
THE UNIVERSITY OF WESTERN AUSTRALIA

# Open Source Tools & Shell Programming

## Lecture 1

Michael J. Wise

# Open Source Philosophy

- Free to use software ≠ Open Source software
  - *Cf Google search (free to use, but not is not OS)*
  - *LibreOffice, Mozilla (Firefox)*
    - Can get source code or binaries
- Virtues (beyond cost)
  - *Transparency and, potentially, greater security*
  - *Flexibility*
    - Create your own version, e.g. Moodle @ UWA
  - *As good, perhaps better quality*
    - Wikipedia vesus Encyclopedia Brittanica
- Licensing
  - *GNU or Creative Commons versus copyright*

# History of UNIX

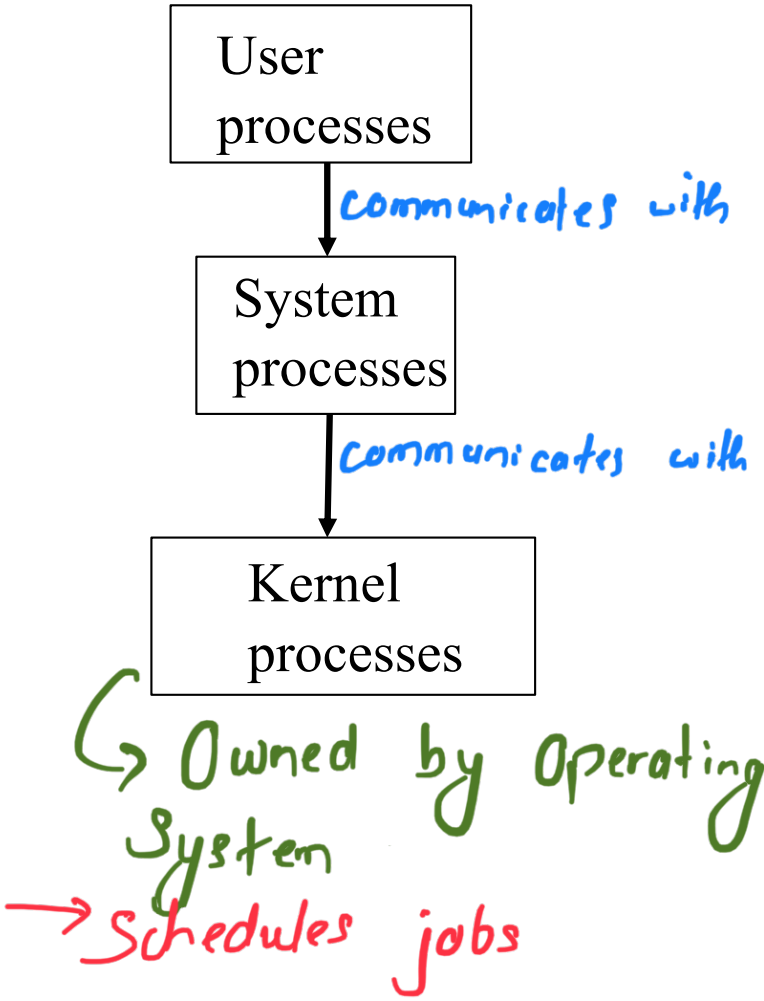
- Created in 1969 at AT&T Bell Labs to run on departmental computers
  - *PDP11 tiny by today's standards*
    - 16 bit words; 64Kb/128Kb memory; Removable disk pack 2.5Mb
- Originally coded in assembler language, then C (1973), which was developed for that purpose
- → Designed for efficiency

# UNIX Philosophy - processes

- Programs are compiled into objects (binaries, executables) → Static
- Programs (or binaries) in execution are processes
  - *Process image*

# UNIX Philosophy - processes

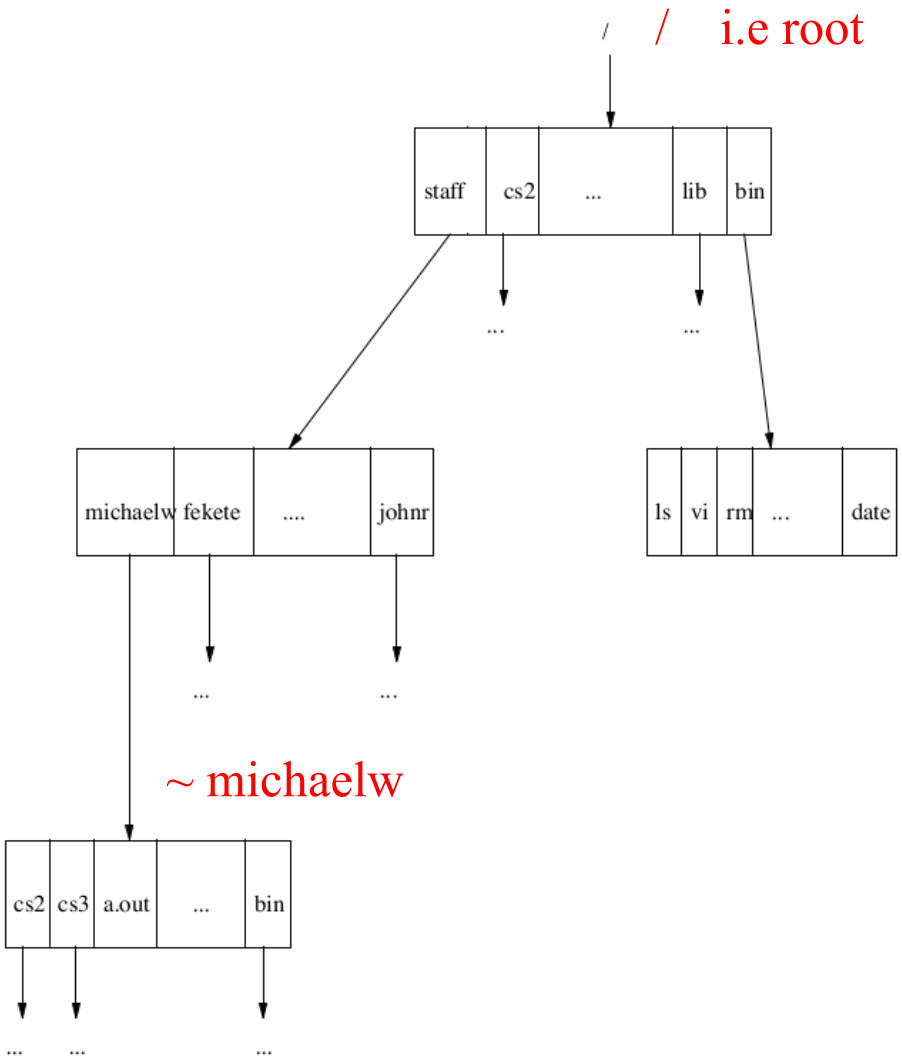
- Small kernel (the real Unix/Linux), a master process, provides services to system and user processes
- Simple data type/format (text) and simplified inter-process communication, allows chaining of processes, versus monolithic programs
- Separation between ordinary users, and a super-user which has special access privileges
  - *Opening a file*
  - *Sending a file the printer*



# UNIX Philosophy - files

- Unix views almost everything as a file
  - *There is even a way of viewing processes as files*
  - *Directories are files that point to other files*
- The Unix file system is structured as a tree. The topmost directory is called the root, written as /
- Every user has their own tree structured nest of directories, each with its own root, called HOME, and written as ~

# Directory Structure



# Filenames

- Unlike Windows, file name suffix is conventional, but not compulsory, e.g. `.txt` for a text file, `.py` for a Python file
- File names can be any non white-space character; no blanks in names, so "_" often substituted.
- The current directory is written as `.` (dot)
- The parent of the current directory is written as `..`
  - *Grandparent directory* `../..`
  - `/staff/michaelw/a.out`
- A file name beginning with `.` (dot) is a "hidden" file, in that you don't see it when doing plain `ls`

# Unix Basix: What is a UNIX command?

*<Command name> [<Options>] [<Objects>]*

Command name

- The verb. Often, all that is required is the name of the command, e.g `ls` lists the names of the files in the current directory. `ls fred` (does file `fred` exist?)

Options

- Act like adverbs, modifying the meaning of the command. Generally begin with a '–', e.g. `ls -l` gives a more verbose listing

Objects

- Act like nouns, i.e. the objects the commands act on, e.g. `ls -l fred`

# The Shell

- You execute commands via a command interpreter called the shell:

Prompt

```
% ls Alice_in_Wonderland.txt
```

Alice_in_Wonderland.txt

- The shell locates the command, locates the object and, all being well, executes the command, e.g. `ls`
  - *User process* ↔ *System process* ↔ *Kernel*
- Typing commands one by one can be tedious, so you can create Shell scripts via a shell scripting language
  - *First was sh (Bourne Shell, 1976; Bash (Bourne Again Shell), 1989; ksh (Korn Shell), 1983,1993*

# The most useful command

`man` *<name>*

Returns the manual page for the named command

E.g. `man ls`

```
LS(1)                    BSD General Commands Manual                    LS(1)

NAME
     ls -- list directory contents

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls displays its name as
     well as any requested, associated information.  For each operand that names a file of type
     directory, ls displays the names of files contained within that directory, as well as any
     requested, associated information.

     If no operands are given, the contents of the current directory are displayed.  If more than
     one operand is given, non-directory operands are displayed first; directory and non-directory
     operands are sorted separately and in lexicographical order.

     The following options are available:

     -@      Display extended attribute keys and sizes in long (-l) output.

     -1      (The numeric digit ``one''.)  Force output to be one entry per line.  This is the
             default when output is not to a terminal.

     -A      List all entries except for . and ..  Always set for the super-user.

     -a      Include directory entries whose names begin with a dot (.).

     -B      Force printing of non-printable characters (as defined by ctype(3) and current locale
             settings) in file names as \xxx, where xxx is the numeric value of the character in
             octal.

     -b      As -B, but use C escape codes whenever possible.

     -C      Force multi-column output; this is the default when output is to a terminal.
```

And the rest ……

# Unix Commands Starter Pack

- `mkdir` <*directory*> ...

Create one or more new directories as named by the argument(s).

- `cd` <*directory*>

Move to named directory (i.e. it becomes the current directory). If no directory is named, move to home directory. The directory can also be `..` which is the parent directory.

- `pwd`

Print name of current (working) directory (for when you get lost)

# Unix Commands Starter Pack

- `ls` *<file or directory names>*

For each directory, the contents of the directory are listed; for each file, the name of the file is repeated. Other information will be added depending on the options that are specified. Most common options:

`-l` Produce a long listing giving detail of file size, ownership, etc.

`-C` Produce multicolumn output.

`-t` List in order of decreasing time-stamp (latest first).

`-r` Reverse the listing order

`-a` List all files, including hidden files (filenames beginning with a `.` (dot))

# Unix Commands Starter Pack

- `mv` *<file1> <file2>*

Change the name of file1 to be file2. If file2 already exists, it is removed

- `mv` *<files> <directory>*

Move the listed file(s) into the directory.

- `cp` *<file1> <file2>*
- `cp` *<files> <directory>*

As for mv, but the original file(s) remain.

- `ln` *<file1> <file2>*

A link is made between *file1* and *file2*. Saves space by allowing the file-name to be present in more than one directory or be known by more than one name, without copying.

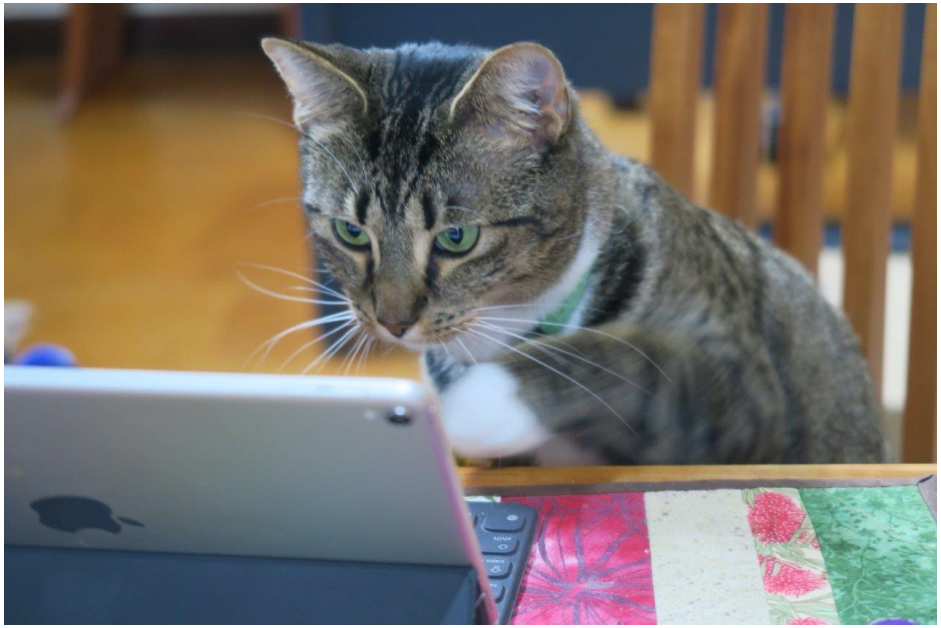
# Unix Commands Starter Pack

`cat` <*files*>

For each of the listed files, write their contents to standard output. Mainly used for con**cat**enating multiple files into a single file.

`rm` <*files*>

Removes the named files, but not directories. (-r recursive)



Sherlock was unimpressed with that last tweet

What if I called the file `-a` rather than `_a` ?

# Unix Commands Starter Pack

- `head` *–<N> <file>*
- `tail` *–<N> <file>*

*Display the first/last N lines of the named text file. Useful if the file is very large to grab the first/last few. For example, debugging output*

- `less` *<file>*

*Allows you to scroll through the file (*`less` *is an improved version of* `more`*)*

# Demo

- Try the following:

```
mkdir week1
cd week1
echo "Hullo world"  > _a
```

- Which command gives you the file size of `_a` ?

```
ln _a _b
```

- How many files are there?
- What is the file size of `_a` ?
- What if I called the file `-a` rather than `_a` ?

```
cd ..
```

- What do you see now?

# Demo

- Run the command:

`wget`
`https://teaching.csse.uwa.edu.au/units/CITS4407/L0_demo/Alice_in_Wonderland.txt`

Which will fetch the Alice text into the current directory.

Use `head`, `tail` and `less` to look at the file. Use `wc` to count the number of lines, words and characters.