be empowered to maintain it as he sees fit. Open source software can be maintained without such arrangements, but by making this arrangement I make it easier for the software to outlive me. However, even if this arrangement doesn't work out, the license makes it possible for anyone to help maintain this code beyond my lifetime.

Related Sites:

- [David A. Wheeler's Home Page](#)
- [Ada95 Lovelace Tutorial](#)
- [GNU Ada Homepage](#)
- [Ada for Linux \(ALT\) Home](#)
- [Ada Home](#)
- [Ada Power](#)
- [Generic_CGI](#), another Ada binding to CGI. Its author prefers a style different than mine, which is fine. It's free for use for many uses, and it has a small interface. However, note that it's not quite open source and it has some odd conditions (with legal ambiguities that might get users into trouble later). For example, it requires that "if you make a lot of money with the help of Ad^alib, you must contribute to its continuing development"; it's also only permitted for use to "make good software (and not anything else)". Also, it doesn't support cookies.

You can [go up to see David A. Wheeler's site](#).

Vim Ada Mode

Here is the [latest vim Ada syntax mode \(April 2001\)](#).

First, some background: [Vim](#) is a neat text editor intentionally similar to the venerable vi. It includes ``syntax highlighting" - it can color text you're editing for a variety of languages. [Ada](#) is a computer language emphasizing compile-time detection of errors.

I maintain the vim Ada mode. If you're using vim and Ada, here's the latest version of this mode; this will eventually get into the official vim sources, but if you you're using Ada and vim, you might want it *now*. In particular, I'd like feedback before this mode is shipped with a new version of vim (my email address is in the header of the mode file).

Here are some of the advantages of this vim Ada mode version (April 2001) over its predecessor (which was shipped in vim 5.7):

1. Based numbers are highlighted correctly (e.g., 16#FFFF#).
2. Operators (+, *, etc.) are colored as operators.
3. The +/- operators are colored differently than numeric signs (-5). So, the "-" in "A-5" is colored as an operator, while the same character in "A:=-5" is colored as a number.
4. Conditionals and repeats are grouped differently. Thus, you can specially color all loop keywords (if you did this, "for", "loop", and "end loop" would be colored differently than "if" and "end if").
5. The "with" and "use" clauses, when used to reference other compilation units, are colored differently (like "#include" in C).
6. Keywords aren't incorrectly colored as "preprocessor" colors, unless you turn that on as an option.
7. Standard exception names are highlighted as such.
8. Types in package Standard can be optionally highlighted.
9. Certain erroneous notation (e.g., "//") is immediately highlighted.
10. Various other options are available, e.g., for highlighting bad spaces.
11. Unlike the March 2001 version, if you're using vim 6.0, this version won't highlight leading spaces in certain cases involving "with" and "use" (avoiding this required advanced pattern-matching features not available in vim before version 6.0).

This vim Ada mode works on both vim version 5.7 (deployed version) and 6.0z (development version); it's been tested on both.

So, please download and enjoy:

[Click here to download ada.vim](#)

To use it, just replace your current Ada mode with this new version. For vim 5.7 on Red Hat Linux 7, this file is located at /usr/share/vim/vim57/syntax/ada.vim. On some systems, change "/usr/share" to "/usr/local/share"; for vim 6.0z, change "vim57" to "vim60z". Then just run "vim" or "gvim" and edit an Ada file. If you've turned off syntax highlighting, you can turn it back on with the command ":syntax on".

Here are the "help" instructions that I will propose with this mode when it's officially incorporated into vim:

Ada

This mode is designed for the 1995 edition of Ada ("Ada95"), which includes support for objected-programming, protected types, and so on. It handles code written for the original Ada language

("Ada83" or "Ada87") as well, though Ada83 code which uses Ada95-only keywords will be wrongly colored (such code should be fixed anyway). For more information about Ada, see <http://www.adapower.com>.

The Ada mode handles a number of situations cleanly. For example, it knows that the "-" in "-5" is a number, but the same character in "A-5" is an operator. Normally, a "with" or "use" clause referencing another compilation unit is colored the same way as C's "#include" is colored. If you have "Conditional" or "Repeat" groups colored differently, then "end if" and "end loop" will be colored as part of those respective groups. You can set these to different colors using vim's "highlight" command (e.g., to change how loops are displayed, enter the command ":hi Repeat" followed by the color specification; on simple terminals the color specification ctermfg=White often shows well).

There are several options you can select in this Ada mode. To enable them, assign a value to the option. For example, to turn one on:

```
let ada_standard_types = 1
```

To disable them use ":unlet". Example:

```
unlet ada_standard_types = 1
```

You can just use ":" and type these into the command line to set these temporarily before loading an Ada file. You can make these option settings permanent by adding the "let" command(s), without a colon, to your "~/.vimrc" file.

Here are the Ada mode options:

Variable	Action
ada_standard_types	Highlight types in package Standard (e.g., "Float")
ada_space_errors	Highlight extraneous errors in spaces...
ada_no_trail_space_error	but ignore trailing spaces at the end of a line
ada_no_tab_space_error	but ignore tabs after spaces
ada_withuse_ordinary	Show "with" and "use" as ordinary keywords (when used to reference other compilation units they're normally highlighted specially).
ada_begin_preproc	Show all begin-like keywords using the coloring of C preprocessor commands.

Even on a slow (90Mhz) PC this mode works quickly, but if you find the performance unacceptable, turn on ada_withuse_ordinary.

You're also welcome to look at my incomplete [Ada indent file](#). I can't seem to get this to work at all, so I'm not planning to work further on it. Help would be appreciated.

If you want to see my website, go to my home page at <http://www.dwheeler.com>.

Software Innovations

This web page is dedicated to identifying *software innovations*. To learn more, see the paper [*The Most Important Software Innovations*](#).

Feel free to examine my home page at <http://www.dwheeler.com>.

Program Library HOWTO

This is the main web site for the *Program Library HOWTO*. This HOWTO for programmers discusses how to create and use program libraries on Linux using the GNU toolset. A "program library" is simply a file containing compiled code (and data) that is to be incorporated later into a program; program libraries allow programs to be more modular, faster to recompile, and easier to update. Program libraries can be divided into three types: static libraries, shared libraries, and dynamically loaded (DL) libraries.

This document is to be part of the [Linux Documentation Project](#) (LDP), and distributed in various Linux distributions. However, note that the LDP's version or the version in a CD-ROM distribution may not be as current as the main (master) web site at "<http://www.dwheeler.com/program-library>".

You can see the HOWTO in [HTML \(good for on-line browsing\)](#), [PDF](#), [RTF](#), [Postscript](#), [ASCII Text \(not recommended\)](#), or [SGML \(DocBook DTD\)](#) formats. You can also see the [ChangeLog](#), and users of the SGML format may find the [Makefile](#) useful. The document includes a lot of trivial examples; you can download [the examples as a tarball](#). Or, to try out various library approaches, you can directly view [demo_dynamic.c](#), [demo_use.c](#), [libhello.c](#), and [libhello.h](#).

If you have comments, proposed improvements, or intend to translate it to another human language, please send email to dwheeler@dwheeler.com.

Other files are available here too. If you don't believe that `/lib` is searched before `/usr/lib`, you can run the test script originally developed by Michael Kerrisk at [usrlib-test](#), which requires source files [mod1.c](#) and [sltest.c](#). You can also view various ELF-related specifications; see [elf-linux.ps](#) and [elfspec.ps](#).

Regarding translations: A German translation is available at <http://www.freeco.de/linux/program-library>. A Japanese translation is available at <http://www.linux.or.jp/JF/JFdocs/Program-Library-HOWTO/index.html>. I hope to mention or link to other translations as I learn about them. Please contact me before translating, so that duplicate work can be avoided (for example, perhaps multiple translators could divide the work). If one becomes available, I *cannot* guarantee that the translations accurately reflect the original English work; I'm sorry, but I'm simply not qualified to judge that.

Feel free to see my main web site at <http://www.dwheeler.com>.

Linux URIs

This page presents the `uri(7)` man page, defining common URIs on GNU/Linux systems. This includes schemes common on GNU/Linux but not necessarily formally defined, such as the `man:` and `info:` schemes that both GNOME and KDE implement. If you're running a Linux system, you can just ask for "`man:uri(7)`" through a browser or type "`man 7 uri`" at the command line to view this information.

You can view `uri(7)` in one of the following formats:

1. [HTML](#)
2. [Postscript](#)
3. [PDF](#)
4. [man page format](#)

You might also want to look at related material, such as:

1. [Addressing Schemes](#) by Dan Connolly (a quasi-complete list of schemes)
2. [The Linux Documentation Project \(LDP\)](#)
3. [GNOME](#)
4. [KDE](#)
5. The [Secure BROWSER convention](#) (a convention for invoking user-defined browsers on Unix-like systems)

You can contact David A. Wheeler at dwheeler@dwheeler.com.

You are viewing <http://www.dwheeler.com/uri>.

ChessClub

"ChessClub" is a set of scripts useful as building blocks for creating web sites for Chess Clubs. These scripts allow you to:

1. Enter and edit chess games via the web, including comments. It will automatically add the names and ECO codes of openings, so it also includes an openings database for that purpose (which you might find useful for other reasons). Entering and editing are controlled by passwords, and multiple directories are supported. You can see [a non-live demo of the form](#).
2. Automatically index those games.
3. Automatically annotate those games using a program (it's set up to use Crafty). As a result, people can enter their games and later on they'll see an annotated commentary on their game!

These scripts are open source/free software. The auto-indexing and annotation are designed to work with Unix-like systems (including Linux); they might work with Windows-based systems if you add various programs to support sh shells, but I've not tried it. The scripts are under the GPL (with some minor extensions for the addgame script); the openings database is under the LGPL. There is no warranty. I've made some effort to make these scripts secure, but there's no guarantee that the results are secure. Please look over the scripts yourself and please send me any correction. See the scripts for details.

You can get the latest version of the scripts from <http://www.dwheeler.com/chessclub/chessclub.html>.

You'll find the scripts and other information available in a [chessclub tarfile](#). You can also see the [individual files through the web](#).

For more information, contact David A. Wheeler at dwheeler@dwheeler.com.

mm2frame

This is the home page for "mm2frame", a program for translating documents using the troff/mm macros into Adobe Framemaker. This software is released as open source/free software under the General Public License (GPL). I've only tested the program on some Unix machines; I don't know what else it runs on.

This is a VERY simplistic translator. It does not handle preprocessor commands, such as tables, equations, or pictures (tbl, eqn, pic), it ignores many parameters, it ignores all MM variables, and handles only a limited subset of the MM commands. Commands the translator doesn't understand are included in the resulting Framemaker document as regular text.

Nevertheless, this is helpful for documents which use (at least mostly) a subset of the MM commands. It certainly beats doing it ALL by hand, and I'm unaware of any other tool that does this. I've found it handy when I needed it, and over 30 people that I know of have requested it from me (there may be more people who have used it, since I don't restrict redistribution).

The translator handles the following MM commands: italics (\fI, .ft I or .I), bold (\fB, .ft B or .B), normal (\fP, \fR, .R or .ft R), headers (.H 1-3), dashed and bold lists (.DL/.BL - .LI - .LE; can't embed them), don't fill/fill (.nf/.fi), paragraph (.P), and center (.ce no arguments, creates a Title paragraph). The vertical space commands (.sp, .SP, .SK, and .bp) are interpreted as new-paragraph commands. The table begin and end commands (.TS/.TE) and displays (.DS or .DF /.DE) will cause each line to be its own Framemaker paragraph (as don't fill/fill do) in order to make translating easier. Special characters such as double-backslash, angle brackets, dagger, logical not, em dash, and double quote marks (both directions) are translated as well.

I'm no longer maintaining this software, If you're interested in taking over maintenance of it, let me know. The software includes extensive design documentation, which should help you get started.

You can look at the [installation instructions](#) and the [user and design documentation](#). Most importantly, you can [download the actual mm2frame software \(as a shar archive\)](#).

You can go up and look at [my main web site](#).

David A. Wheeler's Miscellaneous Files

This area is a collection of miscellaneous files. The following files are available:

- [db2latex, a translator that translates DocBook \(db\) files to LaTeX \(latex\) files.](#)
- [klinton, an old tty game \(circa 1976\) reimplemented in Python \(requires Python\).](#)

David A. Wheeler's Essays


This area is a collection of my random essays, which can be of truly any subject. The following essays are available:

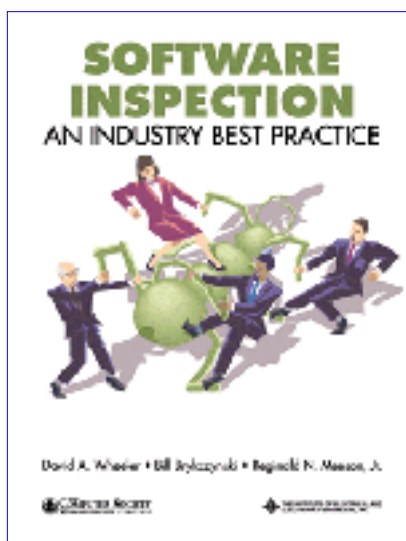
- [Way Off Base](#) (Mathematical recreation)
- [Write It Secure: Format Strings and Locale Filtering](#) (this article is published in E-Security Journal at <http://www.eSecurityJournal.com>)



David A. Wheeler

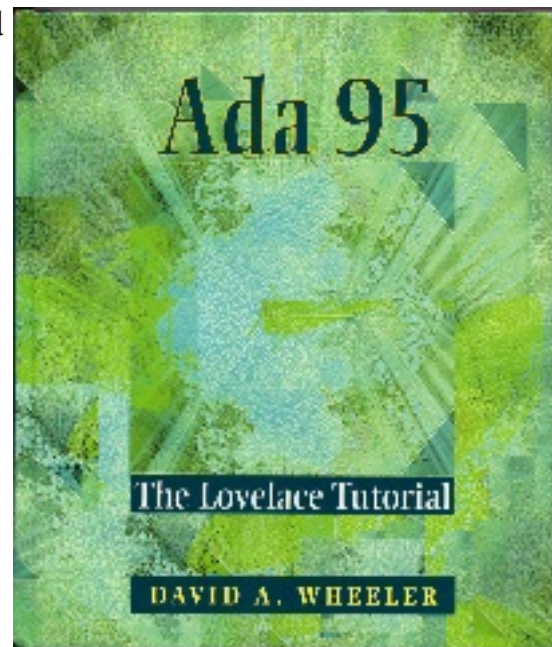
My professional interests are in improving software development practices for higher-risk software systems (i.e., ones which must be secure, large, and/or [safety-critical](#)). I've done a good deal of security work, especially in the area of [writing secure programs](#). Other areas of interest include [inspections](#), Internet/web standards and technologies, software risk assessment, Ada, C++, Java, POSIX, and Linux.

 Most of my written work is *not* available publicly. However, here are two books I've published:



I'm an author and editor of [Software Inspection: An Industry Best Practice](#) by David A. Wheeler, Bill Brykczynski, and Reginald N. Meeson, Jr.. This book, published by the IEEE Computer Society Press, describes the software inspection process and includes a number of papers on the topic, including results from many different users of the process. This book is listed in the "Best Sellers" category in the IEEE Computer Society "New Releases" Fall 1996 catalog. It is ISBN 0-8186-7340-0, IEEE Catalog Number BP07340, and Library of Congress Number 95-41054. You can order this book from the IEEE by emailing to cs.books@computer.org, calling U.S. (714) 821-8380, or faxing to U.S. (714) 821-4641. [You can find more information about the book from the IEEE.](#)

Another book of mine is *Ada 95: The Lovelace Tutorial* by David A. Wheeler, now available as a hardcover book. The publisher is Springer-Verlag and its ISBN number is 0-387-948-01-5. The book is about 292 pages long and was published in 1997. This book is a tutorial on the Ada95 computer programming language; it assumes you know some other computer programming language. You can order this book by calling Springer-Verlag; in the U.S.A., Canada, or Mexico call (800) 777-4643; otherwise call their Berlin, Germany office at 49 30 827 870.



To contact me, send me email at dwheeler@dwheeler.com. You can also see my website at <http://www.dwheeler.com>.