OS X Command Line Interface Tips - Customizing the Bash Shell

Posted At : October 28, 2009 1:56 PM | Posted By : Bob Silverberg
Related Categories: Git, OS X, bash

I've been working on a Mac for around four months now, and never really had much reason to open up Terminal and use the command line interface (CLI), other than for starting and stopping Tomcat. Now that I've started trying to learn about Git, I'm using the command line more and more, and finding out new stuff daily, so I'm going to write the occasional post to share some of this info.

About Bash

The operating system that most of us run on Macs is called OS X, and it's based on Unix. The way that one interacts with Unix is via a command shell, and the default shell for OS X is called Bash. It allows us to interact with our operating system without going through the graphical user interface (GUI) that sits on top of the OS. According to Wikipedia Bash stands for *Bourne-again shell* as it is a successor to the Bourne shell. So, when you open up Terminal, or iTerm which is an enhanced Terminal alternative, you are interacting with the Bash shell.

Customizing the Shell

One thing that I picked up while toying with Git was the fact that you can customize the behaviour and appearance of the shell by placing entries into a file called .bash_profile in your home directory. To do this, open Terminal, at which point you should be in your home directory and type:

```
nano .bash_profile
```

This will open up the file .bash_profile in *nano*, a text editor accessible from the shell. *vi* is another text editor that you can use from the command line, but I find nano to be a bit easier to use as a noob, as it displays the shortcut keys at the bottom of the screen. If you do not already have a .bash_profile file in your home directory it will create one when you save your changes. Just type the commands to customize your shell into the file, then use ctrl-O to save the file followed by ctrl-X to exit from nano. After making changes to .bash_profile you need to open a new Terminal session for the changes to take affect.

Customizing History

You may have noticed that you can press the up-arrow button to bring back the previous command that you typed into the terminal, and that you can keep pressing the up-arrow to bring back more commands. There is a file in your home directory called .bash_history which records all of your commands so they can be used again. One thing that I found annoying is that if I use the same commands repeatedly they keep getting added to my history, so that if I want to get back to an older command I have to up-arrow many times to get to the command I want. There are a number of ways of recalling a command more directly than using the up-arrow, which I hope to cover in a future blog post, but for now I want to point out a .bash_profile customization that can help. Adding the following line to your .bash_profile:

```
export HISTCONTROL=erasedups
```

will eliminate any duplicate entries from getting added to your history, so now, when you use the up-arrow, you'll only get one instance of a command back, not the same command over and over again. You can also control the number of entries stored in your history by using HISTSIZE. For example, to store 10,000 entries in your history, add this line to .bash_profile:

```
export HISTSIZE=10000
```

Using Aliases for Commands and Directories

You can create shortcuts to commonly used commands by adding an alias for the command to your .bash_profile. Here are some examples:

```
alias gs='git status'

alias gr='cd /Users/robertsilverberg/Documents/gitRepos'

alias hm='cd /Users/robertsilverberg'
```

With those aliases in place, I can see the status of my Git repo by simply typing "gs", I can switch to the directory that holds all of my Git repos by typing "gr", and I can switch back to my home directory by typing "hm". I'm sure you can think of lots of other shortcuts that you could define.

Customizing Your Prompt

By default, on my machine anyway, my shell prompt displays my hostname followed by a colon, then the current working directory followed by a space, and then my username followed by a dollar sign. So when I first open up terminal I see a prompt that looks like:

```
Macintosh:~ robertsilverberg$
```

This can be customized in a number of ways, and this was brought to my attention via a tweet by John Benyon. I saw his example and decided that I needed to figure out how to do that. Thanks to our friends at Google, I found a whole bunch of posts about customizing the shell prompt, including some specific to adding the current Git branch to your prompt. I ended up using some code from a .bash_profile file that I found in a repository on GitHub itself. It's not exactly the way I'd like the prompt to look, but it's close, and it includes not only the name of the current branch of the Git repo, but also an indicator of whether there are uncommitted files in the repo. Here's the code:

```
RED="\[\033[0;31m\]"

     YELLOW="\[\033[0;33m\]"

    GREEN="\[\033[0;32m\]"

       BLUE="\[\033[0;34m\]"

  LIGHT_RED="\[\033[1;31m\]"

LIGHT_GREEN="\[\033[1;32m\]"

      WHITE="\[\033[1;37m\]"
```

```
 LIGHT_GRAY="\[\033[0;37m\]"

 COLOR_NONE="\[\e[0m\]"

       GRAY="\[\033[1;30m\]"



function parse_git_dirty {

  [[ $(git status 2> /dev/null | tail -n1) != "nothing to commit (working directory clean)" ]] && echo "*"

}

function parse_git_branch {

  git branch --no-color 2> /dev/null | sed -e '/^[^*]/d' -e "s/* \(.*\)/[\1$(parse_git_dirty)]/"

}



function set_prompt {

  export PS1="${COLOR_NONE}\w${RED}$(parse_git_branch)${COLOR_NONE}$ "

}

PROMPT_COMMAND='set_prompt'
```

Notice that the author made it very easy to change colours by creating a bunch of colour constants in the file. Try adding that code to your .bash_profile and see how your prompt changes. If you don't like it, don't worry. Just remove the code from .bash_profile and you'll have your old prompt back.

I hope that some other Mac/Shell noobs benefit from this. I'm going to try to continue to post about useful CLI stuff that I find.