Linux Systems Administration

Trainer: Ken Marr

Overview

This course gives an introduction to Linux systems administration. Linux is a versatile and popular Unix like operating system used in business, on high-end workstations and on many of the servers on the internet.

Linux is easy to learn and use and we expect this course to be a fun and interesting way to master it.

The course includes the installation and configuration of the operating system, installing updates and packages, startup and shutdown, managing services, printers, users and the network, running backups, the partitioning of file systems and the use of logical volumes as well as troubleshooting and the use of monitoring tools.

Although based on Red Hat (CentoS) and Ubuntu the commands and principles learnt may be applied to all other versions of Linux.

Aims and Objectives

The main aim is that the delegate leaves the course feeling comfortable with the operating system, ready and able to administer the system on a daily basis.

At the end of this course the delegate will be able to:

- install and configure the Linux software
- install and upgrade software packages
- startup and shutdown the system
- manage services and printers
- create and manage user accounts
- backup and recover files and partitions
- manage files systems and logical volumes
- manage the network and the NFS file system

Schedule

The times shown may be changed as agreed with the tutor:

- Start time is 9.30am; end time is 4.15pm approx
- Break times, morning and afternoon
- Lunch will be for 1 hour at 12.30pm approx
- Please ensure that you **adhere** to the agreed times
- For an on-site course, **interruptions** should be kept to a minimum
- If you have a **mobile phone**, please turn it off now

Your tutor will point out the location of the following:

- The men's and women's toilets
- The fire exit and the action to be taken in the event of a fire

Introductions

Before the course begins, your tutor will ask each delegate to introduce themselves to the rest of the class.

In your introduction, please include the following:

- Name and company
- Job title and responsibilities
- Previous experience with the software to be used

Also consider answers to the following questions:

- I have come on the course because?
- What would I like to gain from the course?
- What would I like to be able to do by the end of the course?

Pre-requisites

Some previous knowledge of Linux or Unix is required.

Responsibilities

The course is divided between lecture, demonstrations and exercises. The idea behind this is three fold. The delegate will:

- listen and learn
- see and understand
- do and remember

There are many exercises and workshops throughout this course. Please note that collaboration whilst undertaking these is perfectly acceptable.

- COOPERATION is encouraged, competition is not
- all the exercises may be undertaken in PAIRS
- delegates may, if they prefer, work TWO to a terminal
- if sharing try to achieve a 50% - 50% SPLIT at the keyboard

During the course, in order to aid the learning experience, the delegates should endeavour to:

- try out the course EXAMPLES
- attempt to complete each EXERCISE
- EXPERIMENT and try out new things
- not be afraid to ask QUESTIONS
- CONFER with their neighbours
- LEAN and be leaned on; the best way to learn is to help others
- learn by your (and their) MISTAKES
- ENJOY themselves!

Contents

Installation

Objectives

At the end of this section the delegate will be able to:

- install the Linux system
- describe the role of the systems administrator
- log on as root, the super user
- use the su and sudo commands
- examine common log files

Red Hat Enterprise Linux

Obtaining RHEL
Although an evaluation copy is available for production use an active subscription is required before downloading RHEL.from the Red Hat portal.

RHEL versions
For older editions it is often assumed that the branding ES, AS, and WS stand for "Entry-level Server", "Advanced Server" and "Work Station" respectively. The reason for this is that the ES product is indeed the company's base enterprise server product while AS is the more advanced product. However, nowhere on its site or in its literature does Red Hat say what AS, ES and WS stand for.

In newer versions of Red Hat Enterprise Linux there are new editions that substitute for the former Red Hat Enterprise Linux AS/ES/WS/Desktop editions:

- Red Hat Enterprise Linux Advanced Platform (former AS)
- Red Hat Enterprise Linux (former ES) (limited to 2 CPUs)
- Red Hat Enterprise Linux Desktop with Workstation and Multi-OS option
- Red Hat Enterprise Linux Desktop with Workstation option (former WS)
- Red Hat Enterprise Linux Desktop with Multi-OS option
- Red Hat Enterprise Linux Desktop (former Desktop)

RH Certification Program
This is from the official Red Hat website:

“Skeptical of IT certifications? Red Hat certifications are different. To earn a Red Hat certification, you must pass a hands-on, practical exam in which you complete real-world tasks using our technologies rather than just being asked questions about the technology.

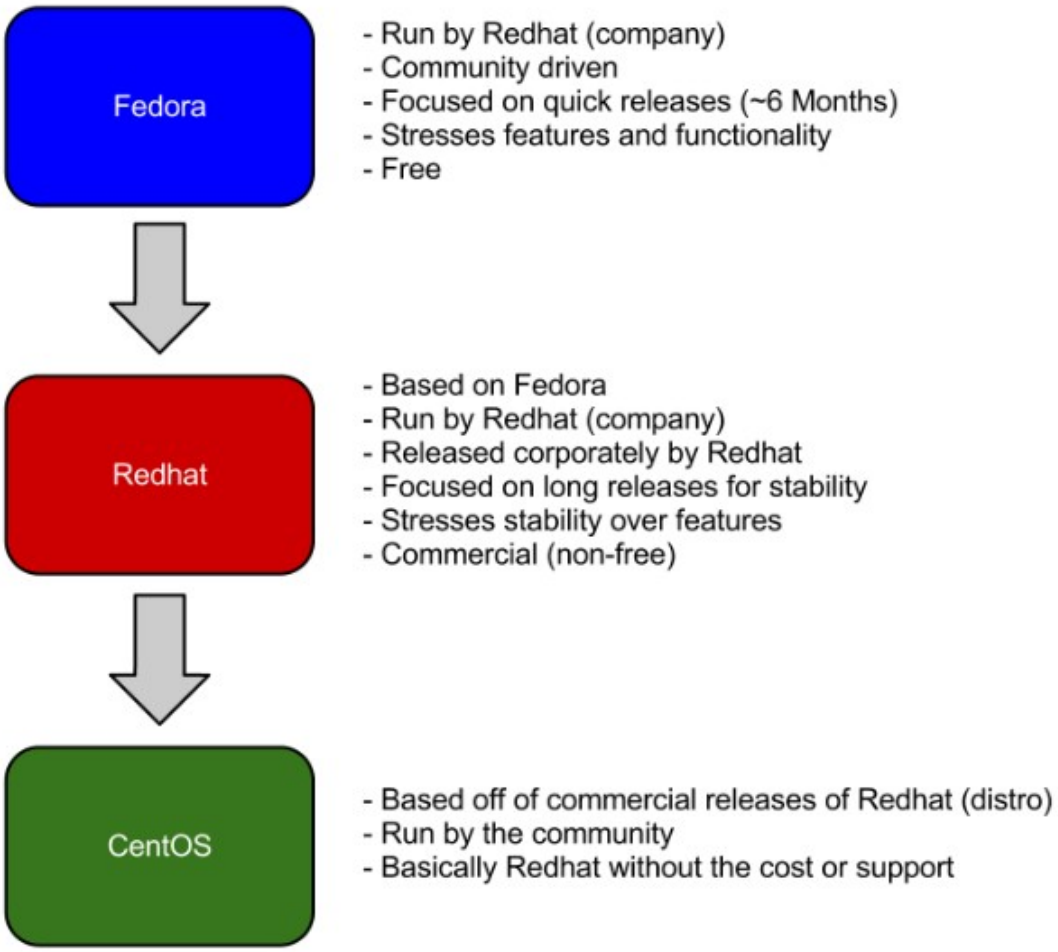
Our certification program gives employers a way to find and develop qualified professionals and allows technical professionals to prove their skills and build their careers”.

More information can be found here:
http://www.redhat.com/training/certifications/

What are Red Hat, CentOs and Fedora?

There is often some confusion over the relationship between Fedora, Red Hat and CentOS. For example, are they owned by the same company, is one a version of the other, which one is the more up to date?

A simple diagram and description:

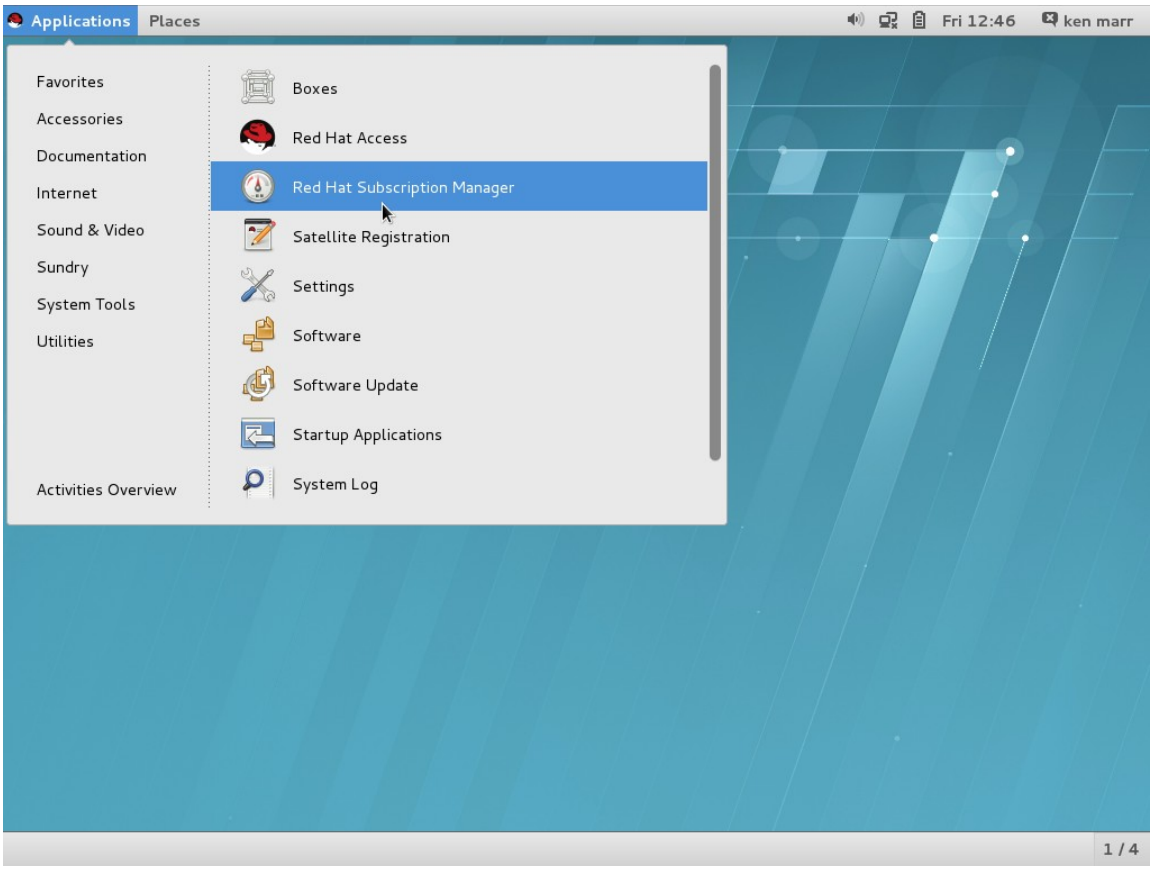


Red Hat, the actual company, market two Linux distributions: Fedora (formerly known as Red Hat) which is free and for desktops and Red Hat Enterprise Linux, RHEL, which is the higher end more robust version for servers. They're very similar in appearance and workings, the major differences being:

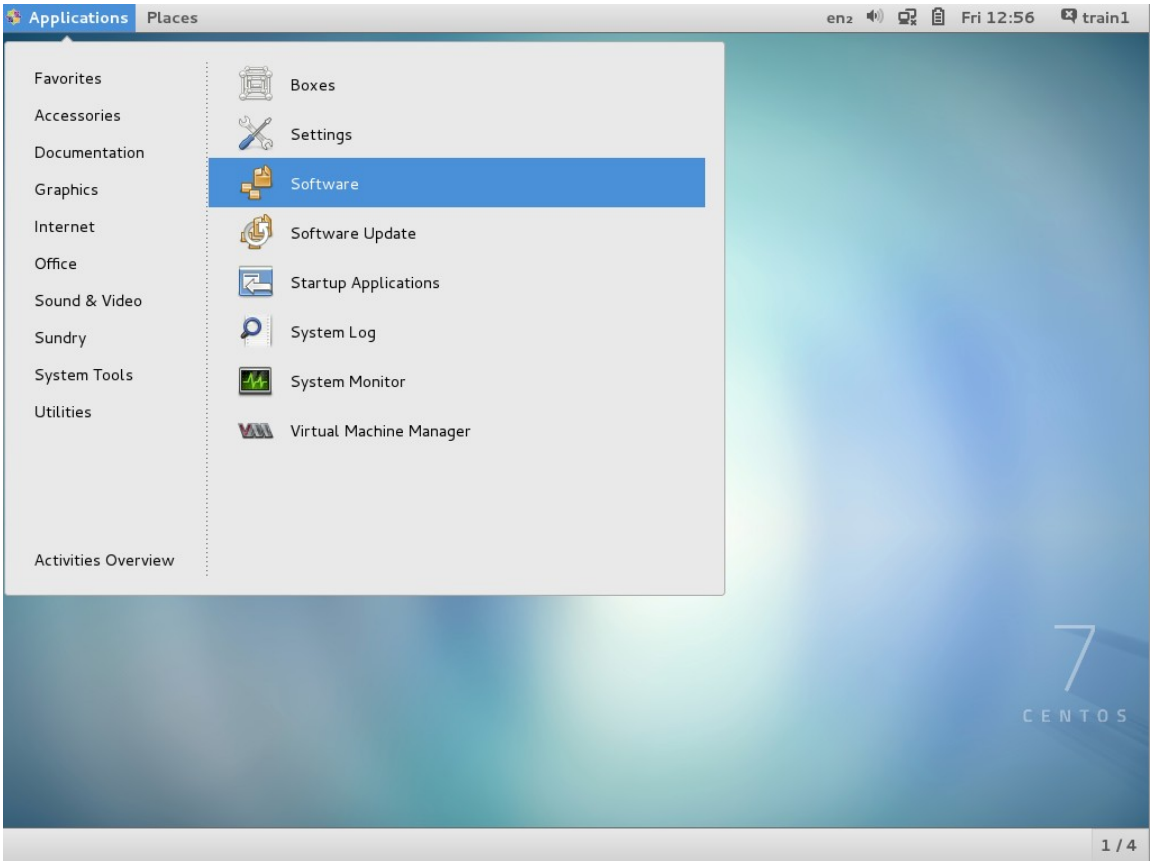
- the cost of support , Fedora is free
- the hardware limitations; RHEL supports more processors and larger amounts of RAM

CentOS, the Community Server OS, is based on the RHEL base code rather than on Fedora and so enjoys the stability and robustness offered by RHEL and it's free. Several options are missing including Red Hat Subscription Manager and RHN Registration.

Red Hat 7



CentOS 7



Installation

Hardware Compatibility

Companies can certify their hardware with Red Hat Enterprise Linux to provide their customers with the confidence of knowing that their systems are supported by Red Hat. During the certification process, Red Hat offers an automated test suite, support and access to online updates while the results are being reviewed.

To see the Certified Hardware catalog and to learn more about the Hardware Program visit: http://www.redhat.com/rhel/compatibility/hardware/

Linux can be installed onto a hugh variety of hardware including laptops and PCs or can be installed into a VM, a Virtual Machine. The installations of different flavours are very similar and this course shows how to install CentOS, the unbranded version of Redhat, onto a VM.

KickStart - Automated Installation

An installation can run unattended by using Kickstart. Here a Kickstart file specifies settings for an installation. Once the installation system boots, it can read the Kickstart file and carry out the installation process without any further input from a user. After a normal installation a kickstart file can be viewed as follows:

```
less  /root/anaconda-ks-cfg
```

Network Based Installation

Linux can be installed as a network installation via NFS, FTP, or HTTP or via local storage.

Pre Installation

Before installation there are several choices which must be made including:

- Architecture, 32 or 64 bit
- For a VM, memory (50% of host) and disk size
- Hostname and root password
- Partition layout and swap size
- Installation type, full, development or minimal

Partition Layout
This involves deciding how many partitions to have (for root (/), boot, home and tmp for example) and how big each should be.

Swap Size
The term *swap* describes both the act of moving memory pages between RAM and disk and the region of a disk where pages are stored. Many administrators follow the old rule that the swap partition should be twice the size of the main system RAM. This rule probably comes from old systems where memory mangers were badly designed. Today systems have very smart and intelligent memory managers.

As a rule of thumb for a normal server (web/mail etc):

- RAM < 2GB, swap size = RAM size
- RAM > 2GB, swap size = 2GB

For something such as a heavy duty Oracle server with fast storage such as RAID 10:

- RAM < 8GB, swap size = RAM size
- RAM < 16GB, swap size = 8GB
- RAM > 16GB , swap size = 0.50 times the size of RAM

So if RAM is 5GB, swap is 5GB, if RAM is 10GB, swap is 8GB, if RAM is 32GB, swap is 16GB.

Installation Summary

Installation may take between 30 and 45 minutes. The installation as a VM can be summarised as follows:

- Install the VM software, Oracle VirtualBox
- Install Linux from a DVD iso image
- Choose the correct architecture, Red Hat, 32 or 64 bit
- Set memory to 1500, choose a disc size of **30GB**
- Select language, basic storage devices, hostname (linux.train)
- Set time zone, root password (root123)
- Choose Create A Custom Layout
- Create swap 2000, /boot 200, root (/) 8000, /home 500
- Format and write changes to disc
- Install Software Development Workstation
- Reboot
- Add a new user

The XWindow system

The X Window System, X11, X and sometimes informally X-Windows is a windowing system for bitmap displays common on UNIX-like computer operating systems.

X provides the basic framework for a GUI environment: drawing and moving windows on the display device and interacting with a mouse and keyboard. X does not mandate the user interface; this is handled by individual programs. As such, the visual styling of X-based environments varies greatly; different programs may present radically different interfaces.

X originated at the Massachusetts Institute of Technology (MIT) in 1984. The protocol version has been X11 since September 1987. The X.Org Foundation leads the X project, with the current reference implementation, X.Org Server, available as free and open source software under the MIT License and similar permissive licenses.

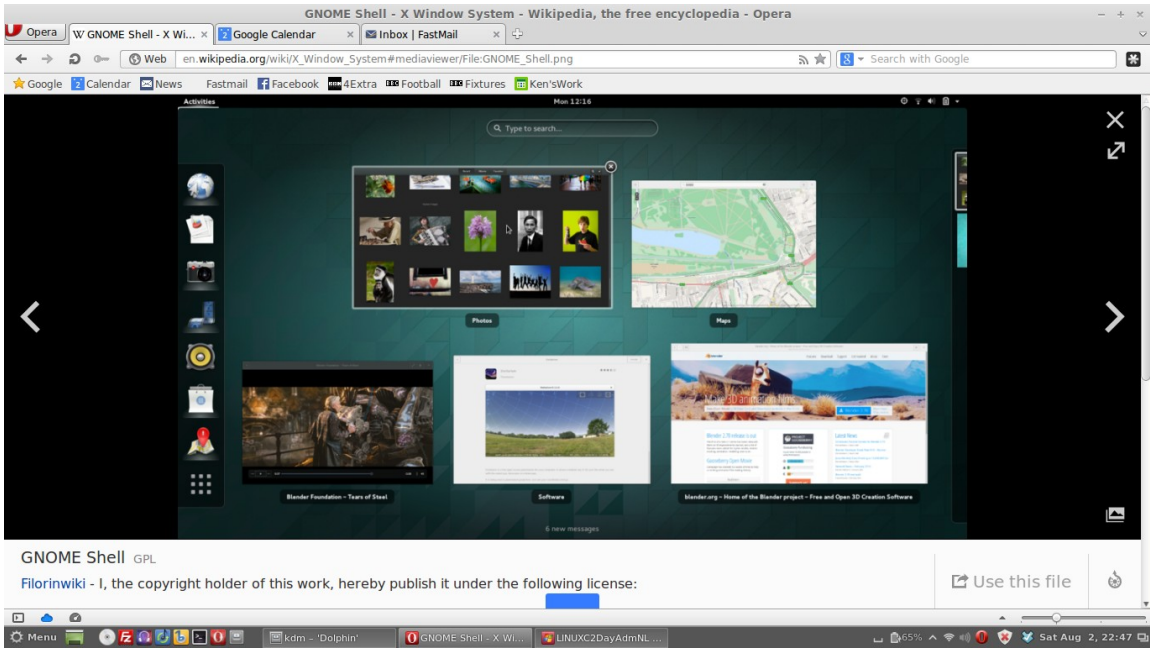
User interfaces
X primarily defines protocol and graphics primitives; it deliberately contains no specification for application user-interface design, such as button, menu, or window title-bar styles. Instead, application software, such as window managers, GUI widget toolkits and desktop environments, or application-specific graphical user interfaces, define and provide such details. As a result, there is no typical X interface and several different desktop environments have become popular among users.

A window manager controls the placement and appearance of application windows. This may result in desktop interfaces reminiscent of those of Microsoft Windows or of the Apple Macintosh. Some interfaces eschew the desktop metaphor altogether, simplifying their interfaces for specialised applications. Window managers range in sophistication and complexity from the bare-bones (e.g., twm, the basic window manager supplied with X, or evilwm, an extremely light window-manager) to the more comprehensive desktop environments such as Enlightenment and even to application-specific window-managers for vertical markets such as point-of-sale.

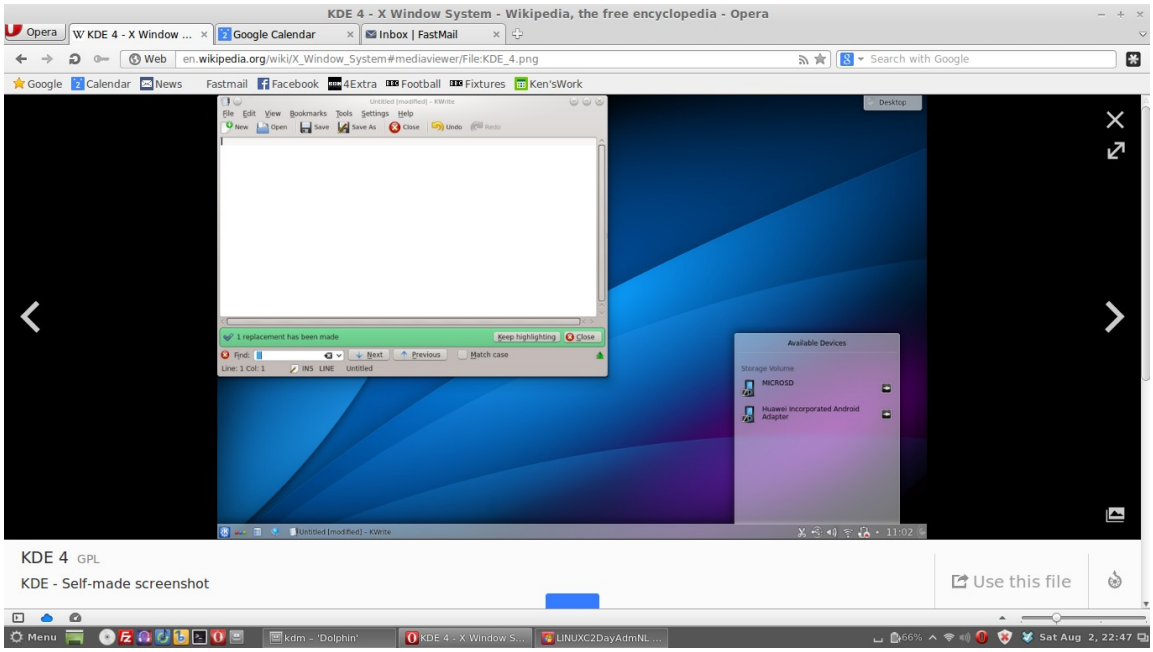
Many users use X with a desktop environment, which, aside from the window manager, includes various applications using a consistent user-interface. Popular desktop environments include **GNOME** and KDE. The UNIX98 standard environment is the Common Desktop Environment (CDE) used on Solaris.

More information can be found here:
http://en.wikipedia.org/wiki/X_Window_System

The Gnome window manager



The KDE window manager



The System Administrator

The typical duties of a system administrator vary depending on the number of systems supported and how the duties are divided up. It is not uncommon for system administrators to be experts in administering one or more areas and be inexperienced in others.

For example, some administrators specialise in network administration, others in managing user accounts.

The following is a list of typical system administration duties:

- Administering user and group accounts
- Administering file systems
- Backing up and restoring files and partitions
- Administering print services
- Administering network and mail services (not covered here)
- Identifying problems with search paths, permissions and ownership

To accomplish these tasks, they need to know when and how to perform the following tasks:

- Install packages and third party software
- Shut down and start up the system
- Gain full access to all file systems and resources
- Start and stop services and printers
- Monitor users, files and processes
- Use command line tools including vim, find and grep
- Use and understand shell scripts
- Grant users permissions for specific tasks

Understanding Superuser Status

The superuser is a privileged user with unrestricted access to all files and commands on the system. The user name for this account is **root** and the terms root and superuser have the same meaning and can be used interchangeably. The user has the special UID, user ID, of 0 in the password file. The account may be displayed as follows:

```
head -1  /etc/passwd
```

An administrator must be root to perform many system administration tasks, such as mounting and unmounting file systems, changing ownership or permissions for a file or directory, backing up and restoring file systems and shutting down the system.

When a user has superuser privileges, the shell provides a special #, hash sign, prompt to remind them that they have extra access to the system.

A user can become superuser in a number of ways.

- When logged in as another user, by typing the ***su***, switch user, command with no arguments, and then typing the root password
- Rather than switching into the user root, the ***sudo*** command allows a permitted user to execute a command as the superuser
- At the login prompt by typing root and then the root password, although this method is often disabled and is **not recommended**

A system is far more secure if access to the *root* user and to the *su* command are restricted. Later in this section we will look at restricting access.

su – Switch User

A user should become superuser only when it is required, and avoid doing any routine work as superuser. When a task requires them to be root, they should switch user to root, perform the required tasks and exit superuser status when the task is complete. To switch to root use:

```
su -
```

Because unauthorised access to root can be a serious security breach, the user should always have a password, which, for enhanced security, should be changed frequently.

Note that if the hyphen is not used with this command, as in:

```
su
```

this is not the same as logging in as the new user. This format of the command does **not** set the new user environment but retains some of the settings of the original user.

Post Installation

The current version installed can be viewed as follows:

```
cat /etc/redhat-release
```

Guest Additions
If installing Linux as a VBox VM, in order to use options such as Shared Folders, Shared Clipboard, Drag and Drop and to correct the behaviour of the mouse, the Guest Additions should be installed.

Choose Devices from the VBox menu, then Install Guest Additions. Double click the CD icon on the desktop then choose OK and Run.

As an alternative, at the command line as root enter:

```
cd  /media/VBox* ; sh ./VBoxLinux*.run
```

Shared Folders
Shared folders are created via the VBox Devices menu. To automount a new shared folder, either reboot or as root enter:

```
service vboxadd-service restart
```

To mount a shared folder at the command line, use a command of the form:

```
mkdir /windows

mount -t vboxsf linux /windows
```

Exercise

1. Login as your own user and install the VBox guest additions.

2. With the help of your tutor switch into *root* and setup a shared folder.

Customising the Shell

When customising the Bash shell, the hidden file *.bash_profile*, if it exists, will be executed in preference to the file *.profile* on login. The bash shell may also use the file ***.****bashrc* which is run every time a Bash shell is invoked.

For example, if added, the lines below set the prompt to the current directory in the colour red and add two aliases:

```
PS1='\[\e[01;31m\]$PWD: \[\033[01;34m\]\[\033[00m\]'
alias c='clear'
alias vi='vim'
```

Exercise

- Switch user into *root*, edit the hidden file *.bashrc* and alter the prompt and add several aliases. Test the new changes.

To add a common alias for all users, add the alias to the file /etc/bashrc:

```
less  /etc/bashrc
```


sudo - Commands as Root

Rather than switching into the user root, the *sudo* command allows a permitted user to execute a command as the superuser.

The command sudo requires that users authenticate themselves with a password; this is the user's password, not the root password. Once a user has been authenticated, a timestamp is updated and the user may then use sudo without a password for a short period of time.

For example, the command *cat* may be used to list the contents of the file */etc/passwd*, a special file which contains an entry for every user login on the system:

```
cat  /etc/passwd
```

However, this file does not contain encrypted passwords. These are held in a restricted file which cannot be viewed by a user other than *root*:

```
cat  /etc/shadow
```

The sudo command, if available, may be used to view the file:

```
sudo  cat  /etc/shadow
```


Backing Up Configuration Files

An administrator will often have to alter configuration files. Before any file is edited, it should first be copied to create a backup file.

A quick way to do this is to use the following syntax:

```
cp  /etc/sudoers{,.bak}
```

As an alternative (or as well as), make a backup of the /etc directory:

```
cd  /etc ; tar  czf  /backup/etc.tar.gz  .
```

sudo – Allowing Use

To allow the use of the *sudo* command the administrator should first backup the two files to be altered. Then edit the files */etc/sudoers* and */etc/group* as below.

To allow the use of the *sudo* command, this file must be amended using the command *visudo*:

- the */etc/sudoers* file should contain the following lines:

  ```
  ## Allows people in group wheel to run all commands
  %wheel  ALL=(ALL)       ALL
  ```

- the user must also be added to the wheel group:

  ```
  usermod  -aG  wheel  train1

  grep  wheel  /etc/group
  ```

To test the command, **logout** and **login** again and run:

```
sudo  cat  /etc/shadow
```

To check for illegal use of the *sudo* command view the following log file:

```
grep sudoers /var/log/secure
```


Review Exercise

1. Make a *tar* backup of the */etc* directory.

2. Make a backup of the files */etc/sudoers* and */etc/group*.

3. Amend these two files and setup and test the *sudo* command for your user.

sudo - Root Password

When installing Red Hat, the user is asked for a root password. This is not the case on **Debian** based systems. To change the root password on Ubuntu for example, run the following:

```
sudo passwd
```


Log Files

Almost all logfiles are located under the */var/log* directory and its subdirectories. For example, view the log file /var/log/messages using any one of the commands tail, less or more:

```
tail /var/log/messages
```

Common Log Files

/var/log/messages	General messages, system related
/var/log/auth.log	Authentication logs
/var/log/kern.log	Kernel logs
/var/log/cron.log	Crond logs (cron job)
/var/log/maillog	Mail server logs
/var/log/httpd/	Apache access and error logs directory
/var/log/boot.log	System boot log
/var/log/mysqld.log	MySQL database server log file
/var/log/secure	Authentication log
/var/log/wtmp	Login and logout records file
/var/log/btmp	Bad login records file
/var/log/yum.log	Yum log file

To view the login records file use the following command:

```
last | less
```

To view failed logins use:

```
lastb
```

In summary /var/log is the location for all Linux logs file. However some applications such as httpd and samba have a directory within /var/log/ for their own log files:

```
ls /var/log
```

Log files can be rotated using *logrotate* software and, if installed, monitored using *logwatch* software.

GUI Tools

There are other ways of managing Linux besides using the command line. These include via the graphical (GUI) tools (if installed) and via a web interface. For example, to view the services using a GUI from the menu bar choose:

System > Administration > Services

Some GUI tools may also be started from the command line, for example:

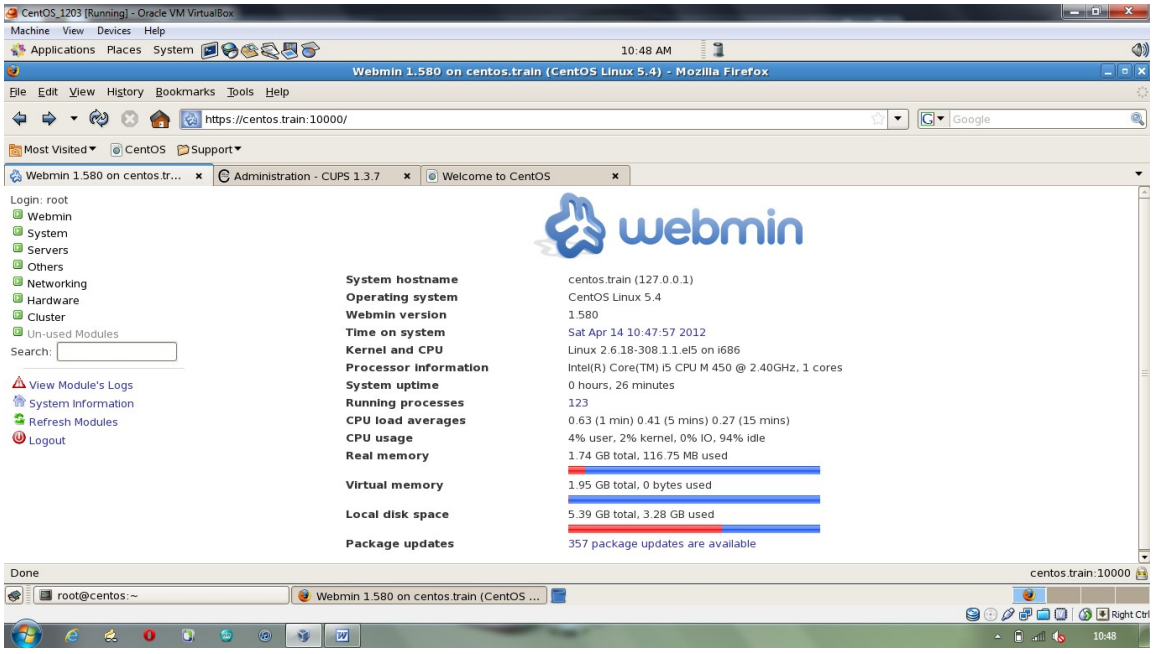
system-config-services

However, many GUI tools vailable in Red Hat 5 have been removed in later versions.

Webmin

Webmin is an excellent web-based management platform for Linux and several other operating systems. It simplifies and streamlines standard administrative tasks and can be used to configure very complex implementations of software such as Apache and MySQL. This is installed later in the course.

http://localhost:10000/



Patch and Package Management

Objectives

At the end of this section the delegate will be able to:

- update the operating system using packages
- use the yum and rpm commands on Red Hat systems
- compile packages from source
- use the apt-get and dpkg commands on Debian systems

Managing Packages

Most modern Linux distributions have their own package managers.

On Red Hat based systems the most popular are:

- Yellowdog Update Manager, yum
- Red Hat Package Manager, rpm

The *pirut* graphical package manager front end on Red Hat provides a set of graphical tools for managing software.

Debian based systems such as Ubuntu and Linux Mint use:

- Advanced Packaging Tool, apt-get
- Debian Package Management, dpkg
- A package conversion tool, alien

Useful Packages

The following are a selection of useful packages:

• vim	vi editor improved
• gparted	partition manager
• dolphin	file manager
• wine	run Windows applications
• filezilla	file transfer
• lrzip	zip utility
• soundconverter	convert audio files
• audacity	edit audio files
• avidemux	edit video files

Yellowdog Update Manager

The **yum**, Yellowdog Update Manager, command is the most sophisticated way to manage packages on Red Hat based systems. One of the primary benefits of yum is that it will automatically discover dependencies and then install them. The only drawback is that it needs a connection to the internet.

There is also a graphical update tool available through Applications -> System Tools -> Software Updater or from the command line using:

```
pup &
```

Create a Local Yum Repository
If the internet is **not** available a local yum repository may be created using the install DVD iso file. The steps involved are as follows.

Create a mount point and mount the ISO image:

```
mkdir  /yumlocal
mount  -o  loop  /wiso/C*DVD*iso  /yumlocal
```

Save the current entries in /etc/yum.repos.d:

```
cd  /etc/yum.repos.d
mkdir  save
mv  C*  save
ls
```

Create entries in a new repository file:

```
cd /etc/yum.repos.d
```

```
echo "[OL64]
name=Oracle Linux 6.4 x86
baseurl=file:///yumlocal
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
gpgcheck=1
enabled=1" > local.repo
```

```
cat local.repo
ls -R
```

Clean up the yum cache:

```
yum clean all
```


Configuration
If needed, edit the yum configuration file */etc/yum.conf* and add a line of the following form for a proxy server:

```
http_proxy=192.168.1.21:8080
```

Test access to the repository:

```
yum repolist
```

Update
After the initial installation of Linux, to list all updates run:

```
yum  list  updates
```

To update a single package:

```
yum  update  bc
```

To update all packages:

```
yum  update   #  NOTE - THIS WILL TAKE SOME TIME
```

Upgrade
To upgrade just the kernel:

```
yum upgrade kernel*
```

List
To list installed packages:

```
yum  list  installed | less
```

To determine if a particular package is installed:

```
yum  list  installed | grep bc
```

List All
To list all available packages:

```
yum  list  all
```

Remove
To remove a package and its dependencies:

```
yum  remove  sysstat  bc
```

Install
To install a package:

```
yum  install  bc
```


Whatprovides
To install a package that cannot be found:

```
iostat

yum  whatprovides  */iostat

yum  install  sysstat

iostat
```

yum Error
An error such as the following may arise when running *yum*:

```
Loaded plugins: fastestmirror
Existing lock /var/run/yum.pid: another copy is running as pid 2923.
Another app is currently holding the yum lock; waiting for it to exit...
  The other application is: yum
    Memory :  13 M RSS ( 22 MB VSZ)
    Started: Fri May  4 09:07:24 2012 - 22:31 ago
    State  : Sleeping, pid: 2923
```

To clear the error, find and kill the conflicting process:

```
ps -ef  |  grep 2923

kill  2923
```

For more information on yum go to:

http://www.cyberciti.biz/faq/rhel-centos-fedora-linux-yum-command-howto/

Exercise

1. Examine some of the options of the *yum* command.

2. Use the *yum* command to find and install the commands *locate* and *updatedb*.

Oracle VirtualBox Pre-requisites

After installation of Linux as a VM, the following may be required in order to install VirtualBOX Guest Additions:

```
yum install kernel-devel kernel-headers gcc

yum upgrade kernel*
```


Appling RHEL Updates

Systems must be registered before updates from RHN can be applied. This can be done via System -> Administration -> Software Update. It can also be accessed via the command line by using the command *rhn_register*.

Registration requires local root access and a valid RHN login that has an unused Update or Management entitlement for the new system to be registered.

Sometimes it may be necessary to apply only certain updates, for instance, those advisories that are tagged as fixing security issues (RHSA) or bug fixes (RHBA).

To manually select updates:

- Log into Red Hat Customer Portal.
- On the Subscriptions -> Classic Management -> Registered Systems page, select which system you want to update.
- Click the Software tab.
- Click the Errata tab.
- Select the errata you want to apply.
- Click "Apply Errata" at the bottom right of the page.

On the actual system to be updated run the following command:

```
rhn_check
```

This will apply the selected errata to the system.

Red Hat Package Manager

The **rpm**, Red Hat Package Manager, utility has become very popular and can be run on CentOS, Red Hat, Ubuntu and even SUSE systems. It is most often used to install local files and can also determine the state of installed files. However, unlike yum it will NOT automatically discover dependencies.

An RPM file is named using the following convention:

```
name-version-release.architecture.rpm
```

For some example files run:

```
ls  /package
```

List
To list available packages:

```
rpm -qa | less
```

To determine if a particular package is installed:

```
rpm -qa | grep figlet
```

Remove
To remove the same package (note the missing rpm suffix):

```
rpm -e figlet
```

or

```
rpm -e figlet-2.2.1-1.2.el4.rf
```

Install
To install a file with verbose output (-v) and showing hashes (-h) to denote installation progress:

```
cd  /package

rpm  -ivh  figlet*
```

Test the install with:

```
figlet  hello
```


To discover what package an installed file comes from:

```
which  figlet

rpm -qf /usr/bin/figlet
```

Dependencies
To ignore dependency problems and install the file:

```
rpm --nodeps figlet-2.2.1-1.2.el4.rf.i386.rpm
```

Checksig
To verify the signature on an RPM file to be installed:

```
rpm --checksig figlet-2.2.1-1.2.el4.rf.i386.rpm
```

Exercise

1. Examine some of the options of the *rpm* command.

2. Use the *rpm* command to install the package *sl* (steam locomotive) from the directory /package.

   Hint: Resolve the dependency issue using *yum*.

3. Use the *rpm* command to install the package *webmin* test the package. Test it using localhost:10000.

Installing Software From Source

Some software packages cannot be installed with yum or rpm but are downloaded as tarballs, a file with the tar.gz extension. These packages need to be compiled from the source files.

The Procedure
The installation procedure for software that comes in tar.gz packages usually involves the following steps:

- unpack the file — tar xvzf pkg.tar.gz
- change to the directory — cd pkg
- run configure — ./configure
- run make — make
- run make install — make install

These simple commands will unpack, configure, compile and install the software package.

Unpack
If the package has the extension tar.gz it is a compressed tar archive also known as a tarball. When making the package, the source code and the other needed files were put together in a single tar archive, hence the tar extension. The archive was then compressed with gzip, hence the gz extension.

Some files are compressed with bzip2 instead of gzip. In these cases the package has a tar.bz2 extension. Install these packages in exactly the same way as tar.gz packages but use bunzip2 to unzip the package.

The following example installs the game *lbreakout2*. First check that the game is not installed:

```
lbreakout2
```

Then unpack the downloaded tarball with this command:

```
cd  /package ; ls

tar  xvzf  lbreak*.tar.gz
```

In most cases a directory with the package's name is created:

```
cd  lbreak*3 ; ls
```

Read any documentation in this directory, such as README or INSTALL files, before continuing!

```
less README
```


Configure
Usually, but not always (check the README and INSTALL files) the next step is to run the configure script. To save a log of the output and check for errors use:

```
./configure | tee /tmp/conf.log

tail  /tmp/conf.log

grep  error /tmp/conf.log
```

The configure script will display several screens of messages but it doesn't actually compile anything; it just checks the system and assigns values for system dependent variables. These values are used for generating a Makefile. The Makefile in turn is used for generating the actual binary files.

Check for error messages such as:

```
configure: error: libz is needed
```

and install any missing packages.

NOTE: If sdl-config cannot be found, use yum to install the package SDL-devel.

Rerun configure as there must be no errors before the next step is run and the Makefile must exist:

```
ls  -l  Makefile
```

Make
To actually build the binary, the executable program, from the source code run the make command:

```
make
```

Note that make needs the Makefile for building the program otherwise it doesn't know what to do. This is why it's important to run the configure script successfully first.

This step will also display several screens of messages and may take some time depending on how big the program is and the speed of the server.

Make Install
Finally install the compiled program with the make install command:

```
make install
```

The program will be installed by default in /usr/local/bin. Test the install with:

```
lbreakout2
```

Cleaning Up
To remove temporary files run:

```
make clean
```

Uninstalling
First read the doumentation to see if it says anything about uninstalling. Attempt the following:

```
make uninstall
```

If this fails then remove the program files manually.

Debian Package Management

The **apt-get** and **apt-cache** commands are the most sophisticated way to manage Debian packages on systems such as Ubuntu. One of the primary benefits of apt-get is that it will automatically discover dependencies and then install them. This feature is different from RPM, which will simply notify a failure and then quit.

The command uses the */etc/apt/sources.list* file which contains a default set of repositories. It is possible to edit this file and add various repositories, depending on what is to be installed.

```
less  /etc/apt/sources.list
```

The distribution can be either the release code name (lenny, etch, squeeze, maverick) or the release class (stable, oldstable, testing, unstable) respectively. If tracking a release class use the class name, if tracking a Debian point release, use the code name.

Once added, the user can issue commands to install applications and daemons.

Run this command after changing the */etc/apt/sources*.list file:

```
apt-get clean

apt-get update
```

After installation from a DVD, to upgrade all installed packages:

```
apt-get upgrade
```

To find a package to install:

```
apt-cache search vim | grep vim-tiny
```

To show information about the package:

```
apt-cache show vim-tiny | less
```

To check to see if it is already installed:

```
apt-cache policy vim-tiny
```

To install the package:

```
apt-get install vim-tiny
```


To uninstall everything in the same package, except for the configuration files:

```
apt-get remove vim-tiny
```

To uninstall the entire package, including its configuration files:

```
apt-get --purge remove vim-tiny
```

Using dpkg

The dpkg command is not as sophisticated as apt-get. It will not install dependencies as easily and it is best for installing local files. However, it must be used when you cannot get a package using apt-get or when you want to list information about installed packages. It is also useful when you want to list the contents of a package before installing it.

A Debian file is named using the following convention:

```
name-version-release.architecture.deb
```

For example:

```
fakeroot_1.12.1ubuntu1_i386.deb
```

To search the listing of packages:

```
less  /var/lib/dpkg/available

grep  Package  /var/lib/dpkg/available  |  less
```

To list packages related to a package:

```
dpkg  -l  “*cups*”
```

To see the entries in /var/lib/dpkg/available of a package:

```
dpkg  --print-avail cups vim-tiny | less
```

To install a package, first find it in an archive or on a CDROM:

```
cd  /media/U*/pool/main/f/fakeroot ; ls

man  fakeroot

dpkg -i fakeroot*.deb
```


To remove an installed package:

```
dpkg -r fakeroot
```

To uninstall all elements in the package provided there are no dependencies:

```
dpkg --purge vim-tiny
```

The /var/lib/dpkg/ directory contains various files, including /var/lib/dpkg/status, which dpkg uses to remember what has been installed. If this file is missing, dpkg will not work properly. Recover the file from the /var/lib/dpkg/status-old file.

Using alien
The alien command converts packages from other managers to Debian packages. It can convert the following packages:

- RPM
- Slackware (.tgz)
- Solaris (pkg)

For example, to convert the figlet rpm file issue the following command:

```
alien figlet-2.2.1-1.2.el4.rf.i386.rpm
```

Alien will convert the file into a new file with a .deb ending. You can then install it using dpkg. You can also use alien to convert Debian packages to RPM and other formats.

On Debian systems such as Ubuntu the Synaptic Package Manager is an excellent tool for finding, fetching and installing packages. Press System > Administration > Synaptic Package Manager to start Synaptic.

Startup and Shutdown

Objectives

At the end of this section the delegate will be able to:

- describe how the system boots
- describe the system files used at start up and shutdown
- start up and shutdown the system

Booting the System

The first sector of the hard disk is reserved for booting and is called the master boot record (**MBR**). When booting from a hard disk, the system BIOS loads and executes the boot loader code in the MBR. The MBR then needs to know which partitions on the disk have operating systems which can be loaded.

To do this the MBR boot loader loads a second stage boot loader. This then reads the data in the **configuration file** which lists all the available operating systems and their booting parameters. When this is complete, the second stage boot loader displays a splash screen that lists all the configured operating systems available.

GRUB is the default boot loader, but the older boot loader LILO is available for those who prefer it.

One common task performed by GRUB in the Linux world is to allow users to dual boot Linux and Microsoft Windows on a single PC. The typical pattern for creating a dual boot system is to install Windows, if it is not already installed, then install Linux on another partition. Most Linux distributions will detect the Windows installation and automatically install and configure the boot loader.

GRUB - GRand Unified Bootloader

GRUB, the GRand Unified Bootloader, is an integral part of many Linux systems and it is this that displays the boot menu and starts the Linux kernel.

Changes to the boot menu can be made in one of two ways:

- simply edit the *menu.lst* file
- enter the GUI interface by typing an '*e*' when the GRUB menu is invoked

The GRUB menu file can be viewed as:

```
less /boot/grub/menu.lst
```


This file is a plain text file with a set of directives and configuration parameters.

- *default* specifies which entry is the default. An entry comprises, at least, *title*, *root*, and *kernel* directives. Entry numbers start at 0 and increment upward.

- *timeout* specifies how long, in seconds, the menu will be displayed before the default entry is executed.

- *splashimage* is the image that will be displayed.

- *title* is the text that is displayed in the menu for the entry that follows.

- *root* tells GRUB on what device and partition it can find the kernel for this entry.

- *kernel* specifies what kernel will be booted if this entry is selected. Options after this directive are passed to the kernel for processing. Here, *ro* means read-only (the device is not physically made read-only; it just tells the kernel not to attempt writing), *quiet* indicates not to display debug information and *splash* means show a splash screen while booting.

- *initrd* tells GRUB what to run after the kernel has been loaded. When this directive is executed, GRUB passes off control of the system to the operating system.

Note that a new version of GRUB, GRUB2, stores the file here:

```
less /boot/grub/grub.cfg
```


The Kernel

Once the second stage boot loader has determined which **kernel** to boot, it locates the corresponding kernel binary in the /boot directory. The boot loader then loads the appropriate **init**ial **R**AM **d**isk image, called **initrd**, into memory. Once the kernel and the initrd image are loaded into memory, the boot loader hands control of the boot process to the kernel. The kernel file can be viewed thus:

```
ls  -l  /boot
```

When the kernel is loaded, it immediately initializes and configures memory and then configures the various hardware attached to the system including all processors, I/O subsystems and storage devices.

It then looks for the compressed initrd image in a predetermined location in memory, decompresses it, mounts it and loads all necessary drivers.

Next, it initializes virtual devices related to the file system, such as LVs or software RAID before unmounting the initrd disk image and freeing up all the memory the disk image once occupied.

The kernel then creates a root device, mounts the root partition read-only, and frees any unused memory.

At this point, the kernel is loaded into memory and operational. However, since there are no user applications that allow meaningful input to the system, not much can be done with it. In order to set up the user environment, the kernel then executes the /sbin/init program.

init Process

The */sbin/init* program (also called init) coordinates the rest of the boot process and configures the environment for the user.

When the init command starts, it becomes the parent or grandparent of all of the processes that start up automatically on the system. First, it runs the */etc/rc.d/rc.sysinit* script, which sets the environment path, starts swap, checks the file systems and takes care of everything the system needs to have done at system initialization.

The init command then runs the */etc/inittab* script, which describes how the system should be set up for each runlevel and also sets the default runlevel.

init Process

In Red Hat 5 the *init* process looks at the file */etc/inittab* to find out which processes to start on startup and also when and how to start them. The *init* process will also monitor its child processes. When one terminates, it will again consult the *inittab* file to find out what to do next.

```
less /etc/inittab
```

init - Upstart Process

In Red Hat 6, *init* is now synonymous with *upstart*, an event-based init system. This system handles the starting of tasks and services during boot, stopping them during shutdown and supervising them while the system is running. For more information on *upstart* itself, refer to the init man page.

Processes are known to Upstart as jobs and are defined by files in the /etc/init directory:

```
ls  /etc/init
```

With *upstart* the */etc/inittab* file is deprecated and is now used only for setting up the default runlevel. Other configuration is done via *upstart* jobs in the /etc/init directory.

Red Hat 5 /etc/inittab

These extra lines exist in Red Hat 5 only:

```
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm –nodaemon
```


/etc/inittab

An explanation of this file will help explain how startup works. Each line in the *inittab* file defines a process to run and has four fields separated by a colon, for example:

```
LABEL : RSTATE : ACTION : PROCESS

l5:     5:      wait:   /etc/rc.d/rc 5
```

The Label Field

Each line must have its own unique label. Whenever *init* starts or stops a process, it uses this label to identify the process.

The Rstate Field

The *init* command uses run levels or run states to identify when to run processes. Whenever a run state changes, *init* consults the *inittab* file and any lines which have a matching run state in this field will then be run.

Run states can be listed as single values, multiple values or if the field is left blank this means all run levels.

Linux supports seven different run levels, typically used in the following manner:

0	Halt
1(S)	Single user mode
2	Multi-user without NFS
3	Multi-user mode with NFS
4	Unused
5	Multi-user mode, with X11
6	Reboot

The time of the last boot up and the current run level can be found as follows:

```
who  -br
```


The Action Field

The action field defines how to run the process. The following are some of the actions that can be used:

initdefault
This is a special flag, typically on the first line of *Inittab*, which tells *Init* what run state to set when it first starts. The run state to be set is specified in the run state field. DO NOT set it to 0 or 6!

sysinit
Used for programs that initialise devices, usually during bootup.

wait
Runs the specified process then waits for termination before reading next line.

ctrlaltdel
Issues a shutdown command if the key combination CTRL & ALT & DEL is pressed.

powerfail
Some machines have a capability to signal a power failure. If this signal is seen, run the specified process.

respawn
Runs the specified process. If the process ever terminates restart it. Used mostly on getty processes for terminals and modems.

Examine the file *inittab* to see what steps are taken once *init* has started.

The Process Field

This field defines the command to be executed. Typically this is a special run control script. The *init* command uses the same control script for each run level. The script may be viewed as follows:

```
less  /etc/rc
```

The control script is actually a link located in the /etc directory and is passed a different parameter for each run level.

```
ls  -ld  /etc/rc*
```

The script executes files in a set of directories in the /etc directory. These define the sequence in which scripts are performed within each run level. For example, the /etc/rc5.d directory contains files that start and stop processes for run level 5. To view the files for run level 5:

```
ls   /etc/rc5.d
```

The scripts here have the following characteristics:

- The scripts are always run in ASCII sort order
- The names of the scripts have the form [K,S] [0-9] [A-Z] [0-99]
- Files beginning with S are run to start a system process
- Files beginning with K are run to terminate or kill a system process

Note that here the file S99local is for installation of additional software:

```
cat  /etc/rc5.d/S99local
```


Shutdown Commands

Shutdown
This is an executable shell script that calls the *init* process in order to shut down the system. To see the options available enter:

```
shutdown  --help
```

The command is recommended over commands such as *halt* and *reboot* because users are notified of the impending shut down as are the systems that are mounting resources from the server being shut down.

The useful option -k only tests the command, it does not actually shutdown the server.

Shutdown - Halt
This command shuts down and halts the system in 10 minutes:

```
shutdown  -k  -h  +10  LINUX IS SHUTTING DOWN
```

Shutdown - Reboot
This command reboots at a particular time:

```
shutdown  -k  -r 15:00  LINUX IS REBOOTING
```

Shutdown - Single User Mode
To bring the system down to single user mode, for example in order to backup a file system, use:

```
shutdown  now
```

To check the current run level use the *runlevel* command or:

```
who  -r
```

To bring the system back up to the default run level enter *exit*.

Halt, Reboot, Init
These commands exist but are **not** recommended on a shared system.

Note that ordinary users should **not** have access to these or the *shutdown* command.

Booting to Command Prompt

If a problem occurs with the GUI login screen it may be necessary to make a temporary change and boot to a command prompt by going to single user mode in order to login. Do this as follows:

- Shutdown the system and then reboot.
- Type an ‘e’ to enter edit mode when the GRUB menu appears.
- Select the *kernel* to be edited and again type an ‘e’.
- Select the line to be edited and type an ‘e’ again. Add an S to the end of the line:

  kernel /vmlinuz-2.6.32-358.el6.i686 ro root=UUID…… rhgb quiet S

- Press enter to save the change and then type a ‘b’ to boot the system.

Review Exercise

1. Familiarise yourself with the *shutdown* command.
2. Now use the command to go down to *single user* mode.
3. Bring the server back up to the *default* run level.

   If you have time:

4. Add a message of the form *SYSTEM COMING UP* to the file S99local for run level 5.

   Hint: use the command *figlet* and redirect the output

5. Test the script at the command line:

   ./S99local

6. Use the *shutdown* command to *reboot* the server and check that the message appears.

Service and Printer Management

Objectives

At the end of this section the delegate will be able to:

- mangage, start and stop services
- monitor services and resources
- install and manage printing services

Overview of Services

Services are programs or daemons that once started run continuously in the background. Some wait for input and some monitor changes in the server and respond to them. For example the Apache server has a daemon called httpd (the d is for daemon) that listens on port 80 on the linux server and when it receives a request for a page it sends the appropriate data back to the client machine.

Many services are required to run all the time. However, many can be safely turned off for both security reasons, as running unnecessary services opens doors into the server, and for performance reasons. If the user turns off a service and loses some functionality they can just turn it on again without doing any harm. For example, the user may decide to start services such as Apache, MySQL and Samba manually as required.

There are 2 main commands used to control services.

chkconfig
This controls which services are set to start on boot up; by their nature these settings are saved and are applied at next boot. Changing these settings will not start the service immediately it will just flag the service to be started from the next boot up.

service
This controls the starting and stopping of services during a session; these settings are not saved. If Apache is started this way but is not set to start on boot up, it will continue to run but on the next boot up will not start automatically.

/etc/rc.d/init.d scripts

In order to understand how chkconfig works, it's necessary to review how scripts and run level directories are set up. The scripts to run services are all located in /etc/rc.d/init.d on Red hat and CentOS systems although this may differ on other systems.

There are a set of run level directories under /etc named rc0.d, rc1.d, and so on that map to the various run levels. In order to execute a script (start a service) in a specific run level, a symbolic link to the service's script in /etc/rc.d/init.d is created in the directory of interest. So, for example, if the isdn script is to be run for run levels 4 and 5, but not for 0-3 or 6, a symlink to /etc/rc.d/init.d/isdn is created in both /etc/rc4.d and /etc/rc5.d.

This way, if the script for a service needs to be modified, only the original script in /etc/rc.d/init.d needs to be dealt with. The symlinks for each run level automatically pick up the changes.

In order to have a script execute for a certain run level, all that's needed is to create a symlink in that run level's 'rc' directory to the original script file. The command chkconfig, by virtue of the arguments passed to it, takes care of this.

chkconfig Command

To get information about services use the --list option to the chkconfig command. To see the status of all services type:

```
chkconfig  --list
```

This returns a long list of all services, each column refers to a different run level. In most cases the server is booted into level 5 so this is the column of interest. The ‘on’ and ‘off’ status refers to whether the service is set to start on boot up, it does not state whether the service is currently running.

To query the status of just one service, use grep to filter the returned data. Here we use chkconfig but only want to see the Apache service.

```
chkconfig  --list | grep httpd
```

This shows that Apache is not set to start on boot in all run levels.

Use grep again to see only those services not set to start on boot for a particular run level. Here we see services not set to start in run level 5.

```
chkconfig  --list | grep 5:off | less
```

To see services that are set to start in run level 5:

```
chkconfig  --list | grep 5:on | less
```

To alter which services start at boot time run chkconfig with different arguments. For example to set Apache to start on boot in run level 5:

```
chkconfig  --level 5 httpd on
```

To set Apache to start on boot in run levels 3, 4 and 5:

```
chkconfig  --level 345 httpd on
```

```
chkconfig  --list | grep httpd
```

To stop Apache starting on boot up, replace 'on' with 'off':

```
chkconfig  --level 345 httpd off
```


To remove services from chkconfig control use:

```
chkconfig  --del  sshd

chkconfig  --list | grep sshd
```

This will remove the symlinks in all of the run level directories, which effectively turns the service off at startup. To add the script back in for its' default run levels:

```
chkconfig  --add  sshd

chkconfig  --list | grep sshd
```

So how does chkconfig know what to do? If the --add argument is passed, how does *chkconfig* know which run levels are the default for the particular script?

The chkconfig program looks inside the original script (inside /etc/rc.d/init.d) for the comment line that begins #chkconfig:

```
grep  chkconfig  /etc/init.d/sshd
```

This might return:

```
# chkconfig: 2345 55 25
```

The first argument identifies which run levels are the defaults, for example 2, 3, 4, and 5.

The second argument is the order of the script during startup, for example 55.

The third argument is the order that the service should be run when killed, for example 25

These arguments, 55 and 25, are used for naming the symlinks as S55sshd and K25sshd for this script:

```
find  /etc/r*/rc*  -name  "*sshd"  2>/dev/null
```

Review Exercise

1. Use the *chkconfig* command to check the status of the *nfs* service to see at which run levels it is set to on.

2. Set the *nfs* service to start at run levels 3, 4 and 5.

Service Command

The user can manually start a service when required. This way they can turn off many services not actually required for the normal operation of the server and just start them when the need arises.

To check to see if a service is running use:

```
service  httpd  status
```

To start a service simply use the service command, this example uses Apache (httpd service) as an example but the command is the same for any service:

```
service httpd start
```

```
service httpd status
```

Stopping a service is just as easy:

```
service httpd stop
```

Restarting just uses restart in place of start or stop:

```
service httpd restart
```

Note that this will start a service for this session but after rebooting this service may not automatically restart.

To view a list of all the services use:

```
service  --status-all
```

To view a list of all the currently running services:

```
service  --status-all | grep running
```

To see a list of all stopped services:

```
service  --status-all | grep stopped
```


init.d

The service command with the ‘start’ option runs the startup script for the service. The script for httpd for example may be viewed thus:

```
less /etc/init.d/httpd
```

If the service command is not available, the service may be started and stopped by directly running the script:

```
/etc/init.d/httpd start

/etc/init.d/httpd stop
```

inetd, xinetd - Internet Daemon

The **I**nter**NET D**aemon, **inetd**, is a super-server daemon on many Linux and Unix systems that manages Internet services such as ftp and telnet. The e**X**tended **I**nter**NET D**aemon, **xinetd**, is an open-source daemon which also manages Internet-based connectivity but offers a more secure version of inetd.

The xinetd daemon performs the same function as inetd; it starts programs that provide Internet services. However, instead of having such services started at system initialisation time and then have them dormant until a connection request arrives, xinetd is the only daemon process started and it listens on all service ports for the services listed in its configuration file, */etc/xinetd.conf*. When a request comes in, xinetd starts the appropriate server.

To find out if the server is running either of these daemons run:

```
ps -ef | egrep '[xi]netd'
```

GUI Tools

System > Administration > Services

Review Exercise

1. Use the *service* command to view a list of all services running.

2. Check the status of the *nfs* service to see if it is running.

3. Start the *nfs* service and re-check the status.

4. Now stop the *nfs* service and re-check the status.

Manage Debian Services

Traditionally, Debian systems such as Ubuntu provided various tools to manage services:

- /etc/init.d/service
- rcconf
- update-rc.d

Under Red hat and Centos the chkconfig command can be used to configure Sys V style init script links and the service command can stop, start and restart services.

To use the chkconfig and service commands on Debian distributions, first install the following packages:

```
apt-get install chkconfig  sysvinit-utils
```

This package also installs a simple GUI for managing run levels:

```
sysv-rc-conf
```

System Monitoring Tools

These built-in commands can be used to find hardware values as well as storage, CPU or memory bottlenecks. Some provide metrics which can be used to get information about system activities and to find the possible causes of a performance problem.

Note that some of these tools may need to be installed and that the *man* command can be used to view the options available.

dmidecode
This is a tool for dumping a server's DMI or SMBIOS table contents in a human readable format. The table contains a description of the system's hardware components as well as other useful pieces of information such as serial numbers and BIOS revision:

```
dmidecode
```

top - Process Activity Command
The top program provides a dynamic real-time view of a running system i.e. actual process activity. By default, it displays the most CPU-intensive tasks running on the server and updates the list every five seconds:

```
top
```

Commonly used hot keys:

?	Displays help
t	Displays summary information off and on
m	Displays memory information off and on
A	Sorts the display by top consumers of various system resources. Useful for quick identification of performance-hungry tasks on a system
f	Enters an interactive configuration screen for top. Helpful for setting up top for a specific task
o	Enables you to interactively select the ordering within top
r	Issues renice command
k	Issues kill command
z	Turn on or off color/mono

Also available is **atop**, a bit like top on speed. The full description is AT Computing's System & Process Monitor, an interactive utility to view the load on a Linux system.

w - Who Is Logged On
w command displays information about the users currently on the machine and their processes,

```
who

w
```

wall – Notify Users
The wall command sends a message to all users logged on:

```
wall  SYSTEM GOING DOWN IN 10 MINUTES
```

uptime - System Running Time
The uptime command can be used to see the current time, how long the system has been running, how many users are currently logged on and the system load averages for the past 1, 5, and 15 minutes.

```
uptime
```

A system load average of 1 can be considered as an optimal load value. The load can change from system to system. For a single CPU system a load value of 1 - 3 might be acceptable.

ps - Displays Processes
ps command reports a snapshot of the current processes. To select all processes:

```
ps  -ef
```

ps is just like top but provides more information.

Other useful tools include:

free - Memory Usage
The command free displays the total amount of free and used physical and swap memory in the system as well as the buffers used by the kernel.

pmap - Process Memory Usage
The command pmap displays a memory map of a process. Use this command to find out causes of memory bottlenecks. Use *man pmap* to examine the options.

vmstat - System Activity
The command vmstat reports information about processes, memory, paging, block IO, traps and cpu activity.
Here the utility runs 10 times, reporting every 10 seconds:

```
vmstat -n 10 10
```

iostat - Average CPU Load, Disk Activity
Part of the sysstat package the command iostat reports Central Processing Unit (CPU) statistics and input/output statistics for devices, partitions and network filesystems (NFS).
Here the utility runs 10 times, every 10 seconds, reporting extended statistics:

```
iostat  -x 10 10
```

sar - Collect and Report System Activity
The sar command is used to collect, report and save system activity information.

To monitor CPU utilization, every 2 seconds, 10 times:

```
sar -u 2 10
```

To monitor disk activity, 10 iterations, every 5 seconds:

```
sar -d 5 10
```

There is also a graphical interface for sar called Ksar.

Review Exercise

1. Run the *top* command and try out some of the options.

2. Try some of the other commands, if installed, for example *iostat* and *sar*.

3. Browse several system logs.

CUPS Printing System

CUPS is the Common UNIX Printing System. It is aimed at providing a common printing interface across a local network masking differences among the printing systems on each computer. It also provides interactivity with Windows printers and allows dynamic printer detection and grouping.

CUPS is a replacement for the LPD printing system. It replaces the lpr command with its own command lp and the LPD printer drivers with its own versions. However, CUPS is similar to LPD in that it uses PostScript as its underlying language for page descriptions. Linux and UNIX programs don't know the difference between CUPS and LPD.

Installation

If CUPS is not installed, install the package as follows:

```
yum install cups

service cups status
```

After a short delay the CUPS software will be installed and the scheduler will be started automatically.

If a real printer is not available a device may be used. To enable this facilty first check the file */etc/cups/cupsd.conf* as follows:

```
grep FileDevice /etc/cups/cupsd.conf
```

If the line is missing update the file */etc/cups/cupsd.conf* with the line:

```
FileDevice Yes
```

as follows:

```
echo  "FileDevice Yes" >> /etc/cups/cupsd.conf
```

Then restart the service:

```
service cups restart
```


Managing Printers

Each printer queue has a name associated with it; case is not significant. Printer queues also have a device associated with them. The device can be a parallel port, a network interface and so forth. Devices within CUPS use Uniform Resource Identifiers (URIs) which are a more general form of Uniform Resource Locators (URLs) that are used in a web browser.

For example, the first parallel port in Linux usually uses a device URI of:

```
parallel:/dev/lp1
```

For a complete list of supported devices run:

```
lpinfo  -v
```

The first word in each line is the type of device, direct, file, network, or serial. This is followed by the device URI or method name for that device.

Finally, printer queues usually have a PostScript Printer Description (PPD) file associated with them. PPD files describe the capabilities of each printer, the page sizes supported, etc., and are used for PostScript and non-PostScript printers.

lpadmin - Adding a Printer

CUPS provides several methods for adding printers: a command-line program called lpadmin, a GUI tool and a Web interface. The lpadmin command allows the user to perform printer administration tasks from the command-line and is located in /usr/sbin.

To view the print queue run:

```
lpstat  -t
```

To add a printer run the *lpadmin* command with the -p option:

```
lpadmin -p lp1 -v /dev/pts/2

lpadmin -p lp2 -v /dev/pts/3

lpstat -t
```

Note that any new printer is both *disabled* and is *not accepting* print requests.

Accept and Reject

The *accept* and *reject* commands accept and reject print jobs for the named printer. For example:

```
accept  lp1 ; accept lp2; lpstat -t
```

A printer can be stopped but can still be accepting new print jobs. A printer can also be rejecting new print jobs while it finishes those that have been queued. This is useful when maintenance is needed on the printer and it will not be available to users for a long period of time.

Set Default Printer

Run the *lpadmin* command with the -d option to set a default printer:

```
lpadmin -d lp1
```

Printing Files

The *lp* command sends files to the default or a named print queue:

```
lp /etc/passwd; lp /etc/group

lp /etc/hosts -d lp2 ; lpstat -t
```

Enable and Disable

The *cupsenable* and *cupsdisable* commands start and stop printer queues. For example:

```
cupsenable  lp1 ; lpstat  -p  lp1
```

Printers that are disabled may still accept jobs for printing, but won't actually print any files until they are restarted.

lpmove - Moving Files

The *lpmove* command can be used to move waiting prints from one printer to another:

```
lpmove  lp1-1  lp2 ; lpstat -t
```

To move all prints use:

```
lpmove  lp1   lp2 ; lpstat -t
```

Deleting a Printer

Run the lpadmin command with the -x option to delete a printer:

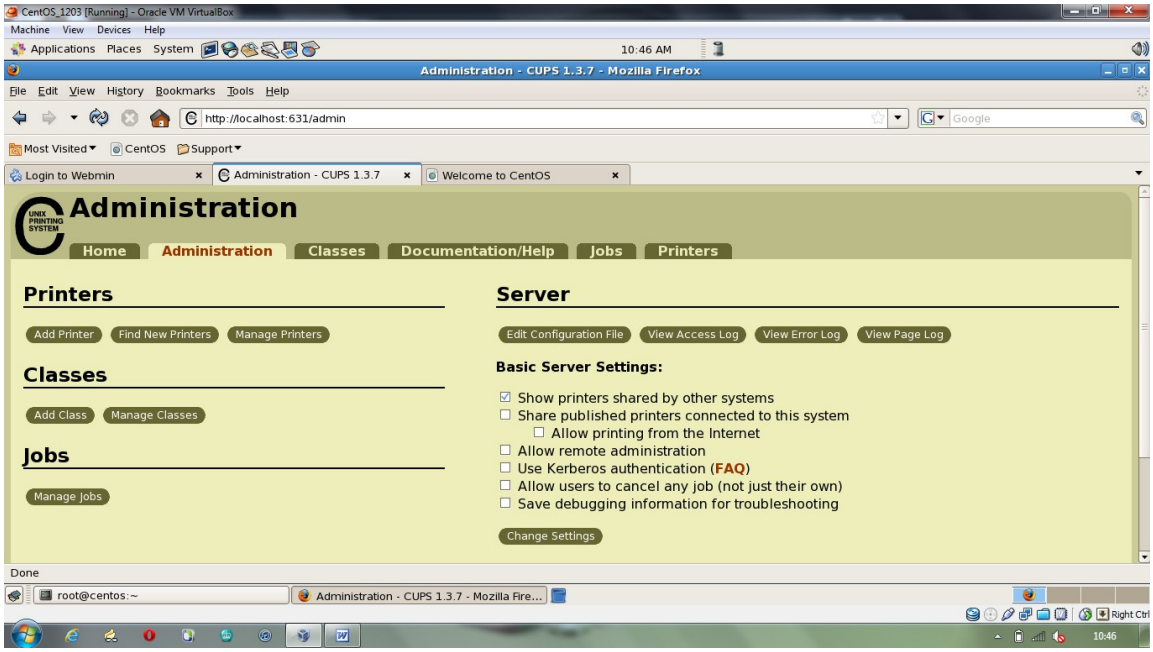
```
lpadmin -x lp1
```

GUI Tools

System > Administration > Printing

Web Tool

The web interface for lpadmin is located at: http://localhost:631/admin



Review Exercise

1. Check that the *cups* service is running and that the line 'FileDevice Yes' exists in the cups *configuration* file.

2. Open a second terminal screen and run the *tty* command.

3. Add a new printer named *mylp* using as the device the output from the *tty* command, for example /dev/pts/2.

4. Make this new printer the *default* printer for the system and confirm this by viewing the print queue.

5. Print several files to the new printer and view the print queue.

User Management

Objectives

At the end of this section the delegate will be able to:

- describe relevant system files for managing users
- add, modify and remove users and groups
- restrict access to the su command
- describe how to control access to the system
- describe and use ACLs, Access Control Lists

/etc/passwd File

Each line in the */etc/passwd* file defines system and normal user accounts. In order to add a user to the system this file must be modified. View the file using:

```
less  /etc/passwd
```

Each user entry is split into seven fields as follows, each separated by a colon:

Login : Password : Userid : Groupid : Description : Home Directory : Program

Every user on the system must have read permission on the */etc/passwd* file but no one should have write permission including root, since this will help prevent accidental corruption or deletion. If the file becomes corrupted, then no one will be allowed to log onto the system. Unless a root login is already running, the system will have to be booted from CD or DVD to cure the problem.

Login
This is the account name for the user and can be up to eight characters long. Any lower case letter or digit may be used in the name, but it is worth avoiding punctuation and non-printing characters. Login names must be unique.

The following command may be used to check whether a particular login exists:

```
grep  '^root'  /etc/passwd
```


Password
This field is no longer used and will contain an x character. Historically it contained the users encrypted password. It was then very easy for any user to look at the password file and find all users who did not have a password set.

This is now impossible as the encrypted password is stored in another file called */etc/shadow*. This file is read permission only to the super user.

```
less  /etc/shadow
```

Users can assign themselves passwords using the *passwd* command.

Userid
This is a unique number assigned to each user to identify them to the system. The system always stores the identity of a user as a number and then looks up the user login when necessary. When adding a new user the next available *userid* should be used but users should not have the same *userid*.

Note that all users with a *userid* of zero are super users so as well as *root* there can be other super user logins.

Groupid
This field contains the number of the group to which the users belongs. Valid groups are defined in the */etc/group* file.

```
less  /etc/group
```

Description
This field is available for comments about a particular account.

Home Directory
This field specifies the full path name of the users home directory. Note that if the directory does not exist or a mistake is made in the pathname, then a user will not be able to login.

Program
This is the full path to the shell program which the user will run upon entry to the system. When this program terminates, the user will be logged off.

This field, although typically a shell of one kind or another, can be an application or any other program required to start at login.

Checking the passwd File

The command *pwck* may be used to check the password file for errors and errors may be corrected with *pwconv*.

```
pwck
```


Adding a User

The process of adding a new user can be carried out in one of several ways:

- create the user manually by entering commands
- use the command *useradd*
- use the GUI or a web interface

Creating a User Manually

Although it is not normally done this way, the steps involved in manually adding a new user to the system are as follows.

- Log in as *root* and if required, edit the */etc/group* file and add a new entry
- Edit the */etc/passwd* file and add the new user
- Create a new home directory for the user
- Change ownership and group of this directory for the new user using the commands *chown* and *chgrp*
- Create a new password for the user
- Log out and log in as the new user

useradd - Creating a User

A new user may be added to the system using the command *useradd*. This command has many options including the following:

-c "comment"	Use open and close double quotes for the comment
-d directory	Full pathname of the new user's home directory
-m	Create the new home directory if it does not already exist
-g group	The group the user is placed in; the default is a new group with the name of the user
-s shell	Full pathname of the shell on login, the default is /bin/bash
username	The new username

The following examples add new users:

```
useradd -c "User userb" -d /home/userb -m -s /bin/bash userb

useradd -g users -s /bin/csh userc

useradd userd
```

To check the existence of a new entry in the passwd file:

```
tail /etc/passwd
```

To check that the users home directory exists:

```
ls -l /home
```

Before the account can be used, a new password must be added for the user:

```
passwd userb
```

Exercise

1. Create two new users with a home directory in */home* and a default group and shell. Allocate each user a password.

Default Values

The default values for the useradd command can be listed or changed by using useradd with the -D option:

```
useradd  -D
```

To change the default home directory or default shell for example, use:

```
useradd  -Db  /home/users

useradd  -Ds  /bin/ksh
```

Customising the Environment

Depending on the shell to be used, the following files are executed for each user when they log in and these files can be customised for each user.

/etc/profile	All
.bash_profile	Bash
.profile	sh,ksh,bash
.login	Csh

These files are executed for each user when they fork a new shell:

/etc/bashrc	All
.bashrc	Bash
.csh	Csh

/etc/profile, /etc/bashrc Files

These files contain system wide settings for all users:

```
less  /etc/bashrc
```

/etc/skel Files

The directory */etc/skel* contains files including hidden files which are automatically copied into any new user's home directory.

Any standard user variables, aliases or functions should be stored here:

```
cat  /etc/skel/.bashrc
```

/etc/profile.d

The directory */etc/profile.d* includes files that customise the colours used by commands such as *ls*.

/etc/shadow File

Each line in the */etc/shadow* file defines a password for a user account. In order to add a user to the system this file must be modified. View the file as:

```
less  /etc/shadow
```

Each entry is split into nine fields, each separated by a colon. For example:

```
train1 : erf130hj kd2Z : 15399 : 0: 99999 : 7 : : :
```

The fields are defined as follows:

1. userid
2. a 13 character encrypted password, or no characters signifying no password
3. date last changed; the number of days between 1 Jan 1970 and the date the password was last modified
4. minimum number of days between password changes
5. maximum number of days the password is valid
6. warn users number of days before the current password is due to expire
7. inactivity, number of days of inactivity allowed for that user
8. expire, an absolute data string when the login may no longer be used
9. flag set to zero for future use

Password Ageing

Users should change their passwords on a regular basis, perhaps every two or three months and the *root* user at least once a month. This can be enforced by using the password ageing facility using information stored in the */etc/shadow* file.

/etc/group File

Users on the system can be split into different groups to allow some users to share files while excluding other users. When a new user is created, the user is assigned a *groupid,* which is a number that identifies a specific group located in the */etc/group* file.

```
less  /etc/group
```

Each line in this file defines a group and has the following structure:

```
Name : Password : Groupid : Userlist
```

Name
The name of the group, up to eight characters long.

Password
The group password is rarely used since there is no easy way of applying a password to a group.

Groupid
A unique number used to identify the group.

Userlist
A list of users who may change into this group. This is not the same as those already belonging to the group as defined by the password file. On some occasions a user may need to access more than one group. By including their names in a comma-separated list, users who can change into this group may be defined. Even if there is no password on the group, a user cannot change to this group unless their name appears in this list.

Checking the group File

The command *grpck* may be used to check the password file for errors and errors may be corrected with *grpconv*.

```
grpck
```

usermod - Modifying a User

The details for a user may be modified using the command *useradd*. For example, the following alters the login shell for the user *userc*:

```
usermod  -s  /bin/bash  userc
```

Removing Users

Removing users is simple. However it is probably better to lock an account rather than remove it entirely. The first step is to lock the login. This can be done by using the password command:

```
passwd  -l  userb
```

The username should then be deleted from the /etc/group file and the user's files backed up before they are deleted. However, it may be best just to leave the account locked in case any of the files are needed.

userdel - Removing Users

The command to delete a user is *userdel*. This command deletes a user's login name from the system and makes changes to the system files and to the file system.

For example:

```
userdel  userb
```

The option -r may be used to remove the user's home directory from the system. The directory must exist but this format should not be used unless there is a certainty that the files owned by the user are no longer required:

```
userdel  -r  userc
```


groupadd - Creating a Group

A new group may be added to the system using the command *groupadd*. The following steps add a new group to the system:

- Add the new group, specifying the group name

```
groupadd  dev
```

- Check the existence of a new entry in the group file

```
tail  /etc/group
```

groupdel - Removing a Group

A group may be deleted from the system using the command *groupdel* as follows:

```
groupdel  dev
```

groupmod - Modifying a Group

The details for a group may be altered using the command *groupmod.*

newgrp - Using Groups

The command *newgrp* may be used to change a user's group identification. When a user runs the command, the system places them in a new shell and changes the name of their group to the group specified.

The changes only last for the current session and a user can only change their real group name to a group they are already a member of. However, the root user can change their real group to any group regardless of whether they are a member of it or not.

First, add the user to a secondary group:

```
groupadd  dev

usermod  -aG  dev  userc

tail  /etc/group
```

Note that if the -a flag is not used the user is added to a new group but is **removed** from all other secondary groups.

Now that train1 is a member of the group dev, they can change groups as follows:

```
newgrp dev
```

Any files created will now be in the dev group.

This command forks a new shell. To leave the shell enter:

```
exit
```

Note that the newgrp command does not take input from standard input and cannot be run from within a script.

Review Exercise

1. Create the following users each with a home directory in */home* and with a default group and shell. Allocate each user a password.

   train1, train2, mary, jane

2. Also create a new user for your own use.

3. Tail the */etc/passwd* file to see the new users and log in as your user.

4. Edit the following file in the directory */etc/skel*.

   vi .bashrc

   Customise this file, for example, add the prompt PS1 and create an alias as follows:

   ```
   PS1='$PWD: '
   alias c='clear'
   ```

5. Create the following users each with a home directory in */home*, a default group of *users* and a default shell of *bash*:

   fred, alan

6. Log in as fred and check that the *.bashrc* file is executed.

7. Amend the users *mary* and *jane* to have a default group of *users*.

8. Remove the users *jane* and *fred*.

9. Add a new group dev and check that it exists in the group file.

10. Modify the users *alan* and *mary* to use this new group.

11. Check that the users may use the new group.

Access control lists

The basic Linux permission model lets you specify permissions for the file's owner and group and all others. The Access Control List (ACL) feature extends the model to allow much finer control; you can specify permissions for each individual user and group defined in your system.

Consider this scenario: your server supports multiple office departments: sales, marketing, and helpdesk. Each department has a manager and one or more staff members.

First define a group for each department (these comprise a manager and staff members) sales, marketing and helpdesk. Then define a managers only group: managers.

It is normal that some departments need to share files among each other but not with all departments. For instance, suppose sales needs to share a file with marketing, but not with helpdesk. To set that up using only the basic permissions, you can define yet more groups: sales-marketing, sales-marketing-managers, etc.

Alternatively, a better way is to use ACLs to assign permissions to individual groups and users.

By default ACL is not in use To explicitly turn it on for certain partitions edit the file /etc/fstab. Find the partition that you want ACL enabled for and add the mount option *acl*. For example:

```
LABEL=/home   /home   ext3  defaults,acl  0  2
```

Then dynamically remount the partition:

```
mount -o remount,acl /home
```


Suppose you have a file /home/train1/logfile that you want to share between sales, marketing, and a user named trainx.

Use setfacl -m to set Access Control List for the file.

```
setfacl -m group:sales:rw- logfile
```

The group:sales:rw- parameter specifies Read/Write permissions for the group: sales.

To enable the Read/Write permissions for the Marketing department, and trainx user:

```
setfacl -m group:marketing:rw-,user:trainx:rw- logfile
ls  -l
```

Note that ls -l does not display the actual ACL of a file. It only tells you that ACL is defined for that file: a plus character (+) is displayed to the right of the permissions.

To examine the actual ACL, run getfacl.

```
getfacl  logfile
```

A user can also specify the Read/Write/Execute permissions for any group or user on a directory. In addition, you can specify the default permissions for any file created under this directory.

Adding New Commands

Whenever a command is run the shell looks in two places:

Hash table	This is held in memory and contains a list of commands which have previously been executed in the current session.
PATH variable	This environment variable is checked and each directory is searched in turn for the specified command.

Whenever a new shell is generated a new hash table is created. Also, if the PATH variable is changed, the hash table will be reset. The contents of the hash table may be displayed as follows:

```
hash
```

The contents of the PATH variable can be displayed thus:

```
echo $PATH
```

If no command is found then the shell returns the appropriate error. If a command is located then it is loaded into memory and executed. This new program becomes known as a child process, and the original shell is the parent. Once the child process terminates, the parent process then resumes control.

Note that a command may be available for use but may not be found in the current path. Here the user may specify the full path name. For example:

```
runlevel

/sbin/runlevel
```

If the user needs access to the command then the directory should be added to their PATH variable.

If new shell scripts or commands for users are needed, they can be placed in a bin directory such as */usr/local/bin* rather than altering system directories. If the directory is not already in the PATH varable it should be altered thus:

```
PATH=$PATH:/usr/local/bin
```

GUI Tools

System > Administration > Users and Groups

Restricting root Access

PAM (Pluggable Authentication Modules) is a framework that assists applications in performing authentication-related activities. The core pieces of PAM are a library (libpam), a collection of PAM modules in a dynamically linked library (.so) and a set of configuration files:

```
ls /etc/pam.d
```

To tell if a particular program uses PAM see if the program is linked against the PAM library. Do this with a command such as:

```
ldd /bin/login | grep libpam
```

To substantially strengthen a system for users of root, make sure that the user must first log in as a member of the wheel group and then use su to become root. Here knowledge of the root password, by itself, is not sufficient to gain root access.

PAM may be used to control this behaviour as follows.

Disable Direct Root Login

A system is more secure if users are prevented from logging in directly as root so that they must first login as a normal user then use su to achieve root status.

Virtual Terminals
To prevent root logins on virtual terminals, edit the file */etc/pam.d/login* and add as the first line the entry:

```
auth    required    pam_securetty.so
```

This module will prevent root login on **terminal** devices that are not listed in the file */etc/securetty*. Empty this file as follows:

```
echo "" > /etc/securetty
```

Note that having an empty /etc/securetty file is quite different from not having one at all. The former disables root logins on all terminals, the latter enables root logins on all terminals.

Desktop
To prevent root logins on the desktop, add the same line at the top of the file /etc/pam.d/gdm.

```
auth    required    pam_securetty.so
```

Try logging in as *root* on the graphical desktop. Verify that a non-root user can still login.

Allow only members of the wheel group to use su

A system may be made more secure if only users of the wheel group can use the command *su*. To achieve this edit the file */etc/pam.d/su* and add or uncomment these lines:

```
auth    sufficient    pam_rootok.so
auth    required      pam_wheel.so use_uid
auth     include      system-auth
```

Log in as a user added to the wheel group and try to use su. Now try to use su whilst logged in as a user who is not a member of the wheel group.

Prevent all users from using su

Edit the file */etc/pam.d/su* so that the only line relating to the auth stack reads:

```
auth     requisite    pam_deny.so
```

Log in as a normal user and verify that any attempt to become root now fails.

File System Backup

Objectives

At the end of this section the delegate will be able to:

- describe reasons for taking backups
- use the backup utilities such as tar and dd
- describe how to backup and restore file systems
- use the crontab to schedule jobs

Why Do We Backup?

Backing up files means making copies of them, usually on removable media, as a safeguard in case the original is lost or damaged. Backup are convenient for restoring accidentally deleted files but they are essential in case of serious hardware failures or other disasters.

Backing up files is one of the most crucial system administration functions. To do a full backup on a file system, make sure all users are logged out. Then bring the system down to single-user mode before running the backup.

The administrator must plan and carry out a procedure for regularly scheduled backups of file systems for three major reasons.

- To ensure file-system integrity against a possible system crash
- To protect user files against accidental deletion
- To act as an important safeguard before reinstalling or upgrading a system

Backups are the only practical way of restoring corrupted or deleted files on a Linux system. Unlike Windows, there is no recycle bin!

What is a Backup?

A backup, or archive, is simply a copy of a file, files or entire file system stored on another medium. Backups can be made to any type of storage media including magnetic tapes, DAT tapes and disc files.

Types of Backup

There are three types of backup in common use:

- FULL The entire system is backed up, typically performed after major system changes.
- PARTIAL All files on part of the disk are backed up, i.e. anything less than a full backup.
- INCREMENTAL Only files that have been changed since the last backup.

These types of backups are often used in combination. A typical scenario would be:

- Full backup once a month
- Partial backup weekly
- Incremental daily

However this will depend on a variety of factors, such as:

- The type of backup media available
- The size of file systems to be backed up
- The availability and usage level of the system

It is recommended, that, if possible, a full backup is taken every day.

Logging Backups

It is very useful to log all backups, particularly if an incremental system is in use. This should include when and how the backup was taken and the format used. This may save confusion if a file needs to be restored.

At least two sets of media should be used and rotated, backups being logged in a book. It is also very important to store backups in a safe, clean environment, preferably a fire-proof safe, off site.

Backing Up Files - Review

The command *tar*, tape archive, was covered earlier. Here, we review the command. First, as user *root* create a shared directory to be used to store the backups:

```
mkdir   /backup
chmod  777  /backup
chmod  +t  /backup
```

In order to have some test files to backup, as the user *train1* copy the following files to the user's home directory:

```
cd; cp /etc/[a-c]*  .
```

Then as user *train1*, create the tar archive using:

```
tar  cvf  /backup/train1.tar  .
```

To view the tar archive:

```
tar  tvf  /backup/train1.tar  |  more
```

To restore a file use:

```
tar  xvf  /backup/train1.tar  ./oldfile
```

If the backup is to be run on a regular basis and create several new backup files, the date may be appended to the filename. This example also zips the output file:

```
tar  cvzf  /backup/train1_$(date +%m%d%H%M).tar.gz  .
```

Remember that the *cron* daemon may be used by users to schedule a task such as this to run at a specified time.

File Systems

The servers' discs are formatted into partitons or file systems and each file system is mounted on a directory. To view the current disk layout use:

```
df  -h
```

Backing Up File Systems - dump

The command *dump* can back up complete or partial file systems to local or remote tape drives The tape device can be on any system in the network to which the user has access. This command works quickly because it is aware of the structure of the UFS file system type and it works directly through the raw device interface.

It can also back up incremental file system changes, only those files that were changed since a previous backup. The tape drive is known to as */dev/st0*. Although it is usual to backup to tape, backups can also be made to disc files.

The command may take the following arguments:

- 0 full dump
- 1-9 incremental, since the last lower increment
- u add entry to /etc/dumpdates
- f output device or file

For example, the following takes a full dump of the file system on which */home* is mounted:

```
dump  0uf  /backup/home.bak  /home
```

Now view the dumpdates file as follows:

```
cat  /etc/dumpdates
```

and the dump file:

```
file  /backup/home.bak

ls  -lh  /backup/home.bak
```


Live Systems

When running a file system backup in a live situation, the server should be in **single user** mode and the file system should be **un-mounted** (covered in a later section). The administrator should **automate** backups of file systems by using the crontab utility and a backup script that runs the dump command.

Restoring File Systems - restore

The command *restore* can restore individual or complete file systems from a local or remote tape drive or disk file.

The command may take the following arguments:

- r recursive, restore all files
- i interactive restore
- v verbose
- t table of contents
- f output device or file

For example, to view the contents of a dump file:

```
restore  tvf  /backup/home.bak
```

Full Restore

In the event that the a file system is corrupted or has to be re-sized, the following example restores the complete file system on which */home* was mounted:

```
cd  /home

restore  rvf  /backup/home.bak
```

Interactive Restore

This is the easiest way to restore individual files and directories. The following example restores files to the directory /rest.

```
mkdir  /rest ; cd  /rest
```

To initiate an interactive restore enter:

```
restore  ivf  /backup/home.bak
```

Then at the prompt, enter a ? for help:

```
restore > ?
```

Change directories:

```
restore > cd  train1
```

List the files available:

```
restore > ls
```

Add files to the list to be extracted:

```
restore > add  aliases
restore > add  anacrontab
restore > ls
```

Note that the files to be extracted are marked with an *. To extract the files:

```
restore > extract
Specify next volume #: 1
set owner/mode for '.'?  n
restore > quit
```

Note that if *set owner/mode* is set to y, this will alter the permissions on the /rest directory. Now view the restored files:

```
ls  -l
```

Review Exercise

1. Use the *dump* command to take a full backup of the file system on which */home* is mounted. If the command *dump* cannot be found, use *yum* to install it.

2. Create and change to the directory */rest* and use the *restore* command to retrieve several files from the dump.

crontab - Scheduled Jobs

The *cron* daemon may be used by the root user to schedule a task to run at a specified time. To find out if it's running do the following:

```
service  crond  status
```

Scheduled commands for users are stored in the directory */var/spool/cron*. The main config file for root is /etc/anacrontab. The following is an example of the file:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=15
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-8

#period/days delay/min    job-identifier      command
1            5            cron.daily          nice run-parts /etc/cron.daily
7            25           cron.weekly         nice run-parts /etc/cron.weekly
@monthly     45           cron.monthly        nice run-parts /etc/cron.monthly
```

Each cron entry consists of 4 fields and if a script is placed into one of the directories in /etc it will then be run either daily, weekly or monthly.

The cron daemon is often used to schedule regular system backups which may be run at night when the system is lightly loaded. The output from cron is mailed to the owner of the process or to the person specified in the MAILTO variable.

The START_HOURS_RANGE variable sets the time frame when the job should be started. In this example:

```
cron.daily will run once a day          at 3:05am
cron.weekly will run once a week        at 3:25am
cron.monthly will run once a month      at 3:45am
```

If the RANDOM_DELAY environment variable is set, then a random value between 0 and RANDOM_DELAY minutes will be added to the start up delay of the job.

For example a RANDOM_DELAY set to 45 would therefore add, randomly, between 0 and 45 minutes to the user defined delay. So for the cron.daily entry the delay will be 5 minutes + a RANDOM_DELAY of up to 45 minutes.

Note that in Red Hat 5 the config file for root is /etc/crontab. This is because Red Hat 6 includes the *cronie* package as a replacement for *vixie-cron* in Red Hat 5.

Controlling Access to cron

The cron has a built in feature allowing the root user to specify who may and who may not use it. It does this by use of the two files */etc/cron.allow* and */etc/cron.deny*. The implications are:

- *cron.allow* file exists - the user **must** be listed therein in order to use the command.
- *cron.allow* file does not exist, *cron.deny* file exists - the user **must not** be listed in the *cron.deny* file in order to use the command.
- *cron.allow* file does not exist, *cron.deny* file exists but is empty - allows access to all users.
- If neither file exists, only the super user will be allowed to use this command.

Review Exercise

1. Check to see if the files *cron.allow* and *cron.deny* exist.

```
ls  -d  /etc/cron*
```

2. Now check to see if user *train1* can use the crontab?

```
crontab  -l
```

3. Create the file *cron.allow* and check the use of the crontab for user *train1* again. Can they use it now?

4. Remove the file *cron.deny* and check the use of the crontab for user *train1* again. Can they use it now?

```
touch  /etc/cron.allow
```

5. Add the user *train1* to the file *cron.allow* and check again. Can the user *train2* use the crontab?

```
echo train1  >  /etc/cron.allow
```

6. Delete the file *cron.allow* and create the file *cron.deny*. Now all users may use the crontab again.

```
rm  /etc/cron.allow
touch /etc/cron.deny
```

Backing Up Files - dd

The Disk to Disk copy program, *dd*, is a byte-for-byte copier which means that the target file is an exact copy of the source file. Although, it is not a true archiving program it is very useful if a swap file is to be created.

This command will only copy one file at a time, however that file could be a partition device. It is commonly used to copy from one device to another, tape to tape or disk to tape.

The simple format of the command is:

dd *options* *input filename* *output filename*

For example:

dd if=/etc/passwd of=password

would copy the file */etc/passwd* to a file called *password* in the current working directory.

Changing the Block Size

The *dd* program maintains two block sizes, the input block and the output block size. These have a default size of 512 bytes. The speed of the copy can be significantly improved by altering the block sizes as follows:

- ibs=NX Alters the input block size
- obs=NX Alters the output block size
- bs=NX Alters both blocks overriding any previous setting

where N is an integer and X can be B for multiplies of 512 or K for multiplies of 1024.

The Conversion Options

The *dd* program also has the ability to change the file format during transfer. This is particularly useful when moving files between different makes of machine. The following options are available:

- ucase Convert file to uppercase
- lcase Convert file to lowercase
- ascii Convert from ebcdic to ascii format
- ebcdic Convert from ascii to ebcdic format

For example, the following would convert the *password* file to upper case:

```
dd  if=/etc/passwd  of=password  conv=ucase
```


File System Management

Objectives

At the end of this section the delegate will be able to:

- mount and unmount file systems
- check file systems for consistency
- create and format new file systems
- increase the available swap space

File Systems

This section looks at creating, formatting and mounting file systems. Before it can be used any **disk** on the server must be formatted into **partitons** or file systems and each file system must be mounted on a directory. To view the current disk layout use:

```
df  -h
```

Disc Usage

The command *du* shows the disk usage of each file in each subdirectory of a file system.

For example, to obtain a list of the size of each directory under root use:

```
cd  /;du  -s  *
```

Disks
On a running Linux system, disks are represented by entries in the /dev directory. Hard disk descriptors in /dev begin with *hd* (IDE) or *sd* (SCSI); a SCSI tape would be *st.* Since a system can have more than one block device, an additional letter is added to the descriptor to indicate which device is in use.

A set of disks might be known as /dev/hda, /dev/hdb or /dev/sda, /dev/sdb etc.

Partitions
A disk such as /dev/sda can be formatted or partitioned into file systems for example /dev/sda1, /dev/sda2 etc. Each will have a size in MBs and be made up of sectors/tracks and cylinders.

Disks normally allow only 4 *primary* partitions, one of which can be *extended*. The extended partition can be further divided into *logical* partitions. There can be a maximum of 64 partitions on an IDE disk and 16 on a SCSI disk.

To list available disks and partitions use:

```
fdisk -l
```


Mounting File Systems

Before it can be used a file system must first be mounted on a directory. A list of the mounted file systems and the directories they are mounted on can be displayed as follows:

```
mount
```

proc
This special directory holds all the details about the Linux system including its kernel, processes and configuration parameters. Under Linux, everything is managed as a file; even devices are accessed as files in the /dev directory. Although files are usually either text or binary the /proc directory contains a special type: virtual files. These files are listed, but don't actually exist on disk; the operating system creates them on the fly.

Most virtual files always have a current timestamp, which indicates that they are constantly being kept up to date. The /proc directory itself is created every time the server is booted. Although almost all the files are read-only, a few writable ones notably in /proc/sys allow the changing of kernel parameters.

The /proc directory is organised into virtual directories and subdirectories and it groups files by similar topic:

```
ls  /proc
```

The numbered directories correspond to each running process; a special self symlink points to the current process. Some virtual files provide hardware information, such as /proc/cpuinfo, /proc/meminfo and /proc/interrupts. Others give file-related info, such as /proc/filesystems or /proc/partitions. The files under /proc/sys are related to kernel configuration parameters.

sysfs
Sysfs is a virtual filesystem exported by the kernel, similar to /proc. The files in sysfs contain information about devices and drivers. Some files in Sysfs are even writable, for configuration and control of devices attached to the system. Sysfs is always mounted on /sys.

devpts
Pseudoterminal slave devices are created in the /dev/pts directory.

tmpfs
Reading from RAM is a lot of faster than reading from a hard drive and it reduces disk I/O. The file system tmpfs can be used to store files in memory which is ideal for file caches and other temporary data. Access is fast and the data is lost on power down or reboot. It is normally mounted at /dev/shm.

sunrpc
An implementation of Sun's remote procedure call protocol.

Mounted Systems

A mounted file system is attached to the system directory tree at the specified mount point and becomes available to the system. The root file system is always mounted. Any other file system can be connected or disconnected from the root file system.

The system tracks the mounted file systems in the mount table (the file /etc/mnttab on Red Hat 5). A list of the mounted file systems may be displayed as:

```
mount
```

Disk Slices

The file system on which a directory is mounted is known as a *disk slice*. The following file contains details of all disk slices that are automatically mounted at startup:

```
cat  /etc/fstab
```

umount

The umount command does not unmount a file system that is busy. A file system is considered busy if a user is in a directory in the file system or if a program has a file open in that file system.

The following example unmounts a file system:

```
umount  /dev/sda1
```

or

```
umount  /boot
```

To force an unmount if the file system is busy use:

```
umount  -f  /boot
```

To unmount all file systems use:

```
umount –a
```


mount

The mount command does not mount a read/write file system that has inconsistencies. If an error message is received from the mount command, the file system will need to be checked.

This example mounts the file system:

```
mount  /dev/sda1  /boot
```

or

```
mount  /boot
```

```
df  -h
```

To mount all file systems use:

```
mount  -a
```

Automounter

The command *autofs* controls the operation of the automount daemons running on the Linux system. Usually autofs is invoked at system boot time with the start parameter and at shutdown time with the stop parameter.

Checking File Systems

A system program is responsible for checking the file systems during startup. To do this, it invokes another program called e2*fsck*.

The command e2*fsck* must be run on an unmounted file system as follows:

```
e2fsck  -f  /dev/sda1
```

The command can be used to check each of the file systems for any errors.

When checking a file system, *fsck* may encounter errors and may ask various questions during this process. As far as this course is concerned, the answer should always be **yes** to these questions.

Preparing Disks

If using an Oracle Virtual Box VM, at this point add **three new discs** to the image as follows:

- Power off the VM and open Virtual Box.
- Choose Storage, click on the Controller, click on the Plus sign and add a new SCSI controller.
- Click on the Plus sign and add a new Virtual Disk1.
- Click on the Plus sign and add Virtual Disk2 and Virtual Disk3.
- Click OK and restart the image.

Creating Disk Slices

The interactive menu command *fdisk* may be used to create new disk slices. In a real world situation this command should only be used in single user mode. With the help of your tutor, work through the following example which will create a new disk slice.

To list available disks use:

```
fdisk -l
```

To invoke the command for a particular disk use:

```
fdisk /dev/sdb
```

To list the commands available enter:

```
m
```

To display the current partition table:

```
p
```

Create Partition 1
To create partition 1 enter:

```
n

p  (for primary)

1
```

Enter the start cylinder as 1 and the size as +1G.

To display the new partition table:

```
p
```


Create Partition 2
To create partition 2 enter:

n

p (for primary)

2

Accept the default for the start cylinder (return) and enter a size of +2G.

To display the new partition table:

p

Again list available commands:

m

Now write the new partition table to disc:

w

List available disks:

ls /dev/sd*

For Red Hat 6 it may be necessary to reboot before the changes take effect.

Creating a File System

The disk is now labelled but the new slice must have a file system built onto it before it can be used. It will not yet appear here:

```
df  -h
```

The command *mkfs.ext4* (ext3 for Red Hat version 5) will create a file system on each new slice:

```
mkfs.ext4  /dev/sdb1

mkfs.ext4  /dev/sdb2
```

As an alternative the following may be used:

```
mkfs  -t  ext4  /dev/sdb1
```

Mounting the File System

The disk slice now has a file system built onto it but before it can be used the *mount* command must set up a connection from the new slice to an actual directory.

For example, to create two new directories and mount the new slices:

```
mkdir  -v /myusers  /myaccts

mount  /dev/sdb1 /myusers

mount  /dev/sdb2 /myaccts

df  -h
```


The /etc/fstab File

The file */etc/fstab* contains details of all disk slices which are to be mounted automatically on boot up. To ensure that a new disk slice is mounted when the server is next rebooted, an entry must be added to this file.

First create a backup of the file:

```
cp  /etc/fstab{,.bak}
```

Now add entries of the following form:

```
/dev/sdb1    /myusers    ext4        defaults      1  2
/dev/sdb2    /myaccts    ext4        defaults      1  2
```

Test the new file as follows:

```
umount -a

df

mount  -a
```

Adding Swap Space

The available swap space can be increased in one of three ways:

- Increase the current swap slice size by re-slicing the disk
- Use an extra hard slice of the disk
- Use a special file as outlined below

First show the current swap space available:

```
swapon  -s
```

Swap File
To add extra swap space using a file, first create the file in the appropriate place and of the desired size using the command dd as follows:

```
dd if=/dev/zero of=/var/swap bs=1024 count=1024000
```

This creates a new 1Gb swap file:

```
ls -l /var/swap
```

Use the mkswap command to initialise the file as swap area:

```
mkswap  /var/swap
```

Now use swapon to bring it into use

```
swapon  /var/swap

swapon -s
```

To remove the file from use:

```
swapoff  /var/swap
```

Add an entry of the following form in /etc/fstab if the swap area is to be brought into use at each boot:

```
/var/swap               swap            swap defaults         0  0
```

Now reboot to test all of the changes made.

Swap Partition
To use a new swap partition (/dev/sda6), first create the partition using fdisk then run the following:

```
mkswap  /dev/sda6

swapon  /dev/sda6

swapoff  /dev/sda3

swapon -s
```

Add an entry to the /etc/fstab file and reboot to test the changes made.

Review Exercise

This exercise moves the /home directory to a new partition.

1. Login as root, un-mount the directory */home* and make a *backup* of the directory */home*.

2. Use the *mkfs* command to format the partition on which */home* was mounted. This will delete all of the data and make it unusable!

3. Use the *fdisk* command to create a new partition on /dev/sdb.

4. Create a file system on the new disk partition.

5. Re-mount the directory */home* on the new file system.

6. Now *restore* all files for the */home* diecrtory.

7. To test the restore login as the user *train1*.

8. Amend the file */etc/fstab* to include the new partition and then reboot.

Logical Volume Management

Objectives

At the end of this section the delegate will be able to:

- manage and use logical volumes

Logical Volume Manager (LVM)

With LVM, the user can create logical partitions that can span across one or more physical hard drives. First, the hard drives are divided into physical volumes, then those physical volumes are combined together to create the volume group and finally the logical volumes are created from volume group.

To create a logical volume, run through the following steps:

- select the physical storage devices and create suitable disc slices
- create the Physical Volumes (PV)
- create the Volume Group (VG) from the Physical Volumes
- create Logical Volumes (LV) from Volume Group

Create Device

First choose the physical volumes that will be used to create the LVM and create suitable disc slices for use:

```
fdisk /dev/sdc
```

Create two 1GB slices, sdc1 and sdc2.

```
fdisk /dev/sdd
```

Create two slices sdd1 and sdd2, using the same options. List the new partition table:

```
fdisk -l
```

Physical Volumes (PV)

Create the physical volumes using the *pvcreate* command as shown below.

```
pvcreate /dev/sdc1 /dev/sdc2 /dev/sdd1 /dev/sdd2
```

When the physical volumes are created, they can be viewed using the *pvscan* command:

```
pvs
```

The list of physical volumes with attributes like size, physical extent size, total physical extent size and the free space etc. can be viewed using *pvscan* or *pvdisplay*:

```
pvscan

pvdisplay
```


Volume Groups (VG)

Volume groups are nothing but a pool of storage that consist of one or more physical volumes. Once the physical volumes are created, the user can create the volume group (VG) from these physical volumes (PV).

In this example, the command *vgcreate* is used to create the volume group *vg* from the two physical volumes:

```
vgcreate vg /dev/sdc1 /dev/sdd1
```

The commands *vgs, vgscan* and *vgdisplay* can be used to list the created volume groups:

```
vgs

vgscan

vgdisplay
```

Logical Volumes (LV)

Now the logical volumes, with the names *share* and *mine*, can be created from the volume group using the *lvcreate* command:

```
lvcreate --name mine  --size 300M vg

lvcreate --name share --size 500M vg
```

Use the *lvs*, *lvscan* and *lvdisplay* commands to view the available logical volumes with their attributes:

```
lvs

lvscan

lvdisplay
```


Make File System & Mount

Before the LV can be used its' filesystem must be created and it must be mounted on a directory. For example, for the LV *mine*:

```
mkfs.ext4 /dev/vg/mine

mkdir /mine

mount /dev/vg/mine /mine
```

And for *share*:

```
mkfs.ext4 /dev/vg/share

mkdir /share

mount /dev/vg/share /share
```

List them with:

```
df –h
```

Automount

To ensure that the new disk slice is mounted when the server is next rebooted, an entry of the following type must be added to the */etc/fstab* file:

```
/dev/mapper/vg-mine     /mine     ext4     defaults     1  2
/dev/mapper/vg-share    /share    ext4     defaults     1  2
```

Test the new file as follows:

```
umount -a

mount  -a
```


Extend the LV

Extend the size of the logical volume after creating it by using the command *lvextend*:

```
lvextend -L +100M  /dev/vg/mine
```

This command does not actually increase the physical size of volume; to do that issue the command:

```
resize2fs  /dev/vg/mine;df -h
```

Reduce the LV

Reduce the size of the logical volume as follows:

```
umount  /mine

e2fsck  -f  /dev/vg/mine

resize2fs  /dev/vg/mine  200M

lvreduce  -L  -200M  /dev/vg/mine

mount /dev/vg/mine /mine
```

Remove the PV

To remove the physical volume, first unmount it and then use the command *pvremove*:

```
umount /mine

umount /share

vgremove  vg

pvremove  -ff  /dev/sdc1 /dev/sdc2 /dev/sdd1 /dev/sdd2

pvscan;vgs;lvs
```


Recovery Workshops

Workshop I

This exercise recovers from an error when a user cannot login via the GUI:

Note that when using the *restore* command if *set owner/mode* is set to y, this will alter the permissions on the directory to which the files are being restored.

1. To simulate the error first login as root and restore a file to /tmp; ensure that *set owner/mode* is set to y:

   ```
   cd  /tmp ; ls  -ld  .

   restore  ivf  /backup/home.bak

   ls  -ld  .
   ```

2. Logout and try and login again as the user train1. This error will appear:

   'GDM could not write to your authorization file.'

   This is because the user can no longer write to the /tmp directory!

3. Login again as root and correct permissions on the /tmp directory:

   ```
   ls  -ld  /tmp

   chmod  777  /tmp

   chmod  +t  /tmp ; ls  -ld  /tmp
   ```

4. Login again as user train1.

Workshop II
This exercise recovers the server from a seemingly fatal error which prevents a re-boot.

1. Login as root and make a copy of the file */etc/fstab*:

```
cp  /etc/fstab{,bak}
```

2. Edit the file */etc/*fstab, add a new entry of the following form for a partition that does not exist and save the file:

```
/dev/sde1        /extra        ext4        defaults        1  2
```

3. Re-boot the server. The re-boot will **fail** with an error and prompt for the root password. Enter the password to enter single user mode.

4. Attempt to fix the error by editing the file */etc/fstab.* This will fail as the root file system is mounted read only.

5. Remount the root file system read/write as follows:

```
mount  -o  remount,rw  /
```

6. Correct the error in the file /etc/fstab by editing the file:

```
vi  /etc/fstab
```

7. Re-boot the server and it should boot as normal.

An alternative method involves booting from the DVD:

1. Boot the server from the DVD and choose 'Rescue installed system' mode. Answer the few simple questions required.

   This will mount the root file system at */mnt/sysimage* in read/write mode and present a shell prompt.

2. Correct the error in the file /etc/fstab by editing the file:

```
vi /mnt/sysimage/etc/fstab
```

3. Re-boot the server and it should boot as normal.

Network Management

Objectives

At the end of this section the delegate will be able to:

- describe and find out an IP Address
- describe how routing works
- use network troubleshouting tools
- start and stop networking
- configure and use the Network File System (NFS)

Determining the IP Address

Most modern PCs come with an Ethernet port. When Linux is installed, this device is called eth0. The interface configuration command *ifconfig* can be used to view or change the configuration of an IP interface on the system. This command is used to initialise IP interfaces at startup time and example commands can be found in the TCP startup script for the system, in /etc/rc2.

Once initialised the kernel module aliased to the network interface (eth0) in /etc/modules.conf is loaded and assigned an IP address and a netmask value.

To list all interfaces use:

```
ifconfig
```

On very rare occasions the NIC card may not work because it shares both an interrupt and memory access address with some other device. Look at the contents of the */proc/interrupts* file to get a listing of all the interrupt IRQs used by the system and check that they only have a single entry. If there are conflicts, refer to the manual for the offending device to try to determine ways to either use another interrupt or memory I/O location.

Changing the IP Address

To reset the IP Address and netmask:

```
ifconfig  eth0  inet  10.0.2.15  netmask 255.255.255.0
```

Linux makes life easy with interface configuration files located in the /etc/sysconfig/network-scripts directory. Interface eth0 has a file called ifcfg-eth0, eth1 uses ifcfg-eth1, and so on.

```
cat  /etc/sysconfig/network-scripts/ifcfg-eth0
```

Place the IP address information in these files to auto-configure the NICs when Linux boots. The card will be activated on booting if the parameter ONBOOT has the value ‘yes’.

Two other commands are *ifup* and *ifdown*. The *ifup* utility reads the system’s configuration files in /etc/sysconfig and assigns the stored values for a given interface. To activate the new IP address use:

```
ifdown eth0

ifup eth0
```

At boot time the ethernet card is initialised with the /etc/rc.d/init.d/network script. All the relevant networking files are sourced in the /etc/sysconfig directory. In addition the script also reads the sysctl options in /etc/sysctl.conf; this is where one can configure the system as a router.

/etc/hosts File

The name and IP Addresses of network servers are held in a special file which can be displayed as follows:

```
cat  /etc/hosts
```

This will display lines of the form:

```
127.0.0.1       localhost.localdomain   localhost
10.0.2.15       linux.train             linux
10.0.2.17       solaris.train           solaris
```

The line 127.0.0.1 is the localhost or loop back address.

Here, the current servers are called *linux* and *solaris*. The user may use the command *hostname* to confirm the name of the host server:

```
hostname
```

ping - Contacting a Server

The command *ping* may be issued by a user to confirm that another server is on the network. For example, to send a message to a server use:

```
ping  10.0.2.15
```

If the server is alive and well, the command will return a message otherwise it will hang.

Rather than using the IP Address of the server, the name of the server from the /etc/hosts file may be used instead. For example:

```
ping  linux
```

```
ping  -c4  $(hostname)
```

This second example will send a message to the server and stop after 4 iterations.

Network Configuration

The network interface card (NIC) must be supported by the kernel. Determine which card is being used with one of the following commands:

```
dmesg | grep NIC

cat /proc/interrupts

/sbin/lsmod
```

From the output can be determined the ethernet card's chipset, the i/o address and the IRQ.

Host Information

The file /etc/resolv.conf contains a list of DNS servers:

```
cat  /etc/resolv.conf
```

The following command gives the name of the host:

```
hostname
```

The file /etc/sysconfig/network can define if networking must be started and can also contain the HOSTNAME variable:

```
cat  /etc/sysconfig/network
```

The configuration parameters for a card, for example eth0, are contained in the following file:

```
cat  /etc/sysconfig/network-scripts/ifcfg-eth0
```


File Transfer Between Servers

File transfer involves gaining access to the files on a **target** server and then transferring a file or files from or to the **host** server. The files will be transferred to and from the current working directory on the host. Three steps are involved here:

- Find the IP address of the target server
- Contact the target server to confirm it is available
- Transfer files between the two servers

sftp
With *sftp*, the user name and target server are supplied as follows:

```
sftp  traina@192.168.1.150
```

If this command hangs, make sure the */etc/hosts* file has been updated. A list of commands available on the target server may be displayed with the command:

```
help
```

The commands available include *cd* and *ls*.

To run a command on the host server, use the following syntax:

```
!ls
```

Exercise

1. With the help of your tutor, use *sftp* to login to the given target server as trainN.

put
The command *put* may be used to **move** files to the target machine from the current working directory on the host server.

To move a single file or multiple files use:

```
put  text1
```

or

```
put  p*
```


get
The command *get* may be used to **fetch** files from the target machine into the current working directory on the host server.

For example, to fetch a single file or multiple files use:

```
get  text1
```

or

```
get  p*
```

Exercise

- Transfer several files from the host to the target server.

ssh
The command *ssh* allows a user to logon to another server, much like Putty on Windows:

```
ssh  traina@192.168.1.150
```

Disable Root Login
To disable direct login as root via ssh, edit the file /etc/ssh/sshd_config and add the line:

```
PermitRootLogin no
```

Then restart the service:

```
service sshd restart
```

Associated commnds include *ftp* and *scp*. Most Windows operating systems also offer a version of ftp which can be run from within a command window.

An alternative to the command line is one of the many GUI based alternatives such as Filezilla, SSH Secure Shell and Tunnelier. These allow files to be transferred between servers using simple drag and drop commands.

Common Network Tools

Anyone who is the least bit Internet savvy will be aware that to move data from one point say A to another point B across the Internet, it has to pass through a number of intermediary points say C, D, E etc. What many won't know is that the data is not transferred in one piece when it is sent over the net, Rather, it is split into chunks of about 1500 bytes each, then each chunk is enclosed in what is known as a packet. These contain additional data such as the destination IP address and the port number plus other details which provide the unique identity to the packet.

While the packets travel the path from point A to point B, each packet may take a different path depending upon diverse factors and eventually they are merged together in the same order at the receiving end to provide the document sent in the first place.

The intermediate gateways through which the packets pass before they reach the final destination are known as hops. So for data to travel from point A to point B on the net it has to go through a number of hops.

Linux, being a network operating system, has a number of powerful tools which help the network administrator find out a wealth of data about the network and the Internet. The full list of options to any command may be found in the *man* pages.

route
This command is used to show or manipulate the IP routing table. For example to list the kernel routing table:

```
route -n
```

tcpdump
This is a command used to analyse network traffic by capturing network packets. The following commands illustrate some options:

Let tcpdump autodetect network interface:

```
tcpdump
```

or specify a network interface to capture packets from:

```
tcpdump -i eth0
```


netstat
This gives information on current network connections, the routing table or interface statistics depending on the options used:

```
netstat | less
```

arp (Address Resolution Protocol)
This tool resolves IP addresses to actual MAC addresses. It may be useful to view or alter the contents of the kernel's ARP tables, for example when a suspected duplicate Internet address is the cause of some intermittent network problem. The arp tool was made for situations like this:

```
arp  -a
```

traceroute
This tool is used to find out the potential bottlenecks in between the host computer and a remote computer across the net. For example, to run a trace on a domain:

```
traceroute www.google.co.uk
```

Network File System (NFS)

The Network File System is one of the most widely used network services. NFS is based on a remote procedure call which allows a client to automatically mount remote file systems from a server and transparently provide access to them as if the file system was local.

Scenario
Here we export a directory from the NFS server (10.0.2.15) and mount it on an NFS client (10.0.2.19).

Server & Client Actions

Start NFS Service
First configure NFS on both the server and the client. Start the NFS daemon if it is not already running:

```
service nfs status

service nfs start
```

To check whether the system supports NFS run the following:

```
cat /proc/filesystems | grep nfs
```

If there is no output it means that NFS is not supported or the NFS module has not been loaded into the kernel. To load NFS module execute:

```
modprobe nfs
```

Execute rpcinfo -p to check the correctness of the NFS installation and to confirm that the NFS server is running and accepting calls on a port 2049:

```
rpcinfo -p | grep nfs
```

When installed correctly the NFS daemon should be listening on both UDP and TCP 2049 port and portmap should be waiting for instructions on a port 111.

At this point we should have portmap listening on both the NFS server and NFS client:

```
rpcinfo -p | grep portmap
```


Server Actions

Edit Export File

The directories that are to be shared over the network using NFS must be defined in the file */etc/exports*. Common export options include read write (rw), read only (ro) and synchronized mode (sync).

To export files from the server first create a new directory:

```
mkdir  /home/nfs
```

then copy some files to that new directory:

```
cd  /home

cp  train1/*  nfs

ls  nfs
```

Example Entries

To export the */home/nfs* directory for access from a client with an IP address of 10.0.2.19 with read and write permissions in synchronized mode:

```
/home/nfs/  10.0.2.19 (rw,sync)
```

Note that if correctly configured server names may be used in place of IP addresses:

```
/home/nfs/  linux.train (rw,sync)
```

To export the */home/nfs* directory for access with read only permissions:

```
/home/nfs/  10.0.2.19 (ro)
```

To export the */home/nfs* directory for access from any client with read only permissions in synchronized mode:

```
/home/nfs/  * (ro,sync)
```

Edit /etc/exports File

As an example, add the following entry to the file */etc/exports*:

```
/home/nfs/  *(ro,sync)
```

Export the File System

Once the /etc/exports file has been edited run the command *exportfs*:

```
exportfs -rva
```


Client Actions

Stop the Firewall
Some RedHat systems by default block all incoming traffic to a NFS server via a firewall using iptables rules. Either turn off the firewall or add iptables rules to allow traffic on portmap port 111, NFS port 2049 and random ports for other NFS services. To turn off the firewall and clean up all iptables rules enter:

```
service iptables stop
```

Mount Remote Files
Create a new mount point:

```
mkdir  /home/nfs_local
```

Run the mount command to mount the exported NFS remote file system from the server with an IP address of 10.0.2.15:

```
mount 10.0.2.15:/home/nfs /home/nfs_local
```

If required a filesystem type may be specified:

```
mount -t nfs 10.0.2.15:/home/nfs /home/nfs_local
```

List the files in the newly mounted directory:

```
cd /home/nfs_local/

ls
```

An attempt to create a new file fails as the file system is mounted read only.

```
touch test_file
```

Client - Configure Automount

To make this completely transparent to end users, automount the NFS file system every time the client system boots.

To mount the file system automatically, on the client add the following line to the file */etc/fstab*:

```
10.0.2.15:/home/nfs  /home/nfs_local/ nfs  defaults  0  0
```

