Help Topic: Command Line Aliases

Spring 2024

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

Augmenting the Power of the Command Line with Aliases

Aliases may be the single most powerful, and simplest tool available for command line users. They also allow you to personalize your work flow. They allow you to save time on simple things that you may do very often. For example, to change directories to the directory above the current one, you normally type:

```
$ cd ..
```

No problem, but if do this a lot, this can be simplified by making an alias:

```
$ alias cdd='cd ..'
```

Once this is defined, I can change directories a bit more efficiently with:

```
$ cdd
```

Add aliases to your shell configuration to make them persistent

Although aliases can be defined on the fly on the command line as above, the drawback is that they do not carry over to other shell sessions in another window or at a later date after logging back in. To make more persistent aliases, add them to your shell configuration file. The above example alias can be achieved with the following line in your `.bashrc` file as in:

```
alias cdd='cd ..'
```

Suggestions for useful aliases

Below are some suggestions offered up here partly from firm belief of best practices, and partly based on what I found to work for me over time. In the category of best practices, I recommend altering your `rm`, `mv` and `cp` shell commands to work in the interactive mode:

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

In my opinion, any time you're about to potentially overwrite or remove a file, a prompt should be given to the user. You can always override this by using `-f` on the command line which will override the `-i`. You might have these already if you took the class example shell configuration files. Some others:

```
alias    cdd='cd ..'
alias   cddd='cd ../..'
alias  cdddd='cd ../../..'
alias cddddd='cd ../../../..'
```

The above should be self-explanatory. Try it. The need to change up a directory or three comes up a lot!

```
alias hemacs='/Applications/Emacs.app/Contents/MacOS/Emacs -bg DarkOliveGreen -fg white'
```

This emacs alias explicity invokes the graphical emacs (if you installed it from emacsformacosx.com) rather than the one in `/usr/bin/emacs`. It also sets my background and foreground color the way I like them.

```
alias duh1='du -d 1 -h'               (in OS X)
alias duh1='du --max-depth=1 -h'      (in GNU/Linux)
```

The du command is super useful. It estimates the file space usage for the current directory and each subdirectory. The command line options are slightly different in GNU/Linux and OS X as shown above. By setting the depth (OSX) or max-depth (GNU/Linux) to one, this limits the depth of reporting subdirectories to just the present directory and subdirectories. Otherwise it will recursively report everything, which I find is too much. The first time you fill up your hard drive with a runaway log file, you will find this command indespensible.

```
alias  cdmi='cd moos-ivp'
alias cdmim='cd moos-ivp/missions'
alias cdmis='cd moos-ivp/src'
```

In the example above the moos-ivp aliases are not what's important. The general idea is to give yourself meaningful aliases for jumping through your own file structure. Probaly 70% or more of my aliases are of this type, and may account for most of why I think using the command line is way more efficient than anything else. My "cd aliases" all begin with "cd" and usually what follows is a single letter for each directory in the path, or some heuristic along these lines. If I look over a student's shoulder midway through a course and I see them typing "cd ../../project12/missions" rather than something like "cdp12m", then a feeling of failure rushes over me, like I have failed to pass on something important. Embrace this capability.

```
alias runa='cd moos-ivp/missions/s1_alpha; pAntler alpha.moos'
alias runb='cd moos-ivp/missions/s1_bravo; pAntler bravo.moos'
```

As before, the important thing here is not the actual mission. The key point is that an alias can include *multiple* shell commands separated by a semicolon. When you're working on a particular autonomy problem, you may find yourself launching the same mission over and over again to test little changes. With the above style alias, you can launch the mission from wherever your shell happens to be in the file system. You can even add a timewarp component to launch the mission with time warp 12, for example, with an alias like `runa12`.

Temporarily disabling an alias

Sometimes you may want to temporarily disable an alias like:

```
alias rm='rm -i'
```

and return the `rm` command to the way it was before, without prompting for confirmation. You can do this by typing:

```
$ alias rm='rm'
. . .
```

Reminding yourself of previously defined aliases

If the power of aliases strikes you as it did me, you may find at some point that you have a lot of aliases defined. A quick check on my own computer just now confirms that I have over 850 aliases defined. People ask me how I can remember them all? Well, I don't. But I remember the important ones, and even the lesser used ones are still pretty useful when I come back to working on a piece of code or a paper I haven't touched in a while.

One way to remind yourself of previously defined aliases is to just look in the file where they are defined, e.g., `.bashrc`. But you know there has to be a better way. A slightly better way is to just type the following:

```
$ alias
alias cp='cp -i'
alias mv='mv -i'
alias rm='rm -i'
. . .
```

This will dump all your defined aliases in alphabetical order. At least you didn't have to open an editor to get this information, but the alphabetical order may actually make it harder to find what you're looking for. A better approach is to use `grep` to find what you're looking for. For example, you may remember that you had an alias related to launching emacs, so you could do this on the command line:

```
$ alias | grep emacs
alias hemacs='emacs --geometry=162x83+735+0 -bg DarkOliveGreen -fg white'
alias nem='emacs -nw'
```

This will find only only those aliases that have the word "emacs" as part of their definition. This is such a useful trick that I actually have an alias for that in my shell configuration file as in:

```
alias afind='alias | grep'
```

This can now be used instead to find all aliases with the word "emacs" in the definition. The search list can be further pruned to say only show aliases that have "emacs" and "geometry" in the definition:

```
$ afind emacs | grep geometry
alias hemacs='emacs --geometry=162x83+735+0 -bg DarkOliveGreen -fg white'
```