

Que posição você prefere?

Nesta seção aprenderemos, principalmente, a posicionar o cursor, além de outras facilidades, para que, quando necessitarmos receber os dados via teclado, possamos formatar a tela visando melhorar a apresentação e o entendimento do que está sendo pedido.

Neste ponto, já devemos saber que existe a instrução `clear`, cuja finalidade é limpar a tela e que deve ser o ponto inicial de qualquer rotina de recepção de dados via teclado.

Para formatação de tela, além do `clear` existe uma instrução multifacetada de uso geral, que é o `tput`. Veremos a seguir as principais faces dessa instrução:

- `tput cup (cup → cursor position)` – Cuja finalidade é posicionar o cursor na tela e cuja sintaxe é a seguinte:

```
tput cup lin col
```

Onde `lin` é a linha e `col` a coluna onde se deseja posicionar o cursor. É interessante e importante assinalar que a numeração das linhas e das colunas começa em zero.

- `tput home` – O mesmo que `tput cup 0 0`
- `tput bold` – Coloca o terminal no modo de ênfase, chamando a atenção sobre o que aparecerá na tela a partir daquele ponto até a sequência de restauração de tela.
- `tput smso` – Coloca o terminal no modo de vídeo reverso, a partir daquele ponto até a sequência de restauração de tela.
- `tput rev` – Idêntico ao `tput smso`.
- `tput smul` – Todos os caracteres, a partir daquele ponto até a instrução para restauração de tela, aparecerão sublinhados na tela.
- `tput blink` – Coloca o terminal em modo piscante, a partir daquele ponto até a sequência de restauração de tela (nem todos os *terminfo* aceitam esta opção).
- `tput sgr0` – Restaura o modo normal do terminal. Deve ser usado após um dos três comandos anteriores, para restaurar os atributos de vídeo.
- `tput reset` – Restaura todos os parâmetros do seu terminal voltando suas definições ao *default* do *terminfo* – que está definido na variável do sistema `$TERM` – e dá um `clear` no terminal. Sempre que possível deve ser usado no final da execução de programas que utilizam a instrução `tput`.
- `tput lines` – Devolve a quantidade de linhas do monitor corrente, terminal corrente ou console (se esta for o monitor corrente).

- `tput cols` – Devolve a quantidade de colunas do monitor corrente, terminal corrente ou console (se esta for o monitor corrente).
- `tput ed` (`ed` → *erase display*) – Limpa a tela a partir da posição do cursor até o fim do monitor corrente, terminal corrente ou console (se esta for o monitor corrente).
- `tput el` (`el` → *erase line*) – Limpa a partir da posição do cursor até o fim da linha.
- `tput il N` (`il` → *insert lines*) – Insere `N` linhas a partir da posição do cursor.
- `tput dl N` (`dl` → *delete lines*) – Deleta `N` linhas a partir da posição do cursor.
- `tput dch N` (`dch` → *delete characters*) – Deleta `N` caracteres a partir da posição do cursor.
- `tput civis` – Torna o cursor invisível (produz o mesmo efeito de `setterm -cursor off`).
- `tput cvvis` ou `tput cnorm` – Volta a mostrar o cursor (produz o mesmo efeito de `setterm -cursor on`).
- `tput flash` – Dá uma claridade intensa e rápida (*flash*) na tela para chamar a atenção.
- `tput sc` (`sc` → *save cursor position*) – Guarda a posição atual do cursor.
- `tput rc` (`rc` → *restore cursor to position*) – Retorna o cursor para a última posição guardada pelo `sc`.
- `tput smcup` – Bate uma foto atual da tela, limpa-a e salva-a para posterior recuperação.
- `tput rmcup` – Repõe na tela a foto batida com o comando `tput smcup`.
- `tput setaf` – Especifica a cor da fonte (*foreground*).
- `tput setab` – Especifica a cor de fundo (*background*).
- `tput op` – Restaura as cores de fonte e fundo padrões.
- `tput invis` – Coloca o terminal invisível.
- `stty` – Este comando tem uma série imensa de opções de pouco uso, porém muito fortes, podendo inclusive alterar as definições do *terminfo*. Vale a pena, por ser muito usada, esmiuçarmos a opção `echo`, que serve para inibir ou restaurar o eco das teclas no terminal, isto é, caso o seu *script* possua um comando `stty -echo`, tudo que for teclado daquele ponto até a ocorrência do comando `stty echo` não aparecerá no terminal de vídeo. É de suma importância para recebermos uma **senha** pela tela.

Com esse comando também se pode colorir a tela. Mais adiante, no Capítulo 8, seção “Mandando no terminal”, você verá outras formas de fazer a mesma coisa; acho, porém, esta que veremos agora mais intuitiva (ou menos “desintuitiva”).

Os valores de `setaf` e `setab` variam de 0 a 7, e 9 é o valor que restaura as cores do fonte e/ou do fundo para seu(s) padrão(ões). Veja seus valores e cores correspondentes na tabela a seguir:

Símbolo	Cor
0	Preto
1	Vermelho
2	Verde
3	Amarelo
4	Azul
5	Magenta
6	Ciano
7	Branco
9	Volta a cor <i>default</i>

Como em um livro só é possível utilizar matizes de cinza, fica bastante complicado mostrar esse padrão de cores, por isso a seguir tem um programa que você deveria executá-lo para ver todas as possibilidades de combinação de cores de fonte e de fundo usando o comando `tput` (como já foi dito, existem outras formas, mas essa é a melhor).

```
$ cat cores1.sh
#!/bin/bash
for ((b=0; b<=7; b++))
{
    tput setab 9; tput setaf 9; echo -n "|"
    for ((f=0; f<=7; f++))
    {
        tput setab $b; tput setaf $f; echo -n " b=$b f=$f "
        tput setab 9; tput setaf 9; echo -n "|"
    }
    echo
}
tput setab 9; tput setaf 9
```

Os matizes das cores das fontes variam um pouco se a opção `tput bold` estiver ativa (normalmente a cor com ênfase é um tom mais claro). Execute o programa a seguir que você verá essas diferenças.

```
$ cat cores4.sh
```

```
#!/bin/bash
# cores4.sh - Lista as cores da console com bold
clear
for ((Cor=0; Cor <=7; Cor++))
{
    for Modo in sgr0 bold
    {
        tput $Modo
        tput setaf $Cor
        printf -v Linha "%-${tput cols}s" "-Em-modo-$([ $Modo = sgr0 ]
&& echo Normal || echo Bold)-"
        sed "s/ /█/g" <<< "$Linha"
    }
    tput sgr0
}
}
```

Neste exemplo o **printf** foi usado para formatar a saída, preenchendo-a com espaços em branco até o tamanho da linha do terminal e jogando-a na variável **\$Linha** (por causa da opção **-v**). O **sed** trocou os espaços em branco por este símbolo que preenche todo o espaço destinado a cada letra (█) .

A seguir, um exemplo de como se salva/recupera a tela. Aproveitando a oportunidade, mostramos diversas opções da instrução **tput** .

\$ cat salva_tela.sh

```
#!/bin/bash
((${tput cols} < 110)) && {
    clear; echo Tela precisa de largura mínima de 110 caracteres
    exit 1
}
seq 5000 | xargs
Lin=$((tput lines) / 2)
Col=$((tput cols) / 2)
tput cup $Lin; tput el; tput bold
tput cup $Lin $((Col-55)); tput setaf 4
echo "Em 10 segundos essa tela será fotografada e se apagará"
tput civis
for ((i=10; i!=0; i--))
{
    tput cup $Lin $Col; tput el
    echo $i
    sleep 1
}
tput smcup
clear
tput cup $Lin $((Col-53))
echo "Dentro de 10 segundos a tela inicial será recuperada"
for ((; i<10; i++))
{
    tput cup $Lin $Col
    echo $i
```

Sujando a tela

Calculando linha e coluna centrais da tela

Posicionando e apagando a linha central

Posicionando e colorindo para dar mensagem

Cursor invisível para melhorar apresentação

Posiciona no centro da tela e limpa núm anterior

Tirando uma foto da tela

Poderia ter usado tput reset

Posicionou no centro da tela

```
        sleep 1
    }
    tput rmcup
    tput cvvis;tput setaf 9
```

Restaurou a foto

Restaurou o cursor e cor

A seguir, um *script* que serve para especificar o par de cores (da letra e do fundo).
Veja:

```
$ cat mudacor.sh
#!/bin/bash
tput sgr0
clear
# Carregando as 8 cores básicas para o vetor Cores
Cores=(Preto Vermelho Verde Marrom Azul Púrpura Ciano "Cinza claro")
# Listando o menu de cores
echo "
Opc      Cor
===      ==="
for ((i=1; i<=${#Cores[@]}; i++))
{
    printf "%02d      %s\n" $i "${Cores[i-1]}"
}
CL=
until [[ $CL == 0[1-8] || $CL == [1-8] ]]
do
    read -p "
Escolha a cor da letra: " CL
done
# Para quem tem bash a partir da versao 3.2
#+ o test do until acima poderia ser feito
#+ usando-se Expressoes Regulares. Veja como:
#+ until [[ $CL =~ 0?[1-8] ]]
#+ do
#+     read -p "
#+ Escolha a cor da letra: " CL
#+ done
CF=
until [[ $CF == 0[1-8] || $CF == [1-8] ]]
do
    read -p "
Escolha a cor de fundo: " CF
done
let CL-- ; let CF-- # A cor preta eh zero e nao um
tput setaf $CL
tput setab $CF
clear
```

Exemplo:

```
$ cat tputcup
clear
tput cup 3 6
```

```

echo ".<-"
tput cup 2 10
echo "/"
tput cup 1 12
echo "/"
tput cup 0 14
echo "_____ Este eh o ponto (3, 6)"

```

Executando vem:

```

$ tputcup
$      _____ Este eh o ponto (3, 6)
      /
     /
    .<-

```

Note que, no exemplo citado, propositadamente, a tela foi formatada de baixo para cima, isto é, da linha 3 para a zero. Isso explica a presença do *prompt* (\$) na linha 1, já que, quando acabou a execução do programa, o cursor estava na linha zero.

Por curiosidade, vamos tirar a instrução **clear** da primeira linha, em seguida vamos listar o programa e executá-lo. A seguir está a tela resultante desses passos:

```

$ cat tputcup _____ Este eh o ponto (3, 6)
$ ut cup 3 6/
echo ".<-" /
tput cup.<-10
echo "/"
tput cup 1 12
echo "/"
tput cup 0 14
echo "_____ Este eh o ponto (3, 6)"
$ tputcup

```

Nessa bagunça, vimos que a execução do programa foi feita por cima de sua listagem e, conforme dá para perceber, se não usarmos o **clear** no início do programa que trabalha com **tput cup**, sua tela formatada normalmente ficará comprometida.

Já que as outras formas de **tput** envolvem somente atributos de tela, fica difícil, em uma publicação, apresentar exemplos que ilustrem seus usos.

Para finalizar:

```

$ cat mengo1.sh
#!/bin/bash
# Cria uma bandeira rubronegra na tela

trap "tput sgr0; tput cnorm; clear; exit" 0 2 3 15

clear
for ((i=0;i<$(tput lines);i++))

```

```

{
    tput cup $i 0
    tput setab ${i%2}
    sleep .1
    tput ed
}
for ((i=1;i<10;i++))
{
    tput flash
    sleep .5
}
Lin[0]="mmmm      mmmm  eeeeeee  nnnn   nnn      gggg      0000"
Lin[1]="mmmmm      mmmm  eeeeeee  nnnnn  nnn      ggg  ggg      ooo  ooo"
Lin[2]="mmmmmm      mmmm  eee       nnnnnn nnn      ggg          ooo  ooo"
Lin[3]="mmm  mmmmmmm  mmm  eeeee    nnn  nnnnnn  ggg          ooo  ooo"
Lin[4]="mmm  mmm      mmm  eee       nnn  nnnnn  ggg  ggggg  ooo  ooo"
Lin[5]="mmm          mmm  eeeeeee  nnn   nnnn   ggg  ggg      ooo  ooo"
Lin[6]="mmm          mmm  eeeeeee  nnn   nnn      gggg          oooo"
let ColIni=$((tput cols) - ${#Lin[3]})/2
let LinIni=$((tput lines) - 7)/2
((ColIni < 0 || LinIni < 0)) && {
    zenity --error --text "Para curtir o resto da
animação,\n<b><big>Aumente o tamanho da tela</big></b>"
    exit 1
}
tput setab 1
tput setaf 0
tput civis
clear
for ((i=0; i<=6; i++))
{
    tput cup $((LinIni + i)) $ColIni
    echo "${Lin[i]}"
    sleep .3
}

for ((i=1;i<10;i++))
{
    tput flash
    sleep .5
}
sleep 3

```