UNIX BootCamp

- AfNOG IX
- May 2008
- Rabat, Morocco

AfNOG

BootCamp Summary Time Table

Day	Topic	Instructor
Saturday	Introduction to UNIX	PO
	Introduction to commands	HA
	Introduction to the Unix File System/Hierarchy	PR
	Privileges	HA
	Commands, File system and Privileges together	PR
Sunday	Editing f les (conf guration f les)	HA
	Editing cont. (More advanced)	HA
	Introduction to TCP/IP	PO
	More Networking	PR
	Summary	*

Introduction To UNIX

- AfNOG IX
- May 2008
- Rabat, Morocco

AfNOG

Why use UNIX?

- Scalability and reliability
  - has been around for many years
  - works well under heavy load
- Flexibility
  - emphasises small, interchangeable components
- Manageability
  - remote logins rather than GUI
  - scripting
- Security
  - Windows has a long and sad security history
  - Unix and its applications are not blameless though

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?

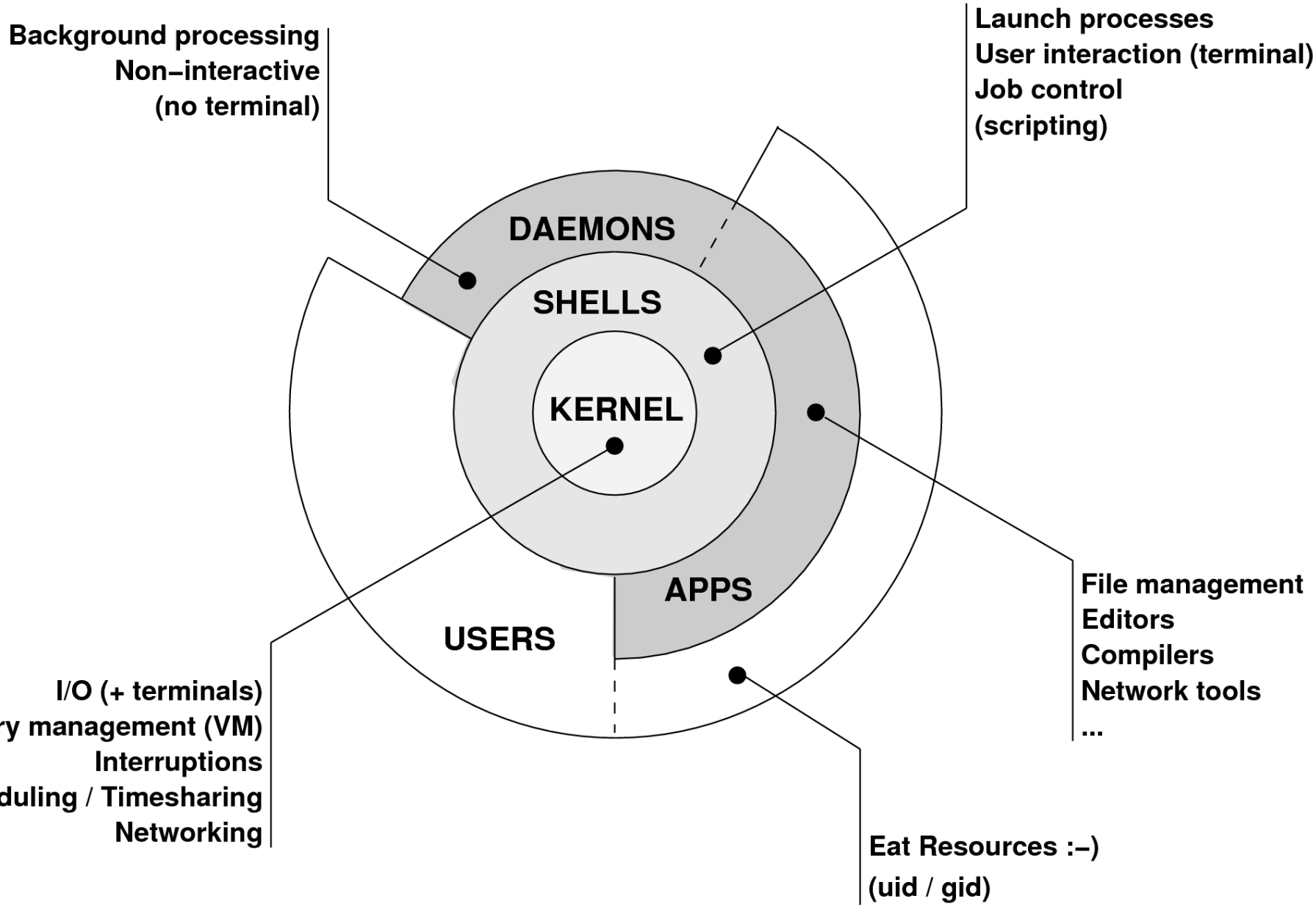
- Core Internet services run on free software
  - BIND Domain Name Server
  - Apache web server (secure SSL as well)
  - Sendmail, Postfix, Exim for SMTP/POP/IMAP
  - MySQL and PostgreSQL databases
  - PHP, PERL, C languages
- Several very high profile end-user projects
  - Firefox, original Netscape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

First topics:

- Unix birds-eye overview
- Partitioning
- FreeBSD installation

AfNOG

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 (multiprocessing)
- Networking stacks - esp. TCP/IP
- Enforces security model

Shell

- Command line interface for executing programs
  - DOS/Windows equivalent: command.com or command.exe
- 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. But distributed under GPL (more restrictive than BSD licence)

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 the base system
  - editors, compilers, system admin tools
- Lots more stuff available to install too
  - packages / ports

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 (other MTA daemon like Exim): 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)
  - /etc/pwd.db (fast indexed database)
- Suitable security rules enforced
  - e.g. you cannot kill a process running as a different user, unless you are "root"

Key differences to Windows

- Unix commands and filenames are CASE-SENSITIVE
- Path separator: / for Unix, \ for Windows
- Windows exposes a separate filesystem tree for each device
  - A:\foo.txt, C:\bar.txt, E:\baz.txt
  - device letters may change, and limited to 26
- Unix has a single 'virtual filesystem' tree
  - /bar.txt, /mnt/floppy/foo.txt, /cdrom/baz.txt
  - administrator choses where each FS is attached

Any questions?

.?

Some reminders about PC architecture

- When your computer turns on, it starts a bootup sequence in the BIOS
- The BIOS locates a suitable boot source (e.g. floppy, harddrive, CD-ROM, network)
- Disks are divided into 512-byte blocks
- The very first block is the MBR (Master Boot Record)
- The BIOS loads and runs the code in the MBR, which continues the bootup sequence

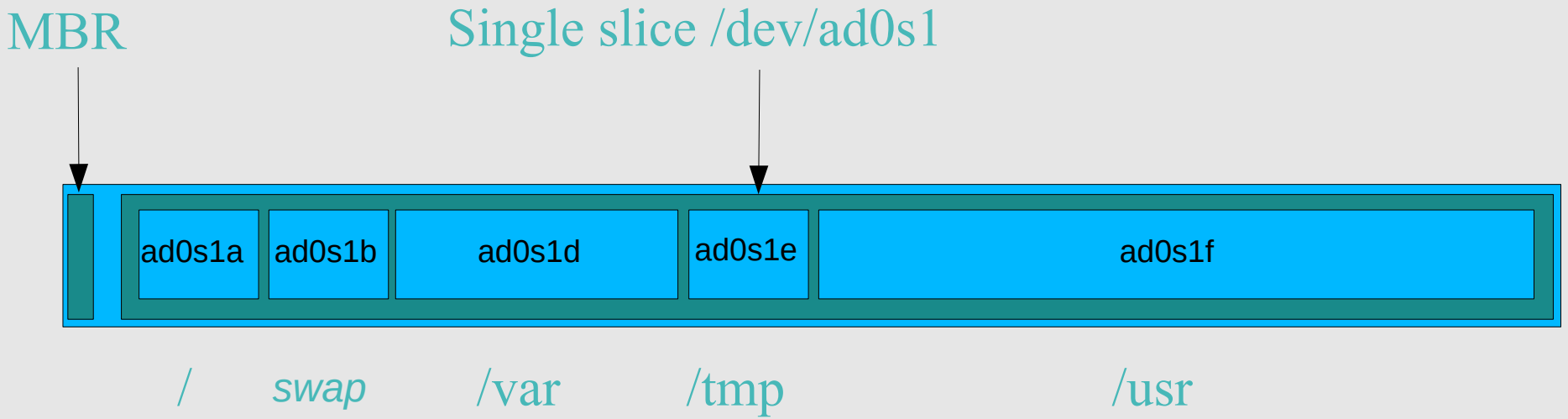
Partitioning

- The MBR contains a table allowing the disk to be divided into (up to) four partitions
- Beyond that, you can nominate one partition as an "extended partition" and then further subdivide it into "logical partitions"
- FreeBSD has its own partitioning system, because Unix predates the PC
- FreeBSD recognises MBR partitions, but calls them "slices" to avoid ambiguity

FreeBSD partitions

- Partitions (usually) sit within a slice
- Partitions called a,b,c,d,e,f,g,h
- CANNOT use 'c'
  - for historical reasons, partition 'c' refers to the entire slice
- By convention, 'a' is root partition and 'b' is swap partition
- 'swap' is optional, but used to extend capacity of your system RAM

Simple partitioning: /dev/ad0




```
/   (root partition)      ad0s1a     256MB
     swap partition       ad0s1b     ~ 2 x RAM
/var                      ad0s1d     256MB (+)
/tmp                      ad0s1e     256MB
/usr                      ad0s1f     rest of disk
```

'Auto' partition does this:

- Small root partition
  - this will contain everything not in another partition
  - /boot for kernel, /bin, /sbin etc.
- A *swap partition* for virtual memory
- Small /tmp partition
  - so users creating temporary files can't fill up your root partition
- Small /var partition
- Rest of disk is /usr
  - Home directories are /usr/home/<username>

Issues

- /var may not be big enough
- /usr contains the OS, 3rd party software, and your own important data
  - If you reinstall from scratch and erase /usr, you will lose your own data
- So you might want to split into /usr and /u
  - Suggest 4-6GB for /usr, remainder for /u
- Some people prefer a ramdisk for /tmp

```
# /etc/fstab: 64MB ramdisk
md   /tmp   mfs   -s131072,rw,nosuid,nodev,noatime   0   0
```

Or, see /etc/rc.conf later today. We can't do this due to limted RAM.

Core directory refresher

- / *(/boot, /bin, /sbin, /etc, maybe /tmp)*
- /var *(Log files, spool, maybe user mail)*
- /usr *(Installed software and home dirs)*
- Swap *(Virtual memory)*
- /tmp *(May reside under "/")*

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

Note...

- Slicing/partition 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 with a separate drive
  - Another reason to keep your data on a separate partition, e.g. /u
  - Remember, "`rm -rf`" on a mirror works very well.

Summary: block devices

- IDE (ATAPI) disk drives
  - /dev/ad0
  - /dev/ad1 ...etc
- SCSI or SCSI-like disks (e.g. USB flash, SATA)
  - /dev/da0
  - /dev/da1 ...etc
- IDE (ATAPI) CD-ROM
  - /dev/acd0 ...etc
- Traditional floppy drive
  - /dev/fd0
- etc.

Summary

- Slices
  - /dev/ad0s1
  - /dev/ad0s2
  - /dev/ad0s3
  - /dev/ad0s4
- Defined in MBR
- What PC heads call "partitions"

- BSD Partitions
  - /dev/ad0s1a
  - /dev/ad0s1b
  - /dev/ad0s1d ...etc
  - /dev/ad0s2a
  - /dev/ad0s2b
  - /dev/ad0s2d ...etc
- Conventions:
  - 'a' is /
  - 'b' is swap
  - 'c' cannot be used

Any questions?

.?

Installing FreeBSD

- Surprisingly straightforward
- Boot from CD or floppies, runs "sysinstall"
- Slice your disk
  - Can delete existing slice(s)
  - Create a FreeBSD slice
- Partition
- Choose which parts of FreeBSD distribution you want, or "all"
- Install from choice of media
  - CD-ROM, FTP, even a huge pile of floppies!

Installing Software in FreeBSD

- Several different methods
  - ports
  - packages
  - source
  - binary
- Meta installation wrapper we recommend is *portupgrade*
- We will go in to detail on these methods later in the workshop.

How Does FreeBSD Start?

- The *BIOS* loads and runs the *MBR*
  - The *MBR* is not part of FreeBSD
- A series of "bootstrap" programs are loaded
  - see “`man boot`”
    - `/boot.config` parameters for the boot blocks (optional)
    - `/boot/boot1` first stage bootstrap file
    - `/boot/boot2` second stage bootstrap file
    - `/boot/loader` third stage bootstrap
- Kernel is loaded, and perhaps some modules
  - controlled by `/boot/loader.conf`

How Does FreeBSD Start?

- The root filesystem is mounted
  - “root” = “/” or something like “ad0s1a”
- `/sbin/init` is run and executes the main startup script `/etc/rc`
- This in turn runs other scripts `/etc/rc.d/*`
  - /etc/rc.conf is used to decide whether a service is started or not and to specify options.

Finding more information

- Our reference handout
  - a roadmap!
- man pages
  - esp. when you know the name of the command
- www.freebsd.org
  - handbook, searchable website / mail archives
- "The Complete FreeBSD" (O'Reilly)
- comp.unix.shell FAQ
  - http://www.faqs.org/faqs/by-newsgroup/comp/comp.unix.shell.html
- STFW (Search The Friendly Web)