Configuring the Bash prompt

# PRIMPED PROMPT

Color-coding your prompt may help avoid configuration errors and data loss. We'll show you how to design your own custom shell prompt with color and control sequences. **BY HEIKE JURZIK**

ANSI control sequences for color and cursor positioning give users the ability to design a customized and functional Bash prompt to suit their own needs.

Command-line fans often find it hard to see the prompt for Xterms. If the prompt on a remote machine looks different from your home prompt, it might save you from inadvertent shutdown commands, and if unprivileged users see a green prompt and root sees a red one, you might be able to avoid configuration errors or even data loss. A shell prompt tailored to match your needs can help prevent confusion.

The default Bash prompt looks pretty much the same on most Linux distributions. SUSE Linux systems set up *user@host:~ >* for normal users, whereas Debian uses *user@host:~$*. The tilde represents the user's home directory. In other words, the prompt shows the current working directory on both systems, giving the user an orientation aid. If you change directory to */var/log* on Debian, the prompt will look like:

```
user@host:/var/log$
```

Most Linux systems change the prompt for the root user. The username is left out and you see a pound sign (*#*) instead of an angle bracket or dollar sign.

The *$PS1* [1] environmental variable defines the appearance of the Bash prompt, which appears after entering a command. You can modify the *$PS1* temporarily in the current shell for test purposes. When the prompt has the look and feel you need, you can make the changes permanent by modifying the Bash configuration file, *~/.bashrc*. Look for the default settings for *$PS1*, comment out the line by inserting a pound sign (*#*) at the start, and add your own construction.

If one of your prompt experiments in the next couple of sections goes haywire, you can close the current Bash session and pop up a new shell or call *source ~/.bashrc* to restore the default prompt.

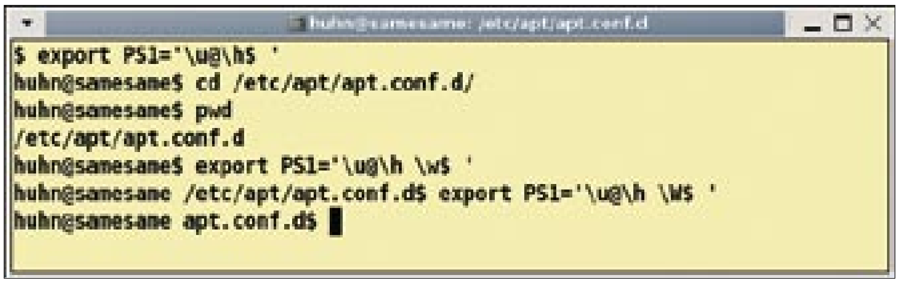
**Listing 1: Time Display Options**

```
chicken@samesame ~$ export PS1='[\t] \u@\h \w$ '
[20:53:03] chicken@samesame ~$ export PS1='[\T] \u@\h \w$ '
[08:53:06] chicken@samesame ~$ export PS1='[\@] \u@\h \w$ '
[08:53 ] chicken@samesame ~$
```

## A New Format

To remove the username and hostname from the prompt and use a short but

Alexandre, Fotolia



```
$ export PS1='\u@\h$ '
huhn@samesame$ cd /etc/apt/apt.conf.d/
huhn@samesame$ pwd
/etc/apt/apt.conf.d
huhn@samesame$ export PS1='\u@\h \w$ '
huhn@samesame /etc/apt/apt.conf.d$ export PS1='\u@\h \W$ '
huhn@samesame apt.conf.d$ 
```

**Figure 1: If you use the escape sequence \W instead of \w in the prompt, you only see the current directory name.**

sweet prompt with a dollar sign, do the following:

```
chicken@samesame:~$ ↵
export PS1=$
$ls
bin/   easy/   user/
```

This does save space, but it also looks a little cramped.

If you would prefer a space between the dollar sign and the command, you can simply put the space and dollar sign in quotes:

```
$export PS1='$ '
$ ls
bin/   easy/   user/
```

This also applies to Bash escape sequences (see Table 1). Whenever the value for *$PS1* includes an escape sequence, a blank, or a non-standard character, you will need to place the expression in quotes.

Escape sequences allow users to really polish the prompt. If you like to see your current username and hostname, you can display them by entering *\u* and *\h*. If you split these names with an "at" sign (@), the results are quite neat:

```
$ export PS1='\u@\h$ '
chicken@samesame$
```

What's missing now is the current working directory.

If you change directory without the current working directory in the prompt, you have to enter *pwd* ("print working directory") to find out where you are in the directory tree. With the escape sequences *\w* and *\W*, this is changed easily. The former displays the full path and can be extremely long, whereas the latter just displays the directory name (see Figure 1).

If you want the prompt to tell you the time, you have three display options: *\t* gives you a 24-hour format (HH:MM:SS) and *\T* (HH:MM:SS) and *\@* (HH:MM) give a 12-hour format, with and without seconds. Square brackets help you format the output (Listing 1).

## Painting by Numbers

The ANSI color codes I covered in last month's article [2] can help you paint the prompt. You need to quote all of these control sequences in *\[\e[* and *\]* (see Table 1).

To paint the prompt with the time from the last section green but display the time in blue:

- start by typing *\[* to open the control sequence,
- define the color (*\e[0;34m*),
- close the control sequence with *\]*,
- set the date (e.g., in 24-hour format) in square brackets: *[\T]*),
- change the color to green for the username and for the hostname

## Prompts for All Occasions

Besides *$PS1*, other environmental variables affect the appearance of the prompt:

- *$PS2*, which appears if you wrap a command line by pressing Enter but without terminating the line (because closing quotes or brackets are missing, for example),
- *$PS3*, which is used by Bash's *select* control element, and
- *$PS4*, which appears at the start of every line while you are debugging a script.

I will focus on *$PS1* in this article because its siblings are too uncommon in daily work with the shell.

## Table 1: Escape Sequences for the Prompt

| Character | Meaning |
|---|---|
| *\d* | Short date format (e.g., *Tu Jul 24*) |
| *\e* | Escape character (^[) |
| *\h* | Short hostname (up to first dot; e.g., *samesame*) |
| *\H* | Long hostname (e.g., *samesame.chickenix.org*) |
| *\j* | Number of jobs running in this shell |
| *\l* | Base name of terminal (e.g., *10* if your terminal is *pts/10*) |
| *\n* | New line |
| *\r* | Carriage return |
| *\s* | Name of shell (e.g., *bash*) |
| *\t* | Time in 24-hour format (HH:MM:SS; e.g., *22:11:55*) |
| *\A* | Time in 24-hour format without seconds (HH:MM; e.g., *22:11*) |
| *\T* | Time in 12-hour format (HH:MM:SS; e.g., *10:11:55*) |
| *\@* | Time in 12-hour format without seconds (HH:MM; e.g., *10:11*) |
| *\u* | Username (e.g., *chicken*) |
| *\v* | Bash version you are using (e.g., *3.1*) |
| *\V* | Bash release with version number and patch level (e.g., *3.1.17*) |
| *\w* | The current working directory with full path name (e.g., *chicken@samesame /etc/apt$*, etc.) |
| *\W* | The current working directory, restricted to the current folder (e.g., *chicken@samesame apt$* if the user is currently in */etc/apt*) |
| *\!* | Order of the command in the Bash history (e.g., *123*, the 123rd command) |
| *\#* | Number of commands in the current shell session |
| *\$* | Shows a dollar sign for non-UID 0 users 0 (=root), a pound sign otherwise (*#*) |
| *\nnn* | Three-digit octal number *nnn* with which you can display any ASCII character (e.g., *\033* for Escape_ |
| *\* | The backslash (\) itself |
| *\[* | Escapes a following control character (e.g., ANSI escape sequences or ANSI cursor control sequences) |
| *\]* | Closes control character input |

(*\[\e[0;32m\]\u@\h*),
- switch back to the standard settings (*\[\e[0m\]*), and
- output the current working directory and everything else (Figure 2).

If you prefer to see a warning when you are working as root, you can use different colors for user and root prompts. To do so, add, for example, to your user account *~/.bashrc*,

```
export PS1='\[\e[01;32m\]\u@\h\[
\e[00m\]:\w\$ '
```

then add the following to the */root/ .bashrc* file:

```
export PS1='\[\e[01;31m\]\h\[\e[
00m\]:\w\$ '
```

Your user prompt will be plain green, whereas the root prompt will be red to keep you on your toes.

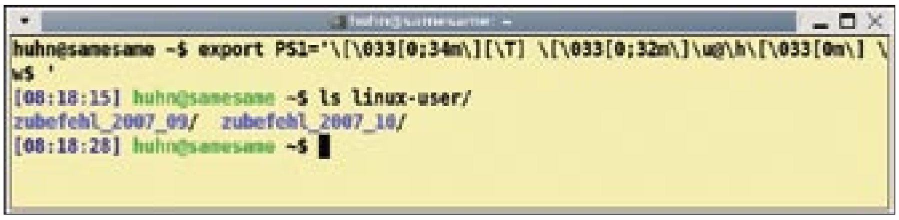
## Cursor Moves

In addition to ANSI color control sequences are cursor positioning sequences. For example, *\e[<n>A* moves the cursor up and *\e[<n>B* moves it down *<n>* lines; *C* moves the cursor right, and *D* moves it left (Table 2).

In combination with color sequences, you can create practical and individual looks for the prompt. If you often find yourself at the bottom of a long path of directories (with 60 to 70 characters), you can design your prompt to display a status line, with the path and time at the top edge of the terminal and a simple prompt with a dollar sign (as a user) or pound sign (as root) (Figure 3).

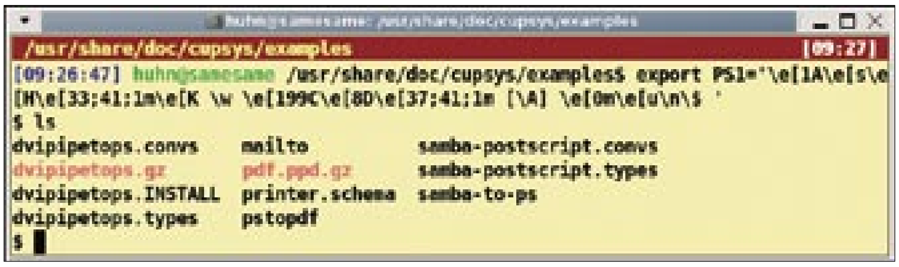
To do this:
- move the cursor one line up (*\e[1A*) to prevent the status line from being pushed out later,
- store the position (*\e[s*),
- move the cursor to the top left corner position *1;1* (*\e[H*),
- color the output yellow on red (*\e[33;41;1m*) and delete from the cursor position to end of the line (*\e[K*),
- output the full path (*\w*),
- move the cursor 199 positions to the right (full right, *\e[199C*),
- move back eight positions left (*\e[8D*),
- output the time in 24-hour format without seconds (*\A*) in white on red (*\e[37;41;1m*),
- reset the colors to the defaults (*\e[0m*),
- use *\e[u* to restore the stored cursor position, and
- insert a new line (*\n*) and output the prompt (*\$*).

The result is always an empty command line; however, you still will know what directory you are working in and what time it is. ■



```
huhn@samesame ~$ export PS1='\[\033[0;34m\][\T] \[\033[0;32m\]\u@\h\[\033[0m\] \
w$ '
[08:18:15] huhn@samesame ~$ ls linux-user/
zubefehl_2007_09/  zubefehl_2007_10/
[08:18:28] huhn@samesame ~$ 
```

Figure 2: Painting the prompt with ANSI escape sequences.



```
 /usr/share/doc/cupsys/examples                                        [09:27]
[09:26:47] huhn@samesame /usr/share/doc/cupsys/examples$ export PS1='\e[1A\e[s\e
[H\e[33;41;1m\e[K \w \e[199C\e[8D\e[37;41;1m [\A] \e[0m\e[u\n\$ '
$ ls
dvipipetops.convs   mailto          samba-postscript.convs
dvipipetops.gz      pdf.ppd.gz      samba-postscript.types
dvipipetops.INSTALL printer.schema  samba-to-ps
dvipipetops.types   pstopdf
$ 
```

Figure 3: Keeping track despite long path names and with the time, too.

**Table 2: ANSI Control Sequences for the Cursor**

| Character | Meaning |
|---|---|
| *\e[<n>A* | *<n>* lines up (defaults to 1 if you do not specify *<n>*) |
| *\e[<n>B* | *<n>* lines down (defaults to 1 if you do not specify *<n>*) |
| *\e[<n>C* | *<n>* spaces right (defaults to 1 if you do not specify *<n>*) |
| *\e[<n>D* | *<n>* spaces left (defaults to 1 if you do not specify *<n>*) |
| *\e[<n>E* | Moves the cursor to the start of a following line (defaults to 1 if you do not specify *<n>*) |
| *\e[<n>F* | Moves the cursor to the start of a previous line (defaults to 1 if you do not specify *<n>*) |
| *\e[<n>G* | Moves the cursor to column *<n>* |
| *\e[<n>;<m>H* | Moves the cursor to line *<n>*, column *<m>* (defaults to 1 if you do not specify *<n>* or *<m>*) |
| *\e[<n>J* | Clears the screen; if *<n>* = 0 or is missing, the instruction clears the screen from the current cursor position to the bottom of the screen. If *<n>* = 1, the command clears the screen from the current cursor position to the top of the screen, and if *<n>* = 2, the command clears the whole screen. |
| *\e[<n>K* | Deletes part of a line; if *<n>* = 0 or is missing, the instruction deletes from the cursor to the end of the line. If *<n>* = 1, the instruction deletes from the current cursor position to the start of the line, and if *<n>* = 2, it deletes the whole line. |
| *\e[<n>S* | Scrolls *<n>* lines up; new lines appear below (defaults to 1 if you do not specify *<n>*) |
| *\e[<n>T* | Scrolls *<n>* lines down; new lines appear above (defaults to 1 if you do not specify *<n>*) |
| *\e[s* | Stores the cursor position. |
| *\e[u* | Restores a stored cursor position. |

**INFO**

[1] Command Line: environmental variables: *http://www.linuxpromagazine.com/issue/80*

[2] Command Line: Paint Your Bash: *http://www.linuxpromagazine.com/issue/83*

**THE AUTHOR**

Heike Jurzik studied German, Computer Science and English at the University of Cologne, Germany. She discovered Linux in 1996 and has been fascinated with the scope of the Linux command line ever since. In her leisure time you might find Heike hanging out at Irish folk sessions or visiting Ireland.