navigation
- Main page
- Getting Involved
- Wiki News
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

page discussion view source history

Color Bash Prompt

i18n

Dansk – **English** – Español – Français – Indonesia – Italiano – Lietuviškai – Magyar – Nederlands – Polski – Português – Română – Slovenský – Suomi – Svenska – Türkçe – Česky – Ελληνικά – Български – Русский – Српски – Українська – עברית – ไทย – 日本語 – 正體中文 – 简体中文 – 한국어

There are a variety of possibilities for Bash's prompt (PS1), and customizing it can help you be more productive at the command line. You can add additional information to your prompt, or you can simply add color to it to make the prompt stand out.

Contents [hide]

Basic prompts

The following settings are useful for distinguishing the root prompt from non-root users.

- Edit Bash's personal configuration file:

```
$ nano ~/.bashrc
```

- Comment out the default prompt:

```
#PS1='[\u@\h \W]\$ '
```

- Add the following green prompt for regular users:

```
[chiri@zetsubou ~]$ _
```

```
PS1='\[\e[1;32m\][\u@\h \W]\$\[\e[0m\] '
```

- Edit root's .bashrc file; copy it from /etc/skel if the file is not present:

```
# nano /root/.bashrc
```

- Assign a red prompt for root:

```
[root@zetsubou ~]# _
```

```
PS1='\[\e[1;31m\][\u@\h \W]\$\[\e[0m\] '
```

Slightly fancier prompts

- A green and blue prompt for regular users:

```
chiri ~/docs $ echo "sample output text"
sample output text
chiri ~/docs $ _
```

```
PS1='\[\e[0;32m\]\u\[\e[m\] \[\e[1;34m\]\w\[\e[m\] \[\e[1;32m\]\$\[\e[m\] \[\e[1;37m\]'
```

This will give a very pleasing, colorful prompt and theme for the console with bright white text.

The string above contains color-set escape sequences (start coloring: \[\e[color\], end coloring: \[\e[m\]) and information placeholders:

- \u - Username. The original prompt also has \h, which prints the host name.
- \w - Current absolute path. Use \W for current relative path.
- \$ - The prompt character (eg. '#' for root, '$' for regular users).

The last color-set sequence, "\[\e[1;37m\]", is not closed, so the remaining text (everything typed into the terminal, program output and so on) will be in that (bright white) color. It may be desirable to change this color, or to delete the last escape sequence in order to leave the actual output in unaltered color.

- A red and blue prompt for root:

```
root ~/docs # echo "sample output text"
sample output text
root ~/docs # _
```

```
PS1='\[\e[0;31m\]\u\[\e[m\] \[\e[1;34m\]\w\[\e[m\] \[\e[0;31m\]\$ \[\e[m\]\[\e[0;32m\]'
```

This will give you a red designation and green console text.

Advanced prompts

Load/Mem Status for 256colors

This isn't even pushing the limits.. other than using 'sed' to parse the memory and load average (using the *-u* option for non-buffering), and the builtin *history* to save your history to your *HISTFILE* after every command, which I have found incredibly useful when dealing with crashing shells or subshells, this is essentially just making BASH print variables it already knows, making this extremely fast compared to prompts with non-builtin commands.

This prompt is from AskApache.com's BASH Power Prompt article, which goes into greater detail. It is especially helpful for those wanting to understand 256 color terminals, ncurses, termcap, and terminfo.

This is for **256 color terminals**, which is where the **\033[38;5;22m** terminal escapes come from.

```
802/1024MB 1.28 1.20 1.13 3/94 18563
[5416:17880 0:70] 05:35:50 Wed Apr 21 [srot@host.sqpt.net:/dev/pts/0 +1] ~
(1:70)$ _
```

```
PROMPT_COMMAND='history -a;echo -en "\033[m\033[38;5;2m"$(( `sed -nu "s/MemFree:[\t ]\+\([0-9]\+\) kB/\1
PS1='\[\e[m\n\e[1;30m\][$$:$PPID \j:\!\[\e[1;30m\]]\[\e[0;36m\] \T \d \[\e[1;30m\][\[\e[1;34m\]\u@\H\[\e
```

List of colors for prompt and Bash

Add this to your Bash file(s) to define colors for prompt and commands:

```
txtblk='\e[0;30m' # Black - Regular
txtred='\e[0;31m' # Red
txtgrn='\e[0;32m' # Green
txtylw='\e[0;33m' # Yellow
txtblu='\e[0;34m' # Blue
txtpur='\e[0;35m' # Purple
txtcyn='\e[0;36m' # Cyan
txtwht='\e[0;37m' # White
bldblk='\e[1;30m' # Black - Bold
bldred='\e[1;31m' # Red
bldgrn='\e[1;32m' # Green
bldylw='\e[1;33m' # Yellow
bldblu='\e[1;34m' # Blue
bldpur='\e[1;35m' # Purple
bldcyn='\e[1;36m' # Cyan
bldwht='\e[1;37m' # White
unkblk='\e[4;30m' # Black - Underline
undred='\e[4;31m' # Red
undgrn='\e[4;32m' # Green
undylw='\e[4;33m' # Yellow
undblu='\e[4;34m' # Blue
undpur='\e[4;35m' # Purple
undcyn='\e[4;36m' # Cyan
undwht='\e[4;37m' # White
bakblk='\e[40m'   # Black - Background
bakred='\e[41m'   # Red
badgrn='\e[42m'   # Green
bakylw='\e[43m'   # Yellow
bakblu='\e[44m'   # Blue
bakpur='\e[45m'   # Purple
bakcyn='\e[46m'   # Cyan
bakwht='\e[47m'   # White
txtrst='\e[0m'    # Text Reset
```

To use in commands from your shell environment:

```
$ echo -e "${txtblu}test"
test
$ echo -e "${bldblu}test"
test
$ echo -e "${undblu}test"
test
$ echo -e "${bakblu}test"
test
```

To use in a prompt (note double quotes and \[\] used by the shell to count proper length):

```
PS1="\[$txtblu\]foo\[$txtred\] bar\[$txtrst\] baz : "
```

Prompt escapes

The various Bash prompt escapes listed in the manpage:

```
Bash allows these prompt strings to be customized by inserting a
number of backslash-escaped special characters that are
decoded as follows:

  \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 `.'
  \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 characters

  The command number and the history number are usually different:
  the history number of a command is its position in the history
  list, which may include commands restored from the history file
  (see HISTORY below), while the command number is the position in
  the sequence of commands executed during the current shell session.
  After the string is decoded, it is expanded via parameter
  expansion, command substitution, arithmetic expansion, and quote
  removal, subject to the value of the promptvars shell option (see
  the description of the shopt command under SHELL BUILTIN COMMANDS
  below).
```

Positioning the cursor

The following sequence sets the cursor position:

```
\[\033[<row>;<column>f\]
```

The current cursor position can be saved using:

```
\[\033[s\]
```

To restore a position, use the following sequence:

```
\[\033[u\]
```

The following example uses these sequences to display the time in the upper right corner:

```
PS1=">\[\033[s\]\[\033[1;\$((COLUMNS-4))f\]\$(date +%H:%M)\[\033[u\]"
```

The environment variable *COLUMNS* contains the number of columns of the terminal. The above example substracts 4 from its value in order to justify the five character wide output of *date* at the right border.

Return value visualisation

WARNING WARNING WARNING WARNING
This seemed to me to have some bugs, see the entry I added to the discussion page.

Add this line if you want to see the return value of the last executed command. This should work with any kind of prompt as long as it does not need PROMPT_COMMAND:

```
PROMPT_COMMAND='RET=$?; if [[ $RET -eq 0 ]]; then echo -ne "\033[0;32m$RET\033[0m ;)"; else echo -ne "\03
```

It will look like this:

```
0 ;) harvie@harvie-ntb ~/ $ true
0 ;) harvie@harvie-ntb ~/ $ false
1 ;( harvie@harvie-ntb ~/ $
```

Zero is green and non-zero is red. There is also the smiley indication (replace it with anything you want); so your prompt will smile if the last operation was succesful.

Advanced return value visualisation

If you want colors, you need to set *$RED* and *$GREEN* values:

```
RED='\e[0;31m'
GREEN='\e[0;32m'
```

You have to specify these values in Bash's configuration files:

```
#return value visualisation
PROMPT_COMMAND='RET=$?;'
RET_VALUE='$(echo $RET)' #Ret value not colorized - you can modify it.
RET_SMILEY='$(if [[ $RET = 0 ]]; then echo -ne "\[$GREEN\];)"; else echo -ne "\[$RED\];("; fi;)'
```

Then you can use *$RET_VALUE* and *$RET_SMILEY* variables in the prompt. Note that you need use double quotes:

```
#prompt
PS1="$RET_VALUE $RET_SMILEY : "
```

This will give you basic prompt:

```
0 ;) : true
0 ;) : false
1 ;( :
```

But you will probably want to use *$RET_VALUE* or *$RET_SMILEY* in your own prompt, like this:

```
PS1="\[$WHITE\]$RET_VALUE $RET_SMILEY \[$BLUE\]\u\[$RED\]@\[$EBLUE\]\h\[$WHITE\] \W \[$ERED\]\\$\[$EWHITE
```

Wolfman's

After reading through most of the Bash Prompt Howto, the author developed a color bash prompt that displays the last 25 characters of the current working directory. This prompt should work well on terminals with a black background. The following code goes in file `~/.bashrc`.

- Add the bash_prompt_command function. If you have a couple directories with long names or start entering a lot of subdirectories, this function will keep the command prompt from wrapping around the screen by displaying at most the last pwdmaxlen characters from the PWD. This code was taken from the *Bash Prompt Howto*'s section on Controlling the Size and Appearance of $PWD and modified to replace the user's home directory with a tilde.

```
################################################
# Fancy PWD display function
################################################
# The home directory (HOME) is replaced with a ~
# The last pwdmaxlen characters of the PWD are displayed
# Leading partial directory names are striped off
# /home/me/stuff          -> ~/stuff               if USER=me
# /usr/share/big_dir_name -> ../share/big_dir_name if pwdmaxlen=20
################################################
bash_prompt_command() {
    # How many characters of the $PWD should be kept
    local pwdmaxlen=25
    # Indicate that there has been dir truncation
    local trunc_symbol=".."
    local dir=${PWD##*/}
    pwdmaxlen=$(( ( pwdmaxlen < ${#dir} ) ? ${#dir} : pwdmaxlen ))
    NEW_PWD=${PWD/#$HOME/\~}
    local pwdoffset=$(( ${#NEW_PWD} - pwdmaxlen ))
    if [ ${pwdoffset} -gt "0" ]
    then
        NEW_PWD=${NEW_PWD:$pwdoffset:$pwdmaxlen}
        NEW_PWD=${trunc_symbol}/${NEW_PWD#*/}
    fi
}
```

- The next fragment generates the command prompt and various colors are defined. The user's color for the username, hostname, and prompt ($ or #) is set to cyan, and if the user is root (root's UID is always 0), set the color to red. The command prompt is set to a colored version of Arch's default with the NEW_PWD from the last function.

  Also, make sure that your color variables are enclosed in double and not single quote marks. Using single quote marks seems to give Bash problems with line wrapping correctly.

```
bash_prompt() {
    case $TERM in
     xterm*|rxvt*)
         local TITLEBAR='\[\033]0;\u:${NEW_PWD}\007\]'
          ;;
     *)
         local TITLEBAR=""
          ;;
    esac
    local NONE="\[\033[0m\]"    # unsets color to term's fg color

    # regular colors
    local K="\[\033[0;30m\]"    # black
    local R="\[\033[0;31m\]"    # red
    local G="\[\033[0;32m\]"    # green
    local Y="\[\033[0;33m\]"    # yellow
    local B="\[\033[0;34m\]"    # blue
    local M="\[\033[0;35m\]"    # magenta
    local C="\[\033[0;36m\]"    # cyan
    local W="\[\033[0;37m\]"    # white

    # emphasized (bolded) colors
    local EMK="\[\033[1;30m\]"
    local EMR="\[\033[1;31m\]"
    local EMG="\[\033[1;32m\]"
    local EMY="\[\033[1;33m\]"
    local EMB="\[\033[1;34m\]"
    local EMM="\[\033[1;35m\]"
    local EMC="\[\033[1;36m\]"
    local EMW="\[\033[1;37m\]"

    # background colors
    local BGK="\[\033[40m\]"
    local BGR="\[\033[41m\]"
    local BGG="\[\033[42m\]"
    local BGY="\[\033[43m\]"
    local BGB="\[\033[44m\]"
    local BGM="\[\033[45m\]"
    local BGC="\[\033[46m\]"
    local BGW="\[\033[47m\]"

    local UC=$W                 # user's color
    [ $UID -eq "0" ] && UC=$R   # root's color

    PS1="$TITLEBAR ${EMK}[${UC}\u${EMK}@${UC}\h ${EMB}\${NEW_PWD}${EMK}]${UC}\\$ ${NONE}"
    # without colors: PS1="[\u@\h \${NEW_PWD}]\\$ "
    # extra backslash in front of \$ to make bash colorize the prompt
}
```

- Finally, append this code. This ensures that the NEW_PWD variable will be updated when you cd somewhere else, and it sets the PS1 variable, which contains the command prompt.

```
PROMPT_COMMAND=bash_prompt_command
bash_prompt
unset bash_prompt
```

KitchM's

These prompts offer a little more flash and visual clarity. Note that the use of red in the root user's prompt should provide ample warning. That's not to say someone couldn't use flashing text or arrow to do even more, but these will give you a good starting point.

First, change the default background in your terminal preferences (this example uses Xfce's Terminal program) to #D2D2D2, and the text color

to #000000. The font is listed as DejaVu Sans Mono Book 12. The cursor color is #00AA00, and the tab activity color is #AF0000.

Second, in ~/.bashrc and right after the PS1= line, enter a new line with the following:

```
PS1='\e[1;33;47m\u \e[1;32;47mon \h \e[1;35;47m\d \@\e[0;0m\n\e[1;34m[dir.= \w] \# > \e[0;0m'
```

And then place a # in front of the first PS1 line to remark it out.

Third, for root user, edit /root/.bashrc in the same manner to include:

```
PS1='\e[1;31;47m\u \e[1;32;47mon \h \e[1;35;47m\d \@\e[0;0m\n\e[1;31m[dir.= \w] \# > \e[0;0m'
```

Don't forget to comment out the old line.

Set xterm window title

You can use the `${XTERM_TITLE}` variable at the beginning of the prompt to set xterm title (if available) to directory@user@hostname. Or you can modify it using regular prompt escape sequences:

```
#set xterm title
case "$TERM" in
  xterm | xterm-color)
    XTERM_TITLE='\[\033]0;\W@\u@\H\007\]'
  ;;
esac;
```

Different colors for text entry and console output

If you don't reset the text color at the end of your prompt, both the text you enter and the console text will simply stay in that color. If you want to edit text in a special color but still use the default color for command output, you'll need to reset the color after you press enter, but still before any commands get run. To do this, install a DEBUG trap in your ~/.bashrc like this:

```
trap 'echo -ne "\e[0m" > $(tty)' DEBUG
```

And add yourself to the tty group:

```
sudo gpasswd -add <username> tty
```

Now bash will execute the DEBUG trap every time before running a command. We have to echo the reset sequence directly to the terminal device (as reported by tty), otherwise the sequence would become part of the command output (and thus part of any redirected data). To be able to write to the terminal device we need to be part of the tty group.

This is a bit of a hack, so use it at your own risk.

External links

- Forum Discussions:
  - BASH prompt
  - What's your PS1?
- http://www.tldp.org/HOWTO/Bash-Prompt-HOWTO/x329.html
- Giles Orr's collection of sample prompts

Categories: Eye candy (English) | Command shells (English) | HOWTOs (English)