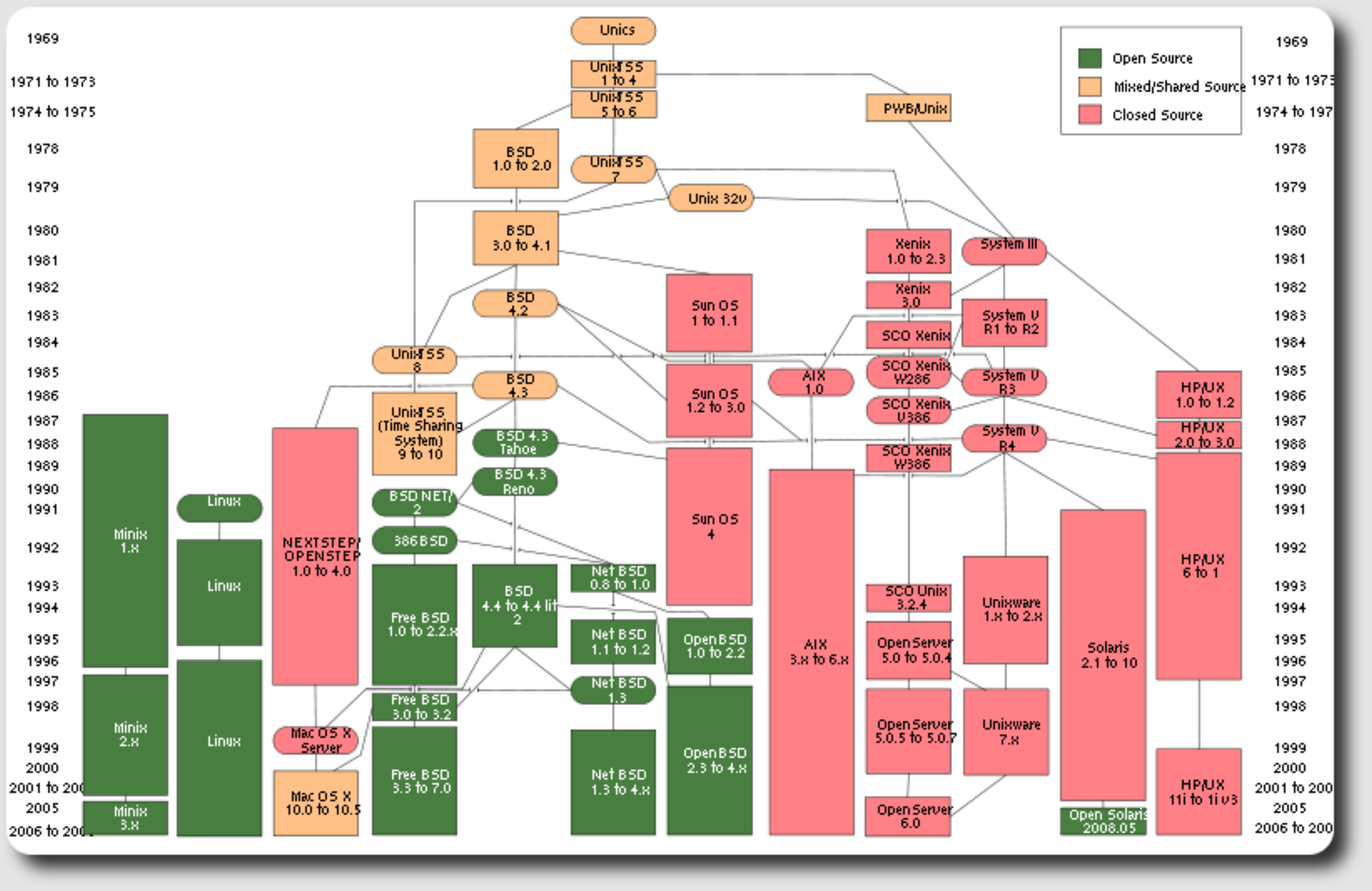
UNIX™/Linux Overview

Track 2 Workshop
November 2011
Nouméa, New Caledonia

UNIX History



Unix vs. Linux

- Are they the same?
  - Yes, at least in terms of operating system interfaces
  - Linux was developed independently from Unix
  - Unix is much older (1969 vs. 1991)
- Scalability and reliability
  - Both scale very well and work well under heavy load
  - (this is an understatement 🙂)
- Flexibility
  - Both emphasize small, interchangeable components
- Manageability
  - Remote logins rather than GUI
  - Scripting is integral
- Security
  - Due to modular design has a reasonable security model
  - Linux and its applications are not without blame

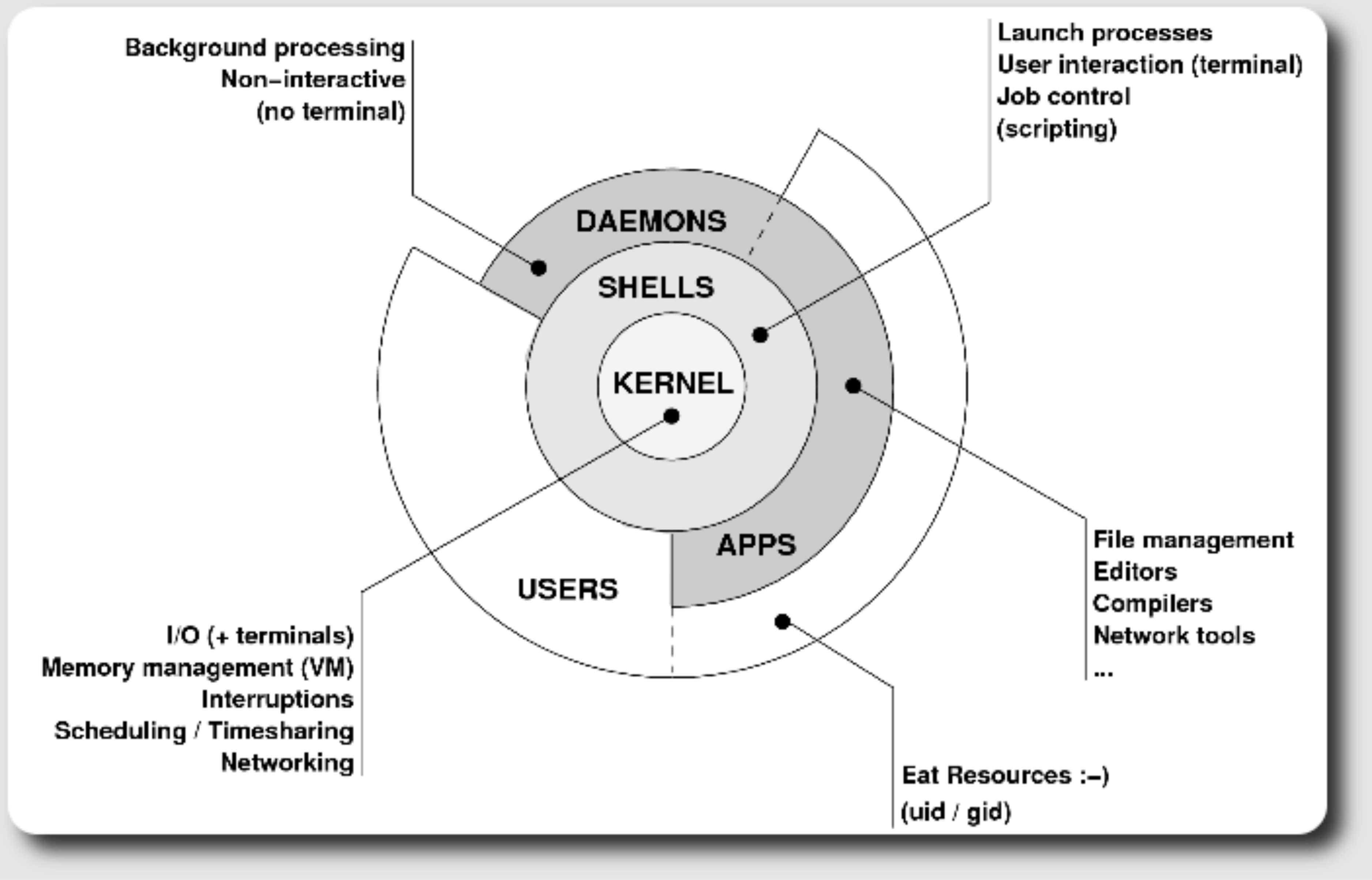
Is free software really any good?!

- The people who write it also use it
- Source code is visible to all
  - The quality of their work reflects on the author personally
  - Others can spot errors and make improvements
- What about support?
  - documentation can be good, or not so good
  - mailing lists; search the archives first
  - if you show you've invested time in trying to solve a problem, others will likely help you
  - http://www.catb.org/~esr/faqs/smart-questions.html

Is free software really any good?!

- Core Internet services run on free software
  - BIND Domain Name Server
  - Apache web server (secure SSL as well)
  - Sendmail, Postfix, Exim for SMTP/POP/IMAP
  - MySQL and PostgreSQL databases
  - PHP, PERL, Python, Ruby, C languages
- Several very high profile end-user projects
  - Firefox, original Netscape browser
  - OpenOffice
  - Thunderbird
  - Ubuntu

The Unix System



Kernel

The "core" of the operating system

Device drivers

communicate with your hardware

block devices, character devices, network devices, pseudo devices

Filesystems

organise block devices into files and directories

Memory management

Timeslicing (multitasking)

Networking stacks - esp. TCP/IP

Enforces security model

Shells

Command line interface for executing programs

DOS/Windows equivalent: command.com or command.exe

Also programming languages for scripting

DOS/Windows equivalent: batch files

Choice of similar but slightly different shells

sh: the "Bourne Shell". Standardised in POSIX

csh: the "C Shell". Not standard, but includes command history

bash: the "Bourne-Again Shell". Combines POSIX standard with command history.

Others: ksh, tcsh, zsh

User processes

The programs that you choose to run

Frequently-used programs tend to have short cryptic names

"`ls`" = list files

"`cp`" = copy file

"`rm`" = remove (delete) file

Lots of stuff included in most base systems

editors, compilers, system admin tools

Lots more stuff available to install too

Using the Debian/Ubuntu repositories

System processes

Programs that run in the background; also known as "daemons" ==>

Examples:

cron: executes programs at certain times of day

syslogd: takes log messages and writes them to files

inetd: accepts incoming TCP/IP connections and starts programs for each one

sshd: accepts incoming logins

sendmail (or other MTA daemon like Postfix): accepts incoming mail

Security model

Numeric IDs

user id (uid 0 = "*root*", the superuser)

group id

supplementary groups

Mapped to names

/etc/passwd, /etc/group (plain text files)

Suitable security rules enforced

e.g. you cannot kill a process running as a different user, unless you are "*root*"

Filesystem security

- Each file and directory has three sets of permissions
  - For the file's uid (user)
  - For the file's gid (group)
  - For everyone else (other)
- Each set of permissions has three bits: rwx
  - File: r=read, w=write, x=execute
  - Directory: r=list directory contents, w=create/delete files within this directory, x=enter directory
- Example: **`brian   wheel   rwxr-x---`**

Filesystem security

- The permission flags are read as follows (left to right)
- **-rw-r--r-- for regular files,**
- **drwxr-xr-x for directories**

We will see permissions in detail later.

Any questions?

?

Standard filesystem layout

/bin	essential binaries
/boot	kernel and boot support
/dev	device access nodes
/etc	configuration data
/etc/default	package startup defaults
/etc/init.d	startup scripts
/home/*username*	user's “home” directory
/lib	essential libraries
/sbin	essential sysadmin tools
/tmp	temporary files
/usr	programs & appl. data
/var	changing files (logs, E-mail messages, queues, …)

Don't confuse the the “root account” (/root) with the “root” (“/”) partition.

More filesystem details

/usr	
/usr/bin	binaries
/usr/lib	libraries
/usr/sbin	sysadmin binaries
/usr/share	misc application data
/usr/src	kernel source code
/usr/local/...	3rd party applications not installed with apt
/var	
/var/log	log files
/var/mail	mailboxes
/var/run	process status
/var/spool	queue data files
/var/tmp	temporary files

Partitioning considerations

Single large partition or multiple ?

A single partition is flexible, but a rogue program can fill it up…

Multiple partitions provides a more “protected” approach, but you may need to resize later, on older filesystems, or without a “Volume Manager”

- Is **/var** big enough ? /tmp?
- How much *swap* should you define?

Note...

Partitioning is just a logical division

If your hard drive dies, most likely *everything* will be lost.

If you want data security, then you need to set up mirroring or RAID with a separate drive.

Remember, "`rm -rf /`" on a mirror will erase everything on both disks ☺

Data Security <==> Backup

/dev

- Virtual files pointing to hardware or other
- e.g. /dev/sda or /dev/hda = the first harddisk (SCSI/SATA/SAS or IDE)
- In modern UNIX, including Linux, entries for each device under /dev are created dynamically
  - e.g. when you plug in a new USB device
- Some "devices" don't correspond to any hardware (pseudo-devices)
  - e.g. /dev/null is the "bit bucket"; send your data here for it to be thrown away
  - or /dev/random, which can be “read” to provide random data (useful for cryptography)

- https://help.ubuntu.com/10.04/installation-guide/amd64/device-names.html

Linux disk management

- Either direct partitioning:

```
# mount
/dev/sda1 on / type ext4 (rw)
```

- Or use of a Logical Volume Manager
  - Sits between the device and the filesystem

```
# mount
/dev/mapper/mail-root on / type ext3 (rw)
```

This allows resizing the volume under the filesystem, and making the device name irrelevant.

How Does Linux boot?

- The *BIOS* loads and runs the *MBR*
  - The *Master Boot Record* points to a default partition, or lets you select the boot partition
- The MBR code then loads the boot loader, LILO or GRUB
- This boot loader then reads its configuration parameters (usually under /boot) and presents the user with options on how to boot the system
- The kernel is loaded and started, filesystems are mounted, modules are loaded
- The init(8) process is started
- The system daemons are started

http://en.wikipedia.org/wiki/Linux_startup_process

Any questions?

?

Administration

The use of the *root* account is by default disabled – it doesn’t have a password!

The `sudo` program should be used to access root privileges from your own account instead.

Important Reads

- man hier
- man man

And, “`man any_unknown_command`” when you are in doubt.

Packages & Exercises

We'll reinforce some of these concepts using exercises...