COMP20200 Unix Programming

Lecture 2

CS, University College Dublin, Ireland

Simplified system

Most basic Unix machine

- Processor
  - CPU
  - Memory
- Keyboard
- Screen

Basic C programme

- -
  - Arithmetic calculations
  - Memory operations, data types
- Input
- Output

The Shell

- User command-line interface is known as the Shell.
  (a.k.a. terminal, command line, prompt)
- Faster to use & more powerful then GUI for power users
  - However takes time to learn
- Shell is a user level program
- Numerous versions of shell (`ksh`, `csh`, `bash`) with slight syntax differences
  - `bash` most common (**B**ourne **A**gain **S**Hell)
- Only desktop systems have GUI (Graphical User Interface) but all systems will have a some form of shell.
- When a server boots:
  - First starts necessary services (file system, network, logs, etc.).
  - Then starts a login shell and services to accept network logins (`ssh`)
  - Users can then login and use the system

The Shell (cont.)

After Unix system boots, user presented with a shell

```
user@host:~/current_path$
```

Or if logged in as root (superuser)

```
root@host:/etc/#
```

The Shell (cont.)

- User types a command line.
- Shell assumes first word is a program and searches for it.
- If it finds it, it runs program and passes remainder of line to program.
- Shell suspends itself until programme terminates
- Then tries to read next command.
- Examples:

```
$ man mkdir
$ mkdir -v testdir
$ gcc -Wall -o hello hello.c
$ cp src dest
```

Running commands

```
$ command arg1 arg2...
```

Example: `ls` lists the contents of a directory and `mv` moves a file.

```
$ ls
dir1  file1  file2  file3
$ mv file1 dir1
$ ls dir1/
file1

 $ typo
typo: command not found
```

Note:

`mv` also used to rename a file (move from old name to new name).

Running commands: Path

- Core utilities are stored in /bin
  eg: ls cp rm su cat
- Other system programmes are stored in /usr/bin
  eg: gcc vi tar ssh

On the default user setup the environment variable $PATH includes /bin /usr/bin

```
$ /bin/ls     #same as running 'ls'
$ /usr/bin/gcc #same as running 'gcc'
```

- Users current directory isn't in $PATH so must do:

```
bob@host:~/lab1$ ./hello
```

. means current working directory.
.. means up one directory level.
~ is users home directory.

Command Arguments

Modify behaviour of programmes by passing arguments.

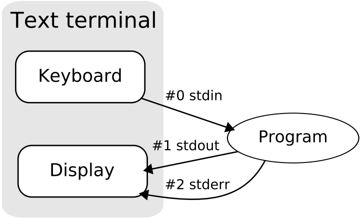
```
  $ ls
  dir1 file2 file3
  $ ls dir1
  file1
  $ ls -l .
drwxrwxr-x 2 bob users 4096 Jan 20 13:46 dir1
-rw-rw-r-- 1 bob users    0 Jan 20 12:57 file2
-rw-rw-r-- 1 bob users    0 Jan 20 12:57 file3
$
```

Everything is a file (or a process)!

- One of the defining features of Unix
- Same tools and APIs can be used on wide range of resources
- For example, in the virtual filesystem `/proc`
  `/proc/cpuinfo` contains info about CPU
  `/proc/uptime` has the length of time the kernel has been running
- All these "files" have standard attributes such as an owner and permissions.
- Therefore, a tool can handle input and output from many sources in the same, predictable way
  - a regular file
  - from keyboard (via stdin)
  - from the kernel (via `/proc`)
  - output to screen (via stdout or stderr)
- Additionally, almost all system settings are in plain text files

stdio - Standard Input/Output

- standard input (stdin)
- standard output (stdout)
- standard error (stderr)



- Abstract devices
- Program doesn't need to know what kind of device it is communicating with
- > redirects stdout into file
- < puts contents of file to stdin
- | pipe, passes output of first programme as input to second - very powerful!
- 2> redirects stderr to a file.
- &> redirects both stdout and stderr to same file.
- >> Appends stdout to a file.

stdio, an example

Example: `wc` - word count, reads from stdin, writes to stdout
(more info: `man wc`)

```
$ wc
This is text typed in. End with Ctrl+D
     1       8      39

$ wc comp20200-L2.tex
     608  2328 22048

$ ls ~/src/somesoftware/*.c | wc
     22      22     1219

$ grep processor /proc/cpuinfo | wc > num_cores
$ cat num_cores
     2       6      28
```

stdout - vs - stderr

```
#include <stdio.h>
int main(){
  fprintf(stdout, "Hello from standard out\n");
  fprintf(stderr, "Hello from standard error\n");
  return 0; }
```

```
$ gcc -Wall output.c -o output
$ ./output
Hello from standard out
Hello from standard error
```

Both outputs appear on screen the same. But see what happens with:

```
$ ./output > file
$ ./output 2> file
$ ./output &> file
```

Between each command do `cat file` to see it's contents.

Where to get help

It is too much to know all arguments to all commands. Manuals are your friend (if you know the name of the command).

Read the manual!

What does `-o` and `-Wall` do on the previous slide?

```
$ man gcc
```

First learn about manuals: `$ man man`

RTFM

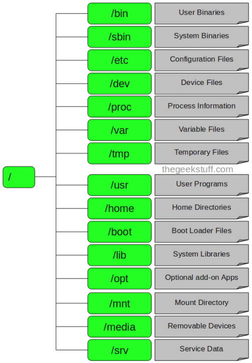
HELLO, 911? I JUST TRIED TO TOAST SOME BREAD, AND THE TOASTER GREW AN ARM AND STABBED ME IN THE FACE!
DID YOU READ THE TOASTER'S MAN PAGE FIRST?
WELL, NO, BUT ALL I WANTED WAS—
CLICK

http://xkcd.com/293/

```
$
```

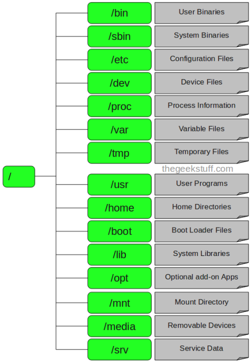
Directory Structure of Linux

1. / - Root
   - Every single file and directory starts from the root directory.
   - Only root user has write privilege under this directory.
   - Note that `/root` is root user's home directory, and is not same as /
2. /bin - User Binaries
   - Contains binary executables.
   - Commands used by all the users of the system are located here.
   - For example: `ps`, `ls`, `ping`, `grep`, `cp`.
3. /sbin - System Binaries
   - Contains binaries for system administrators.
   - For example: `iptables`, `reboot`, `fdisk`, `ifconfig`, `swapon`



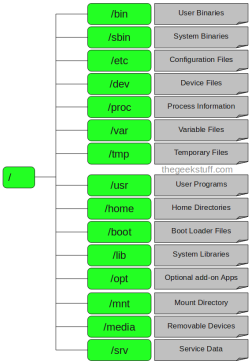
Directory Structure of Linux

1. /lib - System Libraries
   - Contains library files that supports the binaries located under /bin and /sbin
   - Library filenames are either ld* or lib*.so.*
   - For example: `ld-linux.so.*`, `libxtables.so.*`
2. /etc - Configuration Files
   - Contains configuration files required by all programs.
   - The "personality" of a system.
   - This also contains startup and shutdown shell scripts used to start/stop individual programs.
   - For example: `/etc/hostname`, `/etc/fstab`, `/etc/passwd`



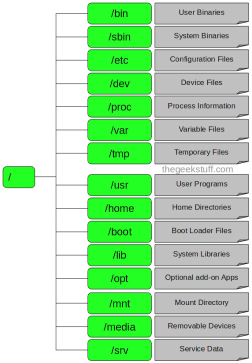
Directory Structure of Linux

1. /dev - Device Files
   - Contains device files.
   - These include terminal devices, usb, or any device attached to the system.
   - For example: `/dev/sda1` `/dev/tty1` `/dev/cdrom` `/dev/null`
2. /proc - Process Information
   - Contains information about system process.
   - This is a pseudo filesystem contains information about running process. For example: `/proc/{pid}` directory contains information about the process with that particular pid.
   - This is a virtual filesystem with text information about system resources.
   - For example: `/proc/uptime` `/proc/meminfo` `/proc/cpuinfo`



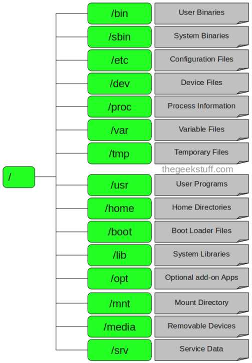
Directory Structure of Linux

1. /var - Variable Files
   - Content of the files that are expected to grow can be found under this directory.
   - This includes - system log files (`/var/log`); packages and database files (`/var/lib`); emails (`/var/mail`); print queues (`/var/spool`); lock files (`/var/lock`); temp files needed across reboots (`/var/tmp`);
2. /tmp - Temporary Files
   - Directory that contains temporary files created by system and users.
   - Files under this directory are deleted when system is rebooted.



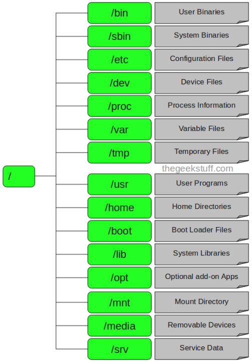
Directory Structure of Linux

1. /usr - User Programs
   - Contains binaries, libraries, documentation, and source-code for second level programs.
   - /usr/bin contains binary files for user programs. For example: `at`, `awk`, `cc`, `less`, `scp`
   - /usr/sbin contains binary files for system administrators. For example: `cron`, `sshd`, `useradd`, `userdel`
   - /usr/lib contains libraries for `/usr/bin` and `/usr/sbin`
   - /usr/man Manual pages
   - /usr/include Header files (for C/C++ . . .)
   - /usr/local contains users programs that you install from source.



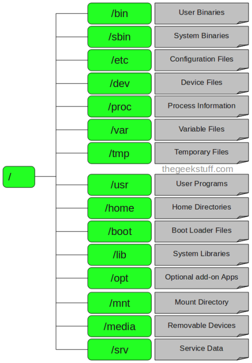
Directory Structure of Linux

1. /home - Home Directories
   - Home directories for all users to store their personal files.
   - For example: `/home/dave`, `/home/john`
   - can use ∼/ to access your own directory. Also `∼dave/` is equivalent to `/home/dave`
2. /boot - Boot Loader Files
   - Contains boot loader related files.
   - Kernel initrd, vmlinux, grub files are located under /boot
   - For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic



Directory Structure of Linux

1. /opt - Optional add-on Applications
   - Contains add-on applications from individual vendors.
2. /mnt - Mount Directory
   - Temporary mount directory where sysadmins can mount filesystems.
3. /media - Removable Media Devices
   - Similar to /mnt, temporary mount directory for removable devices.
   - For examples, `/media/cdrom`
4. /root - Root user's home directory



Finally

- Command line tip(s) of the day:
  - Use tab to auto complete.

    `$ cp som`

    type tab and you get:

    `$ cp somelongfilename`

    then finish typing command. `$ cp somelongfilename new_name`
  - Use up arrow to get previous command
- Vi tip(s) of the day:
  - Vi is definition of steep learning curve, but well worth it.
  - First lesson, know how to quit:

    Type Esc, followed by :q

    When in doubt - hit Esc.