Introduction to Linux — Lecture 2

Chris P. Jobling

School of Engineering
University of Wales Swansea

EG-253 — 2006.10.11

Outline

Previously

- Introduction to course.
- History of Unix & Linux.
- System overview — kernel, shells, commands, GUI, etc.
- Getting started / Essential commands.
- Some bash tips: history & tab completion.

Today

- Files & the filesystem.
- Tour of the filesystem hierarchy.
- Mount points, links, home directories, paths.
- Users, groups, ownership, permissions.
- Commands for filesystem management.

Next Week

- Closer look at plain text files.
- Processes & process management.
- I/O, redirection, & pipes.
- More bash tips (environment, path, aliases).
- Networking tools.
- Power tools.

Files in Unix

There is a concept in Unix that ‘everything is a file or a process’. File types:

- Regular files
- Directories
- Links — like shortcuts in Windows.
- Special files — `/dev/` and `/proc/`
- Sockets — file-like objects for networking.
- Named pipes — redirection pipes which live on disk.

A path — a string of characters pointing to a file.

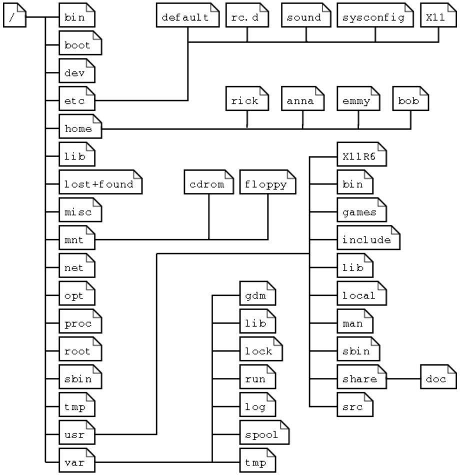
Filesystem Organisation

- MS Windows — A: drive, C: drive, D: drive, etc.
  - Each is effectively a standalone hierarchy.
- Linux — **only one** hierarchy of files and directories.
  - Everything hangs somewhere off /
  - What about multiple partitions, CD-ROMs, etc?
  - Answer is **mount points** — each filesystem hooks into the root filesystem.
  - Effectively creates a single all-encompassing ‘drive’.

Some Important Directories

- `/bin`, `/usr/bin`, `/sbin` — binary executables.
- `/dev` — device files (eg hard disk, printer).
- `/etc` — system configuration files.
- `/lib`, `/usr/lib` — libraries of shared code (compiled).
- `/mnt` — mount points for removable media.
- `/proc` — information on running processes, system, etc.
- `/tmp` — temporary area, open to everyone.
- `/var` — variable files, eg mail spools, system logs.

Some Important Directories (2)



Home Directories

- Every user has a **home directory**
  - Shouldn't be writable by others.
  - Readable? Not sure — see later to find out how to check.
- Generally `/home/username`
- `root` is special, theirs is `/root`
- Refer to your own home directory as ~
- Refer to someone else's like `~csandy`
  - Tab completion should expand these — woo hoo!

Mounting & Mount Points

- A **mount point** is just a directory.
- **Mount** a filesystem using `mount`
  - Its root now appears at the mount point.
  - eg: `mount /dev/cdrom /mnt/cdrom`
- **Unmount** using `umount`.
  - `umount /mnt/cdrom`
- nb: Mount point can be any old directory.
  - Previous contents are hidden by mounting.
- Floppy/CD mount/unmount can be made automatic.

Links

- **Links** — like shortcuts in MS Windows.
- **Soft links** (aka symbolic links):
    - A file which points to another file by storing its path.
    - Exactly like shortcuts in Windows.
- **Hard links**:
    - Inodes — every file uniquely identified by an inode.
    - So, many paths pointing at one inode; multiple paths to same file.
    - That's a hard link: a second path pointing at some inode.
    - Can't span filesystems (inode uniqueness), hence need soft links.

Listing Links

- Use `ls -l` to see a soft link.
  ```
  mycp -> /bin/cp
  ```
- Use `ls -i` to see inode numbers (and thus hard links).
  ```
  3506978 a.txt 3506980 b.txt 3506978 c.txt
  ```
- Use `stat` to see how many links to a file.
- Also: `rm` removes links, only deletes file when last link removed.

More on Paths

- Current directory: `.` Parent directory: `..`
- eg, `hello.txt` == `./hello.txt`
  - `ls -a` to show files whose names begin with `.`
  - `ls -A` — like `-a` except skips `.` and `..`
- Absolute path — begins with `/`, unambiguous, relative to root directory.
- Relative paths — no `/` at start, relative to current directory, sensitive to context.

Users & Groups

- Users uniquely identified by a **uid** (integer) and associated **username** (string).
  - **username** is what we usually think of as user ID.
- Also **groups** of users, for aggregated control of permissions, etc.
  - Again, unique **gid** (integer) and **group name** (string).
  - A user can (and often does) belong to more than one group. If group has permission, user has permission.
  - A user always belongs to at least one group.

User & Group Commands

- `id` — get uid, username, and gid, group name for all groups.
- `logname` — print username you logged in as originally.
- `whoami` — print username you're *currently* logged in as.
  - Could differ from `logname` if called `su`.
- `groups` — print names of groups you belong to.
- `users` — print names of users currently logged on.
- `who` — like `users`, but with extra information.

File Ownership

- Every file is owned by a user and a group.
- Usually from the user who created it.
- Can be changed using `chown` command.
- Use `ls -l` or `stat` to see ownership information.

File/Directory Permissions

- Read/Write/Executable permissions in three classes:
  - User — ie user who owns this file.
  - Group — ie group who owns this file.
  - Other — everyone else.
- If file is a directory:
  - read == list
  - write == create/delete files
  - execute == move into.
- Sticky bits, setuid bits, setgid bits. . .

`ls` — list a file or a directory

- `-l` for long information, `-i` for inode.
- `-a` for all files, `-A` for all except two.
- `-1` — 1 item per line.
- `-F` — append a character telling type of file.
- `-R` — list directories **recursively**.
- `-d` — list directory names but not contents.
- `-t` — sort output by last modification.

cd, pwd, pushd, popd

- `cd` — trivial, changes directory.
    - No destination specified: go home.
- `pwd` — print out present working directory.
- `pushd` — like `cd` but pushes directory moved *from* onto a **stack**.
- `popd` — pop the stack, go back to that directory.

cp — copy files

- `cp src dest`
  - What if `dest` is a directory?
- `cp src1 src2 ...srcn dest_dir`
- Skips directories unless specify `-r` or `-R`
- Silent unless `-v` (**verbose**)
- Read the man page.

mv — move/rename files

- `mv src dest`
  - What if `dest` is a directory?
- `mv src1 src2 ...srcn dest_dir`
- Not many options — an easy man page to read.
- Move within filesystem — quick, because just moving references to files on disk.
- Move across filesystem — slower, because must actually move data of files.

mkdir — make a directory

- `mkdir dir_name`
- `-m` to specify `mode` of directory — see `chmod`
- `-p` to create missing parents.
- `-v` to be verbose.

ln — make a link

- `ln src dest` — create a hard link
  - What if on different filesystems?
  - What if `dest` exists already?
- `ln -s src dest` — create a soft link.
- `-f` option to force overwriting existing file.
- Several other options.

rm & rmdir — remove files and directories

- Danger, danger, danger!
- **There is no undelete in Linux!**
- `rm file1 file2 ...filen`
- `-r` to recurse into directories.
- `-i` for interactive mode.
- `-v` for verbosity.
- Remember: removes *references* to files.
- `rmdir` fairly redundant?

touch — update a file’s modification time

1. Update the timestamp of the file.
2. Create a zero-length file.

chown — change ownership of a file

- `chown username filename`
- `chown username.groupname filename`
- `-R` to operate recursively.

chmod — change permissions (mode) of a file

- `chmod perms filename`
- `-R` to operate recursively.
- Symbolically — `chmod og+rx filename`
  - `u` for user, `g` for group, `o` for other, `a` for all
  - Can be absolute (=) or relative (+, −)
  - `r` for read, `w` for write, `x` for execute
- Octal 4 digits (absolute) — `chmod 0555 filename`
  - 2nd, 3rd, 4th digits correspond to user, group, other.
  - 4 = read, 2 = write, 1 = execute

stat — get file or filesystem status

- `stat filename`
  - Filename, size, block count, file type
  - Inode number, number of links, device
  - Access rights, uid, gid
  - Last access, modify, change
- `stat -f filename` — report on filesystem where file lives
  - Location, max name length, blocks available, used, etc.
  - Inodes used, available, . . .

du — report on disk usage

- `du` — disk usage of current directory
- `du dir_name` — disk usage of specified directory
- `-max-depth` to set maximum depth *reported on*
  - Counting is still done to full depth.
  - Use 0 to just get a final total.
- Default unit is block, use `-h` for human-readable
- `-x` option to constrain to one filesystem

which & find

- `which command` — where is command on disk?
  - Searches **path** — see next lecture.
- `find` — find files according to various criteria.
  - `find` — list all files off current directory.
  - `find . -nane 'hello.txt'`
  - `find /tmp -name '*.txt'`
  - `find . -name '*.txt' -exec rm -vf {} ';'`
  - `find . -type d -name 'no*' -links 3`
  - Monster man page!

df — report on filesystems & disk usage

- See which filesystems are mounted on which mount points.
- `-h` for human readable sizes.

```
[csandy@cspcag csandy] df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/root             7.5G  4.2G  3.0G  59% /
/dev/hda3             9.9G  6.4G  3.6G  65% /usr
/dev/hda5             5.0G  1.2G  3.6G  25% /home
/dev/cdroms/cdrom0    223M  223M     0 100% /mnt/cdrom
```