STATS 769
Prospering in Linux

Paul Murrell

The University of Auckland

August 13, 2019

Overview

- In the last topic, we learned how to **survive** in the linux shell (how to do things we already do in other operating system environments)
- In this section of the course we will learn how to **prosper** in the shell (how to work better/smarter than we do in other operating system environments)

- NOTE that we are working on the filesystem rather than in RAM, so the consequences are more permanent (compared to running R code).

Globs and command substitution

- Filenames can contain "wildcards" that specify patterns.
- `*` matches any number of characters.
- `?` matches any one character.

- Anything within `$()` is executed first and the result is used

Escape sequences

- Space is important in shell commands
  (it is used between arguments)
- Use backslash (\) to escape a space
  (or a wildcard).
- Use double quotes (") to escape spaces and wildcards
  (but keep command substitution)
- Use single quotes (') to escape everything.

Redirection

- > redirects output from a program to a file
  (for programs that normally print output to the screen)
- >> redirects output from a program **and appends it** to a file.
- < redirects input to a program from a file
  (for programs that normally accept input from the keyboard)
- | "pipes" output from one program to another program.

Loops

```
for variable in list
do
    cmd $variable
done
```

- `list` is a space-separated list of one or more values (typically file names).
- `variable` is a shell variable that gets each value in `list`, one at a time.
- `cmd` is called for each value in `list`.

GNU tools

Every Linux distribution comes with a basic set of useful programs.

`wc`	Count lines, words, and characters in text files.
`grep`	(Regular expression) text search (across multiple files).
`find`	(Recursively) search for file names matching a pattern.
`tar`	Create compressed archives containing multiple files and directories.
`awk`	Text processor.
`make`	Build automation tool.

AWK

- awk implicitly loops over each line in a file and breaks each line into fields
- predefined variables for each field in the line
  - `$0` is the whole line, `$1` is field 1, `$2` field 2, ...
  - `NR` is current row, `NF` is num fields
- an awk program is a series of ...

```
condition { action }
```

- conditions are `A == B`, or `/regular expression/`, or `BEGIN`, or `END`
- most common action is `print()`

Shell scripts

- Put commands in a file, `cmd.sh`, then run `bash cmd.sh`
- Within script, `$1` refers to first argument in call (e.g., `arg1` in `bash cmd.sh arg1`)
- Within script, `$@` refers to all arguments in call.
- Can also change file to executable with `chmod` then run `./cmd.sh`

Make

- Create a `Makefile`.

  ```
  target: dependencies
      cmd
  ```

- `make target`
- `target` is only built if one of `dependencies` is newer than `target`.
- The can be more than one `cmd`.
- Both `target` and `dependencies` can be variables and patterns
  (`%.suffix`)
  and those patterns can be referred to in the commands
  ($< means the name of the first dependency,
  $* means the "stem" of the target)

Reading

- The Unix Shell (Sections 4, 5, 6 and 7)
  `http://swcarpentry.github.io/shell-novice/`
- Automation and Make
  `http://swcarpentry.github.io/make-novice/aio.html`