



# Outline

- 1 Quick Introduction to Init Scripts
- 2 The daemons
  - sshd, apache2, cups, mysql, postgresql, ...
- 3 Notes about firewalls
- 4 Other stuff I can talk about if time
  - 1 Linux on laptops
  - 2 Accelerated graphics (OpenGL)
  - 3 Gentoo
  - 4 NFS
- 5 Summary & Question time

# Init Scripts

- Used to stop or start a daemon (service)
  - most often at reboot
  - But just because you install or change something doesn't mean you reboot!
- Init scripts (usually) live in `/etc/init.d/`
- You rarely have to make one
  - rely on the ones released by the distro
- They accept arguments: start, stop, restart, status, pause, zap
  - And sometimes others: reload, save

# Arguments

- start** Start the service and its dependencies (if not started)
- stop** Stop the service and those that depend on it
- restart** Usually calls stop, then start
- status** Tells you whether it thinks it is stopped or started
- pause** Like stop, but doesn't stop dependants
  - zap** Reset a stopped service's state that is still marked as "started" (a Zombie<sup>1</sup>)
- reload** Where supported, re-read changed config files
  - save** Save settings preserved across a reboot (e.g. sound mixer settings)

---

<sup>1</sup>When a daemon turns into a *zombie* it generally becomes hard to kill

# Ports

- Check out `(cat) /etc/services`
- Most will use port numbers  $\leq 1024$ .
  - Only the super user has permission to run servers on these ports
  - $> 1024$  are *ephemeral* ports





# SSH Security I

/var/log/messages

```
Jul 1 01:27:16 beef sshd(pam_unix)[32244]: authentication failure; logname= uid=0 euid=0
tty=ssh ruser= rhost=server1.pirch.com user=root
Jul 1 01:27:18 beef sshd[32244]: Failed password for root from ::ffff:69.44.59.145 port 48541
Jul 1 01:27:20 beef sshd[32246]: Invalid user admin from ::ffff:69.44.59.145
Jul 1 01:27:20 beef sshd(pam_unix)[32246]: check pass; user unknown
Jul 1 01:27:20 beef sshd(pam_unix)[32246]: authentication failure; logname= uid=0 euid=0
tty=ssh ruser= rhost=server1.pirch.com
Jul 1 01:27:23 beef sshd[32246]: Failed password for invalid user admin from
::ffff:69.44.59.145 port 48739 ssh2
Jul 1 01:27:24 beef sshd[32248]: Invalid user test from ::ffff:69.44.59.145
Jul 1 01:27:24 beef sshd(pam_unix)[32248]: check pass; user unknown

...

Jul 1 01:27:23 beef sshd[32246]<> admin from ::ffff:69.44.59.145 port 48739 ssh2
Jul 1 01:27:27 beef sshd[32248]<> test from ::ffff:69.44.59.145 port 48883 ssh2
Jul 1 01:27:31 beef sshd[32251]<> guest from ::ffff:69.44.59.145 port 49041 ssh2
Jul 1 01:27:35 beef sshd[32253]<> webmaster from ::ffff:69.44.59.145 port 49188 ssh2
Jul 1 01:27:44 beef sshd[32258]<> oracle from ::ffff:69.44.59.145 port 49538 ssh2
Jul 1 01:27:48 beef sshd[32260]<> library from ::ffff:69.44.59.145 port 49715 ssh2
Jul 1 01:27:52 beef sshd[32262]<> info from ::ffff:69.44.59.145 port 49882 ssh2
Jul 1 01:27:57 beef sshd[32264]<> shell from ::ffff:69.44.59.145 port 50053 ssh2
Jul 1 01:28:01 beef sshd[32267]<> linux from ::ffff:69.44.59.145 port 50239 ssh2
Jul 1 01:28:05 beef sshd[32269]<> unix from ::ffff:69.44.59.145 port 50413 ssh2
Jul 1 01:28:09 beef sshd[32271]<> webadmin from ::ffff:69.44.59.145 port 50578 ssh2
Jul 1 01:28:18 beef sshd[32276]<> test from ::ffff:69.44.59.145 port 50916 ssh2
Jul 1 01:28:26 beef sshd[32280]<> admin from ::ffff:69.44.59.145 port 51242 ssh2
Jul 1 01:28:31 beef sshd[32283]<> guest from ::ffff:69.44.59.145 port 51398 ssh2
Jul 1 01:28:36 beef sshd[32285]<> master from ::ffff:69.44.59.145 port 51584 ssh2
```



# SSH Security II

/var/log/messages

```
Jul 1 01:29:14 beef sshd[32306]<> admin from ::ffff:69.44.59.145 port 53101 ssh2
Jul 1 01:29:18 beef sshd[32308]<> admin from ::ffff:69.44.59.145 port 53271 ssh2
Jul 1 01:29:22 beef sshd[32310]<> admin from ::ffff:69.44.59.145 port 53439 ssh2
Jul 1 01:29:27 beef sshd[32312]<> admin from ::ffff:69.44.59.145 port 53599 ssh2
Jul 1 01:29:40 beef sshd[32319]<> test from ::ffff:69.44.59.145 port 54112 ssh2
Jul 1 01:29:44 beef sshd[32322]<> test from ::ffff:69.44.59.145 port 54281 ssh2
Jul 1 01:29:48 beef sshd[32324]<> webmaster from ::ffff:69.44.59.145 port 54452 ssh2
Jul 1 01:29:52 beef sshd[32326]<> user from ::ffff:69.44.59.145 port 54627 ssh2
Jul 1 01:29:57 beef sshd[32328]<> username from ::ffff:69.44.59.145 port 54800 ssh2
Jul 1 01:30:01 beef sshd[32331]<> username from ::ffff:69.44.59.145 port 54972 ssh2
Jul 1 01:30:05 beef sshd[32345]<> user from ::ffff:69.44.59.145 port 55136 ssh2
Jul 1 01:30:14 beef sshd[32350]<> admin from ::ffff:69.44.59.145 port 55487 ssh2
Jul 1 01:30:18 beef sshd[32352]<> test from ::ffff:69.44.59.145 port 55658 ssh2
Jul 1 01:30:40 beef sshd[32363]<> danny from ::ffff:69.44.59.145 port 56540 ssh2
Jul 1 01:30:44 beef sshd[32366]<> sharon from ::ffff:69.44.59.145 port 56715 ssh2
Jul 1 01:30:49 beef sshd[32368]<> aron from ::ffff:69.44.59.145 port 56886 ssh2
Jul 1 01:30:53 beef sshd[32370]<> alex from ::ffff:69.44.59.145 port 57080 ssh2
Jul 1 01:30:57 beef sshd[32372]<> brett from ::ffff:69.44.59.145 port 57256 ssh2
Jul 1 01:31:02 beef sshd[32375]<> mike from ::ffff:69.44.59.145 port 57428 ssh2
Jul 1 01:31:06 beef sshd[32377]<> alan from ::ffff:69.44.59.145 port 57607 ssh2
Jul 1 01:31:10 beef sshd[32379]<> data from ::ffff:69.44.59.145 port 57778 ssh2
Jul 1 01:31:14 beef sshd[32382]<> www-data from ::ffff:69.44.59.145 port 57947 ssh2
Jul 1 01:31:19 beef sshd[32384]<> http from ::ffff:69.44.59.145 port 58121 ssh2
Jul 1 01:31:23 beef sshd[32386]<> httpd from ::ffff:69.44.59.145 port 58296 ssh2
Jul 1 01:31:36 beef sshd[32393]<> backup from ::ffff:69.44.59.145 port 58822 ssh2
Jul 1 01:31:40 beef sshd[32395]<> info from ::ffff:69.44.59.145 port 59009 ssh2
Jul 1 01:31:44 beef sshd[32398]<> shop from ::ffff:69.44.59.145 port 59195 ssh2
```









## Private Key Authentication

```
fridge@pc-g33-9 ~ $ ssh-keygen -t dsa <enter>
Generating public/private dsa key pair.
Enter file in which to save the key (/home/fridge/.ssh/id_dsa): <enter>
Created directory '/home/fridge/.ssh'.
Enter passphrase (empty for no passphrase): <enter>
Enter same passphrase again: <enter>
Your identification has been saved in /home/fridge/.ssh/id_dsa.
Your public key has been saved in /home/fridge/.ssh/id_dsa.pub.
The key fingerprint is:
18:09:1b:c9:48:99:bd:13:cc:19:f9:a1:e5:45:eb:ad fridge@pc-g33-9
fridge@pc-g33-9 ~ $ scp .ssh/id_dsa.pub tapted@shade.ug.it.usyd.edu.au: <enter>
The authenticity of host 'shade.ug.it.usyd.edu.au (129.78.8.221)' can't be established.
RSA key fingerprint is 63:8c:b4:7f:7c:19:15:13:29:c9:31:63:e9:aa:0b:18.
Are you sure you want to continue connecting (yes/no)? yes <enter>
Warning: Permanently added 'shade.ug.it.usyd.edu.au,129.78.8.221' (RSA) to the list of known
hosts.
tapted@shade.ug.it.usyd.edu.au's password: <enter>
id_dsa.pub 100% 605 0.6KB/s 00:00
fridge@pc-g33-9 ~ $ ssh tapted@shade.ug.it.usyd.edu.au <enter>
tapted@shade.ug.it.usyd.edu.au's password: <enter>
Last login: Mon Jun 27 15:01:05 2005 from pc-g33-9.cs.usy
Sun Microsystems Inc. SunOS 5.6 Generic August 1997
bash$ cat id_dsa.pub >> .ssh/authorized_keys <enter>
bash$ logout
Connection to shade.ug.it.usyd.edu.au closed.
fridge@pc-g33-9 ~ $ ssh tapted@shade.ug.it.usyd.edu.au <enter> <no password!>
Last login: Fri Jul 1 11:58:03 2005 from pc-g33-9.cs.usy
Sun Microsystems Inc. SunOS 5.6 Generic August 1997
bash$
```







# Basic Configuration I

- Search for these lines – there should be templates already

## /etc/apache2/conf/httpd.conf

```
Listen 80                                # the port we listen on
ServerRoot "/usr/lib/apache2"           # base of relative paths in config
Include conf/modules.d/*.conf           # put more configuration files here
User apache                             # sometimes 'nobody' or 'www-data'
Group apache
ServerAdmin root@localhost              # this is you
DocumentRoot "/var/www/localhost/htdocs" # i.e. dir for 'http://host/'
<Directory />                            # default policy - very restrictive
    Options -All -Multiviews
    AllowOverride None
    <IfModule mod_access.c>
        Order deny,allow
        Deny from all
    </IfModule>
</Directory>
```













# HTTPS - Brief Skim I

- First, you need a server certificate

## Generating an SSL Certificate

```
cd /etc/apache2/conf/ssl
openssl genrsa -out webmail.mydomain.com.key 1024
openssl req -new -key webmail.mydomain.com.key -out webmail.mydomain.com.csr
```

- Then you need to sign it
  - Sign it yourself
  - Or go to [cacert.com](http://cacert.com)
  - If you don't want a "this certificate is not signed by a trusted authority" message you will need to pay and verify your identity to [Versign](http://Versign.com) or [Thawte Consulting](http://Thawte.com) or something
    - Or convince someone to put [cacert.com](http://cacert.com) or your own server in their root certificates





# CUPS

## Features

- Talks to local printers, network printers and Samba (windows shared) printers
- Can usually discover shared printers on the network automatically

## Bad Bits

- Works best with postscript printers (expensive!)
- “Windows Printers” might not have drivers
  - e.g. old Lexmarks and very cheap printers

# Your Friend

Common UNIX Printing System - Mozilla Firefox

File Edit View Go Bookmarks Tab Tools Help

http://localhost:631/

hlines SharePic AlsaPlayer CH 2006 | Call for P...

Loading...

ESP Administration Classes Help Jobs Printers Software

[Do Administration Tasks](#)

[Manage Printer Classes](#)

[On-Line Help](#)

[Manage Jobs](#)

[Manage Printers](#)

[Download the Current CUPS Software](#)

The Common UNIX Printing System, CUPS, and the CUPS logo are the trademark property of [Easy Software Products](#). CUPS is copyright 1997-2005 by Easy Software Products. All Rights Reserved.

Waiting for localhost...

Prompt

Enter username and password for "CUPS" at http://localhost:631

User Name:  
root

Password:  
\*\*\*\*\*

Use Password Manager to remember this password.

Cancel OK

# CUPS - Configuration

- Configuration is in `/etc/cups/printers.conf`, `/etc/cups/cupsd.conf`, etc
- But you don't want to touch that – use the web interface
  - From here it's reasonably straightforward
    - at least for postscript printers
    - Check for your model; if it's there your set
      - whether it is postscript or otherwise
- For other postscript printers, get the Windows XP PS driver
  - crack open the archive and look for a 'ppd' file
  - put it in `/etc/cups/ppd/`, restart cups
  - then it should appear in the model list of the web interface
- For other printers, try your luck
  - Pick your USB port "number", pick a 'close' model, cross fingers.. or Google

# Network printers

- That's all for a local printer (i.e. connected to your Linux box)
- For network printers, you still need cups to print from Linux
- It might just magically discover the printer
  - but not any necessary authentication
- Otherwise try “Windows printer via SAMBA”
  - and `smb://host/printer`
  - `smb://user:pass@host/printer` might work
    - but don't bet on it
- But you will probably still need a driver...

## Actually printing

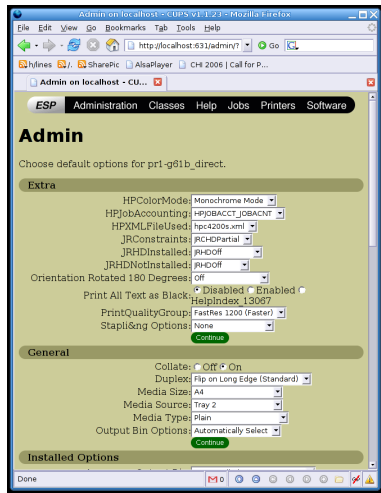
- There are GUIs
  - xprint, gnome-print, gimp-print, acroread, OpenOffice
  - These will query CUPS for available printers
  - But we can't always use these
    - and some need a print command specified anyway
- Ultimately, we can fallback on `/usr/bin/lpr`
  - Only prints postscript
  - `$ lpr -Pprinter_name filename.ps`
    - No space between '-P' and 'printer\_name'
    - printer\_name specified in CUPS config
    - If no filename, use standard input
  - Wrappers for text: `enscript`, `a2ps3`, `/usr/bin/*2ps`
    - pipe to `lpr` or a filename

---

<sup>3</sup>'any' to postscript

# Print options

- What about 2up, duplex, print quality, page size, etc. ?
- Set system-wide default settings in CUPS Printer Configuration
- Run `/usr/bin/lpoptions`
  - As root: edits `/etc/cups/lpoptions`
  - As user: edits `~/.lpoptions`
  - `'lpoptions -l'` lists available options and current settings
- These are derived from the `.ppd` file
- 2up, 4up, etc: use `/usr/bin/mpage`
- can also try `gnome-cups-manager`



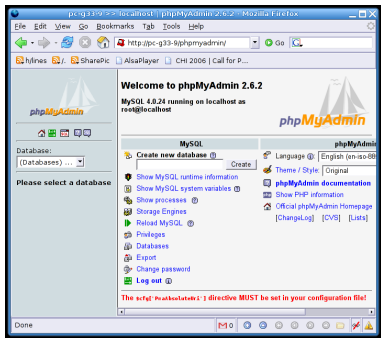
# PHP

- Two flavours, usually separate installs
  - ① mod\_php is the Apache module
  - ② [just] php is the interpreter (/usr/bin/php)
  - no problem having both
- Configuration lives at /etc/php/apache2-php4/php.ini or /etc/apache2/conf/php.ini
  - Very well documented
  - I've never actually had to touch mine
- Setup usually just involves telling apache to load it
  - APACHE2\_OPTS="-D PHP4 -D DAV -D SVN -D SVN\_AUTHZ"
  - Note: PHP4, not just PHP (for upcoming release of PHP5)
- PHP webapps should do most of their configuration locally
  - might need you to tweak global resource limits though

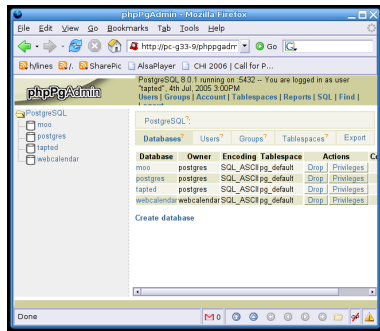
# Sick of Config Files?

php\*admin

- phpMyAdmin



- phpPgAdmin



# MySQL

## Advantages

- Popular
- Admin tools are very mature
- Lots of very good documentation and support forums

## Disadvantages

- Until recently didn't have *views* or *correlated subqueries*
  - What! Call yourself a database??
- Function support not as good as postgres (e.g. `stdev(x)` )

# MySQL

## Setup

- Install
- Config file: `/etc/mysql/my.cnf`

### MySQL Initialisation

```
# /usr/bin/mysql_install_db
# /etc/init.d/mysql start
# /usr/bin/mysqladmin -u root password 'new-password'
# /usr/bin/mysqladmin -u root -h pc-g33-9 password 'new-password'
```

- last line only if we bind non-local address in `my.cnf`
- phpMyAdmin

```
mysql -u root -p <
/.../phpmyadmin/.../sqlscripts/mysql/..._create.sql
```

  - `http://localhost/phpmyadmin/`
    - login as root

# PostgreSQL

## Advantages

- Slightly faster
- More mature (grew out of pre-SQL 'Postgres' database)
- More functionality
  - stored procedures, etc.

## Disadvantages

- Not as popular – not as many tools/frontends

# PostgreSQL

## Setup

- Install
  - Initially somewhat confusing – only user ‘postgres’ has access
  - MySQL: root runs `/usr/bin/mysqladmin`

### PostgreSQL Initialisation

```
root@ # /etc/init.d/postgresql start
root@ # su postgres
postgres@ $ createuser root <not essential>
Shall the new user be allowed to create databases? (y/n)
Shall the new user be allowed to create more new users? (y/n)
CREATE USER
postgres@ $ createuser username
...
postgres@ $ createdb username
CREATE DATABASE
```

- Use `createuser -P username` to prompt for a password
- Interactive terminal – run `/usr/bin/psql` as *username*
  - By default you connect to a database called `$USER`
  - Can also pipe SQL commands as for `/usr/bin/mysql`

# Databases

## Summary

- There's no problem having both
- Often applications requiring a database will offer SQL scripts and interfaces for either
- For my own stuff, I've preferred postgresql
  - But haven't played much with mysql-4 (finally has views)
- ODBC
  - There's a unixODBC package, but it's not very good
  - There's also ODBC drivers for postgresQL and MySQL
    - also not good
  - Also no good programming libraries
  - 'nuff said

# Samba I

- Kerberos
  - That three-headed dog that guards the gates of hell
  - Also Windows' security protocols
- To start...
  - `/etc/init.d/samba start`
  - But first we need to configure it...
- There should be a configuration template hovering around
  - maybe `smb.conf.example.gz`
  - time to put `/usr/bin/locate` to use
  - Ubuntu starts with a default config in `/etc/samba/smb.conf`

# Samba II

`/etc/samba/smb.conf` - *grep for these*

```
[global]
workgroup = WORKGROUP
printcap name = cups # Make all CUPS printers
load printers = yes # available to Windows machines
encrypt passwords = yes
smb passwd file = /etc/samba/private/smbpasswd
[homes] # Allow access to //host/username with that user's password
comment = Home Directories
browseable = no
writable = yes
[public]
path = /usr/somewhere/else/public
public = yes
only guest = yes
writable = yes
printable = no
```

- ... LDAP and lots more, and many caveats
  - use the template and read the manual!

# Samba mounts I

- `smbmount`, `smbumount` – allows users to mount
  - usually only root can use `/bin/mount`
  - username defaults to `$USER`

## Mounting Samba Drives

```
fridge@pc-g33-9 ~ $ mkdir ugrad tapt-ugrad
fridge@pc-g33-9 ~ $ smbmount //ugitsmb/fridge ugrad
Password:
fridge@pc-g33-9 ~ $ smbmount //ugitsmb/tapted tapt-ugrad \
-o username=tapted
Password:
...
fridge@pc-g33-9 ~ $ smbmount ugrad; smbmount tapt-ugrad
```

- Automatically mounting
  - We need a password but `/etc/fstab` is world readable
  - Solution: credentials file

# Samba mounts II

## Automating Samba Mounts

```
# cat > /root/.ugrad-sambacreds
username=fridge
password=gotmilk?
# chmod og-wrx /root/.ugrad-sambacreds
//ugitsmb/fridge /mnt/ugrad smbfs \
credentials=/root/.ugrad-sambacreds,uid=fridge,gid=users,dmask=0770,fmask=0660

# mount -a
```

- Makes the mount read/writable by people in the 'users' group
- **dmask, fmask is inverted**
  - umask and other file system types would use 0007/0117

# SSH Tunnelling

- SSH gives cryptographically secure traffic and authentication
  - but only for system users
- Some programs talk over ssh directly
  - just open a pipe and run themselves over ssh in server mode
    - `scp` Works like copy, but accepts host:path<sup>4</sup>
    - `rsync` Works like scp, but only sends updates
    - `sftp` Works like ftp, but over ssh (no ftp server required!)
    - `cvs` Source code management (SCM), best used over ssh
    - `svn` SCM, not so great over SSH (but it's there)
    - `sancho` Can talk to mldonkey over SSH

---

<sup>4</sup>username@host:path  $\implies$  \$HOME/path (*for that user*)

# rsync is cool

- rsync is usually the best way to transfer files
  - `rsync -rtP /home/fridge/ tapted@shade:backups/fridge/`
- Usually with arguments '-rtP'
  - recursive, preserve **t**imestamps, show **p**rogress, resume **p**artials
- a trailing slash on the *source* path causes different behaviour
  - “don't create a new directory at the destination”
- Why it's so cool
  - When file size and times match. . .
  - Cut file into blocks, compare checksums, send differences
- Lots more arguments
  - e.g. ignore times, assume equal size/time means no change
  - check the man page

# Port Tunnelling I

- What is it?
  - Listen on any TCP/IP port for incoming packets
    - on either the client or the server
    - but only ephemeral ports if you are not root
  - Forward them (encrypted) through the established ssh connection
  - To a server:port that the other side has access to
    - e.g. if you can't access it, or want your traffic encrypted
- Use it to poke holes in firewalls
  - so long as port 22 is accessible
  - or even if it's not. . . . .

## Port Tunnelling II

- `ssh -L2525:example.com:25 me@host`
  - Forward traffic arriving at `localhost:2525` to `example.com:25` via host
  - '-g' allows traffic *not* generated locally to be tunnelled
- `ssh -R2525:nearby.com:25 me@host`
  - Forward traffic arriving at `host:2525` to `nearby.com:25` via localhost
  - There is no '-g' equivalent for remote tunnels
    - Planned for SSH protocol v3
- What if I want to forward stuff (or log into) a machine behind a firewall?
  - i.e. *all* incoming ports are blocked
  - Initially, you will need physical access to the machine
  - You will also need a machine you *can* access *from the blocked machine*
    - But you don't need superuser privileges

## Port Tunnelling III

- from the blocked machine...
  - `me@blocked ~ $ ssh -R2222:localhost:22 me@accessible`
- some other time...
  - `me@accessible ~ $ ssh -p 2222 localhost`
    - never had to specify IP address of *blocked*!
  - can also forward other ports now
    - e.g. `accessible:8080  $\implies$  blocked:80`

```
me@accessible ~ $ ssh -p 2222 -g  
-L8080:localhost:/etc/ssh/ssh_config80 localhost
```

- You will probably want to daemonize ssh
  - Use `-f` (fork) and `-n` (no input) arguments to ssh
    - Or use *autossh*
  - Make sure `AliveIntervals` are set
  - We also can't prompt for a password so you'll need keypairs

# Netkit

- A bunch of antiquated servers from Berkeley software distribution (BSD)
  - fingerd, rsh, rwall, talk, tftp, bootpd, routed, rusers, rwho, telnetd, timed
- Typically use host-based authentication (or none at all)
  - `/etc/hosts.allow`, `/etc/hosts.deny`
- Only use them behind a rock-solid firewall
  - fingerd is probably the most famous victim of a buffer overflow
  - it's fixed now
- Use ssh instead, with public/private key pairs

# X11 I

- Yes, X11 is a *server*
  - The client is whatever computer you are logged into
  - The host is always the computer in front of you
- Authentication is via a 'MAGIC-COOKIE' in `~/.Xauthority`
  - SSH can set this up for us...
  - Alternatively, disable or use host-based authentication with `'xhost +'`
    - not wise

# X11 II

- First, enable X11 Forwarding in SSH (`/etc/ssh/ssh_config`)

```
/etc/ssh/ssh_config
```

```
Host *
ForwardX11 yes
Compression yes
ServerAliveInterval 55
```

- Then 'export ForwardX11=yes' in your shell, or
  - `ssh -X me@somewhere`
- Then you can run X11 apps on the remote computer
  - windows will be displayed locally
- You might also need to tell your local X11 server to accept TCP connections
  - e.g. `DisallowTCP=false` in `/etc/X11/gdm/gdm.conf`

# Firewalls

- Incoming connections on the server port must be enabled
  - get the port number from `/etc/services`
  - or whatever you told the server to run on
- If you are behind a NAT, you'll also need a port forward
  - i.e. traffic to port X on the NAT gets sent to host:Y
    - where *host* is an intranet address
- If your linux box *is* the firewall/NAT
  - time to get intimate with `iptables`
    - or check out *firestarter*
  - you'll probably need to recompile your kernel, too

# Linux on Laptops I

- Four main obstacles
  - ① Snuggling up to an NTFS partition that you can't ditch
  - ② Getting a framebuffer working
  - ③ Wireless networking
  - ④ ACPI
- **Look out for recovery partitions!**
  - older live CDs don't recognise these and may overwrite them
  - most won't let you resize NTFS partitions anyway
    - get the 'System Rescue LiveCD' and use ntfsresize or qtparted
    - you will need to convince Windows' defrag to leave the tail end of your hard disk clear
- Expect some stuff not to work (maybe)
  - extra buttons
  - bluetooth chips
  - multiscard readers

# Linux on Laptops II

- Wireless Networking
  - orinoco adaptors supported via kernel option
  - Intel adaptors via ipw2100, ipw2200 packages
    - kernel module, not part of official kernel
  - You will need all the kernel encryption libraries too
  - Many cards still unsupported
    - e.g. obscure Dell laptop wireless adapters
  - Configuration is done via `/usr/sbin/iwconfig`
    - WEP keys, choosing a base station
- You will very likely want to recompile your kernel
  - enable `{dell, IBM, compaq}|laptop` extras
  - enable the framebuffer driver for your laptop card
  - enable PCMCIA (yenta sockets)
  - enable ACPI, disable APM
  - ditch the crud

# Linux on Laptops III

- Other stuff
  - touchpad: use the 'synaptics' package and xorg mouse *Driver*
  - extra keys: play with gnome keyboard shortcuts and see what registers
    - might be able to bind volume keys this way
  - closed-source ATI and Nvidia Xorg drivers usually support mobility extensions
    - e.g. switch between LCD VGA and S-Video outputs
  - sound is sometimes fiddly
    - e.g. to switch between headphones and speakers
    - try "switch front / surround" in *alsamixer*
  - kernel patch for non-VESA frame buffers (*vesafb-tng*)

# Power management

- Some stuff is usually done at the hardware level
  - turning on or speeding up fans
  - turning off panel when lid closed
  - dimming panel when on battery (or via function keys)
- Other stuff is not
  - put wireless adapter, graphics card into powersave mode
  - turning off hard drive when idle
    - you will want a non-flushing system logger like *metalog*
  - changing the 'soft' runlevel
    - e.g. stop non-essential services (sshd, apache, etc.)
  - CPU frequency scaling
- Write your own scripts, or wait for someone to make a nice ACPI manager
- **Sleep states are supported badly at best**
  - Linux isn't really designed for it
  - but kernel-level laptop-linux extensions are on their way...

# Talking to NTFS

- Linux really doesn't want to
  - the kernel driver is reverse-engineered
  - can only write to existing files and can't change their size
  - reading is usually OK
- Try and squeeze in a *FAT32* partition (vfat in Linux)
  - for sharing files between the operating systems
- Other options
  - CaptiveNTFS in Linux, uses Windows binary drivers directly
    - haven't tried it myself...
  - explore2fs in Windows, lets you read your Linux partition
    - also ext2fsd
    - nothing yet for advanced Linux filesystems
    - e.g. ReiserFS, Reiser4, XFS, etc.

# Accelerated Graphics

- Nvidia and ATI have closed-source Linux drivers available
  - Some older ATI cards are not supported
- Usually comes down to
  - install drivers
  - compile a kernel module
  - autoload the module at boot
    - `/etc/modules.autoload.d/kernel-2.6`
  - change your *Driver* from 'nv' or 'ati' to 'nvidia' or 'fglrx'
    - in "Device" section of `/etc/X11/xorg.conf`
  - test with `glxinfo`
    - look for 'direct rendering: Yes'
  - get > 1000fps using `glxgears`
  - drool over `xscreensavers`
    - get `rss-glx` screensavers
    - drool some more

# NFS: Network File System

- Only good for talking to other \*nix boxes
- Compile your kernel with NFS server support
- Install the NFS tools
  - and portmap
- Set up host-based authentication files
  - /etc/hosts.allow, /etc/hosts.deny

```
/etc/hosts.allow
```

```
ALL:192.168.
```

- Set up exports

```
/etc/exports
```

```
/usr/portage 192.168.0.12(rw,root_squash,sync)
```

# Summary

- Enjoy<sup>5</sup>

---

<sup>5</sup>I can't be bothered writing a summary

