CM1102
Introduction to Unix/Linux

Martin Caminada

Cardiff University

UNIX: the origins

UNIX history

- Original UNIX: 1970s AT&T Bell Labs
- UNIX clones: AIX, HPUX, SunOS, Xenix
- UNIX standards: BSD, System V, Posix
- Start of Open Source Movement
  - Free Software Foundation (GNU)
  - Minix (Andy Tanenbaum)
  - Linux (Linus Torvalds)

Multi User / Multi Tasking

Terminals and Terminal Emulators
How the Shell works

- the shell is an interactive environment for executing UNIX commands
- it is started up when logging into a terminal, or when opening a terminal window
- different kinds of shells: sh, tcsh, bash, *etc*
- basic idea: key in a command, wait for it to complete, key in another command, *etc*
- use the [TAB] key to auto-complete a command and the [UP] arrow to bring back a previous command

Some practical UNIX commands

ls	list contents of directory
cd	change directory
pwd	print working directory
cp	copy file
mv	move file
ln	create a new link to file
rm	remove file
mkdir	make a new directory
cat	concaternate file contents
man	view manual page

Using ln to create links

cp file1 file2

- creates a copy of file1 named file2
- file1 and file2 have the same contents but are different files

ln file1 file2

- creates a link of file1 named file2
- file1 and file2 point to the same file (so editing one will affect both)

Hard Links versus Symbolic Links

ln file1 file2

- creates a *hard* link
- both files will have the same inode (ls -i)
- works only within the same file system

ln -s file1 file2

- creates a *symbolic* link
- file2 will point to file1
  (creates problem when file1 is removed)
- works within the same of different filesystems

Users and File Permissions (1/3)

- each file has an *owner* and *group*
- file permissions (*read*, *write* and/or *execute*) are set for the owner, group and others
- examples (*ls -l*):

  `-rw-r-----`

  owner can read and write file,
  other group members can only read,
  all others have no access at all

  `-rwxr-x--x`

  owner can read, write and execute file,
  other group members can read and execute,
  all others can only execute

Users and File Permissions (2/3)

- for directories
  - *r* is the ability to see the contents (so to run ls)
  - *w* is the ability to alter the contents (add and remove files)
  - *x* is the ability to enter the directory and access its files

`drwxr-x--x`

owner can see contents, add/remove files and access files,
other group members can see contents and access files,
all others cannot see contents, and can only access files
of which they already know the name

Users and File Permissions (3/3)

- numerical file permissions: *r*=4, *w*=2, *x*=1
  chmod 751 myprogram
  yields permissions `-rwxr-x–x`

- *u*=user, *g*=group, *o*=others
  chmod u+w,g+w,o-r myfile
  turns `-r--r--r--` *into* `-rw-rw----`

- *umask 644*
  sets permissions to `-rw-r--r--` for new files

Users and File Permissions

- *chown kirsty myfile.txt*
  changes owner of myfile.txt to user kirsty
- *chgrp lecturers myfile.txt*
  changes group of myfile.txt to lecturers
- For the above two commands to work, you need superuser (*root*) privileges, so either:
  - login as root (not recommended)
  - change to root using *su* (also not recommended)
  - execute the command using *sudo*
    sudo chown kirsty myfile.txt
    (you might be asked for a password)

Redirected I/O

- *standard input* is normally read from the keyboard
- *standard output* is normally written to the screen
- it is possible to *redirect* standard input and and standard output from and to a file
- examples:

sort < somedata.txt (*sort* gets input from file)
ls -al > myfile.txt (*ls* writes output to file)
ls -al >> myfile.txt (*ls* appends output to file)
*ls -al *.gif 2> myfile.txt* (*ls* writes errors to file)
ls -al | *grep myfile.txt* (output of *ls* is input of *grep*)

cat myfile.txt | *tail -n 10* | *sort*

Shell Scripts

- the shell also provides a programming environment

```
for i in *.txt; do
  echo $i;
  cat $i;
done
```

- use quotation marks (“...” and ‘...’) to avoid expanding wildcards (like *.txt) and variable names (like $i)
- when writing a shell script, start your file with *#!/bin/bash*
- more info: *man bash*

Environment Variables

- some shell variables have special meaning:
  $HOME $SHELL $TERM $PATH
- these variables are usually set in files like */etc/profile* and *~/.profile*
- setting an environment variable:
  export capital=’cardiff’
- viewing an environment variable:
  echo $capital

UNIX File System Structure: Where to Find What

/bin	essential programs
/usr	more system resources (read-only)
/var	logfiles and other system stuff (r/w)
/etc	configuration files
/home	user files
/dev	devices (everything is a file)

.	the current directory
..	the parent directory
~	the user's home directory

Useful commands: mount, df, du | xdu

Processes (1/2)

- process: program that is running
- each process has a PID (and a parent)
- some processes are *daemons* (*inet*, *cron*, *getty*, *apache2*, …)
- processes run in *user space* whereas the kernel runs in *kernel space*
- useful commands:
  jobs
  ps -ef
  top

Processes (2/2)

starting a process in the background:
cp hugefile1 hugefile2 &

stopping a process and
resuming in the background:
cp hugefile1 hugefile2
^Z
bg

killing a process:
kill %1 (or any other job number)
kill 123 (or any other process number)
kill -9 123 (if your process is still not listening)

Some examples of other useful UNIX tools

type ls
grep something myfile.txt
find . -name 'test'*
head -n 3 myfile.txt
tail -F myfile.txt
touch myfile.txt
diff myfile1.txt myfile2.txt
wc myfile.txt
od -t cx1 myfile.txt
telnet www.martincaminada.net 80
curl http://www.martincaminada.net
ssh -X scmmc1@lapis.cs.cf.ac.uk
sudo shutdown -h now

vi: The Editor

- *vi* is the standard editor on UNIX systems
- *vi* is fast and elegant, but also has a steep learning curve
- *vi* distinguishes between *walking mode* and *editing mode*
- in walking mode, the keyboard keys serve as editor instructions (like for moving around)
- in editing mode, the keyboard keys serve for entering input to the file
- walking mode → editing mode: press [i] (or [a])
- editing mode → walking mode: press [ESC]

More Info

- The Linux Documentation Project
  Guides + HOWTOs (www.tldp.org)
  - Bash guide for beginners
  - GNU/Linux command-line tools summary
  - the Linux system administrator's guide
  - Linux filesystem hierarchy
- Books from O'Reilly, like "UNIX power tools" (Powers & Peek)
- *en.wikibooks.org/wiki/Learning_the_vi_Editor*

  (note: these additional resources are not exam material)