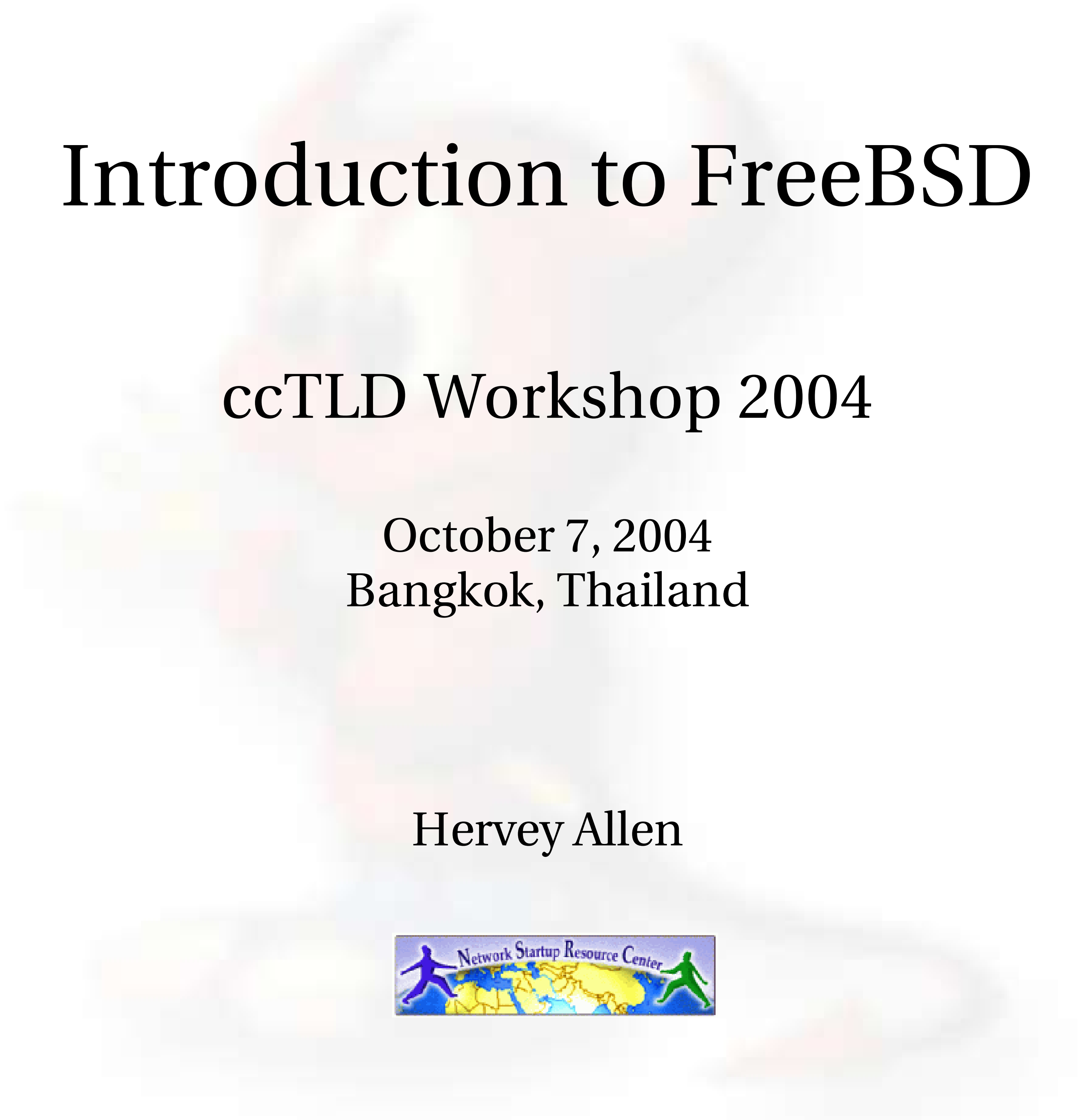


Introduction to FreeBSD

ccTLD Workshop 2004

October 7, 2004
Bangkok, Thailand

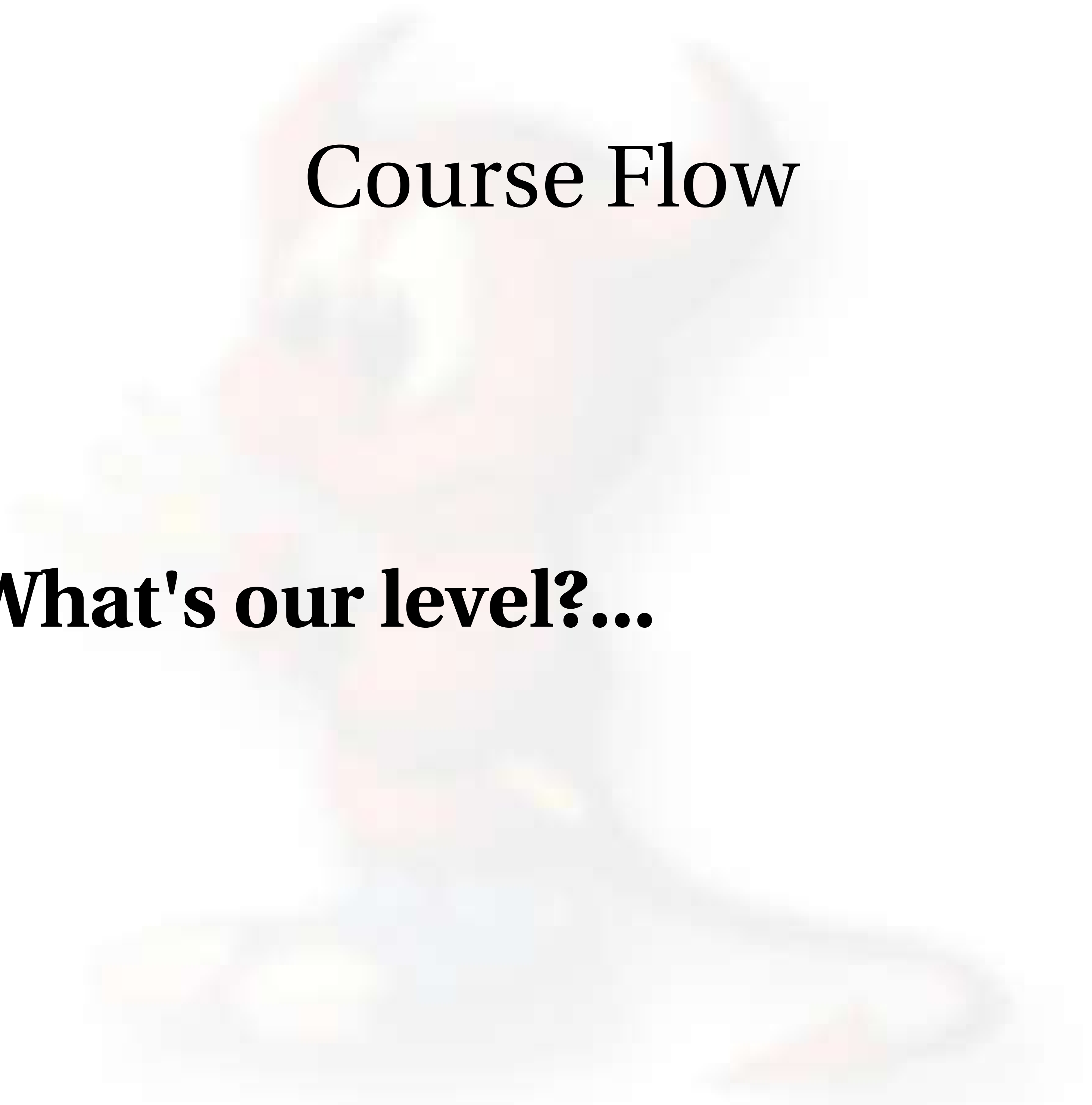
Hervey Allen

Welcome

- Introduction
  - Instructors:

    Hervey Allen
    Krit Witwiyaruj
    Pensri Arunwatanamongkol
    Eduardo Sztokbant
- Level of this session
- How the class runs
- What we'll do today

Note: this presentation is given in conjunction with exercises to reinforce the topics presented.



Course Flow

What's our level?...

Outline

- Why FreeBSD.
- Why UNIX.
- The World of FreeBSD.
- FreeBSD 5.2.1 installation.
- FreeBSD disk paritioning.
- FreeBSD directory structure.
  - /devfs
- The FreeBSD Unix File System (UFS/UFS2).
- How FreeBSD boots (man boot).
- Installation of FreeBSD.
- After Install /stand/sysinstall
- Commands, commands, commands…

Outline continued

- Create and remove user accounts.Discuss /etc/passwd, /etc/group, /etc/master.passwd.
- Basic editor usage with vi.
- Use of 'su' command for 'root', and /etc/sudoers.
- Use of the FreeBSD package facility (pkg_add).
- Install a package using /usr/ports.
- Other install methods, including:
  - From source
  - Binary
  - CVS
- Finding information about your system.
- Mounting filesystems.
- File permissions. Review commands “chmod” and “chown”.
- “ifconfig” to configure network cards and interfaces.

Outline continued

- The /etc/hosts file.
- Commands, programs, shells and paths.
- Shutdown and restart the server. Discuss initialization levels.
- Discuss FreeBSD services and how to know what's running.
- /etc/crontab and crontab format.
- The FreeBSD kernel and how to recompile it.
- Hardware support and kernel modules.
- Firewalls using ipfw
- Gnome vs. KDE and XWindows. Differences. Good for a Server?
- Logs and where they exist. We'll inspect logs and note /etc/syslog.conf.
- Summary.
- More resources.

Linux != UNIX

Why FreeBSD?

(This is our conclusion as well...)

- FreeBSD is built in a modular manner.
- Access to source code.
- Aimed at stability not user desktops.
- Industrial strength TCP/IP stack.
- Very, very good track record for stability and security.
- Scales to very large sizes for services.
- Superior file system.
- Superior password store (hashed db for passwords).
- Has a very rich collection of available software.

Why UNIX?

Note that FreeBSD and "UNIX" are very similar systems. That is, if you use Solaris, any other "BSD", etc. then understanding FreeBSD is of a great help.

It's important to understand this idea:

Linux != UNIX

Linux and UNIX look very similar, but underlying design is different. Still, if you know Linux or UNIX well, then using the other should be conceptually easy.

Why UNIX cont.?

Along with the strengths of FreeBSD, when you use UNIX you get (in general):

- Basic services scale to huge numbers.
- Incredibly stable (crashing is unusual).
- Security model is modular and relatively easy to implement.
- Extremely few memory leaks in core services.
- Very mature multi-processor and multi-process subsystems at the kernel level.
- Does not require a GUI to provide services.
- Extremely interoperable as standards are followed.

Why FreeBSD Part 2

Taken directly from http://www.freebsd.org/features.html

- Merged virtual memory and filesystem buffer cache
- Compatibility modules (Linux, SCO UNIX, NetBSD, etc.)
- Kernel Queues.
- Accept filters in the kernel.
- Soft Updates.
- Support for IPsec and IPv6.
- Security services such as:
  - Kerberos authentication
  - Vritual servers using “jails”
  - Access lists for TCP wrappers.

The World of FreeBSD

Start here: http://www.freebsd.org/

- RELEASE (4.10 and 5.2.1)
- STABLE (4.10)
- CURRENT (5.2)
- Ports
- Documentation
  - FreeBSD Handbook

Installing FreeBSD (5.2.1)

- How can you install? (FreeBSD Handbook section 2.2.6)
  - A CDROM or DVD
  - Floppy disks (including preconfigued install)
  - An FTP site, going through a firewall, or using an HTTP proxy, as necessary
  - An NFS server
  - A DOS partition on the same computer
  - A SCSI or QIC tape
  - A dedicated parallel or serial connection

FreeBSD Disk Organization

If you wish to understand how FreeBSD organizes and views disks then read section 3.5 of the FreeBSD handbook for an excellent and succinct description.

If you come to disk partitioning from a Windows perspective you will find that UNIX (FreeBSD, Linux, Solaris, etc.) *partitions* data very effectively and easily.

In FreeBSD a “slice” is what you may consider to a “partition” under Windows.

FreeBSD Partition Schemes

Partition	Usage
a	Root partition (/)
b	swap partition
c	Not used for filesystems.
d	Supposedly not often used.
e/f	/tmp, /usr, etc...

View information using "`df -h`" and "`swapinfo`"

FreeBSD Disk Slices

Sample Output to view disk slices from "`fdisk -s`"

```
/dev/ad0: 77520 cyl 16 hd 63 sec
Part        Start        Size Type Flags
   1:          63     8385867 0x0b 0x80
   2:     8385930     8385930 0xa5 0x00
   3:    16771860      208845 0x83 0x00
   4:    16980705    61159455 0x0f 0x00
```

This is a 40GB disk with 3 operating systems spread across four slices. The operating systems include Windows 2000 (1), FreeBSD (2), Linux (3) and the 4th partition is a DOS swap slice for Windows 2000.

FreeBSD Partitions in a Slice

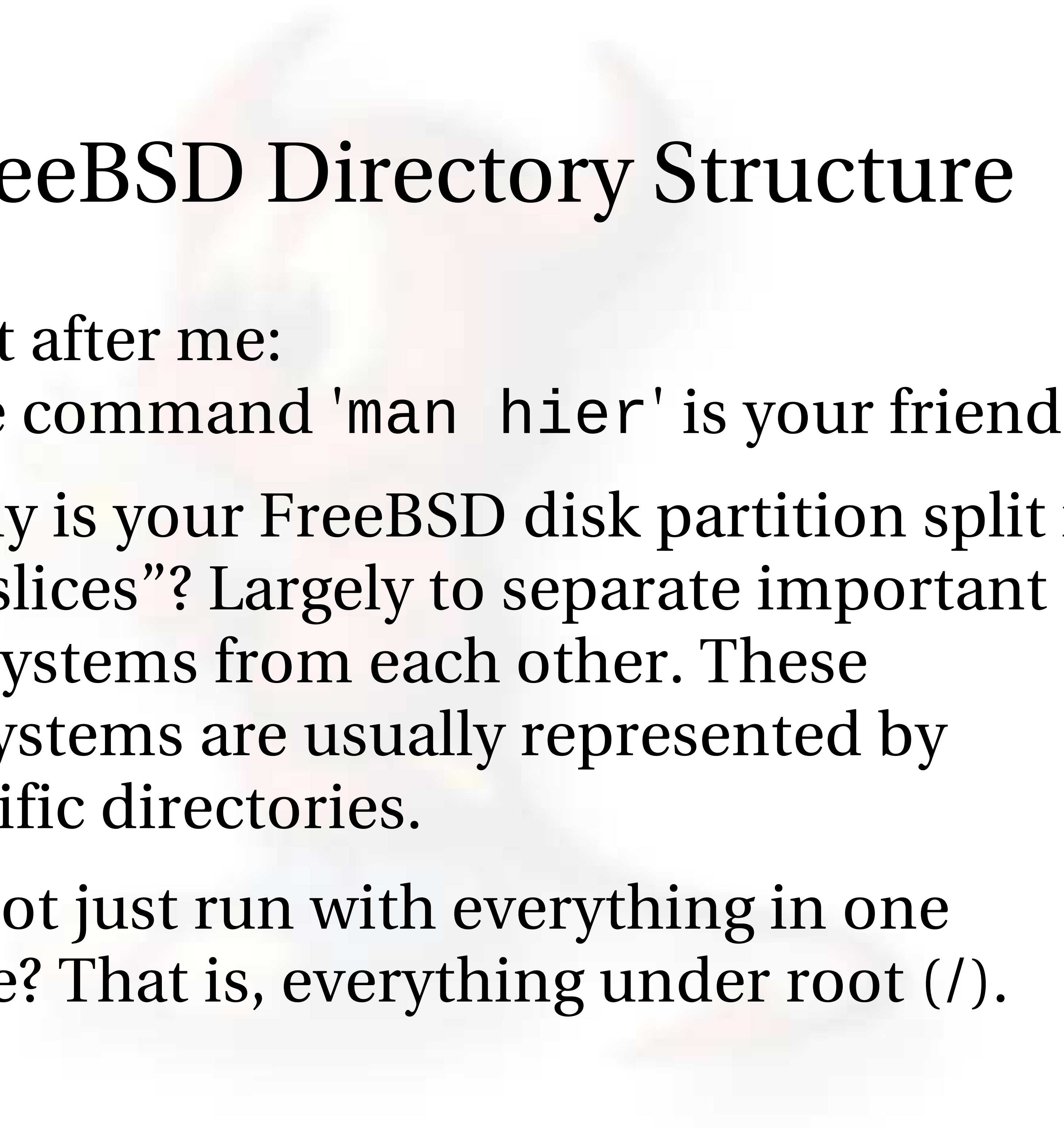
You can see more detailed information about your disk slices by just typing "`fdisk`"

To see the partitions in a FreeBSD slice use "`df -h`":

```
Filesystem     Size   Used  Avail Capacity  Mounted on
/dev/ad0s2a    248M   194M    34M     85%    /
devfs          1.0K   1.0K     0B    100%    /dev
/dev/ad0s2e    248M    18K   228M      0%    /tmp
/dev/ad0s2f    2.4G   2.0G   278M     88%    /usr
/dev/ad0s2d    248M    16M   212M      7%    /var
```

And use "`swapinfo`" to see the swap partition:

```
Device           1K-blocks     Used    Avail Capacity
/dev/ad0s2b         760616        0   760616     0%
```



FreeBSD Directory Structure

Repeat after me:
  “The command '`man hier`' is your friend.”

So, why is your FreeBSD disk partition split in to “slices”? Largely to separate important file systems from each other. These filesystems are usually represented by specific directories.

Why not just run with everything in one place? That is, everything under root (`/`).

FreeBSD Directory Structure cont.

Advantages of a single filesystem:
- Easier to resize if you want to make it larger.
- Easier conceptually for some people.

Advantages of multiple filesystems:
- If one system fails other systems can still work:
  - User fills up disk with runaway program.
  - Power failure only damages one file system.
- FreeBSD can optimize layout of files based on the use for the filesystem.
- Logical separation of functionality, thus improving security. I.E. root can be read only.

A Few FreeBSD Directories

- Structure of partitions/directories:
  - /   ("root")
  - /usr
  - /var
  - /tmp
  - /home
  - devfs
  - swap

“/” Root

The root partition is where critical system files live, including the programs necessary to boot the system in to “single user” mode.

The idea is that this part of the system does not grow or change, but rather stays isolated from the rest of the operating system.

A simplistic install might look like this:

- /boot (approx. 100Mb)
- swap (1 to 2 x installed RAM)
- / (the rest of the hard drive)

/usr

Is used for system software like user tools, compilers, XWindows, and local repositories under the /usr/local hierarchy.

If one has to expand* this partition for additional software, then having it separate makes this possible.

*We'll discuss this. We don't always install FreeBSD with a separate /usr partition.

/var

This is where files and directories that consistently change are kept. For example, webserver logs, email directories, print spools, etc.

On a server it is a good idea to have /var in a separate partition to avoid having it fill your other filesystems by accident.

/tmp

This is where users and application can save temporary files. By default quotas are not enabled in FreeBSD, thus it's possible for a user, or program, to fill the /tmp partition on purpose or by accident.

Sometimes /tmp resides in /var/usr/tmp, but many applications expect to have access to /tmp.

Another convention in the Unix world is to use /scratch, or /var/tmp instead of /tmp.

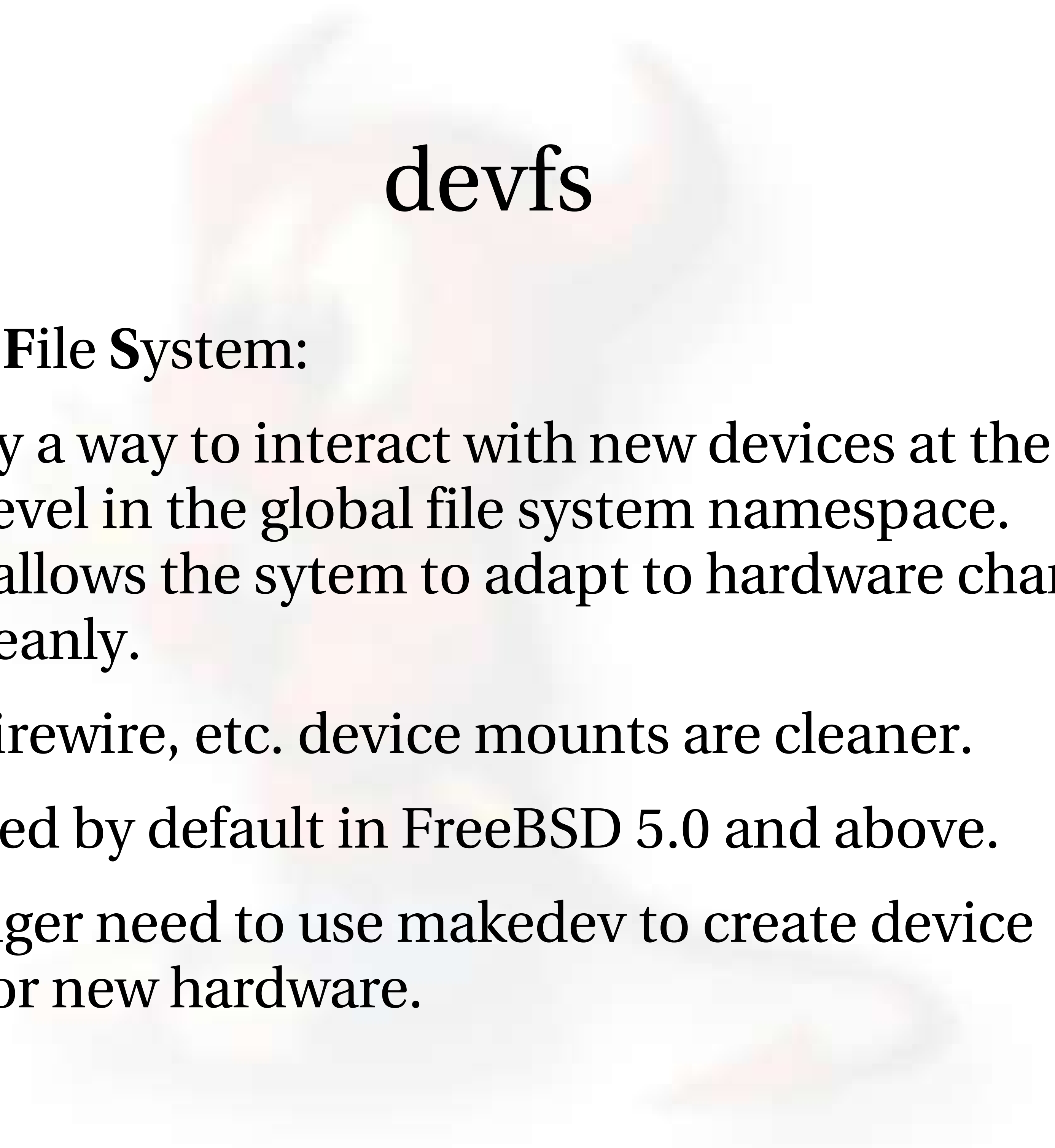
*Consider a logical link from /tmp to /var/tmp

/home

This is where user directories are kept. Often user's email is not kept in this partition.

If you don't use quotas then users can easily fill this partition.

An example: a user writes a program that produces a text file. On accident they create an infinite loop that writes text to a file and they don't notice this. This could fill your /home partition rapidly.



devfs

DEVice **F**ile **S**ystem:

Basically a way to interact with new devices at the kernel level in the global file system namespace. DEVFS allows the sytem to adapt to hardware changes more cleanly.

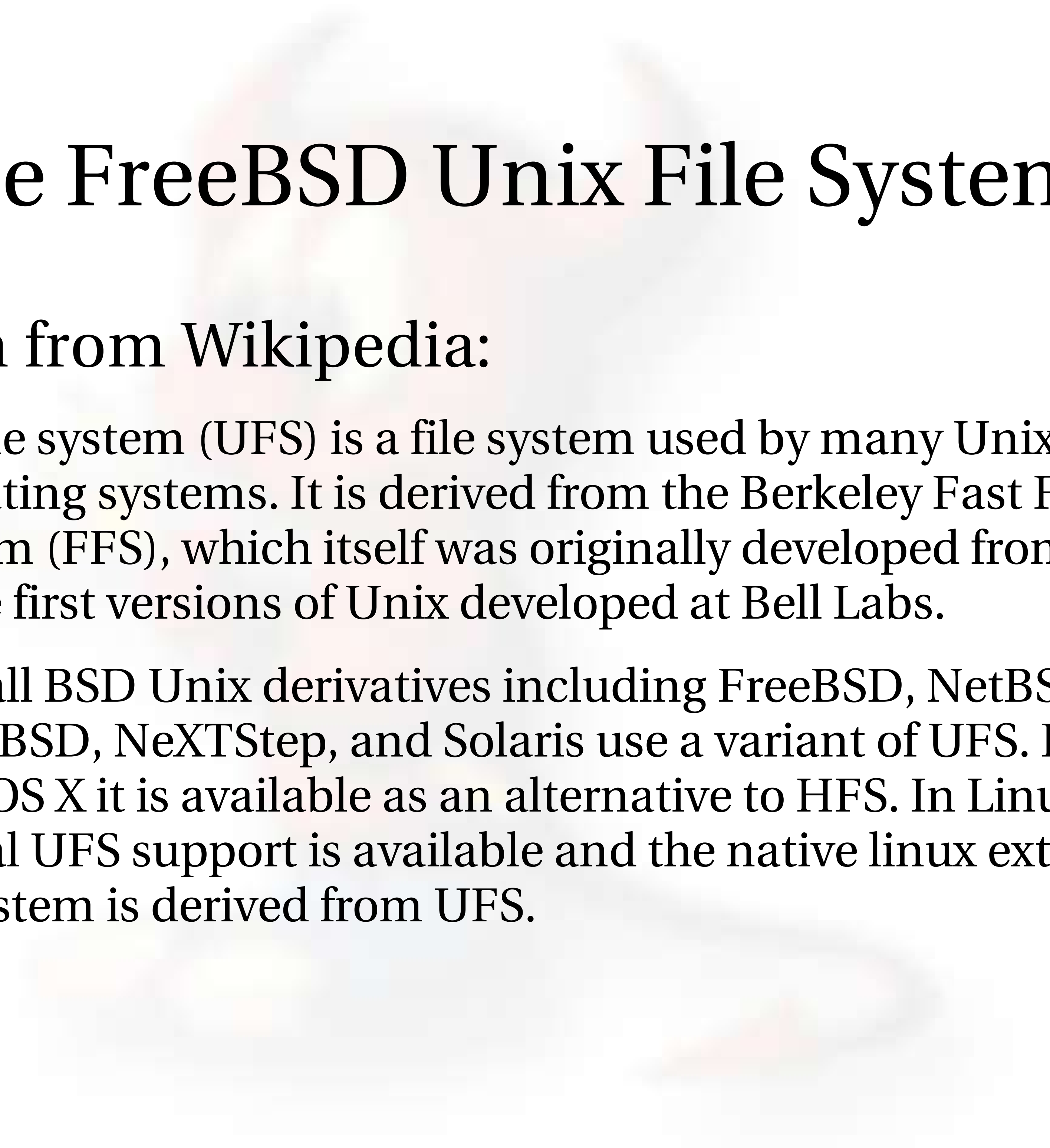
- USB, firewire, etc. device mounts are cleaner.
- Included by default in FreeBSD 5.0 and above.
- No longer need to use makedev to create device nodes for new hardware.

swap

Swap is where virtual memory lives. Swap is it's own filesystem.

You can run without swap, and your PC may run faster, but this is dangerous if you run out of memory.

There are several opinions about what is the optimal swap size. This can depend on what type of services you run (databases need more swap). The general rule of thumb is that swap size should be somewhere between your RAM and twice your server's RAM.



The FreeBSD Unix File System

Taken from Wikipedia:

UNIX file system (UFS) is a file system used by many Unix operating systems. It is derived from the Berkeley Fast File System (FFS), which itself was originally developed from FS in the first versions of Unix developed at Bell Labs.

Nearly all BSD Unix derivatives including FreeBSD, NetBSD, OpenBSD, NeXTStep, and Solaris use a variant of UFS. In Mac OS X it is available as an alternative to HFS. In Linux, partial UFS support is available and the native linux ext2 filesystem is derived from UFS.

FreeBSD UFS cont.

UFS2 and Soft Updates make for a powerful combination:

- Data is clustered on cylinders to reduce fragmentation.
- Block level fragmentation to avoid wasting disk space when large block sizes are used.
- Extended attribute support.
- Support for 1TB file systems.
- Fast file system creation using “lazy” inode initizializtion.
- Soft updates to dramatically improve metadata operations.
- UFS is journaled so no need for fsck on large drives.

FreeBSD UFS cont.

To learn more about UFS and Soft Updates:

UFS Definition from Wikipedia:

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

Little UFS2 FAQ:

http://lists.freebsd.org/pipermail/freebsd-current/2003-April/001444.html

Disk Tuning (Soft Udpates):

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/configtuning-disk.html#SOFT-UPDATES

Inode Definition from Wikipedia:

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

How FreeBSD Boots

Initial boot items are in /boot (this resides under "/", or in it's own partition).

boot0:

- Copy of MBR is in /boot/boot0. MBR is at start of the boot disk and is 512 bytes in size. If you use lilo, grub, or other MBR then this is not relevant.

boot1/boot2 or Stage 1 and 2:

- /boot/boot1 is 512 bytes in size and runs /boot/boot2.
- /boot/boot2 is more complex and runs /boot/loader.

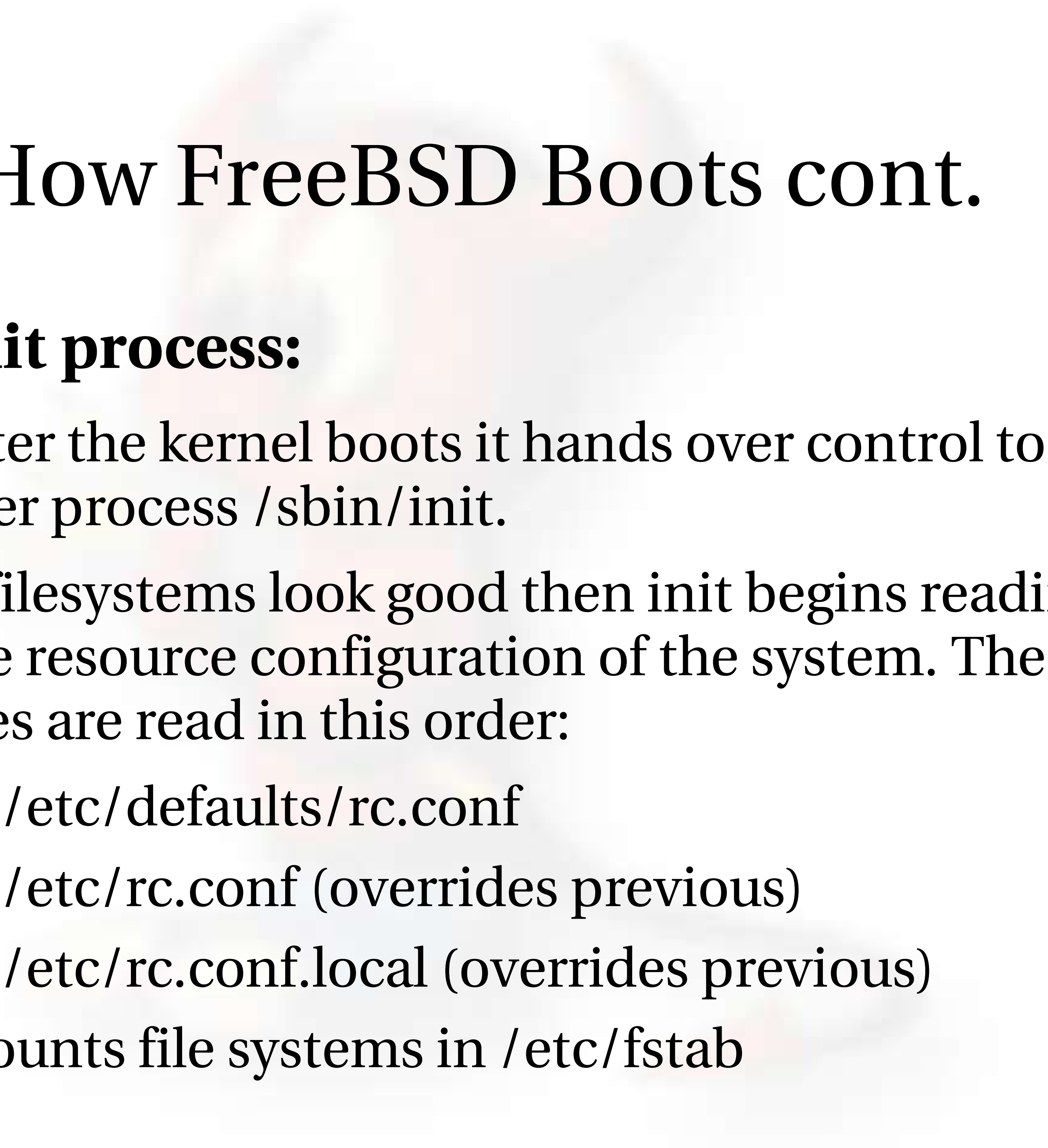
How FreeBSD Boots cont.

Stage 3 or /boot/loader:

- Probes for consoles and disk
- Reads in this order:
  - /boot/loader.rc
  - /boot/defaults/loader.conf
  - /boot/loader.conf to override previous
- Kernel and modules are loaded after a 10 second wait for key press. Interactive prompt available.

For more discussion and examples see:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/boot-blocks.html



How FreeBSD Boots cont.

The init process:

- After the kernel boots it hands over control to the user process /sbin/init.
- If filesystems look good then init begins reading the resource configuration of the system. These files are read in this order:
    - /etc/defaults/rc.conf
    - /etc/rc.conf (overrides previous)
    - /etc/rc.conf.local (overrides previous)
- Mounts file systems in /etc/fstab

How FreeBSD Boots cont.

The init process cont.:

- Once file systems are mounted then the following starts:
  - Networking services
  - System daemons
  - Locally installed package daemons (/usr/local/etc/rc.d scripts)

Init process and shutdown:

- When `shutdown` is called then init runs the scripts /etc/rc.shutdown.

Installing FreeBSD (5.2.1)

Sample install session...

Boot from CD-ROM

Pick default FreeBSD install

Choose Express install option

Say OK to geometry warning

Delete any slices laying around

Use "A" (entire disk) for FreeBSD slice

Q to finish disk partition

Install FreeBSD BootMgr

Create partitions like this:

- / 1GB
- /var 1GB with SoftUpdate on
- swap 1GB
- /usr (rest of disk with Softupdates on)

Q to finish partition creation

Choose 5, or X-Developer for install

Choose to install Ports

Click Exit twice to get to media dialogue
Choose to install from CD/DVD
Say "Yes" to last chance to set options

- set root password
- add users
- configure addtion network interfaces
    - configure dc0
        - no dhcp
        - host = int/labnn
        - domain = workshop.th
        - ipv4 gw = 203.159.31.1
        - nameserver = 203.159.0.1
        - ipv4 address = 203.159.31.nnn
        - netmask (calc) = 255.255.255.0
- configure your mouse, turn on mouse daemon
- configure xfree86
    - monitor at 1280x1024 @ 74 Hz
    - define screen size
    - save /etc/X11/XF86Config
- Configure Gnome
    - Get from CD/DVD
- Add package bash
    - install packages
        - shells
- Add package sudo (under security)
- Set timezone

remove CD-ROM
reboot

Installing FreeBSD cont.

First pick the type of install:

- Standard
- Express
- Custom

During install you must partition and slice.

After install use Configure to:

- Install Distribtutions
- Packages
- Configure network, accounts, XFree86, Timezone, mouse, startup, etc.

After Install /stand/sysinstall

To configure most any aspect of FreeBSD after installation use:

```
    /stand/sysinstall
```

You can configure network interfaces, install packages, distribution sets, add new users, set timezones, configure your X server, and much more.

Almost all of this functionality is available to you from the command line as well.

The FreeBSD project is working on a new installer program that is graphically based.

Basic Commands

- cp, cd*, ls, mkdir, mv, rm y man
  - (*built in command shell commands).
- Where are commands located?
- /bin, /usr/bin, /usr/local/bin, /sbin, /usr/sbin
  - The difference between “sbin”, “bin” and “ /usr”
- If you know DOS:
  - cp = copy
  - cd/chdir = cd/chdir
  - ls = dir
  - mkdir = mkdir
  - mv = move (before it was copy and delete/erase)
  - rm = del[ete] and/or erase

Basic Commands cont.

Not a command, but we'll practice starting a detached process.

To do this you use the “&” symbol *after* the command you wish to run that opens a separate window.

For example, to open another terminal from within your terminal under XWindows (using KDE) you would type:

```
konsole &            (/usr/local/bin)
```

or for an old-style xterm:

```
xterm &                       (/usr/X11R6/bin)
```

More Commands

ps	ProceSs list. Show information for running processes
cat	ConCATenate a file to the default ouput (screen)
less	Display file pausing each page & allowing movement
more	Display file pausing each page, but no movement
tail	Display the end of a file (see "-f" option)
gzip	Compress file(s) using Lempel-Ziv coding
gunzip	Decompress zip'ped files
bunzip2	Uncompress files compressed with bzip2
tar	Manipulate Tape ARchive files.
grep	Search text/files for patterns (many variations)

Even More Commands

- apropos
- bg
- bzip2
- chgrp*
- chmod
- clear
- chown*
- “ctrl-u”
- date
- exec
- df
- dmesg
- du
- export
- file
- find

- gcc
- hexdump
- history
- id
- ifconfig*
- info
- init*
- kill
- ln
- locate
- lsof**
- mkdir
- “|” pipe
- man
- mkisofs

- mount*
- netstat
- nmap**
- ping
- pkg_add
- pkg_delete
- pkg_info
- printenv
- ps
- pwd
- reset
- route*
- rmdir
- script
- set

- su
- sysinstall
- sysctl
- swapinfo
- tcpdump
- top
- touch
- traceroute
- uname
- unset
- unzip
- users
- watch
- whereis
- which
- whoami

*root only for changes **Not installed by default in FreeBSD

Create, Remove, Update User Accounts

(FreeBSD Handbook section 8.6)

- chsh, chgrp, chpass, passwd, pw
- /etc/passwd, /etc/group, /etc/master.passwd, /etc/sudoers (note vipw)
- /usr/sbin/adduser
- /usr/sbin/rmuser
- /usr/share/skel
- /etc/profile
- /var/mail
- Note: /usr/compat/linux

/etc/passwd

The /etc/password file has the following format:

```
hervey:x:500:500:Hervey Allen:/home/hervey:/usr/local/bin/bash
```

i.e.:

```
user:pw:UID:GID:name:directory:shell
```

Using /etc/master.passwd the “pw” is represented by an “x”. If the user entry is actually something like a service, then the “shell” is represented with “/sbin/nologin”.

/etc/master.passwd

This file is used to hide encoded user passwords. Only root can (or should) read this file. /etc/pwd.db is a Berkeley db password database that is used by most applications for efficient user authentication.

/etc/master.passwd has the following format:

```
hervey:$1$qvAgYWGD$nLf/LpT1r0XXXXXXjMC/:1001:1001::0:0:Hervey
  Allen:/home/hervey:/usr/local/bin/bash
```

i.e.:

- User's login name.
- Users encoded password. If starts with “$1$” it's md5 encyrpted.
- User's ID number.
- User's login group ID.
- User's classification (unused).

/etc/master.passwd cont.

```
hervey:$1$qvAgYWGD$nLf/LpT1r0XXXXXXjMC/:1001:1001::0:0:Hervey
   Allen:/home/hervey:/usr/local/bin/bash
```

- Password change time. (0 means never)
- When the account expires (0 means never)
- General user information (like full name...)
- User's home directory.
- User's login shell.

/etc/group

Format is:

```
wheel:*:0:root,hervey,test
```

–Group name. 8 characters or less.

–Encrypted password. Rarely used. “*” as placeholder.

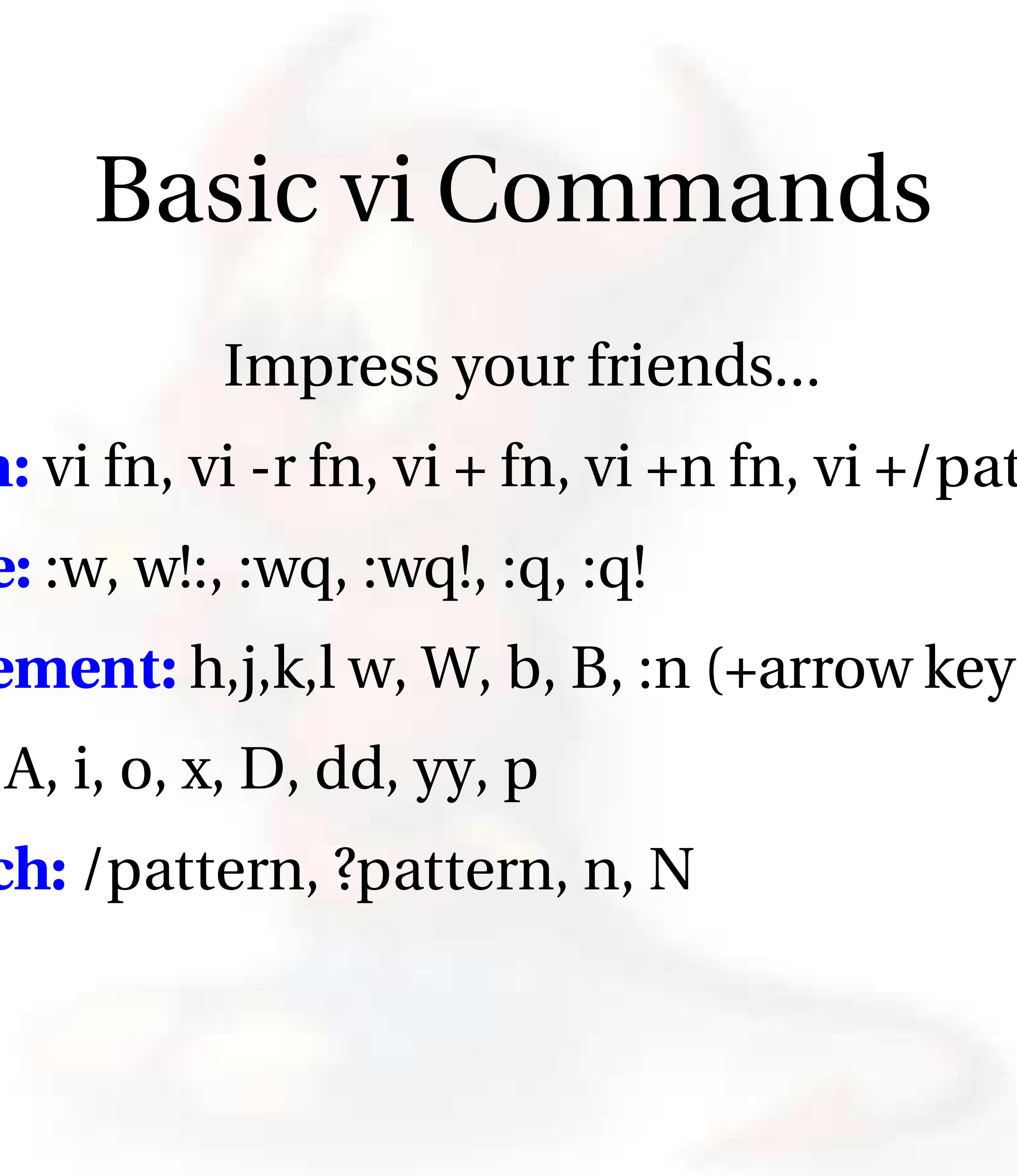
–Group Identifying number (GID).

–List of group members seperated by commas.

–User's login shell.

The vi Editor

- Why use vi? Why not emacs, xemacs, joe, pico, ee, etc.? (*Ask me* about "pico -w")
- vi exists in almost all flavors of Unix and Linux.
- If you have to work on a new machine, then vi will almost always be available to you.
- In reality, you are likely to use a different editor for more complex editing, but let's see what we can do with vi -->



Basic vi Commands

Impress your friends…

- **Open:** vi fn, vi -r fn, vi + fn, vi +n fn, vi +/pat fn
- **Close:** :w, w!:, :wq, :wq!, :q, :q!
- **Movement:** h,j,k,l w, W, b, B, :n (+arrow keys)
- **Edit:** A, i, o, x, D, dd, yy, p
- **Search:** /pattern, ?pattern, n, N

Commands - Programs – Shell – Path

- What's a “command” and a “program”?
- Why can't you always run all commands and programs on a system?
- How do you “fix” this?
- How do you see how things are configured for a user?
  - /usr/share/skel
  - /etc/profile
  - /home/user/.bashrc
  - /home/user/.bash_profile
  - set, printenv, export

Using the su Command

The “su” command is used to become a different userid, like root, without having to logout and log back in.

To use “su” to become root your userid has to be given permission to do this in “/etc/sudoers”.

You can allow users to run specific privileged commands using “/etc/sudoers” and “sudo”.

You can assign users to the “wheel” group and using “/etc/sudoers” you can allow them to run all commands (or some, but unusual).

More Uses for the su Command

Instead of having to open a root shell, you can run a privileged command like this:

```
sudo command
```

For example:

```
sudo less /etc/master.passwd
```

And, if you wish to open a different user shell and run their login scripts do:

```
su – userid
```

Configuring Network Interfaces

During boot if network device is recognized by static kernel module mapped to /etc/defaults/pccard.conf, then appropriate code is used.

After boot, using “ifconfig” you can see that device exists. Look for MAC address.

Initial configuration of specific device can be done with ifconfig, or try “`dhclient dev`”

If device works, edit /etc/rc.conf and put in device specific entries for each boot.

Configuring Network Interfaces cont.

Example lines in /etc/rc.conf for network device:

```
hostname="localhost.localdomain"
ifconfig_wi0="DHCP"
```

Set the hostname and indicate that network device wi0 will use DHCP to get network information. FreeBSD uses device names for each network device. “wi0” indicates the first “Wireless” card.

/etc/hosts

In this file you should have, at least, this line:

```
127.0.0.1   localhost         localhost.localdomain
```

For a private network you could use this file instead of using a DNS server. FreeBSD per default looks in /etc/hosts before asking DNS to resolve an IP address, or you can change this order using /etc/nsswitch.conf.

We'll add an entry in /etc/hosts for our network operations center server, or “noc”.

Shutdown and Restart a Server

How do you shutdown a FreeBSD box?

- shutdown 1 message
- halt
- init 0

And, to restart?

- reboot
- shutdown -r now
- init 6

Run Levels

And, what was “init 0”?

FreeBSD has the concept of run levels (but different from Linux).

Run-level	Signal	Action
0	SIGUSR2	Halt and turn the power off
1	SIGTERM	Go to single-user mode
6	SIGINT	Reboot the machine
c	SIGTSTP	Block further logins
q	SIGHUP	Rescan the ttys(5) file

So, in reality, you either run in single-user mode with “everything off” or your system is up and fulling running in multi-user mode.

Run Levels cont.

Multi-user mode:

- Startup configuration settings in /etc/defaults/rc.conf are executed (scripts in /etc/rc.d correspond).
- Local overrides and system-specific settings go in /etc/rc.conf (/etc/rc.local is deprecated).
- Filesystems are mounted as described in /etc/fstab.
- Network services and system daemons (see “rc.conf”) are started.
- System services are now in /etc/rc.d and can be started/stopped directly. Add service with:
  ```
  serviced_enable=”YES”
  ```
  in /etc/rc.conf. Do this before starting the service.
- Services with startup scripts installed by third party packages are located in /usr/local/etc/rc.d are run.

Running Services and Ports

- To view all services:
  - `ps -aux | more`
- What tcp ports are they using?
  - `sudo /usr/local/sbin/lsof -i`
  - `/usr/bin/netstat -an -f inet`
  - `sockstat -4`
- What starts at boot time? See in /etc/rc.d, /usr/local/etc/rc.d/ along with /etc/rc.conf
  - `Don't forget /etc/inetd.conf`

The “pkg” Commands

In general the pkg_add and pkg_delete facilities allow you to install and remove software on your system in an efficient and consistent manner.

The pkg_info command allows you to see what's installed, quickly, and to get detailed information about each software package that is installed.

Package Installation Using pkg_add

- In FreeBSD you can install by compiling from source, compiling from ports, and by using pkg_add.
- You can get “packages” from local source (a CD), off FreeBSD sites, or your local network.
- If you have a network connection and know the package name you can, literally, just type:

```
pkg_add -r package_name
```

- Set alternate package sites setting PACKAGEROOT, and PACKAGESITE variables, or specify the site explicitly like this:

```
pkg_add ftp://address/dir/package_name
```

Using pkg_info

You can do quite a few things with the “pkg” tool. The most common uses are generally to use pkg_add, pkg_info, and pkg_delete.

Find out if something is already installed:

```
 pkg_info       (list all installed packages)
 pkg_info | grep moz    (find all packages
                                containing
   “moz”)
```

Get more information about an already installed package:

```
pkg_info exec_package_name
```

Using pkg_delete

If you have a package you wish to remove you can simply type:

```
pkg_delete package_name
```

But, if you want to remove the package and all its dependent packages you would do:

```
pkg_delete -r package_name
```

But, *be careful* about doing this. You might want to check what will happen first by doing:

```
pkg_delete -n package_name
```

Installing from Ports

First you must have installed the /usr/ports collection during system installation. Otherwise, use /stand/sysinstall after installation and then choose Configure, Distributions, then Ports.

Once the “ports collection” is installed you can see the entire tree under /usr/ports. There are several thousand software packages available.

This collection contains minimal information so that you can “make” a software package quickly, and easily from a separate CD-ROM or network site containing the port source.

See section section 4.5 of the FreeBSD Handbook.

Installing from Ports cont.

To see if a software package exists as a port:

```
cd /usr/ports
make search name=package
make search key=keyword
```

Let's do this for “lsof” (LiSt Open Files):

```
cd /usr/ports
make search name=lsof (or “whereis lsof”)
```

And the output from this is:

```
Port:   lsof-4.69.1
Path:   /usr/ports/sysutils/lsof
Info:   Lists information about open files (similar to fstat
  (1))
Maint:  obrien@FreeBSD.org
Index:  sysutils
B-deps:
R-deps
```

Installing from Ports cont.

- From the previous page you'll note that the port is in /usr/ports/sysutils/lsof.
- If you have a network connection...
- You can simply type "`make install`"
- But, you might want to do:
  - `make`
  - `make install`

Installing from Ports cont.

You can install from cdrom. If you have a cdrom with the full ports distfiles, then simply mount it. Then you would do:

- `cd /usr/ports/sysutils/lsof`
- `make`
- `make install`

And the port will find the distfile on /cdrom instead of from the internet.

Finally, to fetch from a local server you can force fetch to do this in various ways. For example:

- `export MASTER_SITE_OVERRIDE=”ftp://local.site/distfiles/ fetch”`

Other Software Install Methods

There are three other methods to install software on your FreeBSD system. These are:

1.) The ports collection (/usr/ports)

2.) Installing from source (gcc make)

3.) Installing a binary file (already made)

You are most likely to install from packages, then ports, then from source, and then already made binary applications (e.g. java).

Note that installing from source has its advantages (customization and destination)

Installing from Source

It's likely you'll want to install software that's either not available as a package or port, or that you need to change or reconfigure before installation.

In such cases, you need to compile the software from source code.

It's very typical that software comes as a single “tar” archive that is compressed (tar.gz or .tgz)

An example of how to install from source -->

Installing from Source cont.

- Download a file fn.tar.gz to /usr/src.
- tar -xvzf /usr/src/fn.tar.gz
- cd /usr/src/fn-version
- ./configure
- make
- make install

This is if everything works, but now you don't have any good way to uninstall the software...

Installing a Binary File

This is much less common, but you can precompile a program for a specific version of FreeBSD.

Clearly this would be something that might be done with commercial applications that have restrictive licensing agreements.

Normally installation is done using a shell script that copies compressed files to the appropriate locations and updates configurations as needed.

Adobe's Acrobat Reader, Macromedia Flash plugin, etc. are examples (/usr/local/bin/acroread).

Installing with CVS

CVS: Concurrent Versions System

Somewhat detailed FreeBSD Handbook entry:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvsup.html

- Typical use for CVS and FreeBSD (other than software projects) is to keep your Ports collection up-to-date.
- To do this be sure you have installed the Ports collection at initial installation.
- Now install cvsup-without-gui from source if necessary:

Install cvsup

If you are using KDE or Gnome, then check:

```
pkg_info | grep cvs
```

If CVS is installed you can skip this. Otherwise:

```
cd /usr/ports/net/cvsup-without-gui
```

or

```
cd /usr/ports/net/cvsup
make
make install
make clean
```

Install cvsup cont.

Now copy the cvsup configuration file needed to tell CVS to upgrade your ports collection. A sample is located in /usr/share/examples:

```
    cp /usr/share/examples/cvsup/ports-supfile /root/.
```

Edit this file to look like this (line 50):

```
 # IMPORTANT: Change the next line to use one of the CVSup mirror sites
# listed at http://www.freebsd.org/doc/handbook/mirrors.html.
*default host=cvsup.th.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix
```

Install and Use cvsup

At this point you are ready to update your entire Ports collection with one simple command:

```
cvsup -g -L 2 /root/ports-supfile
```

"`-g`" : don't use graphical interface.

"`-L 2`" : verbosity level. Level 2 is verbose.

CVS Summary

CVS is a powerful and complex tool. For some more hints and information see:

```
man cvsup
info cvs
```

FreeBSD Handbook:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ports-using.html

Looking for More Information

Not only can you use commands to find information about your system, but you can look inside several files, and you can use the sysctl facility as well.

Example of files with useful information:

- /etc/motd
- /etc/resolve.conf
- /etc/services
- /etc/X11/XF86Config
- /etc/fstab

More information cont.

If you are used to "/proc" it's possible to compile support for this in to the kernel, but not normally used ("options LINPROCFS" in kernel conf file).

You can look in /boot/kernel for modules available and use "kldstat" to see what's loaded (kernel loadable modules = "kld").

Use "dmesg" to see what is reported during startup, including hardware and addresses.

Use of sysctl, such as:

```
sysctl -a,
sysctl -aN
sysctl kern.maxproc
```

And, see /etc/sysctl.conf

Logs – How to Know What's Up

- To configure what happens to events that are logged by applications using syslog, edit the file /etc/syslog.conf (see "`man syslog.conf`").
- Take a look at the file /var/log/messages. The "`tail`" command is very useful for this.
- To troubleshoot start by typing:

  ```
  tail -f /var/log/messages
  ```

  and in another terminal start and stop the service you are trying to debug.

Logs cont.

There are many log files. For example, if you run a webserver, like apache, all of the webserver logs are likely to be in /var/log/httpd

sendmail uses /var/log/maillog

There are multiple software packages to read and automatically generate reports based on logfiles. See:

- http://nsrc.org/security/index.html#logging

for some examples of available packages.

Mounting Filesystems

- If you want to mount a filesystem not listed in /etc/fstab then you need to use the mount command.
- First, you need to know what entry in the /dev directory describes the device you wish to mount (a cd, floppy, another hard drive, etc.).
- You, also, need to know what type of filesystem.
- For example, mounting a dos formatted floppy:
  - `mount -t msdos /dev/fd0 /mnt/floppy` or
  - `mount_msdosfs /dev/fd0 /mnt/floppy`

And, a USB flash pen drive:

  - `mount -t msdos /dev/da0s1 /mnt/usb`

Mounting Filesystems cont.

FreeBSD largely ignores the Linux convention of using:

– `mount -t filesystem /dev/dev /mnt/point`

and largely uses individual mount commands like:

```
mount_cd9660           mount_nullfs
mount_devfs            mount_nwfs
mount_ext2fs           mount_portalfs
mount_fdescfs          mount_procfs
mount_hpfs             mount_smbfs
mount_linprocfs        mount_std
mount_msdosfs          mount_udf
mount_nfs              mount_umapfs
mount_ntfs             mount_unionfs
```

So, to mount a cdrom you do (something like):

```
- mount_cd9660 /dev/acd0 /mnt/cdrom
```

File Permissions

- There are five categories and three types of permissions that you need to consider.
- The default file permissions are set with the umask command.
- There are two categories of permissions that relate to the user or group that is going to run a file (setuid, setgid).
- Available access permissions are “r” (read), “w” (write), and “x” (execute).
- You can assign permissions to world (a), group (g), and user (u).

File Permissions cont.

- A file belongs to a user. You can assign a file to another user or another group using the chown ("CHange OWNer") command.
- You can change permissions and/or owners for a group of files or for all files and all files in subdirectories using the chmod and chown commands.
- Finally, you can change directory permissions using the chmod command.

Crontab

- The “cron” service allows you to automatically run programs when you want.
- This is configured in /etc/crontab, and /var/cron/tabs/
- Use the command `crontab` in order to change the files that control how the cron daemon works.

Crontab cont.

- In addition you can specify who may and may not use cronjobs with /var/cron/allow and /var/cron/deny

A cron file that shows how a service is going to run has the following format:

```
Minute Hour Day Month Weekday Command
```

An example:

```
1 4 1 4 * /bin/mail user@dot.com < /home/user/joke
```

Send an email on the first of April.

Additional Topics

- Recompiling the FreeBSD kernel
- Kernel loadable modules and hardware support
- Firewalls
- Other software install methods
  - From source
  - From binary
  - Using CVS
- X Window vs. Gnome vs. KDE
- Summary
- More resources

The FreeBSD Kernel

- You might rebuild a kernel to add hardware support, additional filesystem support, etc.
- Kernel source, if installed, is in /usr/src/sys
  - "If there is not a /usr/src/sys directory on your system, then the kernel source has not been installed. The easiest way to do this is by running /stand/sysinstall as root, choosing Configure, then Distributions, then src, then sys." (FreeBSD Handbook 9.3)
- To rebuild your kernel you use the default configuration file, update settings as needed, then recompile the kernel, installing it in /boot.

Recompiling the FreeBSD Kernel

See FreeBSD Handbook section 9.3

- Config file in `/usr/src/sys/arch/conf`
- Example (old style):
  - `cp GENERIC /root/kernel/MYNEWKERNEL`
  - `ln -s /root/kernel/MYNEWKERNEL`
  - `/usr/sbin/config MYNEWKERNEL`
  - `cd ../compile/MYNEWKERNEL`
  - `make depend, make, make install`

Recompiling the FreeBSD Kernel cont.

Example (new style):

- `cd /usr/src`
- `make buildkernel kernconf=MYNEWKERNEL`
- `make installkernel kernconf=MYNEWKERNEL`

- Kernel installed as /boot/kernel
- Old kernel is in /boot/kernel.old
- If new kernel does not boot, go to boot loader prompt and type:
  - `unload`
  - `boot kernel.old`

Recompiling the FreeBSD Kernel cont.

The kernel config file has many options. For a more complete explanation of the various options see (e.g. on a PC with Intel CPU):

- /usr/src/sys/i386/conf/NOTES

Or look at the FreeBSD Handbook section 9.4 for some more examples.

Kernel and Hardware Support

FreeBSD is moving towards “modularizing” hardware support. That is “drivers” (kernel loadable modules) are loaded at boot time to support your systems' hardware.

Some hardware is still supported by statically loaded software directly in the kernel.

Some hardware use is optimized by setting kernel state using the sysctl facility.

Kernel Loadable & Static Modules

- Static (in conf) – built-in during recompile vs.
- Kernel loadable (kld) /boot/kernel modules.
- Autoloading using /etc/rc.conf directives and/or using loader.conf
- Address security in FreeBSD vs. Linux and modules.
- Commands `kldload, kldstat, kldunload`

Firewalls

Building an appropriate firewall ruleset for your situation requires thought:

- See FreeBSD Handbook section 10.8 to get started.
- Enable IP FireWall support (IPFW) by adding one, or more options to kernel configuration file.
- ipfw was updated to “ipfw2” in July 2002.
- Starting and stopping in /etc/rc.conf and /etc/rc.firewall.
- ipfw rules and firewall set are in /etc/rc.firewall.
- You can dynamically control ipfw as well:
  - `ipfw flush, ipfw enable, ipfw disable, ipfw flush, etc.`

X Windows – Gnome – KDE

The first thing to understand is that Gnome and KDE use the X graphical subsystem. Generally KDE programs run in Gnome and vice-versa.

For a server you do not need to run, or install, any of these.

You can run one, both, or other window managers like fwvm, windowmaker, etc.

X – Gnome – KDE cont.

- Which desktop environment is better? There's no correct answer to this.
- To configure how X runs you specify this in the file /etc/X11/XF86Config.
- You can configure everything using menu-based tools, but if you understand how things work you can use edit /etc/X11/XF86Config directly. (xf86cfg may be required in FreeBSD).
- To exit X you can press ALT-CTRL-Backspace.
- You can, also, go directly to a terminal using alt-ctrl-f2 through f8. alt-ctrl-f9 returns to X. Alt-ctrl-f1 displays X informational messages.

Summary

- FreeBSD is built in a modular manner.
- Access to source code.
- Aimed at stability not user desktops.
- Industrial strength TCP/IP stack.
- Very, very good track record for stability and security.
- Scales to very large sizes for services.
- Well proven and tested file system.
- Superior password store (hashed db for passwords).
- Has a very rich collection of available software.

More resources

This presentation is located here:

http://ws.edu.isoc.org/workshops/2004/ccTLD-bkk/intro/

- http://www.google.com/
- http://www.freebsd.org/
- http://www.freebsd.org/support.html
- O'Reilly books (http://www.oreilly.com/)
- http://www.freshports.org/
- http://www.freebsddiary.org/
- The SANOG mailing list (sanog@sanog.org)