Tip: Prompt magic

Enhancing the system prompt

Daniel Robbins

September 01, 2000

Why stick with the standard boring shell prompt when you can easily make it colorful and more informative? In this tip, Daniel Robbins will show you how to get your shell prompt just the way you like it, as well as how to dynamically update your X terminal's title bar.

As Linux/UNIX people, we spend a lot of time working in the shell, and in many cases, this is what we have staring back at us:

```
bash-2.04$
```

If you happen to be root, you're entitled to the "prestige" version of this beautiful prompt:

```
bash-2.04#
```

These prompts are not exactly pretty. It's no wonder that several Linux distributions have upgraded their default prompts that add color and additional information to boot. However, even if you happen to have a modern distribution that comes with a nice, colorful prompt, it may not be perfect. Maybe you'd like to add or change some colors, or add (or remove) information from the prompt itself. It isn't hard to design your own colorized, tricked-out prompt from scratch.

Prompt basics

Under bash, you can set your prompt by changing the value of the PS1 environment variable, as follows:

```
$ export PS1="> "
>
```

Changes take effect immediately, and can be made permanent by placing the "export" definition in your ~/.bashrc file. PS1 can contain any amount of plain text that you'd like:

```
$ export PS1="This is my super prompt > "
This is my super prompt >
```

While this is, um, interesting, it's not exactly useful to have a prompt that contains lots of static text. Most custom prompts contain information like the current username, working directory, or hostname. These tidbits of information can help you to navigate in your shell universe. For example, the following prompt will display your username and hostname:

```
$ export PS1="\u@\H > "
drobbins@freebox >
```

This prompt is especially handy for people who log in to various machines under various, differently-named accounts, since it acts as a reminder of what machine you're actually on and what privileges you currently have.

In the above example, we told bash to insert the username and hostname into the prompt by using special backslash-escaped character sequences that bash replaces with specific values when they appear in the PS1 variable. We used the sequences "\u" (for username) and "\H" (for the first part of the hostname). Here's a complete list of all special sequences that bash recognizes (you can find this list in the bash man page, in the "PROMPTING" section):

Sequence	Description
\a	The ASCII bell character (you can also type \007)
\d	Date in "Wed Sep 06" format
\e	ASCII escape character (you can also type \033)
\h	First part of hostname (such as "mybox")
\H	Full hostname (such as "mybox.mydomain.com")
\j	The number of processes you've suspended in this shell by hitting ^Z
\l	The name of the shell's terminal device (such as "ttyp4")
\n	Newline
\r	Carriage return
\s	The name of the shell executable (such as "bash")
\t	Time in 24-hour format (such as "23:01:01")
\T	Time in 12-hour format (such as "11:01:01")
\@	Time in 12-hour format with am/pm
\u	Your username
\v	Version of bash (such as 2.04)
\V	Bash version, including patchlevel
\w	Current working directory (such as "/home/drobbins")
\W	The "basename" of the current working directory (such as "drobbins")
\!	Current command's position in the history buffer
\#	Command number (this will count up at each prompt, as long as you type something)
\$	If you are not root, inserts a "$"; if you are root, you get a "#"
\xxx	Inserts an ASCII character based on three-digit number xxx (replace unused digits with zeros, such as "\007")
\\	A backslash
\[	This sequence should appear before a sequence of characters that don't move the cursor (like color escape sequences). This allows bash to calculate word wrapping correctly.
\]	This sequence should appear after a sequence of non-printing characters.

So, there you have all of bash's special backslashed escape sequences. Play around with them for a bit to get a feel for how they work. After you've done a little testing, it's time to add some color.

Colorization

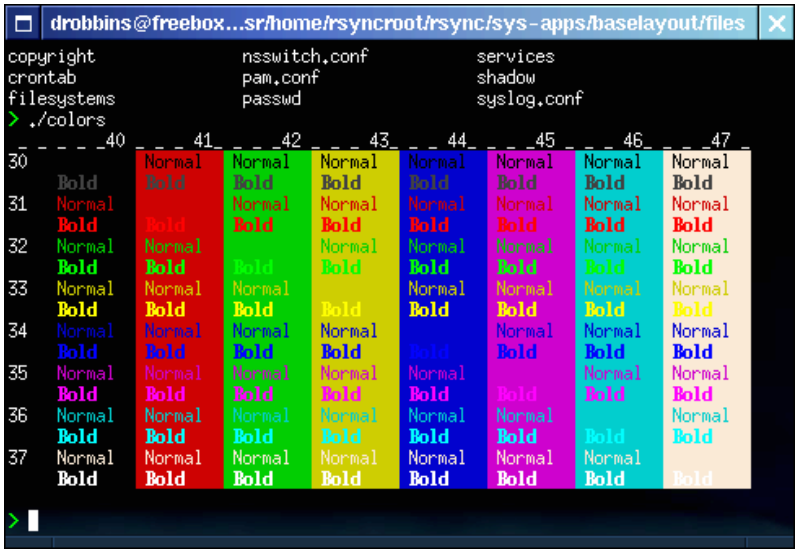
Adding color is quite easy; the first step is to design a prompt without color. Then, all we need to do is add special escape sequences that'll be recognized by the terminal (rather than bash) and cause it to display certain parts of the text in color. Standard Linux terminals and X terminals allow you to set the foreground (text) color and the background color, and also enable "bold" characters if so desired. We get eight colors to choose from.

Colors are selected by adding special sequences to PS1 -- basically sandwiching numeric values between a "\e[" (escape open-bracket) and an "m". If we specify more than one numeric code, we separate each code with a semicolon. Here's an example color code:

```
"\e[0m"
```

When we specify a zero as a numeric code, it tells the terminal to reset foreground, background, and boldness settings to their default values. You'll want to use this code at the end of your prompt, so that the text that you type in is not colorized. Now, let's take a look at the color codes. Check out this screenshot:

Color chart



To use this chart, find the color you'd like to use, and find the corresponding foreground (30-37) and background (40-47) numbers. For example, if you like green on a normal black background, the numbers are 32 and 40. Then, take your prompt definition and add the appropriate color codes. This:

```
export PS1="\w> "
```

becomes:

```
export PS1="\e[32;40m\w> "
```

So far, so good, but it's not perfect yet. After bash prints the working directory, we need to set the color back to normal with a "\e[0m" sequence:

```
export PS1="\e[32;40m\w> \e[0m"
```

This definition will give you a nice, green prompt, but we still need to add a few finishing touches. We don't need to include the background color setting of 40, since that sets the background to black which is the default color anyway. Also, the green color is quite dim; we can fix this by adding

a "1" color code, which enables brighter, bold text. In addition to this change, we need to surround all non-printing characters with special bash escape sequences, "\[" and "\]". These sequences will tell bash that the enclosed characters don't take up any space on the line, which will allow word-wrapping to continue to work properly. Without them, you'll end up with a nice-looking prompt that will mess up the screen if you happen to type in a command that approaches the extreme right of the terminal. Here's our final prompt:

```
export PS1="\[\e[32;1m\]\w> \[\e[0m\]"
```

Don't be afraid to use several colors in the same prompt, like so:

```
export PS1="\[\e[36;1m\]\u@\[\e[32;1m\]\H> \[\e[0m\]"
```

Xterm fun

I've shown you how to add information and color to your prompt, but you can do even more. It's possible to add special codes to your prompt that will cause the title bar of your X terminal (such as rxvt or aterm) to be dynamically updated. All you need to do is add the following sequence to your PS1 prompt:

```
"\e]2;titlebar\a"
```

Simply replace the substring "titlebar" with the text that you'd like to have appear in your xterm's title bar, and you're all set! You don't need to use static text; you can also insert bash escape sequences into your titlebar. Check out this example, which places the username, hostname, and current working directory in the titlebar, as well as defining a short, bright green prompt:

```
export PS1="\[\e]2;\u@\H \w\a\e[32;1m\]>\[\e[0m\] "
```

This is the particular prompt that I'm using in the colortable screenshot, above. I love this prompt, because it puts all the information in the title bar rather than in the terminal where it limits how much can fit on a line. By the way, make sure you surround your titlebar sequence with "\[" and "\]", since as far as the terminal is concerned, this sequence is non-printing. The problem with putting lots of information in the title bar is that you will not be able to see info if you are using a non-graphical terminal, such as the system console. To fix this, you may want to add something like this to your .bashrc:

```
if [ "$TERM" = "linux" ]
then
 #we're on the system console or maybe telnetting in
 export PS1="\[\e[32;1m\]\u@\H > \[\e[0m\]"
else
 #we're not on the console, assume an xterm
 export PS1="\[\e]2;\u@\H \w\a\e[32;1m\]>\[\e[0m\] "
fi
```

This bash conditional statement will dynamically set your prompt based on your current terminal settings. For consistency, you'll want to configure your ~/.bash_profile so that it sources your ~/.bashrc on startup. Make sure the following line is in your ~/.bash_profile:

```
source ~/.bashrc
```

This way, you'll get the same prompt setting whether you start a login or non-login shell.

Well, there you have it. Now, have some fun and whip up some nifty colorized prompts!

Related topics

- rxvt is a great little xterm that happens to have a good amount of documentation related to escape sequences tucked in the "doc" directory included in the source tarball.
- aterm is another terminal program, based on rxvt. It supports several nice visual features, like transparency and tinting.
- bashish is a theme engine for all different kinds of terminals.