Operating systems and concurrency - B01

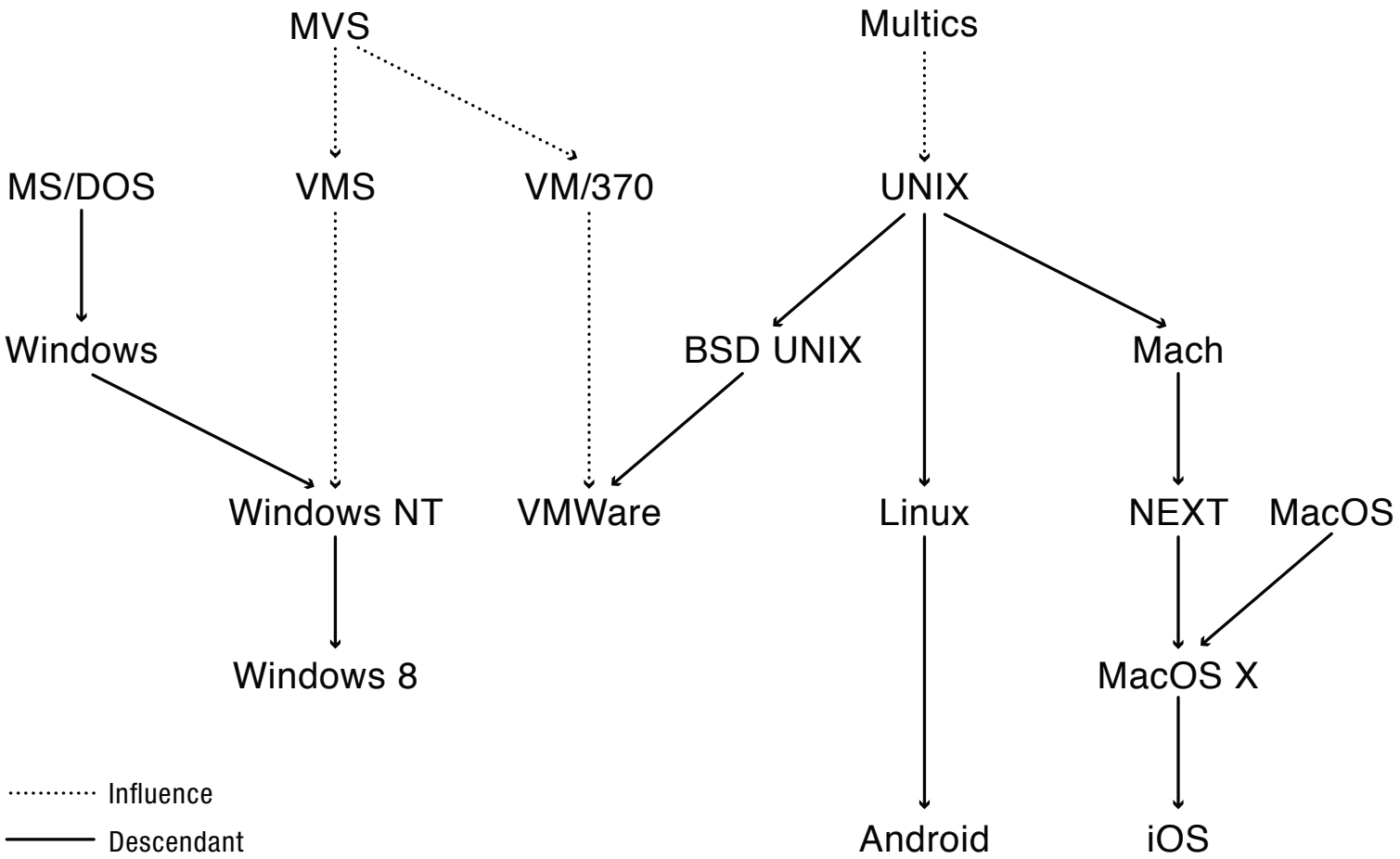
David Kendall

Northumbria University

Outline

- Review the module page (if this is the first lecture of the semester)
- A little OS history
- What is Unix?
- Linux - a free, open-source OS kernel
- GNU - a suite of OS utilities
- Ubuntu - a commonly-used GNU/Linux distribution

Where does Unix fit in OS history?



Anderson, T. and Dahlin, M., *Operating systems: principles and practice*, Recursive Books, 2014

Why is Unix important today?

OS penetration by sector

Type	Unix-based %	Windows %	Other %	Source	Year
Desktop/Laptop	8.61	91.39	0.0	Net Applications	2016
Smartphones	99.3	0.4	0.3	Gartner	2016
Tablets	90.0	10.0	0.1	Strategy Analytics	2015
Web clients	55.25	39.65	5.72	StatCounter	2016
Servers (web)	67.8	32.3	0.0	W3Techs	2015
Mainframe	100.0	0.0	0.0	Gartner	2008
Supercomputer	100.0	0.0	0.0	TOP500	2016
Worldwide device shipments	87.9	11.7	0.4	Gartner	2015

- **Unix-based** Linux, Android, iOS, OSX, Chrome OS, AIX, FreeBSD, HP-UX, Solaris, PS4, PS3, QNX
- **Windows** All Windows OS from Windows 98 to Windows 10, including Xbox and WCE
- **Other** Symbian, Blackberry OS, ...

Projected sales 2017

Approximately 2.14 billion out of a total of about 2.47 billion devices shipped in 2017 will be smartphones and tablets (Forbes)

Unix

Unix was designed to be a:

- portable,
- multi-tasking,
- multi-user,
- time-sharing configuration.

Origins of Unix

- originally developed in 1969 at AT&T Bell Labs by a team including Dennis Ritchie, Ken Thompson and Brian Kernighan
- 1973, coded in C

Unix

Became a “commercial" product in the 1980s. Researchers at University of California, Berkeley continued to develop “academic" version distinct from these.

- BSD Unix
- includes TCP/IP implementations
- adheres to IEEE POSIX standard
- Apple has based Macintosh operating systems on BSD since 1997

Standards

POSIX IEEE based POSIX around the common structure of the major competing variants of the Unix system, publishing the first POSIX standard in 1988

SUS the Common Open Software Environment (COSE) initiative, which eventually became the Single UNIX Specification administered by The Open Group

- Starting in 1998 the Open Group and IEEE started the Austin Group, to provide a common definition of POSIX and the Single UNIX Specification.

The Filesystem Hierarchy Standard was created to provide a reference directory layout for Unix-like operating systems, particularly Linux.

GNU – Gnu's Not Unix

- Richard Stallman announced a project to develop a completely free Unix-like operating system in 1983
- A Unix-like operating system includes a kernel, compilers, editors, text formatters, mail software, graphical interfaces, libraries, games and many other things.
- By 1990, the GNU project had developed all of the major components, except the kernel

gcc	a C compiler
emacs	a text editor
sh	a shell
ls, grep, find	file utilities
...	and many other components.

- In addition, MIT contributed the X Window system, a windowing system that provides the framework for a GUI-environment
- Donald Knuth developed the text processing system, TeX

Linux

Unix-like kernel first developed in early 90s by Linus Torvalds

- Original idea: a complete rewrite of the Unix kernel for “IBM-compatible" PCs
- Now in conjunction with GNU software we have a free, mature PC operating system . . .
- . . . and also an OS for servers, mobile devices, embedded devices
  - routers, satelite decoders, games consoles, mobile phones, tablet computers
- Google is hosted on half a million custom Linux machines
- The Android OS is a Linux variant
- the GNU *General Public License* (GPL)

Linux

- Now packaged in *distributions*
  - Debian, Fedora and openSUSE, ...
  - Ubuntu is a derivative of Debian
- A distribution includes a lot of software libraries including applications and utilites.
  - “office" applications (word processing, spreadsheets, presentation, ...)
  - music, multimedia, graphics, ...
- Distributions intended for desk-top/laptop PC use include a graphical “desktop"
  - based on the *X-Window system*
  - Common ones are *Unity*, *Gnome* and *KDE*
- Distributions intended for servers, embedded systems have only a command-line console.

Ubuntu Linux

- A derivative of the Debian distribution, sponsored by Canonical Ltd who make money by selling technical support, training, etc
- First released October 2004; new version every April, October with a “long-term support" version every 2 years. Versions are named by year and month of release. Both desktop and server versions released.
  - 8.04 “Hardy Heron", 8.10 “Intrepid Ibex",
  - 10.04, “Lucid Lynx", 10.10 “Maverick Meerkat",
  - 12.04 “Precise Pangolin”, 12.10 “Quantal Quetzal”
  - 14.04 “Trusty Tahr”, 14.10 “Utopic Unicorn”,
  - 16.04 “Xenial Xerus”, 16.10 “Yakkety Yak”
- A variety of desktops is available
  - Unity
  - Gnome
  - KDE
  - Mate, Cinnamon, i3wm, . . .

Ubuntu Linux

- Apart from proprietary hardware drivers, software is free and open-source, mostly under GNU General Public License

- Libre Office (Open Office before 11.04) – interoperable with Microsoft Office
- Firefox
- Empathy / Pidgin messaging
- Gimp
- Much more can be installed on demand using the “Synaptic" Package Manager, a pre-installed utility. Choose
  - Free – supported free software
  - Restricted – support non-free software
  - Universe – unsupported free software
  - Multiverse – unsupported non-free software
- *ubuntu-restricted-extras* additionally includes support for some legally restricted software, such as MP3, DVD playback, Microsoft TrueType fonts, Oracleś Java Runtime, support for RAR compression, ...

Ubuntu Linux

System requirements

- currently supports 32 and 64 bit architecures, PowerPC, IA-64, ARM
- current distribution expects $\geq$ 1 GHz CPU and $\geq$ 382 MiB of RAM

Installation is from a bootable CD or USB memory stick

- Run it from the boot medium before deciding to install on the hard drive!
- Download the CD image, (.iso) and burn a CD from it or make a bootable USB “drive" using the UNetbootin utilitiy.
- Wubi (on CD) allows Ubuntu to be run within a Window session.

Unix and KF5010

All practical work in this module will use a Unix-like operating system – typically a version of Ubuntu. In order to make it convenient to use both in the lab and at home, we'll show you how to install the operating system using a *virtual machine* (VirtualBox). More coming in the lab session.

Introduction – Getting started with Linux

- How the operating system starts running
- Terminal, console, shell
- The command line - why bother?
- The file system - managing files
- I/O redirection
- Advanced use of the terminal - tmux

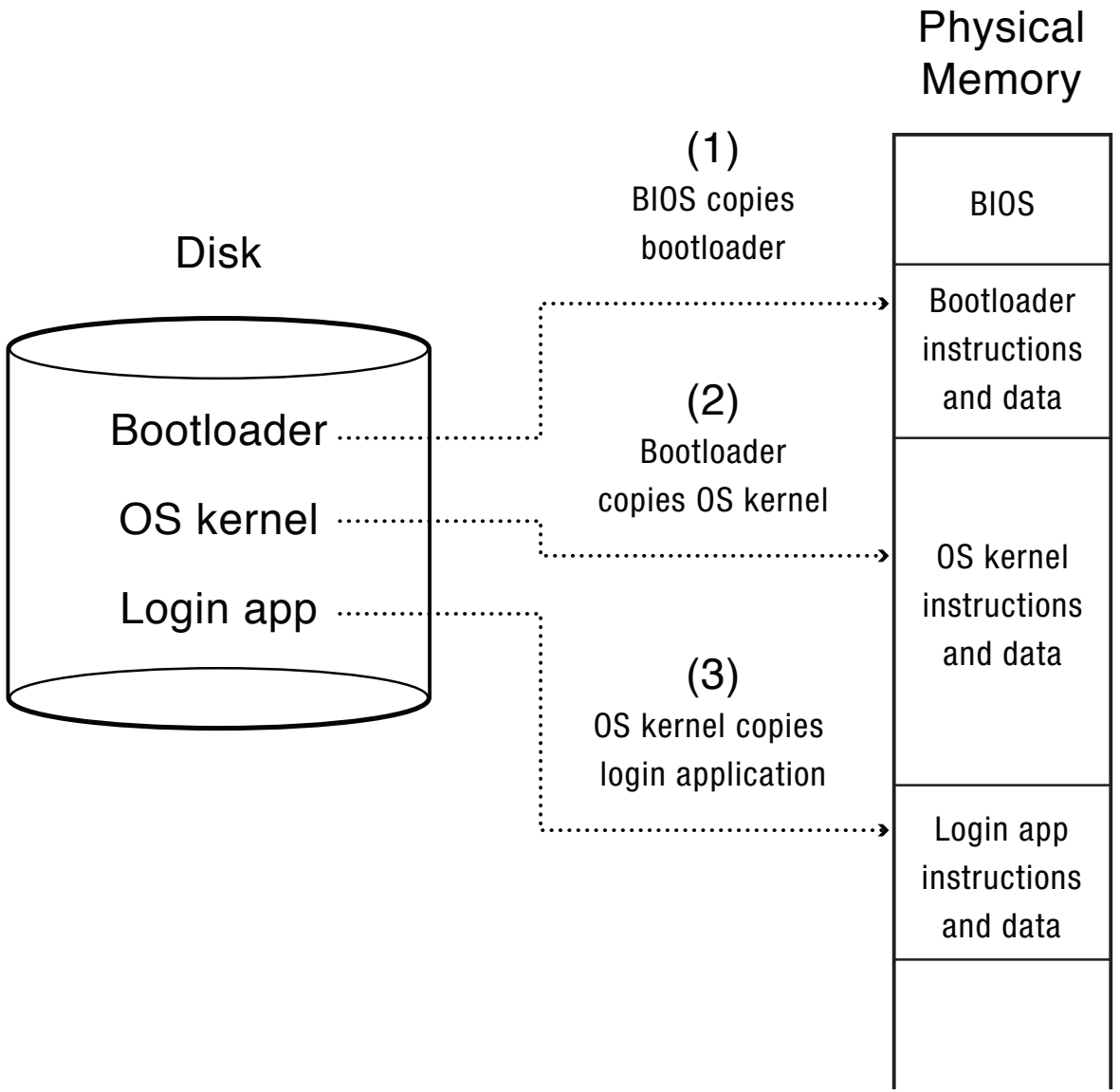
How the operating system starts running

- Turning on the computer (or pressing the reset button) forces the CPU to begin executing instructions from a fixed location in the computer memory
- The contents of memory starting at this location need to be *non-volatile*, i.e. to survive a power down/power on – usually use *ROM* (Read Only Memory).
- The ROM that stores the program used to start the operating system is called the *boot* ROM. On most PCs, the boot program is called the *BIOS* (Basic Input/Output System)
- Don't store the complete OS in the boot ROM
  - ROM is slow and expensive compared to RAM
  - ROM is hard to update
  - OS needs frequent updates

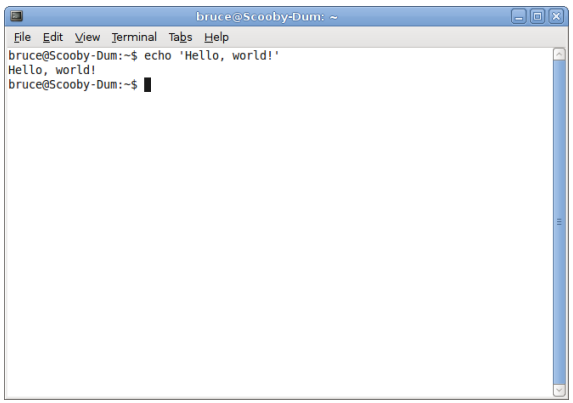
How the operating system starts running

- So load the operating system in stages:
  - The boot ROM contains a small program that is able to read a fixed-size block of bytes from a fixed position on the disk (*the bootloader*)
    - Note the boot program doesn't need to know about the file system – it just has to be able to read a block of raw bytes from a known location
    - The bootloader may have a *digital signature* to ensure that it hasn't been tampered with
  - Once the BIOS has loaded the bootloader into memory, it jumps to its first instruction and starts executing code from there
  - The job of the bootloader is to load the OS kernel into memory
    - The OS kernel is usually stored in the file system on disk, so the bootloader needs to know how to find a file in the file system and read it.
  - Once the kernel has been loaded, it can initialise its data structures and then start the first process (called *init* in Linux), which can start a login process so that the user can login and start work.

How the operating system starts running



Terminal, Console, Shell



- DEC VT220 terminal, popular in the 1980s
- A physical terminal used to communicate with a mini or mainframe computer over a serial communication link
- A console was the keyboard/monitor directly connected to the console port of the computer
- Gnome terminal – a *virtual terminal* that *emulates* the behaviour of a physical terminal – also KDE and xterm
- The *shell* is a program that receives command input from the terminal and makes calls to the computer operating system to execute the commands. Shells include bash, zsh, csh.

Command line – why bother?

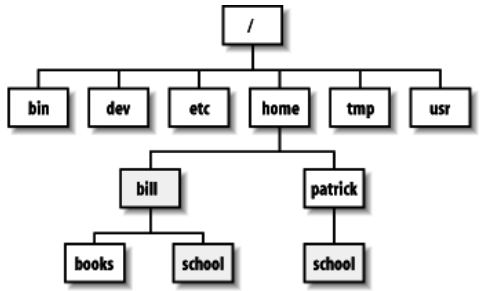
Command line advantages

- Available everywhere – many servers and small embedded systems don't have a GUI
- Uses less resources
- More efficient once you've learnt the commands – faster than clicking and scrolling, then clicking and scrolling, then clicking and scrolling some more just to do something simple
- Can easily compose commands to do complex tasks
- Can automate commands, go away and leave them running

GUI's are also good

- Browse the web using GUI
- Read email using GUI (although there are good terminal mail readers)
- Read pdf documents
- View and edit photos
- Use graphical design tools, stream video, ...

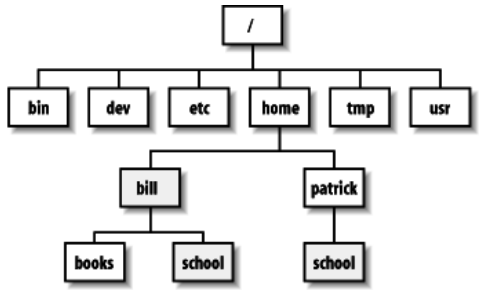
File system tree



http://etutorials.org/shared/images/tutorials/tutorial_95/rh4_0401.gif

- When we first login, we are positioned in the file system directory tree at our *home directory*, e.g. `/home/bill`
- Wherever we are in the tree is called the *current working directory*
- Every directory has a *parent directory* (except the top-level directory, `/`, called the *root directory*). The parent of a directory is the one directly above it in the tree, e.g. `/home` is the parent of `/home/bill`
- The parent of the current working directory can be referred to using the symbol, `..` (the current working directory is referred to using `.`)

File system tree



http://etutorials.org/shared/images/tutorials/tutorial_95/rh4_0401.gif

pwd where am I? (**p**rint **w**orking **d**irectory)

cd go somewhere else (**c**hange **d**irectory)

ls what's in here? (**ls**t contents of directory)

- A *path name* is a description of the location of a directory in the file system tree
- We can use *relative* path names, which start at the current working directory, e.g. `ls books`
- or *absolute* path names, which start at the root of the file system, e.g. `ls /home/bill/books`

Useful commands for working with files

ls lists contents of directory
less display contents of text file
file indicate file type
cp copy files and directories
mv move/rename files and directories
rm remove files and directories

- Command format: `command` *-option arguments*
- e.g. `ls -l /home/bill`

man look up the manual entry for a command

- e.g. `man ls`

Wildcards

Using *wildcard* characters can make the use of the file commands even more powerful

Wildcard	Matches
`*`	Any characters
`?`	Any single character
`[` *characters* `]`	Any character that is in the set *characters*
`[ !` *characters* `]`	Any character that is *not* in the set *characters*

Pattern	Matches
`*`	All files
`s*`	All files beginning with `s`
`f*.txt`	All files beginning with `f` followed by any characters and ending in `.txt`
`Log??`	All files beginning with `Log` followed by exactly two characters
`[xyz]*`	All files beginning with `x`, `y`, or `z`

I/O Redirection – Output

- Most the commands that we've used generate some *output*
- Sometimes the output is the data that we were after, sometimes it's an error message or status information
- All output is sent to a file: good data is usually sent to the *standard output* file, error messages etc. are sent to the *standard error* file
- Usually the standard output file and the standard error file are both mapped to our display, so we see the output appearing on the screen
- We can choose to *redirect* the standard output to a different file, e.g.

```
ls -l /home/bill/books > ls_output.txt
```

- We can also redirect standard error, e.g.

```
ls -l /bill/home/books 2> ls_errors.txt
```

- We can redirect standard output and standard error to the *same* file, e.g.

```
ls -l /home/bill/books > ls_output.txt 2>&1
```

I/O Redirection – Input

- Just as the standard output and standard error files are usually mapped to the display, the *standard input* file is usually mapped to the keyboard
- `cat` is a program that normally reads data from the standard input (keyboard) and sends the data to the standard output (display), e.g.

```
$ cat
Much have I travelled in the realms of gold
Much have I travelled in the realms of gold
```

- We can redirect the output as usual, e.g.

```
$ cat > gold.txt
Much have I travelled in the realms of gold
```

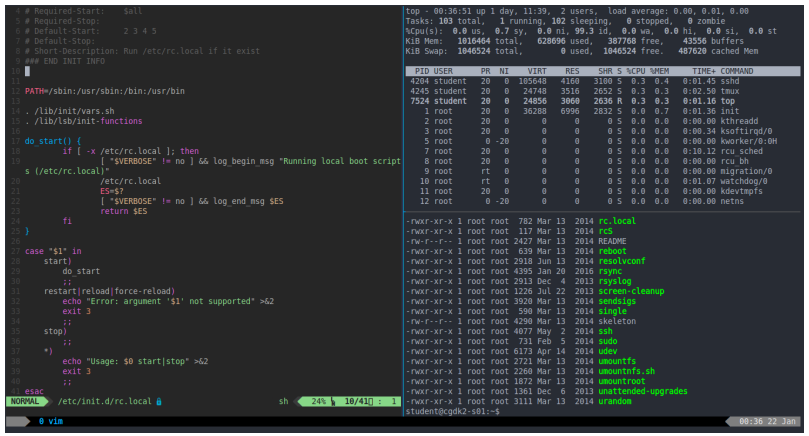
- We can also redirect the *input*, e.g.

```
$ cat < gold.txt > keats.txt
```

Next steps . . .

- We've just scratched the surface of the capabilities of the command line
- You can find out more by reading chapters 1 to 6 and 9 in the The Linux Command Line
- We'll also be covering additional commands in the lab session this week and in later sessions of the module as appropriate
- The key steps to making progress are to read more and to play around with a Linux system as much as possible

Advanced use of the terminal: tmux [Optional]



- **tmux** – **t**erminal **mu**ltiple**x**er
- create sessions, attach and detach from a session, leaving it running exactly as it was
- create multiple windows
- create multiple panes within a window
- Makes it more convenient to use a terminal to manage a remote server
- There's a nice tmux blog post and a tmux home page.