COMP 2718: Alias: The First Taste of Programming

By: Dr. Andrew Vardy

Adapted from the notes of Dr. Rod Byrne

Outline

- `alias` - Customizing Commands or Sequences of Commands
- Multiple Commands
- Listing Defined Aliases
- Removing an Alias
- Limitations

Alias — Chapter 5 of TLCL

We'll cover the `alias` command introduced in chapter 5 of the textbook. This is really the first taste of actual programming using the shell.

First, its useful to see how we can combine multiple commands on one line, just by separating them with semicolons:

```
command1; command2; command3...
```

These commands will be executed in order: `command1`, `command2`, `command3`.

alias - Customizing Commands or Sequences of Commands

In general, the alias command has the following form:

```
alias name='string'
```

This assigns a string to a name and we can then use that name in place of the string anytime the shell expects a command.

A simple example is to create a customized alias to an existing command. For example:

```
alias ll='ls -ltrFh'
```

The previous example can be changed to create an alias called `ls`.

```
alias ls='ls -ltrFh'
```

Note: Once executed the above command will redefine `ls` in the current shell, but as soon as that shell exits (e.g. the terminal window is closed) the alias is gone.

We can make the alias last by putting it in your `.bashrc` file which resides in your home directory. Edit `~/.bashrc` and add your own aliases to the end.

Note on Mac OS X: `~/.bashrc` is not read by default on a Mac. Instead you can put configuration such as aliases in `~/.profile`.

The following comman aliases make potentially dangerous remove, copy, and move operations interactive:

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

(But if you have these defined on your system and then you move to another system, you will be in for a shock!)

Multiple Commands

Alias is particularly useful for executing multiple commands to accelerate a common workflow. For example, the following three:

```
alias m='make'
alias mc='make clean'
alias mcm='make clean; make'
```

Actually, an alias can reference another existing alias, so we could do this instead:

```
alias m='make'
alias mc='m clean'
alias mcm='mc; m'
```

Listing Defined Aliases

You can list all defined aliases just by executing `alias` without arguments:

```
alias
```

Try this on your own system... Where do all of these aliases come from? Many are defined in the user's startup file `.bashrc` (or `.profile` on Mac OS X).

To see how a particular alias is defined, just enter `alias name`. For example:

```
$ alias ll
alias ll='ls -ltrFh'
```

Removing an Alias

Aliases can be removed with `unalias`:

```
unalias name
```

Just like adding an alias, removing an alias does not make a permanent change. If the alias was defined in a startup file (e.g. `.bashrc` or `.profile`) then it will come back to life whenever a new terminal is started.
You can also revert to the original non-aliased command by prepending with `\`. For example:

```
\ls
```

Limitations

`bash` aliases have a number of limitations

- They are not expanded recursively
- They cannot do anything with arguments but pass them along

For advanced uses, it is advisable to use **functions** (covered later).