Bash Shortcuts For Maximum Productivity

15/09/2009 · 972 words · 5 min read

Table of Contents

Maximum

It may or may not surprise you to know that **the *bash* shell has a very rich array of convenient shortcuts** that can make your life, working with the command line, a whole lot easier. This ability to edit the command line using shortcuts is provided by the GNU Readline library. This library is used by many other **nix* application besides *bash*, so learning some of these shortcuts will not only allow you to zip around *bash* commands with absurd ease :), but can also make you more proficient in using a variety of other **nix* applications that use Readline. I don't want to get into Readline too deeply so I'll just mention one more thing. By default Readline uses *emacs* key bindings, although it can be configured to use the *vi* editing mode, I however prefer to learn the default behavior of most applications (I find it makes my life easier not having to constantly customize stuff). If you're familiar with *emacs* then many of these shortcuts will not be new to you, so these are mostly for the rest of us :).

Command Editing Shortcuts

- **Ctrl + a –** go to the start of the command line
- **Ctrl + e –** go to the end of the command line
- **Ctrl + k –** delete from cursor to the end of the command line
- **Ctrl + u –** delete from cursor to the start of the command line
- **Ctrl + w –** delete from cursor to start of word (i.e. delete backwards one word)
- **Ctrl + y –** paste word or text that was cut using one of the deletion shortcuts (such as the one above) after the cursor
- **Ctrl + xx –** move between start of command line and current cursor position (and back again)
- **Alt + b –** move backward one word (or go to start of word the cursor is currently on)
- **Alt + f –** move forward one word (or go to end of word the cursor is currently on)
- **Alt + d –** delete to end of word starting at cursor (whole word if cursor is at the beginning of word)
- **Alt + c –** capitalize to end of word starting at cursor (whole word if cursor is at the beginning of word)
- **Alt + u –** make uppercase from cursor to end of word
- **Alt + l –** make lowercase from cursor to end of word
- **Alt + t –** swap current word with previous
- **Ctrl + f –** move forward one character
- **Ctrl + b –** move backward one character
- **Ctrl + d –** delete character under the cursor
- **Ctrl + h –** delete character before the cursor
- **Ctrl + t –** swap character under cursor with the previous one

Command Recall Shortcuts

- **Ctrl + r –** search the history backwards
- **Ctrl + g –** escape from history searching mode
- **Ctrl + p –** previous command in history (i.e. walk back through the command history)
- **Ctrl + n –** next command in history (i.e. walk forward through the command history)
- **Alt + . –** use the last word of the previous command

Command Control Shortcuts

- **Ctrl + l –** clear the screen
- **Ctrl + s –** stops the output to the screen (for long running verbose command)
- **Ctrl + q –** allow output to the screen (if previously stopped using command above)
- **Ctrl + c –** terminate the command

- **Ctrl + z –** suspend/stop the command

Bash Bang (!) Commands

Bash also has some handy features that use the ! (bang) to allow you to do some funky stuff with *bash* commands.

- **!! –** run last command
- **!blah –** run the most recent command that starts with 'blah' (e.g. !ls)
- **!blah:p –** print out the command that **!blah** would run (also adds it as the latest command in the command history)
- **!$ –** the last word of the previous command (same as **Alt + .**)
- **!$:p –** print out the word that **!$** would substitute
- **!*** **–** the previous command except for the last word (e.g. if you type '_find some*file.txt* /', then **!*** would give you '_find some*file.txt*')
- **!*:p –** print out what **!*** would substitute

There is one more handy thing you can do. This involves using the ^^ 'command'. If you type a command and run it, you can re-run the same command but substitute a piece of text for another piece of text using ^^ e.g.:

```
$ ls -al
total 12
drwxrwxrwx+ 3 Administrator None    0 Jul 21 23:38 .
drwxrwxrwx+ 3 Administrator None    0 Jul 21 23:34 ..
-rwxr-xr-x  1 Administrator None 1150 Jul 21 23:34 .bash_profile
-rwxr-xr-x  1 Administrator None 3116 Jul 21 23:34 .bashrc
drwxr-xr-x+ 4 Administrator None    0 Jul 21 23:39 .gem
-rwxr-xr-x  1 Administrator None 1461 Jul 21 23:34 .inputrc
$ ^-al^-lash
ls -lash
total 12K
   0 drwxrwxrwx+ 3 Administrator None    0 Jul 21 23:38 .
   0 drwxrwxrwx+ 3 Administrator None    0 Jul 21 23:34 ..
4.0K -rwxr-xr-x  1 Administrator None 1.2K Jul 21 23:34 .bash_profile
4.0K -rwxr-xr-x  1 Administrator None 3.1K Jul 21 23:34 .bashrc
   0 drwxr-xr-x+ 4 Administrator None    0 Jul 21 23:39 .gem
4.0K -rwxr-xr-x  1 Administrator None 1.5K Jul 21 23:34 .inputrc
```

Here, the command was the **^-al^-lash** which replaced the **–al** with **–lash** in our previous **ls** command and re-ran the command again.

There is lots, lots more that you can do when it comes to using shortcuts with *bash*. But, the shortcuts above will get you 90% of the way towards maximum *bash* productivity. If you think that I have missed out on an ***essential** bash* shortcut that you **can't live without** (I am sure I have), then please let me know and I'll update the post. As usual, feel free to subscribe to my feed for more tips and opinions on all things software development.

Image by djhsilver

#bash shell #coding #shortcuts

< Ruby Equality And Object Comparison The Secret Of Being A Great Apprentice >

Show Disqus Comments