source: www.thegeekstuff.com

1. Display username, hostname and current working directory in the prompt

The PS1 in this example displays the following three information in the prompt:

- **\u** – Username
- **\h** – Hostname
- **\w** – Full path of the current working directory

```
-bash-3.2$ export PS1="\u@\h \w> "

                ramesh@dev-db ~> cd /etc/mail
                ramesh@dev-db /etc/mail>
```

2. Display current time in the prompt

In the PS1 environment variable, you can directly execute any Linux command, by specifying in the format $(linux_command). In the following example, the command $(date) is executed to display the current time inside the prompt.

```
ramesh@dev-db ~> export PS1="\u@\h [\$(date +%k:%M:%S)]> "

                ramesh@dev-db [11:09:56]>
```

You can also use \t to display the current time in the hh:mm:ss format as shown below:

```
ramesh@dev-db ~> export PS1="\u@\h [\t]> "
                ramesh@dev-db [12:42:55]>
```

You can also use \@ to display the current time in 12-hour am/pm format as shown below:

```
ramesh@dev-db ~> export PS1="[\@] \u@\h> "
                [04:12 PM] ramesh@dev-db>
```

3. Display output of any Linux command in the prompt

You can display output of any Linux command in the prompt. The following example displays three items separated by | (pipe) in the command prompt:

- **\!:** The history number of the command
- **\h:** hostname
- **$kernel_version:** The output of the uname -r command from $kernel_version variable
- **\$?:** Status of the last command

```
ramesh@dev-db ~> kernel_version=$(uname -r)
                ramesh@dev-db ~> export PS1="\!|\h|$kernel_version|\$?> "
                473|dev-db|2.6.25-14.fc9.i686|0>
```

4. Change foreground color of the prompt

Display prompt in blue color, along with username, host and current directory information

```
$ export PS1="\e[0;34m\u@\h \w> \e[m"
              [Note: This is for light blue prompt]


              $ export PS1="\e[1;34m\u@\h \w> \e[m"
              [Note: This is for dark blue prompt]
```

- **\e[** – Indicates the beginning of color prompt
- **x;ym** – Indicates color code. Use the color code values mentioned below.
- **\e[m** – indicates the end of color prompt

Color Code Table:

```
Black 0;30
              Blue 0;34
              Green 0;32
              Cyan 0;36
              Red 0;31
              Purple 0;35
              Brown 0;33
              [Note: Replace 0 with 1 for dark color]
```

Make the color change permanent by adding the following lines to .bash_profile or .bashrc

```
STARTCOLOR='\e[0;34m';
                ENDCOLOR="\e[0m"
```

```
              export PS1="$STARTCOLOR\u@\h \w> $ENDCOLOR"
```

5. Change background color of the prompt

Change the background color by specifying \e[{code}m in the PS1 prompt as shown below.

```
$ export PS1="\e[47m\u@\h \w> \e[m"
              [Note: This is for Light Gray background]
```

Combination of background and foreground

```
export PS1="\e[0;34m\e[47m\u@\h \w> \e[m"
              [Note: This is for Light Blue foreground and Light Gray background]
```

Add the following to the .bash_profile or .bashrc to make the above background and foreground color permanent.

```
STARTFGCOLOR='\e[0;34m';
              STARTBGCOLOR="\e[47m"
              ENDCOLOR="\e[0m"
              export PS1="$STARTFGCOLOR$STARTBGCOLOR\u@\h \w> $ENDCOLOR"
```

Play around by using the following background color and choose the one that suites your taste:

- \e[40m
- \e[41m

- \e[42m
- \e[43m
- \e[44m
- \e[45m
- \e[46m
- \e[47m

6. Display multiple colors in the prompt

You can also display multiple colors in the same prompt. Add the following function to .bash_profile

```
function prompt {
              local BLUE="\[\033[0;34m\]"
              local DARK_BLUE="\[\033[1;34m\]"
              local RED="\[\033[0;31m\]"
              local DARK_RED="\[\033[1;31m\]"
              local NO_COLOR="\[\033[0m\]"
              case $TERM in
              xterm*|rxvt*)
              TITLEBAR='\[\033]0;\u@\h:\w\007\]'
              ;;
              *)
              TITLEBAR=""
              ;;
              esac
              PS1="\u@\h [\t]> "
              PS1="${TITLEBAR}\
              $BLUE\u@\h $RED[\t]>$NO_COLOR "
              PS2='continue-> '
              PS4='$0.$LINENO+ '
              }
```

You can re-login for the changes to take effect or source the .bash_profile as shown below.

```
$. ./.bash_profile
              $ prompt
              ramesh@dev-db [13:02:13]>
```

7. Change the prompt color using tput

You can also change color of the PS1 prompt using tput as shown below:

```
$ export PS1="\[$(tput bold)$(tput setb 4)$(tput setaf 7)\]\u@\h:\w $ \[$(tput sgr0)\]"
```

tput Color Capabilities:

- **tput setab [1-7]** – Set a background color using ANSI escape
- **tput setb [1-7]** – Set a background color
- **tput setaf [1-7]** – Set a foreground color using ANSI escape
- **tput setf [1-7]** – Set a foreground color

tput Text Mode Capabilities:

- **tput bold** – Set bold mode
- **tput dim** – turn on half-bright mode
- **tput smul** – begin underline mode
- **tput rmul** – exit underline mode
- **tput rev** – Turn on reverse mode
- **tput smso** – Enter standout mode (bold on rxvt)
- **tput rmso** – Exit standout mode
- **tput sgr0** – Turn off all attributes

Color Code for tput:

- 0 – Black
- 1 – Red
- 2 – Green
- 3 – Yellow
- 4 – Blue
- 5 – Magenta
- 6 – Cyan
- 7 – White

8. Create your own prompt using the available codes for PS1 variable

Use the following codes and create your own personal PS1 Linux prompt that is functional and suites your taste. Which code from this list will be very helpful for daily use? Leave your comment and let me know what PS1 code you've used for your Linux prompt.

- **\a** an ASCII bell character (07)
- **\d** the date in "Weekday Month Date" format (e.g., "Tue May 26″)

- **\D{format}** – the format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required
- **\e** an ASCII escape character (033)
- **\h** the hostname up to the first part
- **\H** the hostname
- **\j** the number of jobs currently managed by the shell
- **\l** the basename of the shell's terminal device name
- **\n** newline
- **\r** carriage return
- **\s** the name of the shell, the basename of $0 (the portion following the final slash)
- **\t** the current time in 24-hour HH:MM:SS format
- **\T** the current time in 12-hour HH:MM:SS format
- **\@** the current time in 12-hour am/pm format
- **\A** the current time in 24-hour HH:MM format
- **\u** the username of the current user
- **\v** the version of bash (e.g., 2.00)
- **\V** the release of bash, version + patch level (e.g., 2.00.0)
- **\w** the current working directory, with $HOME abbreviated with a tilde
- **\W** the basename of the current working directory, with $HOME abbreviated with a tilde
- **\!** the history number of this command
- **\#** the command number of this command
- **\$** if the effective UID is 0, a #, otherwise a $
- **\nnn** the character corresponding to the octal number nnn
- \\ a backslash
- **\[** begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
- **\]** end a sequence of non-printing character

9. Use bash shell function inside PS1 variable

You can also invoke a bash shell function in the PS1 as shown below.

```
ramesh@dev-db ~> function httpdcount {
              >  ps aux | grep httpd | grep -v grep | wc -l
              > }

              ramesh@dev-db ~> export PS1="\u@\h [`httpdcount`]> "
              ramesh@dev-db [12]>
              [Note: This displays the total number of running httpd processes]
```

You can add the following line to .bash_profile or .bashrc to make this change permanent:

```
function httpdcount {
              ps aux | grep httpd | grep -v grep | wc -l
              }
              export PS1='\u@\h [`httpdcount`]> '
```

10. Use shell script inside PS1 variable

You can also invoke a shell script inside the PS1 variable. In the example below, the ~/bin/totalfilesize.sh, which calculates the total filesize of the current directory, is invoked inside the PS1 variable.

```
ramesh@dev-db ~> cat ~/bin/totalfilesize.sh
                for filesize in $(ls -l . | grep "^-" | awk '{print $5}')
```

```
  do
  let totalsize=$totalsize+$filesize
  done
  echo -n "$totalsize"

  ramesh@dev-db ~> export PATH=$PATH:~/bin
  ramesh@dev-db ~> export PS1="\u@\h [\$(totalfilesize.sh) bytes]> "
  ramesh@dev-db [534 bytes]> cd /etc/mail
  ramesh@dev-db [167997 bytes]>
[Note: This executes the totalfilesize.sh to display the total

    file size of the current directory in the PS1 prompt]
```