tput

通过terminfo数据库对终端会话进行初始化和操作

补充说明

tput命令 将通过 terminfo 数据库对您的终端会话进行初始化和操作。通过使用 tput，您可以更改几项终端功能，如移动或更改光标、更改文本属性，以及清除终端屏幕的特定区域。

什么是 terminfo 数据库？

UNIX 系统上的 terminfo 数据库用于定义终端和打印机的属性及功能，包括各设备（例如，终端和打印机）的行数和列数以及要发送至该设备的文本的属性。UNIX 中的几个常用程序都依赖 terminfo 数据库提供这些属性以及许多其他内容，其中包括 vi 和 emacs 编辑器以及 curses 和 man 程序。

与 UNIX 中的大多数命令一样，tput 命令既可以用在 shell 命令行中也可以用在 shell 脚本中。为让您更好地理解 tput，本文首先从命令行讲起，然后紧接着讲述 shell 脚本示例。

光标属性

在 UNIX shell 脚本中或在命令行中，移动光标或更改光标属性可能是非常有用的。有些情况下，您可能需要输入敏感信息（如密码），或在屏幕上两个不同的区域输入信息。在此类情况下，使用 tput 可能会对您有所帮助。

```
tput clear # 清屏
tput sc # 保存当前光标位置
tput cup 10 13 # 将光标移动到 row col
tput civis # 光标不可见
tput cnorm # 光标可见
tput rc # 显示输出
exit 0
```

移动光标

使用 tput 可以方便地实现在各设备上移动光标的位置。通过在 tput 中使用 cup 选项，或光标位置，您可以在设备的各行和各列中将光标移动到任意 X 或 Y 坐标。设备左上角的坐标为 (0,0)。

要在设备上将光标移动到第 5 列 (X) 的第 1 行 (Y)，只需执行 tput cup 5 1。另一个示例是 tput cup 23 45。此命令将使光标移动到第 23 列上的第 45 行。

移动光标并显示信息

另一种有用的光标定位技巧是移动光标，执行用于显示信息的命令，然后返回到前一光标位置：

```
(tput sc ; tput cup 23 45 ; echo “Input from tput/echo at 23/45” ; tput rc)
```

下面我们分析一下 subshell 命令：

```
tput sc
```

必须首先保存当前的光标位置。要保存当前的光标位置，请包括 sc 选项或“save cursor position”。

```
tput cup 23 45
```

在保存了光标位置后，光标坐标将移动到 (23,45)。

```
echo “Input from tput/echo at 23/45”
```

将信息显示到 stdout 中。

```
tput rc
```

在显示了这些信息之后，光标必须返回到使用 tput sc 保存的原始位置。要使光标返回到其上次保存的位置，请包括 rc 选项或“restore cursor position”。

注意：由于本文首先详细介绍了通过命令行执行 tput，因此您可能会觉得在自己的 subshell 中执行命令要比单独执行每条命令然后在每条命令执行之前显示提示更简洁。

更改光标的属性

在向某一设备显示数据时，很多时候您并不希望看到光标。将光标转换为不可见可以使数据滚动时的屏幕看起来更整洁。要使光标不可见，请使用 civis 选项（例如，tput civis）。在数据完全显示之后，您可以使用 cnorm 选项将光标再次转变为可见。

文本属性

更改文本的显示方式可以让用户注意到菜单中的一组词或警惕用户注意某些重要的内容。您可以通过以下方式更改文本属性：使文本加粗、在文本下方添加下划线、更改背景颜色和前景颜色，以及逆转颜色方案等。

要更改文本的颜色，请使用 setb 选项（用于设置背景颜色）和 setf 选项（用于设置前景颜色）以及在 terminfo 数据库中分配的颜色数值。通常情况下，分配的数值与颜色的对应关系如下，但是可能会因 UNIX 系统的不同而异：

- 0：黑色
- 1：蓝色
- 2：绿色
- 3：青色
- 4：红色
- 5：洋红色
- 6：黄色
- 7：白色

执行以下示例命令可以将背景颜色更改为黄色，将前景颜色更改为红色：

```
tput setb 6 tput setf 4
```

要反显当前的颜色方案，只需执行`tput rev`。

有时，仅为文本着色还不够，也就是说，您想要通过另一种方式引起用户的注意。可以通过两种方式达到这一目的：一是将文本设置为粗体，二是为文本添加下划线。

要将文本更改为粗体，请使用 bold 选项。要开始添加下划线，请使用 smul 选项。在完成显示带下划线的文本后，请使用 rmul 选项。

实例

使输出的字符串有颜色，底色，加粗：

```
#!/bin/bash
printf $(tput setaf 2; tput bold)'color show\n\n'$(tput sgr0)

for((i=0; i<=7; i++)); do
    echo $(tput setaf $i)"show me the money"$(tput sgr0)
done

printf '\n'$(tput setaf 2; tput setab 0; tput bold)'background color
show'$(tput sgr0)'\n\n'

for((i=0,j=7; i<=7; i++,j--)); do
    echo $(tput setaf $i; tput setab $j; tput bold)"show me the money"$(tput
sgr0)
done

exit 0
```

输出格式控制函数：

```
#!/bin/bash

# $1 str       print string
# $2 color     0-7 设置颜色
# $3 bgcolor   0-7 设置背景颜色
# $4 bold      0-1 设置粗体
# $5 underline 0-1 设置下划线

function format_output(){
    str=$1
    color=$2
    bgcolor=$3
    bold=$4
    underline=$5
    normal=$(tput sgr0)

    case "$color" in
        0|1|2|3|4|5|6|7)
            setcolor=$(tput setaf $color;) ;;
        *)
            setcolor="" ;;
    esac

    case "$bgcolor" in
        0|1|2|3|4|5|6|7)
            setbgcolor=$(tput setab $bgcolor;) ;;
        *)
            setbgcolor="" ;;
```

```
    esac

    if [ "$bold" = "1" ]; then
        setbold=$(tput bold;)
    else
        setbold=""
    fi

    if [ "$underline" = "1" ]; then
        setunderline=$(tput smul;)
    else
        setunderline=""
    fi

    printf "$setcolor$setbgcolor$setbold$setunderline$str$normal\n"
}

format_output "Yesterday Once more" 2 5 1 1

exit 0
```

光标属性例子：

```
#!/bin/bash
# clear the screen
tput clear
# Move cursor to screen location X,Y (top left is 0,0)
tput cup 3 15
# set a foreground colour using ANSI escape
tput setaf 3
echo "XYX Corp LTD."
tput sgr0
tput cup 5 17
# Set reverse video mode
tput rev
echo "M A I N - M E N U"
tput sgr0
tput cup 7 15
echo "1\. User Management"
tput cup 8 15
echo "2\. service Management"
tput cup 9 15
echo "3\. Process Management"
tput cup 10 15
echo "4\. Backup"
# Set bold mode
tput bold
tput cup 12 15
read -p "Enter your choice [1-4] " choice
tput clear
tput sgr0
```

```
tput rc

exit 0
```