AfNOG 2005

Track E0:
Unix System Administration

Welcome!

- Who are we?
- Timetable and administrivia
- Objectives for the week
  - Learn your way around Unix/FreeBSD
  - TCP/IP network-based services
  - Security
  - Upgrading and maintenance

This is YOUR workshop!

- Stop us if we're speaking too fast
- Stop us if you don't understand anything
- Ask lots of questions!

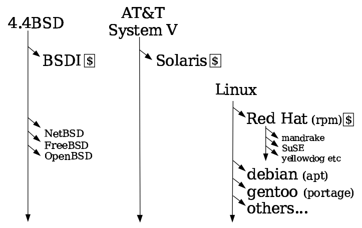
Why use UNIX?

- 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

Windows DOES NOT SCALE

- OK for 100 mailboxes
- But don't try to run 10,000 mailboxes
- Even Microsoft doesn't eat their own dogfood
  - hotmail
- Remote administration is painful
  - It's *still* a desktop OS
- Spend your entire life installing patches?
- Blue screen of death
- Commercial pricing but lousy support

Simplified Unix family tree



Why did we choose FreeBSD?

- It's Free!
- Optimised for performance on i386 hardware
  - NetBSD aims to run on many platforms
  - OpenBSD aims to provide enhanced security
- Well proven in real-world environments

Why not Linux?

- Too many distributions to choose from
- Red Hat used to be the *de-facto* choice for a reliable, free distribution
  - Now it has gone commercial
  - Mandrake and SuSE could follow suit
  - Fedora is "bleeding edge" and has short lifecycle
- BSD includes the kernel *and* the userland utilities in a single source tree
- BSD tends to be more "conservative"
  - emphasises stability and compatibility
  - compare: ipfw, ipfwadm, ipchains, iptables...

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

First topics:

- Unix birds-eye overview
- Partitioning
- FreeBSD installation

Key components of the Unix OS

- Kernel
- Shell
- User processes
- System processes
- Inter-process communication
- Security model
- Filesystem layout

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 other MTA daemon): accepts incoming mail

Inter-process communication

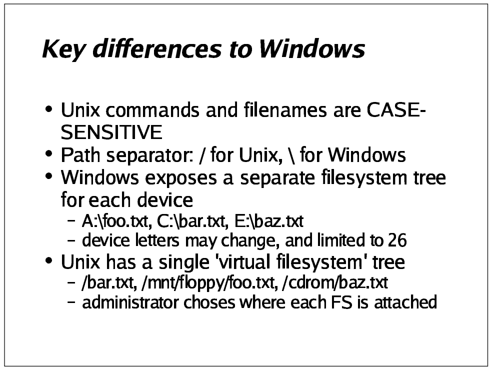
- Pipes: easy to use!
  - `grep hostname /etc/* | less`
- Other, more specialised mechanisms
  - fifos (named pipes)
  - sockets
  - System V IPC and shared memory

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"

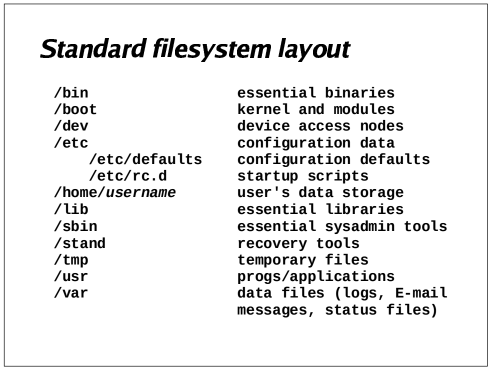
Filesystem security

- 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---`**



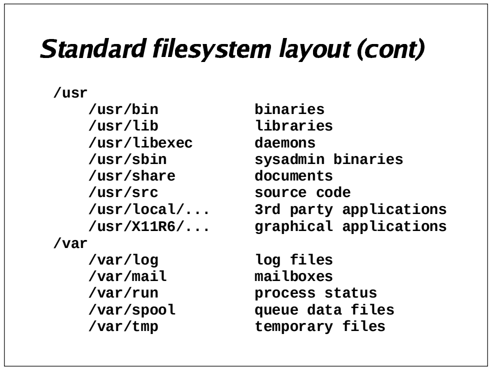
Key differences to Windows

- 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



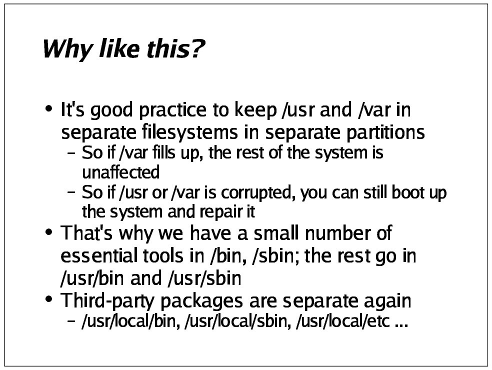
Standard filesystem layout

```
/bin                  essential binaries
/boot                 kernel and modules
/dev                  device access nodes
/etc                  configuration data
    /etc/defaults     configuration defaults
    /etc/rc.d         startup scripts
/home/username        user's data storage
/lib                  essential libraries
/sbin                 essential sysadmin tools
/stand                recovery tools
/tmp                  temporary files
/usr                  progs/applications
/var                  data files (logs, E-mail
                      messages, status files)
```



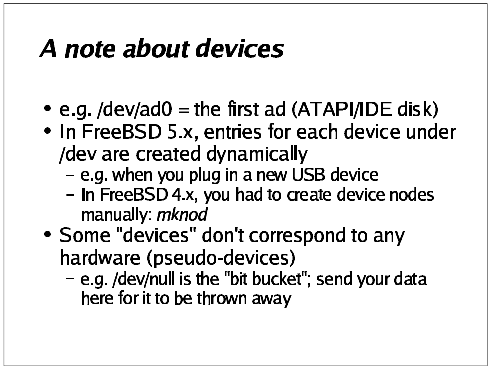
Standard filesystem layout (cont)

```
/usr
    /usr/bin          binaries
    /usr/lib          libraries
    /usr/libexec      daemons
    /usr/sbin         sysadmin binaries
    /usr/share        documents
    /usr/src          source code
    /usr/local/...    3rd party applications
    /usr/X11R6/...    graphical applications
/var
    /var/log          log files
    /var/mail         mailboxes
    /var/run          process status
    /var/spool        queue data files
    /var/tmp          temporary files
```



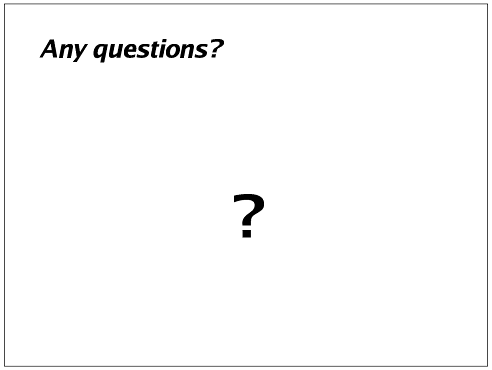
Why like this?

- It's good practice to keep /usr and /var in separate filesystems in separate partitions
  - So if /var fills up, the rest of the system is unaffected
  - So if /usr or /var is corrupted, you can still boot up the system and repair it
- That's why we have a small number of essential tools in /bin, /sbin; the rest go in /usr/bin and /usr/sbin
- Third-party packages are separate again
  - /usr/local/bin, /usr/local/sbin, /usr/local/etc ...



A note about devices

- e.g. /dev/ad0 = the first ad (ATAPI/IDE disk)
- In FreeBSD 5.x, entries for each device under /dev are created dynamically
  - e.g. when you plug in a new USB device
  - In FreeBSD 4.x, you had to create device nodes manually: *mknod*
- 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



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 devided 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

Some legacy problems

- Original PC architecture dates from 1980s
- Early disks were accessed by CHS (cylinder, head, sector). Cylinder was 10-bit number. Hence BIOS could not access beyond 1024 cylinders
- Nowadays we have Linear Block Addressing (LBA). However standard BIOS entry point is limited to 24-bit address. That limits BIOS to accessing 2^24 blocks, or first 8GB of disk

The 8GB problem

- Many OSes won't boot if they are above the 8GB point, since they use this BIOS call
- However, once the OS is booted, the problem goes away
  - FreeBSD talks directly to the hardware
  - "We don't need no steenking BIOS!"
- So only your root partition containing the kernel (/boot directory) has to be below 8GB; the rest is usable for data

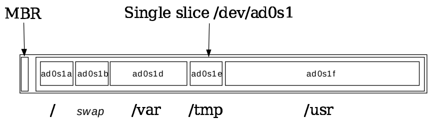
Partitioning

- 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
```

A more complex strategy

- Divide disk into 4 slices
  - s1: 0.5GB
    - Spare. For emergencies, MSDOS etc.
  - s2: 7GB
    - s2a: 0.25GB /
    - s2b: 0.5GB swap
    - s2d: 0.25GB /tmp
    - s2e: 1GB /var
    - s2f: 5GB /usr
  - s3: 7GB
    - same as s2
  - s4: rest of disk
    - s4a /u

Why like that?

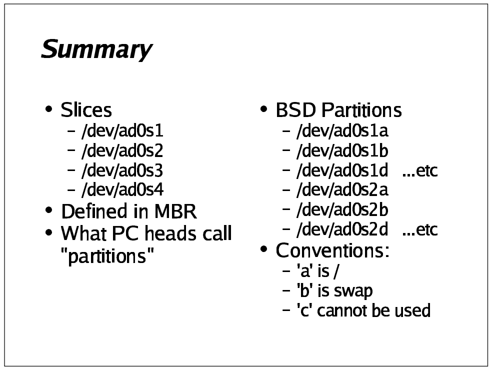
- Both s2a and s3a are below the 8GB level
- Aids in system upgrades/reinstalls
  - You can have a complete FreeBSD installation in s2, and later install a completely new version in s3
  - You can switch between the versions at bootup
  - When setting up configs in s3, you can mount s2 (read-only) to refer back to
- This is just a suggestion however. May not be appropriate or necessary in your case

Note...

- Slicing/partition is juts 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

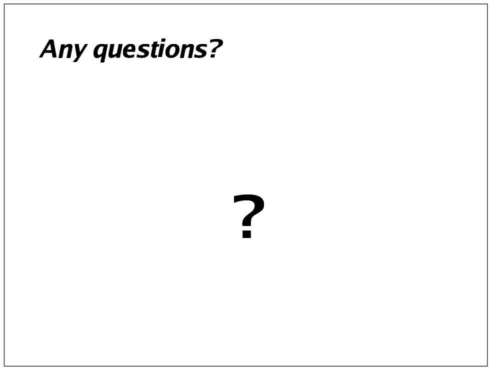
Summary: block devices

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



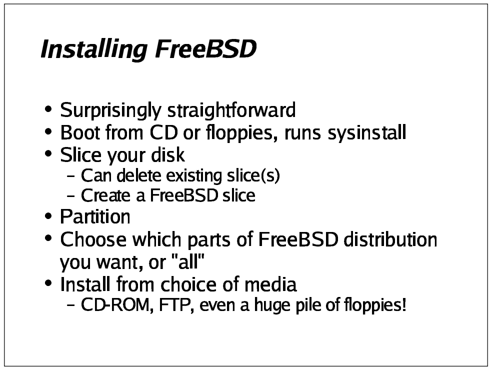
Summary

- 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



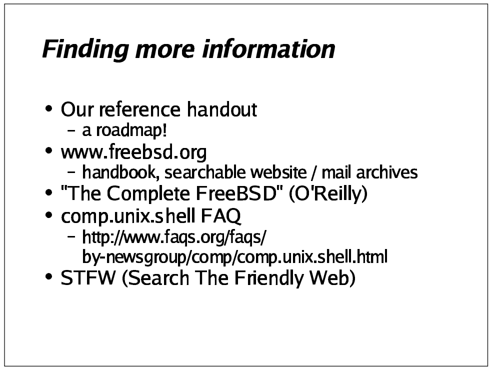
Any questions?

?



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!



Finding more information

- Our reference handout
  - a roadmap!
- 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)