Unix fundamentals

Miles Erickson

January 13, 2017

Lecture Credits

- Ben Skrainka
- Jack Bennetto

Objectives

Morning objectives:

- Perform basic file operations from the command line
- Get help using man
- Configure environment in .bash_profile
- Manage a process with job control
- Write a simple regular expression
- Use grep/sed/awk/cut/paste to process/clean a text file
- Perform “survival” edits using vi

Agenda

Morning: Unix

- About operating systems and unix
- Using a shell
- Files and permissions
- Basic commands

Afternoon: Workflow and Learning

What is an Operating System?

Operating system: the collection of software that directs a computer's operations, controlling and scheduling the execution of other programs, and managing storage, input/output, and communication resources. (Dictionary.com)

Many pieces

- Kernel (CPU, memory, processes, users)
- Device drivers (keyboard, mouse, displays, . . .)
- File manager
- System processes
- User interface (shells and GUI)
- Standard programs

Windows and Unix

Windows (most desktops and laptops)

- Made by Microsoft
- Proprietary
- More GUI based

Unix

- Many versions, made by different companies, groups, or people (including Apple since 2002)
- Originally a commercial product
- Linux and FreeBSD are *open-source* versions of Unix
- More command-line based
- Small, simple commands
- Everything is a file

Processes

A process consists of

- One or more threads
- Program text
- Memory for stack and heap
- File descriptors
- Environment
- Owner and privilege

Shells

A shell is a command-line interface that interprets user input, runs programs, and output results.

- sh (Bourne Shell)
- ksh (Korn Shell)
- **bash** (Bourne-again shell)
- csh (C shell)
- tcsh
- zsh

All are also scripting languages.

(shell ! = terminal)

Redirection and Pipes

Most commands read from STDIN ("standard input") and write to STDOUT and STDERR.

- Redirect a file to STDIN with <
- Redirect STDOUT to a file with >
- Redirect STDERR to a file with 2>
- Append STDOUT to a file with >>
- Connect the STDOUT of one command with the STDIN of another with |

```
grep export < .bash_profile | sort > temp
```

File structure

Basic file commands

- `pwd` (print working directory)
- `ls` (list directory)
- `cd` (change directory)
- `pushd`, `popd` (occasionally useful)

Directories contain files and other directories, with names separated by /

Special directories:

- / root, the top-level directory
- ~ your home directory
- `.` the current directory
- `..` the parent directory

Absolute paths start with / (or ~). Relative paths don't.

Command-line Options

Most commands have can take options.

Usually

- single-character options are preceded by - and can be combined
- full-word options are preceded by -- Usually.

Examples:

- `python --version`
- `ls -l`
- `ls -la`

See `man` *command* for more details.

Permissions

Each file has read, write, and execute permissions for the owner, the group, and the world.

Examine with `ls -la`

```
drwxr-xr-x   7 jdoe  staff     238 Sep 11 21:21 .
drwxr-xr-x  54 jdoe  staff    1836 Sep 11 19:07 ..
-rw-r--r--   1 jdoe  staff    4025 Sep 11 21:21 unix.md
-rw-r--r--@  1 jdoe  staff  128995 Sep 11 21:05 unix.pdf
```

Set with `chmod`

Basic Commands

Most commands Help

- man (read `man man` for more information)
- info (probably not helpful for you)

Files and directories

- `cat` (output file(s) to STDOUT)
- `less` (page through files)
- `head` and `tail` (look at start and end of file)
- `mv` (move file or directory)
- `cp` (copy file or directory)
- `rm` (remove file or directory; use `-rf` *carefully* to remove directory)
- `mkdir` and `rmdir` (make and remove directory)
- `touch` (create empty file, or update timestamp)
- `diff` (compare files)

Common Filters

This can be passed a file, or read from STDIN

- `wc` (count lines/words/characters)
- `grep` or `egrep` (find lines that contain strings)
- `sort`
- `uniq` (combine consecutive identical lines; use `-c` to count)
- `tr` (transpose characters)
- `cut` (select specific columns of csv or tsv)
- `paste` (like SQL join; not that common)
- `sed`, `awk`, and `perl` (languages for file manipulation)

Regular Expressions

The `grep` command (and many languages) use regular expressions to match files.

Most characters match themselves. Some don't.

- `.` matches anything other than a newline
- `*` match zero or more of previous atom
- `+` match one or more of previous atom
- `|` match either previous or next item
- `[abc0-9]` match any of characters within
- `\` escape previous character
- `\(\)` for grouping (use egrep to use without `\`)

Other Useful Commands

- echo (echo to STDOUT)
- `find` (find files that match something and maybe do something with them)
- `ps` (list processes; `ps aux` lists all with more fields)
- kill (kills a process by pid; `kill -9` really kills it)
- tar (archive directories into a single file)

Job Control

Multiple jobs can run from a shell an once. Jobs can be

- stopped (paused)
- foreground (blocks other actions)
- background (can't read input but still prints)

Important commands

- jobs (list jobs)
- `&` (when added to the end of a command, starts job in the background)
- `fg` (puts a stopped/background job in foreground)
- `bg` (puts stopped job in background)
- `^Z` (stops foreground job)
- `^C` (kills foreground job)
- `kill %1` (kills job 1)

Environment Variables

List with `env`

Set with `VAR=foo`, or `export VAR=foo` when run from a script (e.g., .bash_profile)

Basic vi

Two main modes

- Insert mode (typing stuff inserts it in file)
- Command mode (each key has meaning)

The *escape* key brings you to command mode; *i* enters insert mode

Helpful commands

- i (enter insert mode)
- a (enter insert mode after this character)
- dd (delete a line)
- 100G (go to line 100)
- :wq (save and exit)
- :q! (exit without saving)

Editing .bash_profile

```
alias h='history'

alias l='ls -GFh'

alias ll='ls -lFGh'
```