Study Guide for

Linux System Administration 1

Lab work for LPI 101

version 0.2

released under the GFDL by LinuxIT

Copyright (c) 2005 LinuxIT.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with the Invariant Sections being History, Acknowledgements, with the
Front-Cover Texts being "released under the GFDL by LinuxIT".

see full license agreement on p.164

Introduction:

Acknowledgments

The original material was made available by LinuxIT's technical training centre www.linuxit.com. Many thanks to Andrew Meredith for suggesting the idea in the first place. A special thanks to all the students who have helped dilute the technical aspects of Linux administration through their many questions, this has led to the inclusion of more illustrations attempting to introduce concepts in a user friendly way. Finally, many thanks to Paul McEnery for the technical advice and for starting off some of the most difficult chapters such as the ones covering the X server (101), modems (102) and the Linux kernel (102).

The manual is available online at http://savannah.nongnu.org/projects/lpi-manuals/. Thank you to the Savannah Volunteers for assessing the project and providing us with the Web space.

History

First release (version 0.0) October 2003. Reviewed by Adrian Thomasset.
Revised January 2004 after review by Andrew Meredith.
November 2004. Section on expansion cards added in 'Hardware Configuration' chapter by Adrian Thomasset
December 2004. Index and mapped objectives added by Adrian Thomasset.
January 2005. Glossary of terms, command and file review added at end of chapters by Adrian Thomasset
June 2005. Added new entries in line with recommendations from SerNet for the LATM process, by Andrew Meredith with additional text supplied by Andrew D Marshall and review by Adrian Thomasset. Section on Debian tools supplied by Duncan Thomson.
August 2005. "Linux System Administration 1 - Lab work LPI 101 - version 0.2" has been awarded the LATM status by SerNet.

Dramatis Personi

Adrian Thomasset	<adriant@linuxit.com>	http://www.linuxit.com/
Andrew Meredith	<andrew@anvil.org>	http://www.anvil.org/
Andrew D Marshall	<admarshall@gmail.com>	http://h0lug.sourceforge.net/
Duncan Thomson	<thom-ci0@paisley.ac.uk>	http://www.paisley.ac.uk/

Goals

This manuals primary aim is to provide explanations, examples and exercises for those preparing for the Linux Professional Institute (LPI) Certification Programme 1 (LPIC-1), Exam 101.

Three core sources of criteria guide this manual to its primary goals:
- The LPI's Exam-101 "Objectives".
- Its LPI-Approved Training Materials (LATM) criteria.
- The Linux Documentation Project (LDP or TLDP) Author Guide (AG).

The LPI's Exam-101 Objectives and LATM criteria are summarized below. The Objectives are also online at:

http://www.lpi.org/en/obj_101.html

The LDP Author Guide [http://www.tldp.org/LDP/LDP-Author-Guide/] provides a consistent, comprehensive set of guidelines for those wanting to publish HOWTOs, Tutorials and Manuals via the world's largest GNU/Linux documentation system, the LDP.

This manual adopts as its second prime objective, on equal footing with its first, the LDP Author Guide's challenge to prospective LDP authors, "to massage all of the raw data into a readable, entertaining and understandable whole." [LDP-AG, 4.1. Writing the Text]

Intended Training Schedules

The content herein is designed to accompany practical courses preparing for the LPI 101 exam of the LPIC-1 programme. While this material was generally structured to work with a course of 24-32 hours in consecutive 8-hour sessions, it is modularized to also work for shorter or longer sessions, consecutive or otherwise.

Intended Audience & Prerequisites

This manual's material assumes its users will already have:

- Extensive experience (several years) using Intel x86 computers, including a strong knowledge of hardware components and their interaction with basic operating system (OS) components.
- A general knowledge of computing and networking basics such as binary and hexadecimal maths, common units of measure (bytes, KB vs Kb, Mhz, etc), file-system structures, Ethernet and Internet networking operations and hardware, etc.
- More than three cumulative months of practical experience using a GNU/Linux, BSD or Unix OS, logged in and working at the command-line (in a text terminal or console) either locally or remotely.

Those with less experience, however, should not be discouraged from using this manual, if (and only if) they are willing to spend extra time catching up on the prerequisite background skills and knowledge; a challenging task, but not an impossible one.

Further references and examples are provided for the various uses of commands, as well as exercises and accompanying answers demonstrating exam-like problem-solving. All are optional with those most recommended either discussed or referenced in the manual's body.

The LPI Certification Program

There are currently two LPI certification levels. The first level LPIC-1 is granted after passing both exams LPI 101 and LPI 102. Similarly passing the LPI 201 and LPI 202 exams will grant the second level certification LPIC-2.

There are no certification pre-requisites for LPI 101 and 102. However the exams for LPIC-2 can only be attempted once LPIC-1 has been obtained.

Instructor Notice

There are no instructor notes with this manual. The following issues must be considered.
The exercises in the sections **Managing Devices** and **The Linux Filesystem** both assume that a new partition can be created. Make sure during the installation that a large extended partition with at least 100MB free space is available after all the partitions have been created.
The following RPM packages are needed for the exercises:
rpm-build
sharutils

No Guarantee

The manual comes with no guarantee at all.

Resources

www.lpi.org
www.linux-praxis.de
www.lpiforums.com
www.tldp.org

www.fsf.org
www.linuxit.com

Notations

Commands and filenames will appear in the text in **bold**.

The <> symbols are used to indicate a non optional argument.
The [] symbols are used to indicate an optional argument

Commands that can be typed directly in the shell are highlighted as below

```
command
```

or

```
command
```

Contents

Contents

Installation

Prerequisites

None

Goals

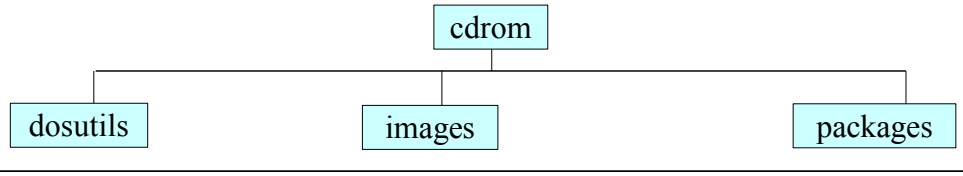
Understand the layout of a typical Linux installation CD
Perform different types of installations
Create a simple partition scheme (see also p.28)

Contents

1. The Installation CD

The various Linux distributions have different names for the directories on the installation CD. The generic structure of the CDROM is as follows:

Generic Installation CD layout



packages: This directory contains the pre-compiled packages. Here are the associated names for the main distrubutions:
debian: **dist**
mandrake: **Mandrake**
redhat: **RedHat**
suse: **suse**

Initially all the software installed on the system comes from these packaged files. See the section on package managers on p.108 for more details.

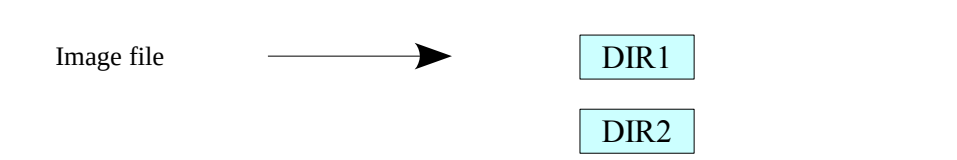
images: This directory contains various “images”. These are special flat files often containing directory structures. An initial ramdisk (initrd) is an example of an image file. There are different types of images necessary to:

- boot the installation process
- provide additional kernel modules
- rescue the system

Some of these files can be copied to a floppy disk when the installation is started using floppies rather than the CDROM. The Linux tool used to do this is **dd**. There is a tool called **rawrite** which does the same under DOS.

The image is a special file which may contain subdirectories (much like an archive file).

Image file structure



An image file can be mounted on a loop device. If the image file name is called *Image* then the following command will allow one to view the content of this file in the **/mnt/floppy** directory:

```
mount -o loop /path/to/Image /mnt/floppy
```

dosutils: this directory contains DOS tools which may be used to prepare a Linux installation such as the

rawrite.exe tool mentioned above. Another tool is the **fips** utility which non destructively partions a C:\ drive in two provided the underlying filesystem type is FAT and not NTFS.

2. Local Installations

The easiest and most common type of installation is a local installation. Most distributions are a CD iso image with an automatic installation script. On machines with no CD-ROM hardware it is still possible to start an installation from a floppy.

CD-ROM installation

Change the settings in the BIOS for the computer to boot from CD. The installation is menu driven and allows for advanced and basic configuration.

Floppy Installation

If for some reason you don't boot using the CD-ROM you will need to create a floppy installation image. This can happen if the CD is not bootable or you have downloaded a non-iso image of the distribution.

Making a bootable installation disk	
`dd if=/path/to/<image_name>  of=/dev/fd0`	on a linux box
`rawrite.exe`	under Windows (not NT)

For RedHat distributions the installation images are in the ***images*** directory. The basic image is ***boot.img***. Other images are more specialised like ***bootnet.img*** or ***pcmcia.img***.
In a Suse distribution the floppy image is in the ***disks*** directory and the image is called ***bootdisk***.

3. Network Installation

For a RedHat installation this is only a specialised floppy installation. Make a bootable floppy using the ***bootnet.img*** image:

```
dd /mnt/cdrom/images/bootnet.img of=/dev/fd0
```

The first part of the installation is text based and will allow you to set up the keyboard and the network parameters needed. The rest of the installation can be done via FTP, NFS or HTTP. Originally protocols that allowed a full mount (NFS) would also allow the install to be done in graphical mode, while file retrieval protocols (FTP HTTP) would only allow text mode. With most modern distributions this is no longer the case.

Also notice that most modern distributions offer network installations directly from the CD (e.g Mandrake disk 2 will start a network type installation or Fedora Core can take the parameter *askmethod* at boot time).

4. Rescue disk

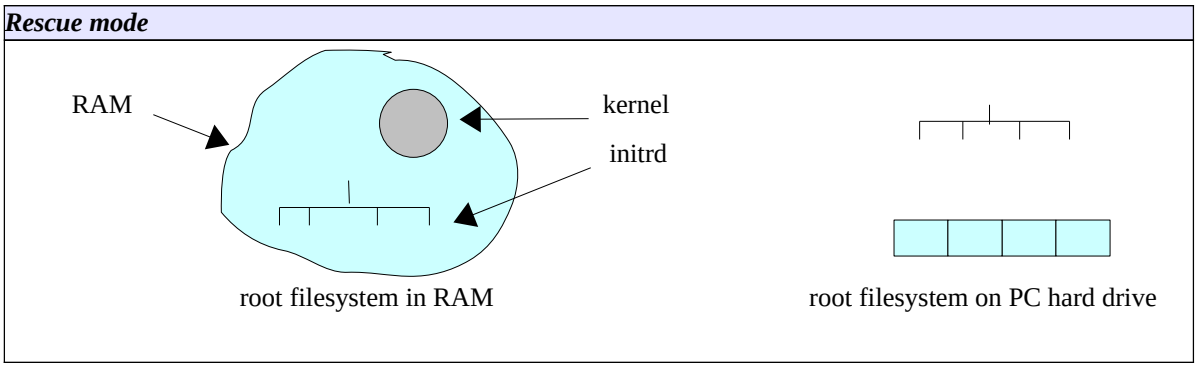
If a Linux system is corrupt it is possible to boot the computer using a rescue disk. This is a small version of Linux that will mount a minimal virtual filesystem into memory.

The Linux operating system runs entirely in RAM. The aim is to access the root filesystem on the PC hard drive. Most rescue disks can determine this automatically. Assuming the root filesystem was found on the first logical partition of the computer's first IDE disk (/dev/hda5), the rescue disk script can then mount this resource on a subdirectory of the filesystem in RAM, say /mnt/system.

Changing perspectives

In this situation we have **two root filesystems** as depicted below. To use the root filesystem on the hard drive as our top directory we need to change our perspective (change root). The **chroot** tool does just that:

```
chroot /mnt/sysimage
```



Getting started

Old Method:

1. Make a bootable floppy using the **boot.img** image file: dd if=boot.img of=/dev/fd0
2. Copy the **rescue.img** image file to a second floppy: dd if=rescue.img of=/dev/fd0
3. Boot the system using with the **boot.img** diskette
4. At the LILO prompt type "linux rescue". You should see something like

   Insert root file system disk:

5. Insert the **rescue.img** diskette and press enter
6. The boot process will continue until you get a shell prompt
7. You may still need to determine where the root filesystem is on the hard drive (not covered)

New Method:

1. Insert the Linux installation disk (Suse, RedHat, Mandrake ...)
2. At the prompt type “linux rescue”

3. Follow the instructions.
4. The instuction should say where the root filesystem is mounted
5. If the root filesystem is mounted on **/mnt/sysimage** then enter the following command

5. Partitioning Schemes

To access resources on a hard drive the operating system uses a mechanism called 'mounting'. For UNIX type operating systems this involves attaching a disk to any directory which is then called a **mount point**.

The figure below shows a possible partitioning scheme. Here many resources (not only local disks and partitions, but possibly network shares, CD-ROMs, etc) are attached on various mount points

To the user the file system layout is simply a tree of directories and subdirectories.

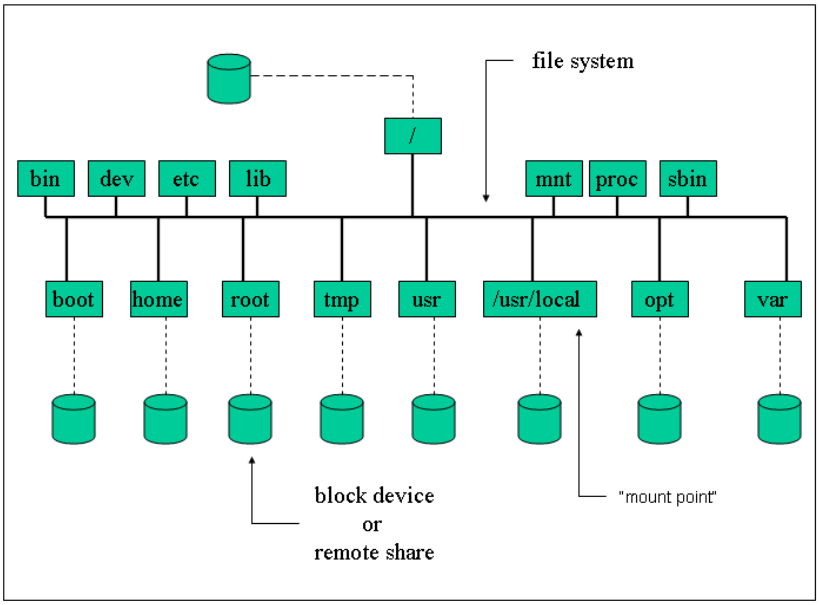
Forming a tree-like filesystem structure

The root of the tree structure is called **root** and is represented by a forward slash "**/**". The root mount point is also the first directory on which the operating system will attach a disk or resource, also called the **root device**.

Once the root is mounted the directories and subdirectories present on the root device can be used as further mount points for other devices, forming a succession of directories ordered like a tree.

The process is made possible as follows:

1. The bootloader will load a kernel telling it where the root device is (also see "Booting Linux" LPI 102)
2. The other directories are mounted following instructions from the **/etc/fstab** file (see p.33)



Mount points on the file system

Creating the Disk Layout

When installing Linux one has to create a partition scheme. This is a particular stage of the installation process and is done most often with a GUI tool such as Yast or DiskDruid. These tools allow one to do three things:

- create partitions of a given size
- select the filesystem type (see p.42)
- assign a mount point for each partition.

Some installations have an 'expert mode' where it is possible to use **fdisk** (see p.29) to create the partitions only.

A minimal partition scheme involves one root device and another partition for swapping. There are no rules when creating a disk layout but one generally takes into account the function of the computer (desktop, mail server, etc).

The SWAP partition

When creating a partition scheme one also has to make decisions about the amount of swap space needed. Once again, there are no rules. The amount of swap space needed depends of the type of applications that will run on the PC (desktop, server, 3D rendering, etc.). However as a rule of thumb, for a 2.4 kernel with an average amount of RAM (e.g less than 256MB) one will generally create a swap space twice as large as the amount of RAM. With older 2.2 kernels one would create a swap partition of the same size as the amount of RAM.

Swapping is generally done using a partition. In the partition table the hexadecimal value for a swap partition is 82.

NOTICE
Unlike partitions used for storing data a swap partition is never mounted. One also doesn't assign a mount point for such partitions. To create a SWAP space during the installation on simply selects the 'filesystem type' labelled 'SWAP'. Once the system is running information about the SWAP partitions is available in /proc/swaps

One can also create SWAP space areas using files rather than partitions (see LPI 201). This is often used for emergencies once a system is running and not during the installation.

6. Easy Dual Booting

(This section is not for exam purposes and can be left out completely).

If Windows9x/2k is already installed on the system the installation setup will automatically configure LILO for dual booting.

Pre-installation:

Before altering the system you should run a defragmentation program over the whole disk. This will make sure that all the blocks used by the Windows operating system are rearranged at the beginning of the disk.

Next, using PartitionMagic or fips, partition the C:\ drive in two. The Windows programs are located at the beginning of the hard disk in the first partition. The second partition must be large enough to hold a Linux installation.

Notice: The average amount of space needed for a Linux distribution is 4GB.

Starting the installation from DOS:

For non-NT systems restart your computer in DOS command mode. If you are installing RedHat then you can run E:\DOSUTILS\AUTOBOOT.BAT. This will start the installation program. Similarly if you are installing Suse you can run E:\setup.exe under DOS.

The hard drive from a Windows' perspective:

When running Windows the OS will only see the FAT and NTFS filesystems. The rest of the disk where Linux is installed will be inaccessible.

The hard drive from a Linux point of view:

When running Linux the Windows partition should be called /dev/hda1 (since it's the first partition on the first physical disk). By default this partition is not mounted. You can make a directory /dos or /mnt/dos and mount this partition. The disk partition corresponding to C:\ is then accessible.

7. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. The **rawrite** tool runs under Linux and is used to copy an image file onto a floppy disk _____

2. When devising a new partition scheme on an empty disk any disk partition can be chosen as the root device _____

Glossary

Term	Description
virtual filesystem	a filesystem is a data structure that allows data on a disk to be organised and accessed by the user. However to the user data is simply located in a series of directories and subdirectories. These directories form a tree structure with a top directory called the root and noted " **/** ". This structure is also called the 'virtual filesystem' because one doesn't need to know anything about the disk layout or partitioning scheme in order to use it. This is different to the situation when using a DOS based system; there if the disk has four partitions all the users will need to know that data can be in either C:\, D:\, E:\ or F:\ and, in this example, that the first CD-ROM is the G:\ device
mount point	a directory where a partition is attached in order to make the device available to the system
partitioning scheme	action performed during the installation to fix the number of partitions and mount points in order to create a standard 'virtual filesystem' on which software is installed. The standard which decides where software components are installed or where user home directories are kept is called the filesystem hierarchy standard (FHS) and should influence our choices when installing Linux (e.g most software is installed in the /usr directory, therefore always make sure that this directory is on a fairly large partition, at least 2 GB in most cases)
rescue mode	action of running a Linux operating system entirely in RAM together with a small root filesystem containing enough tools to access the hard drive. This is generally started with an installation CD
root (/)	the top directory where a first partition is attached. Either all the directories and subdirectories needed can be found on this partition or certain subdirectories of root can be used as mount points to attach further partitions (this depends on the partition scheme chosen during the installation!)

Commands

Command	Description
chroot	change into a directory and consider that directory as the root (/). By default **chroot** tries to run the Bash shell **/bin/bash**, but it is possible to specify any other command (see 'chrooted servers' in LPI 202)
dd	tool used to copy files as well as portions of a device (e.g hard drive, CD-ROM or floppy). An installation CD contains files called 'image files' which are copies of installation or driver disks that can be copied back onto a floppy
fips.exe	a utility found on most Linux distribution CDs that is used to resize a FAT partition in order to make space for a dual boot Windows/Linux system

rawrite	a DOS equivalent of **dd**

Exercises

1. Do a local CD installation. The following points outline a suggested strategy. The **OPTIONAL** points should be attempted only by advanced users familiar with package management and the **vi** editor.

(i) Installation Type: choose "Custom"

(ii) Disk Partitioning Setup: Partition the disk manually with Disk Druid:

This is a suggestion for a partitioning scheme using about 3GB of hard disk space. If you have more space available then make **/usr** larger and consider installing more packages than those suggested in step **(iv)**

IMPORTANT: Leave a free partition of at least 100MB. We will need this later!!

/boot 20M
/ 250M
/usr 2300M
/home 50M
/tmp 100M
/var 150M
SWAP 128M Notice that SWAP is a filesystem type and that no mount point is defined – see p.6

(iii) (OPTIONAL) Install LILO on /dev/hda2 or not at all. In all cases do not use the suggested /dev/hda, which is the MBR.

*We deliberately don't want the installation to boot properly. The bootloader will be fixed in step **2(i)** in rescue mode.*

(iv) Packages to install: (the names may vary from one distribution to another)

"X Window System" + "GNOME desktop environment" OR "KDE desktop environment"
"Editors"
"Graphical Internet"
"Software Development" [This is important, we will need this to compile packages later]

(v) Don't create a bootable floppy

2. (OPTIONAL) Rescue the system:

(i) Reboot with the installation CDROM. At the prompt type:

```
linux rescue
```

(ii) Read all the instructions until you get to a prompt. Use the **chroot** command as suggested.

(iii) You first need to install the **lilo** package. Edit /etc/lilo.conf (use **vi**). You should have

```
boot=/dev/fd0
prompt
linear
timeout=50
image=/boot/vmlinuz-<kernel-version>
        label=linux
        read-only
        root=/dev/<root-partition>
```

(v) Run **/sbin/lilo**. If an error occurs you may have to replace `linear` by `lba32` depending on your disk.

Hardware Configuration

Prerequisites

None

Goals

Understand hardware resource allocation (IRQs, I/O ports and DMA)
Overview hardware devices such as expansion cards, USB and SCSI devices
Detection of network interfaces and printers (no configuration)
Understand basic configuration steps for modems and sound cards

Contents

1. Resource Allocation

To allow peripherals and devices on the PC to communicate directly with system resources, in particular the CPU, the system allocates resources such as lines and channels for each device. These resources are Interrupt Request Lines (IRQ), Input/Output addresses and Direct Memory Access channels (DMA).

IRQs: The Interrupt Request Lines allow devices to request CPU time. The CPU will stop its current activity and process the instructions sent by the device. IRQs range from **0** to **15**.

I/O address: These represent specific addresses in the system's memory map. The CPU will then communicate with the device by *reading and writing to memory* at the specified address.

DMA: Certain devices can access the system's memory through a DMA channel, allowing them to write and process data without accessing the CPU. This can enhance performance.

● Listing Allocated Resources

The kernel keeps information related to allocated resources in the **/proc** directory. The relevant files are:

/proc/dma
/proc/interrupts
/proc/ioports
/proc/pci

Allocated resources can also be listed using tools such as **lspci** and **dmesg**:

lspci: lists chipset information of all attached PCI components. Lists I/O and IRQ settings with the **-v** flag . Also notice the **-b** (BUS centric) option which shows allocations assigned by the BIOS rather than the kernel.

dmesg. Continuously displays kernel messages. It also displays the kernel messages logged at boot time during the " kernel" stage . At this stage the kernel scans all the hardware on the system and can automatically allocate modules (drivers) for given chipsets. These messages are also available in **/var/log/dmesg**.

● Typical Resources

Device	*I/O port*	*IRQ*
/dev/ttyS0	0x03f8	4
/dev/ttyS1	0x02f8	3
/dev/lp0	0x378	7
/dev/lp1	0x278	5
soundcard	0x220	

Manual Resourse Allocation

NOTICE:
This is a very common example, however since kernel modules are only discussed in LPI 102 some may find it difficult. You may skip this example and go to section 2

Example: configuring two ethernet cards

1. For statically compiled modules, parameters can be passed to the kernel at boot time. A typical example is when two ethernet cards are present and only the first one is detected. The following line tells the kernel that:

- there is an ethernet card using IRQ 10 and I/O 0x300
- there is another ethernet card using IRQ 9 and I/O 0x340

```
ether=10,0x300,eth0      ether=9,0x340,eth1
```

You type this line at the LILO/GRUB 'boot:' prompt, or else, as with the RAM settings before, edit **/etc/lilo.conf** (use an `append=` statement) or **/etc/grub.conf**.

Notice that the `ether=` statement is a generic kernel command similar to `root=`, `mem=` or `init=`.
Also notice that you need not specify any information about the ethernet card (Intel, Netgear ...)

2. For dynamically compiled modules, IRQ and I/O address settings can be defined using **/etc/modules.conf** (or **/etc/conf.modules**). Assuming that in the above example both cards where using the `e100.o` kernel module, then **/etc/modules.conf** would contain the following:

```
alias eth0 e100
alias eth1 e100


options eth0 io=0x300 irq=10
options eth1 io=0x340 irq=9
```

2. PC Expansion Cards

ISA and PCI are the most common types of expansion cards. With the latest 2.4 kernel there is very little to be done in order to configure these. In the case of ISA buses however, and only with earlier kernels, it was necessary to scan the ISA bus in order to detect existing expansion cards (sound, ethernet, etc).

The *isapnptools* package provided the **pnpdump** tool which scanned the ISA bus for 'Plug and Play' (pnp) devices. The output would contain the chipset of the card together with I/O port, DMA and IRQ settings. This output would be redirected to **/etc/isapnp.conf** where changes could be made if needed. At boot time the **isapnp** tool would read **isapnp.conf** and would configure these ISA PnP devices.

Since kernel 2.4 PnP initialisation is supported through a kernel module called isapnp.o

3. USB Support

The Universal Serial Bus (USB) is a communication architecture designed to connect devices to a PC. These devices are divided into four classes:

Display Devices
Communication Devices
Audio Devices
Mass Storage Devices
Human Interface Devices (HID)

The devices are plugged into a USB port which is driven by a USB controller. Support for USB controllers is present in the Linux kernel since version **2.2.7** (The Linux USB sub-system HOWTO)

Host Controlers

There are 3 types of USB host controllers:

Host Controler	*Kernel Module*
OHCI (Compaq)	`usb-ohci.o`
UHCI (Intel)	`usb-uhci.o`
EHCI (USB v 2.0)	`ehci-hdc.o`

Once a USB device is plugged into a PC we can list the devices with**lsusb**:

```
lsusb
Bus 001 Device 001: ID 0000:0000
Bus 001 Device 002: ID 04a9:1055 Canon, Inc.
```

Hotplugging

Hotplug is a mechanism used to keep the state of the operating system updated when pluggable hardware devices are added or removed. In most cases the kernel signals an event by passing parameters to the script **/sbin/hotplug**.

This **hotplug** script runs all the scripts in **/etc/hotplug.d** (the default is **default.hotplug**) which in turn starts the appropriate agent listed in **/etc/hotplug**. The names of the agents correspond to different attachment types such as ieee1394, net, pci, scsi and usb.

The following log describes what happens when a USB camera is initialised:

Stage 1: USB kernel modules identify USB event and vendor/product ID:
13:26:19 kernel: hub.c: new USB device 00:07.2-1, assigned address 5
13:26:19 kernel: usb.c: USB device 5 (vend/prod 0x4a9/0x3058) is not claimed by any active driver.

Stage 2:The event arguments are passed to default.hotplug
13:26:19 default.hotplug[10507]: arguments (usb) env (DEVFS=/proc/bus/usb OLDPWD=/ PATH=/bin:/sbin:/usr/sbin:/usr/bin ACTION=add PWD=/etc/hotplug HOME=/ SHLVL=2 DEVICE=/proc/bus/usb/001/005 PRODUCT=4a9/3058/1 TYPE=255/255/255 DEBUG=yes _=/bin/env)

Stage 3: The usb.agent associates the product to a usbcam (using usb.usermap)
13:26:19 default.hotplug[10507]: invoke /etc/hotplug/usb.agent ()
13:26:23 usb.agent[10507]: Setup usbcam for USB product 4a9/3058/1
13:26:23 usb.agent[10507]: Module setup usbcam for USB product 4a9/3058/1
13:26:38 devlabel: devlabel service started/restarted

From this we can see that **Step 1** involves the kernel modules and **Step 2-3** involve the hotplug mechanism. One can also see that the correct USB map must be available in order to fully initialise the device.

The usbmgr tool

On Debian systems an alternative to hotplug is provided with the **usbmgr** package. The main files are:

`/usr/sbin/usbmgr`	The daemon that listens for USB related events
`/usr/sbin/dump_usbdev`	Tool to list USB devices (similar to lsusb)
`/etc/usbmgr/usbmgr.conf`	Configuration file containing vendor/product IDs

4. SCSI Devices

Types of SCSI devices

There are two types of SCSI interfaces:

- an 8-bit interface with a bus that supports 8 devices, this includes the controller, so there is only space for 7 block devices (tapes, disks, etc)
- a 16-bit interface (WIDE) with a bus that supports 16 devices including the controller, so there can only be 15 block devices.

SCSI devices are uniquely identified using a set of 3 numbers called the **SCSI ID**:

a. the SCSI channel
b. the device ID number
c. the logical unit number LUN

The SCSI Channel

Each SCSI adapter supports one data channel on which to attach SCSI devices (disc, CDROM, etc) These channels are numbered from 0 onwards.

Device ID number

Each device is assigned a unique **ID** number that can be set using jumpers on the disk. The IDs range from 0 to 7 for 8-bit controllers and from 0 to 15 for 16-bit controllers.

Logical Units

The Logical Unit Number (LUN) is used to differentiate between devices within a SCSI target number. This is used, for example, to indicate a particular partition within a disk drive or a particular tape drive

within a multi-drive tape robot. It is not seen so often these days as host adapters are now less costly and can accommodate more targets per bus.

Hardware Detection

All detected devices are listed in the **/proc/scsi/scsi** file. The example below is from the SCSI-2.4-HOWTO

/proc/scsi/scsi

```
Attached devices:
    Host: scsi0 Channel: 00 Id: 02 Lun: 00
      Vendor: PIONEER  Model: DVD-ROM DVD-303  Rev: 1.10
      Type:   CD-ROM              ANSI SCSI revision: 02
    Host: scsi1 Channel: 00 Id: 00 Lun: 00
      Vendor: IBM      Model: DNES-309170W     Rev: SA30
      Type:   Direct-Access       ANSI SCSI revision: 03
```

The **scsi_info** tool uses the information in **/proc/scsi/scsi** to printout the SCSI_ID and the model of a specified device. From the file above **scsi_info** would produce the following output:

```
scsi_info /dev/sda
SCSI_ID="0,0,0"
MODEL="IBM DNES-309170W"
FW_REV="SA30"
```

Booting from SCSI disks

The system will boot from the device with SCSI ID 0 by default. This can be changed in the SCSI BIOS which can be configured at boot time.

If the PC has a mixture of SCSI and IDE disks, then the boot order must be selected in the systems BIOS first.

5. Network cards

The network interface card (NIC) must be supported by the kernel. You can get information about your current card using either of the following:

dmesg, **lspci**, **scanpci**, **/proc/interrupts**, **/sbin/lsmod**.or **/etc/modules.conf**:

```
dmesg
Linux Tulip driver cersion 0.9.14 (February 20, 2001)
PCI: Enabled device 00:0f.0 (0004 ->0007)
PCI: Found IRQ 10 for device 00:0f.0
```

```
dmesg
eth0: Lite-On 82c168 PNIC rev 32 at 0xf800, 00:0A:CC:D3:6E:0F,
IRQ 10
eth0: MII transceiver #1 config 3000 status 7829 advertising
```

```
cat /proc/interrupts
▶    0:    8729602          XT-PIC  timer
     1:          4          XT-PIC  keyboard
     2:          0          XT-PIC  cascade
     7:          0          XT-PIC  parport0
     8:          1          XT-PIC  rtc
     10:    622417           XT-PIC  eth0
     11:         0           XT-PIC  usb-uhci
     14:    143040           XT-PIC  ide0
     15:       180           XT-PIC  ide1


/sbin/lsmod
▶    Module                   Size  Used by
     tulip                   37360   1 (autoclean)
```

From the examples above we see that the Ethernet card's chipset is Tulip, the i/o address is 0xf800 and the IRQ is 10. This information can be used either if the wrong module is being used or if the resources (i/o or IRQ) are conflicting.

This information can either be used to insert a module with a different i/o address (using the **modprobe** or **insmod** utilities) or can be saved in **/etc/modules.conf** (this will save the settings for the next bootup).

6. Setting up modems

We first need to detect the modem. If the modem is an external modem all one needs to consider is the serial port it is using. However when dealing with a built-in PCI modem we need information about the I/O port and interrupt used by the device in order to determine which serial device should be configured.

● **The Modem device**

If we have an external modem we can go straight to the next section 'The serial port'.

A PCI modems device can be detected with **lspci**. (the listing below is from PCI-Modem micro-HOWTO):

```
lspci -v
            ----- snip -----
▶  00:0c.0 Serial controller:US Robotics/3Com 56K FaxModem Model 5610 (rev
01)   (prog-if 02 [16550])
      Subsystem: US Robotics/3Com USR 56k Internal FAX Modem (Model 2977)
      Flags: medium devsel, IRQ 11
```

```
      I/O ports at e800 [size=8]
      Capabilities: <available only to root>

            ------ snip ----
```

Notice that the I/O port is 0xe800 and the IRQ is 11

We can now use this information and assign these resources to a serial port device.

● **The serial port**

The modem uses a serial interface for communications. Information is sent through the telephone line as a sequence of bits (serial) over two wires (in and out). Incoming sequential data is translated into parallel data for the PC bus and vice versa for bits of data leaving the computer. The translation is done by a UART chip located on the serial port of the motherboard or inside an internal (PCI) modem.

To see which serial ports were detected at boot time on the system, we do the following:

```
dmesg | grep ttyS

►     /dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
      /dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
```

So far, these are preconfigured serial ports with I/O ports and IRQs generally used by a hardware serial port.

NOTICE
When configuring an external modem one only has to consider serial devices with IRQ 3 or IRQ 4. The I/O ports reported above are also standard addresses used by hardware serial ports

The following table shows the equivalence between DOS COM ports and Linux serial devices.

Serial port equivalence DOS-Linux

DOS	***Linux***
COM1	/dev/ttyS0
COM2	/dev/ttyS1
COM3	/dev/ttyS2

One can also use **setserial** to scan the serial devices. With the **-g** option this utility will tell you which serial devices are in use:

```
setserial -g /dev/ttyS[01]

►     /dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
      /dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
```

Now if we have a PCI modem as the one detected using **lspci** on p.16 we need to remember the I/O port and IRQ setting used:

Hardware setting for the PCI modem on p.16	
I/Oport	0xe800
IRQ	11

This time we will use **setserial** to assign these values to a serial device (other than the hardware serial devices) as follows:

```
setserial /dev/ttyS4 port 0xe800 irq 11 autoconfig
```

The `autoconfig` option automatically sets the correct UART. This command can be saved in a shell script called `serial.rc` and will configure the serial port every time we boot.

A symbolic link called **/dev/modem** pointing to the used serial port is often used to reference the modem.

Manually linking the modem device

```
ln -s /dev/ttyS1 /dev/modem
```

The **setserial** tool is also used to set the speed of the serial port.

setserial speed option	Description
spd_hi	use 56kb instead of 38.4kb
spd_vhi	use 115kb instead of 38.4kb
spd_shi	use 230kb instead of 38.4kb
spd_warp	use 460kb instead of 38.4kb
spd_cust	use the custom divisor to set the speed at 38.4kb (baud rate = baud_base / custom_divisor)
spd_normal	use 38.4kb when a baud rate of 38.4kb is selected

For example setting the speed for the serial port */dev/ttyS4* to 115kb is done as follows:

```
setserial /dev/ttyS4 spd_vhi
```

The **wvdial** commandline tool has a setup script called **wvdialconf** which will scan the system for modems (all serial and USB ports are scanned automatically). Once the script has run a skeleton configuration file is generated:

Sample /etc/wvdial.conf file:

```
[Dialer Defaults]
Modem = /dev/ttyS1
Baud = 115200
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 S11=55 +FCLASS=0
; Phone = <Target Phone Number>
; Username = <Your Login Name>
; Password = <Your Password>
```

A quick way to get started is to replace *Defaults* with the name of your provider say WorldISP, fill in the Usernam/Password entries and type the following:

```
wvdial WorldISP
```

One can also use **minicom** to configure a connection. This tools is first configured with the **-s** switch:

```
minicom -s

                [configuration]
            Filenames and paths
            File transfer protocols
            Serial port setup
            Modem and dialing
            Screen and keyboard
            Save setup as dfl
            Save setup as..
            Exit
            Exit from Minicom
```

All the work done on serial ports will be useful in the '`Serial port setup`' section, whereas dialling information (given by the ISP) can be entered by selecting the menu '`Modem and dialing`'

Once the modem is set up and is capable of dialling the outside world then it is possible to establish a serial connection to a remote host. To get a fully networked connection (i.e assign an IP address to a network interface) we still need to start the **pppd** which will create a *ppp0* network interface and uses the point to point protocol PPP to enable TCP/IP networking. With tools like **wvdial** this is done automatically when needed.

- **WinModems**

If all the above configurations fails then it likely that you have a modem that may only work with some added drivers. Such modems are called *winmodems* (see the winmodem HOWTO for some colourful definitions!).

A winmodem that can be made to work under Linux is also called a *linmodem*. (see the linmodem HOWTO for more details)

- **ISDN**

ISDN is a digital version of the "Plain Old Telephone Service" (POTS). It functions in a similar way, but instead of allowing a single direct analogue path, offers a number of 64KBit/S traffic or bearer channels and a low bandwidth data channel. ISDN2, the basic service offered in many countries, is a so called 2B+D service as it offers two bearer channels and a data channel.

There are a number of ways of using ISDN with a Linux machine. The simplest is to employ an external ISDN device that does the dialing, authentication and session for you, presenting the connection over an Ethernet network.

If the Linux machine is to be directly interfaced to the ISDN connection, a device called a Terminal Adapter (TA) will be required. The details of the various different interfaces to TAs are outside the scope of this course, but fall mainly into the following camps.

Modem Style AT command Interface

With serial connected TAs and some USB devices, the Linux machine is presented with an AT command interface exactly as if the TA were a modem. The TA can therefore be set up as if it were a modem. This has the advantage of being simple to do, but is less efficient than some other methods as it treats the data path as if it were a modem as well. Modems require that some characters are escaped as they have an active effect. ISDN has no such restriction and can pass any character.

PCI/ISA/PCCARD ISDN adapter cards and isdn4linux

A far more efficient way of using your ISDN line is to use an adapter card connected directly to a machine bus. The isdn4linux project seeks to encapsulate a lot of the details of making a connection over ISDN and present the finished connection as just another network interface. The package **isdn4k-utils** contains all the necessary software. Under Red Hat related Linux distributions the tool **"system-config-network"** will set this all up.

- **ADSL**

Asynchronous Digital Subscriber Line (ADSL) has largely replaced ISDN and private leased lines as the mass market higher bandwidth Internet connection method of choice. As with ISDN the Linux user has a number of strategy choices. The simplest, again as with ISDN, is to employ any of a number of different stand alone ADSL Router devices. These present the outside link via an Ethernet router. No special considerations need be taken on the Linux machine. The ADSL router is treated as an ordinary router. Many of these ADSL router devices are actually themselves embedded Linux machines. If the Linux user requires a closer connection to the ADSL service, they will probably need to acquire equipment and an ISP account capable of Point-to-Point Protocol over Ethernet (PPPoE). There are ADSL projects based around particular chipsets, but they have their own specific requirements and configuration methods.

PPPoE

ADSL is not a single protocol but rather a basket of related and interconnected protocols topped off with Point-to-Point Protocol (PPP). If the equipment and the ISP account are compliant, the Linux user can employ PPPoE to form the external connection. The Linux machine would initiate a PPPoE session and aim it at the MAC address of the ADSL equipment. The ADSL equipment would set up the layers beneath the PPP session and pass the PPP frames across those layers. The package "**pppoe**" contains all necessary software and setup information.

7. Printer Configuration

Printing is covered in depth in LPI 102. From a hardware perspective, the printers are detected at boot time automatically and can be seen in the **dmesg** output.

Linux printing happens in two stages. First the raw data is filtered into a postscript format, then the printing itself is handled by the ghostscript, or **gs** utility.

Using printtool (*not examined*)

This utility creates an entry in **/etc/printcap**. The main features which need to be specified are the location of the *input_filter*=if, the *spool_directory*=sd and the *printer_device*=lp.

If the **printtool** fails to detect which parallel port corresponds to the printer device you can use the **dmesg** utility to recall the kernel's initial parallel port scan.

Here is an example of a system with a local printer plugged into the first parallel port /dev/lp0

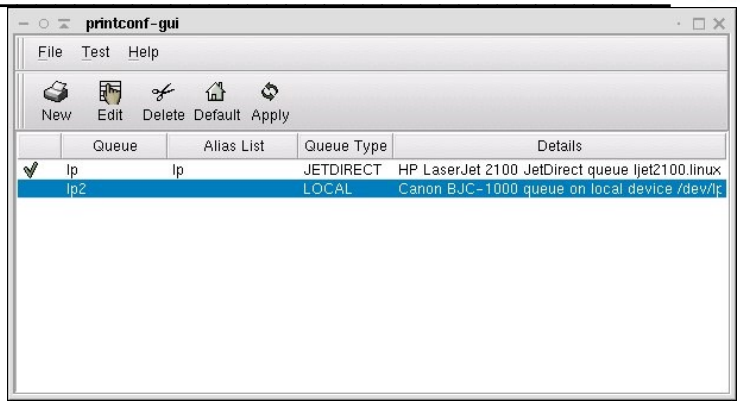
Parallel port scan at the end of dmesg

```
parport0: PC-style at 0x378 (0x778) [SPP,ECP,ECPEPP,ECPPS2]
parport0: detected irq 7; use procfs to enable interrupt-driven operation.
parport_probe: succeeded
parport0: Printer, HEWLETT-PACKARD DESKJET 610C
lp0: using parport0 (polling)
```

Sample /etc/printcap file

```
# This file can be edited with the printtool in the control-panel.
##PRINTTOOL3## LOCAL cdj550 300x300 a4 {} DeskJet550 3 {}
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/lp/filter:
```

Figure 7: The gtk-based printtool GUI



Using cups

Cups is a newer administration and configuration tool for printers. It's main configuration files are stored in **/etc/cups**. The printing process is the same except that **cups** uses its own filters situated in **/usr/lib/cups**.

The configuration tool for CUPS is a Web based GUI running on port **631**.

☝ When using cups **lpd** is replaced by the **cupsd** daemon.

NOTICE
A local printer is physically detected at boot time for both USB and parallel connections. Information on the boot process is displayed at any time with **dmesg**

8. Sound Cards

There are two sound support projects for Linux, namely the open sound system (OSS) and the advanced Linux sound architecture (ALSA). In fact the OSS is a commercial project which supports sound drivers on other UNIX platforms. The original modified OSS drivers where introduced as part of the Linux 2.0 kernel.

The ALSA project is more recent and has only been integrated into the Linux 2.6 kernel. For kernels older than 2.6, using ALSA drivers often means recompiling the kernel except for some Linux distributions such as Suse which adopted ALSA at an early stage.

In most cases the card is configured when the system is installed. Graphical sound configuration tools are also included with all main Linux distributions.

Detecting the sound card

As usual we will use **dmesg** to see if the kernel has detected the sound card as follows:

```
dmesg | grep -i  audio
NeoMagic 256AV/256ZX audio driver, version 1.1p
Initialized NeoMagic 256ZX audio in PCI native mode
```

NOTICE
The command above may return nothing, in which case you must search the output of **dmesg** again and try to determine which device corresponds to the sound card

Using the sndconfig tool (LPI101 objective p.152)

The above sound card would be fully configured if we could find the correct kernel module using the information found with **dmesg**. In the OSS framework this kernel module is then associated to a device name used by applications called *sound-slot-0* (for the first sound device).

This is what a sound configuration tool will do automatically for us. We choose (since it is an LPI101 objective) to discuss **sndconfig**.

This is a RedHat tool that configures audio devices using the OSS modules. You may need to install **sndconfig** as it is no longer installed on most Linux distributions. Then one simply types:

```
sndconfig
```

A graphical menu will be started with a message suggesting it will now probe for audio devices on you system. Select 'OK'.

On our system the following hardware was detected:

Neomagic Corporation | NM2360 [MagicMedia 256ZX Audio]

If no device is detected you will be presented with a list of manufacturers and card models supported by OSS from which to choose. If this happens, you may want to check the output of **lspci** again and also the following site with supported models: http://www.opensound.com/osshw.html

Once a model has been chosen **sndconfig** will attempt to load the associated kernel module and play a sample (surprise!) sound. If this worked then the **/etc/modules.conf** (covered in LPI 102) is automatically modified for us. To illustrate how our particular card has been configured here is the sound module entry:

/etc/modules.conf (entry for sound card used in this section)
`alias sound-slot-0 nm256_audio`

9. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. The root partition of a Linux system must always be on an IDE disk _____

2. A Linux system can support any USB device as long as the kernel has been compiled with USB support _____

Glossary

Term	Definition
DMA	Direct Memory Access allows certain hardware components to access memory to perform read-writes without having to interrupt the CPU
I/O address	a predefined memory range used by hardware devices and the CPU to perform read and write operations
IRQ	signal sent to the CPU by a device in order to interrupt the current process and get it to do something else
resource allocation	collection of DMA, i/o port and IRQ settings allocated to a hardware device
SCSI	interface used to transfer data between a device and the computer bus. For example the device can be a hard drive, a tape drive, a CD-ROM, a CD writer or a scanner
USB	Universal Serial Bus is a standard allowing external hardware devices to be attached to a computer without having to reboot. The design consists of a host controller to which is attached an initial hub device. This hub can then accommodate USB devices or more hub devices allowing to attach up to 127 devices (including hubs) to a single host controller

Resources

The Winmodems-and-Linux HOWTO
The Serial HOWTO
The Modem HOWTO
The Linux USB sub-system (http://www.linux-usb.org/)
SCSI terminology (http://www.scsita.org/terms/scsiterms.html)
The Linux 2.4 SCSI subsystem HOWTO
The Ethernet HOWTO
The Printing HOWTO
The Sound HOWTO
The isdn4linux project (http://www.isdn4linux.de/)
The Roaring Penguin PPPoE project (http://www.roaringpenguin.com/penguin/open_source_rp-pppoe.php)

Files

File	Description
/etc/isapnp.conf	a configuration file for **isapnp** - see **isapnp.conf(5)**
/proc/dma	list of currently used DMA channels
/proc/interrupts	list of currently used interrupts
/proc/ioports	list of currently used i/o ports
/proc/pci	list current information about the PCI bus
/etc/hotplug/usb.usermap	list of recognised USB devices
/var/log/dmesg	log file for current and boot time kernel messages
/proc/scsi/scsi	information about all SCSI devices – see scsi_info(8)

Commands

Command	Description
dmesg	print kernel message since boot time
hotplug	program used by the kernel to handle hardware related events - see **hotplug(8)**
isapnp	tool used to initialise ISA cards prior to kernel 2.4 – see **isapnp(8)**
lspci	list all PCI devices – see **lspci(8)**
lsusb	list all USB devices – see **lsusb(8)**
pnpdump	**pnpdump(8)** – dump ISA Plug-And-Play devices resource information
scsi_info	**scsi_info(8)** – SCSI device description tool
setserial	**setserial(8)** - get/set Linux serial port information
usbmgr	user space daemion which loads or unloads USB modules. It is an alternative to **hotplug** and generally used on Debian based systems
usb.agent	a **hotplug** agent which handles USB related events
usbmodules	**usbmodules(8)** – lists driver modules that may be able to manage interfaces on currently plugged in USB devices. usbmodules may be used by /sbin/hotplug or one of its agents (normally /etc/hotplug/usb.agent) when USB devices are "hot plugged" into the system
wvdial	a PPP dialer – see **wvdial(1)**

Exercises

1. Use the **dmesg** command to view the **/var/log/dmesg** file. Search for keywords such as *USB, tty* or *ETH0*.
- What are the names of the USB controllers used?
- What are the IRQs for the first two serial ports?

2. Investigate the contents of the following files:

```
/proc/ioports
/proc/interrupts
/proc/pci
```

```
/proc/dma
```

3. The PCI bus:
- Investigate the output of **lspci -v** and **scanpci –v**. What type of ethernet card is present?
- Verify that there are as many '*bus* ' entries listed with **lspci** and **/proc/pci**.

4. USB tools:
- Use **lsmod** and **lsusb** to determine which host controller is used on your system, UHCI, OHCI or EHCI (for USB v 2.0).
- Use **usbmodules** to list the kernel module which can handle the plugged in interface.

5. SCSI devices
- Given the following contents of the file **/proc/scsi/scsi** deduce the output of the command **scsi_info** (see p.15):

```
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
Vendor: PHILIPS  Model: CDRW48A          Rev: P1.3
Type:   CD-ROM                           ANSI SCSI revision: 02
```

Managing Devices

Prerequisites

Experience with the Linux installation process (also see the section **Installation** on p.1)

Goals

Understand the difference between a primary, an extended and a logical partition
Use partitioning tools when appropriate (before or after an installation)
Install and customise the boot loaders LILO and GRUB
Understand mount points and the role of the **/etc/fstab** file

Contents

1. Disks and Partitions

Physical disks:

On a running Linux system, disks are represented by entries in the **/dev** directory. The kernel communicates with devices using a unique major/minor pair combination. All major numbers are listed in **/proc/devices**. For example the first IDE controller's major number is **3**:

```
Block devices:
  1 ramdisk
  2 fd
  3 ide0
```

Hard disk descriptors in **/dev** begin with *hd* (IDE) or *sd* (SCSI), a SCSI tape would be *st*, and so on. Since a system can have more than one block device, an additional letter is added to the descriptor to indicate which device is considered.

Table 1	*Physical block devices*
hda	Primary Master
hdb	Primary Slave
hdc	Secondary Master
hdd	Secondary Slave
sda	First SCSI disk
sdb	Second SCSI disk

NB Inserting a new SCSI hard drive with a target number between two existing drives will bump up the device letter of the higher numbered drive. This can cause chaos within a disk system.

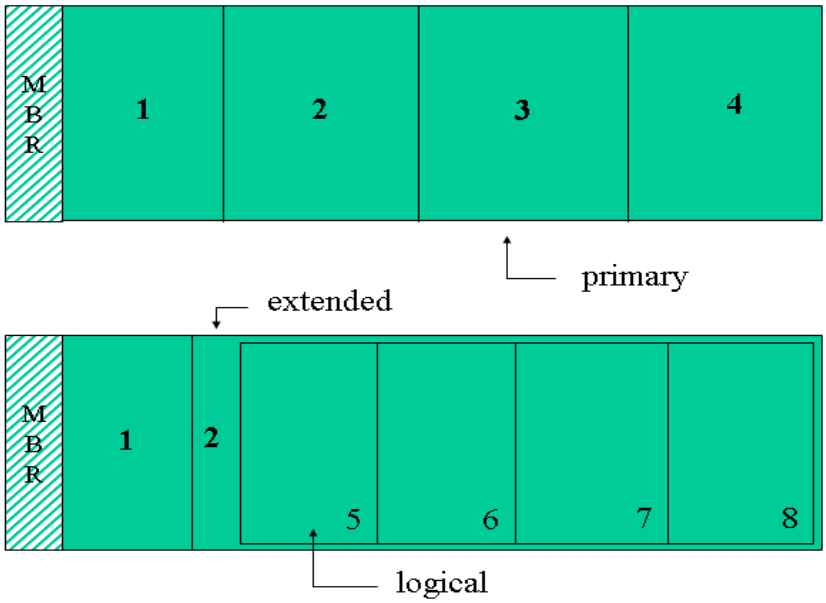
Disk Partitions:

Disks can further be partitioned. To keep track of the partitions a number is added at the end of each physical device.

Table 2	*Partitions*
hda1	First partition on first hard disk
hda2	Second partition on first hard disk
sdc3	Third partition on third SCSI disk

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

Example 1: The primary partitions (1,2,3,4) and (1,2,5,6,7,8)



*Typical output of **fdisk -l***

```
Device     Boot    Start     End       Blocks       Id    System
/dev/hda1   *        1         748        6297448+      b      Win95 FAT32
/dev/hda2            785       788        32130         83     Linux
/dev/hda3            789       2432       13205430      5      Extended
/dev/hda5            789       1235       3590496       83     Linux
/dev/hda6            1236      1618       3076416       83     Linux
/dev/hda7            1619      1720       819283+       83     Linux
/dev/hda8            1721      1784       514048+       83     Linux
/dev/hda9            1785      1835       409626        83     Linux
/dev/hda10           1836      1874       313236        83     Linux
/dev/hda11           1875      1883       72261         82     Linux swap
```

On this system the main feature to notice is that there are 3 primary partitions. The third partition is extended (/dev/hda3) and holds 8 logical partitions. The primary partition /dev/hda3 is not used. In fact /dev/hda3 acts as a 'container' and a filesystem exists only on the enclosed logical partitions.

NOTICE
Make sure to distinguish between primary, extended and logical partitions. Also make sure you understand the naming convention for the IDE disks and controllers.

2. Partitioning Tools:

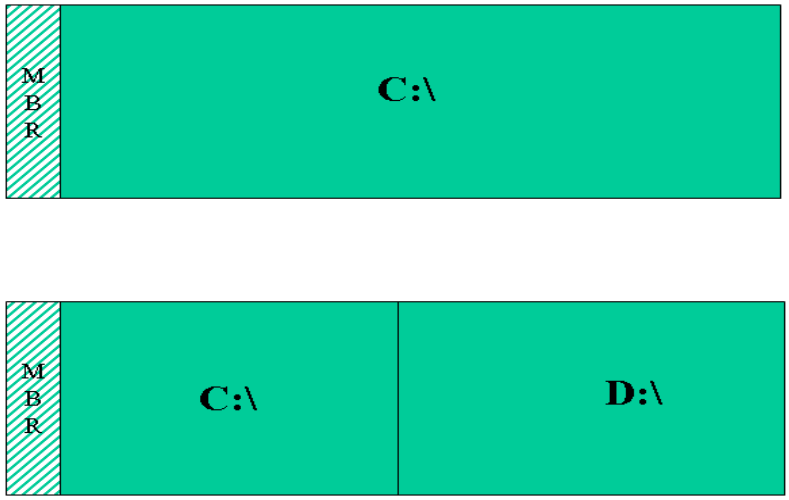
1. Before installation: (not for exam purpose)

PartitionMagic
fips

Notice that **fips** only handles **fat16** and **fat32**. On the other hand, **PartitionMagic** is much more versatile and can handle most common UNIX formats as well.

No partitioning is needed if for example C:\ and D:\ exist and the D:\ drive is empty.

Partitioning before installation:



2. During installation: (not for exam purpose)

While installing Linux you will have the choice of creating new partitions and associating each partition to a mount point (see p.5).

For advanced users this is done in two steps:

1. Use the **fdisk** tool to create new partitions
2. Associate a mount point to each partition

For intermediate users most distributions include a user friendly tool that does both these steps at once:

diskdrake (Mandrake)
DiskDruid (RedHat)

Finally, for beginners and busy sysadmin's, the latest Linux distributions will automatically assign a partition scheme.

3. On a Running System:

Once the operating system is installed you can use the **fdisk** utility to configure new partitions.

We will next look at the basic syntax for **fdisk**

Example:

1) Start partitioning the first hard drive:

```
fdisk  /dev/hda
```

2) Type **m** for help. Then create a new partition with **n**.

3) To write the changes to disk type **w**.

4) REBOOT.

These four points outline the steps you would follow to create new partitions. The last point is often overlooked. This forces the partition table in the master boot record **MBR** to be reread.

NOTICE
You need to create a filesystem on a new partition with **mkfs** or **mke2fs** before using it

This ends the survey of available partitioning tools. We next take a look at bootloaders.

3. Bootloaders

The MBR occupies the first sector of the disk (512 bytes) and contains the partition tables together with a bootloader. At boot time the bootloader reads the partition tables looking for a partition marked "active" and loads the first sector of this partion.

LILO the Linux Bootloader

There are roughly 3 parts involved:

1. LILO

This is the loader itself. LILO is installed on the MBR and loads the second stage bootloader, generally situated in **/boot/boot.b**.

2. /etc/lilo.conf

The main options are specified here

boot*	where LILO should be installed (/dev/hda is the MBR)
install	which second stage to install (**boot.b** is the default)
prompt	give the user a chance to choose an OS to boot
default	name of the image that will be booted by default
timeout	used with prompt, causes LILO to pause (units are 1/10 of a sec)
image*	path to the kernel to boot (one can use 'other' to chain load)
label*	name of the image. This is the name a user can type at the boot prompt
root*	the name of the disk device which contains the root filesystem **/**
read-only*	mount the root filesystem read-only for **fsck** to work properly
append	give kernel parameters for modules that are statically compiled.
linear/lba32	these options are mutually exclusive. Both ask LILO to read the disk using Linear Block Addressing. **linear** is typically used for very large disks. **lba32** is used to allow boot time access to data beyond the first 1024 cylinders (also see p.36)

3. /sbin/lilo

This binary reads it's configuration file /etc/lilo.conf and installs the LILO bootloader.

/sbin/lilo should be run every time a change is made to **/etc/lilo.conf**

GRUB the Grand Unified Bootloader

GRUB is also installed on the MBR. You can either alter this MBR with the **/sbin/grub** shell or use a configuration file called **/boot/grub/grub.conf** which will be read by **/sbin/grub-install**

Detailed information about GRUB can be found in the **info** pages

Main sections in **/boot/grub/grub.conf**:

1. General/Global

default	image that will boot by default (the first entry is 0)
timeout	prompt timeout in seconds

2. Image

title	name of the image
root	where the 2nd stage bootloader and kernel are e.g (hd0,0) is /dev/hda
kernel	path for the kernel starting from the previous root e.g /vmlinuz
ro	read-only
root	the filesystem root
initrd	path to the initial root disk

Example ***grub.conf***

```
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title   Linux (2.4.18-14)
        root (hd0,0)
        kernel /vmlinuz-2.4.18-14 ro root=/dev/hda5
        initrd /initrd-2.4.18-14.img
```

4. Managed devices

At boot time the **/etc/fstab** file assigns mount points for block devices.

The /etc/fstab format

device	mount-point	fstype	options	dump-number	fsck-number

Sample /etc/fstab

```
LABEL=/           /           ext2  defaults          1 1
LABEL=/boot        /boot      ext2  defaults          1 2
LABEL=/home       /home       ext3  defaults          1 2
/dev/fd0          /mnt/floppy  auto noauto,owner      0 0
LABEL=/usr        /usr        ext2  defaults          1 2
LABEL=/var        /var        ext3  defaults          1 2
none              /proc       proc  defaults          0 0
none              /dev/shm    tmpfs defaults          0 0
none              /dev/pts    devpts  gid=5,mode=620  0 0
/dev/hdc9         swap,pri=-1 swap  defaults          0 0
/dev/cdrom         /mnt/cdrom  iso9660  noauto,owner,kudzu,ro 0 0
```

The **mount** command is used to make a particular device available on a specific directory (mount point). The syntax is:

mount -t [FSTYPE] -o [OPTIONS] DEVICE DIRECTORY

For example we can mount a CDROM on the mount point **/mnt/cdrom** with:

```
mount -t iso9660 /dev/cdrom  /mnt/cdrom
```

On a running system the **/etc/fstab** file also acts as a *shortcut* for assigning a resource to a specific directory. For example:

```
mount /dev/cdrom
```

The **mount** utility reads **fstab** and deduces where to mount the resource. Notice that some of the devices are accessed using a label. Labels are assigned to devices with the **tune2fs** tool:

```
tune2fs -L /usr/local /dev/hdb12
```

Option summary for **mount**:	
rw,ro	read-write and read-only
noauto	the device is not mounted at boot time
users	the device can be read and unmounted by all users
user	the device can be unmounted only by the user
owner	the device will change it's permission and belong to the user that mounted it
usrquota	start user quotas on the device
grpquota	start group quotas on the device

NOTICE
Remember that **mount -a** will mount all filesytems in **/etc/fstab** that have not been mounted and do not have the option **noauto**

The **umount** command will unmount a device. Notice that the command is misspelled! The syntax is:

umount DEVICE or **MOUNT-POINT**

For example the following commands will both unmount the CDROM device:

```
umount /dev/cdrom
```

or

```
umount /mnt/cdrom
```

5. Quotas

The quota tools allow administrators to set up quotas without having to reboot. Here are the steps.

1. Edit **/etc/fstab** and add **`usrquota`** to the options

2. Remount the partition:

```
mount  -o remount  <device>
```

3. Start the quota stats:

```
quotacheck -ca
```

The preliminary **aquota.user** database file is generated at the top of the directory.

4. Edit quotas for each user:

```
edquota -u <user>
```

Here a soft/hard limit must be set for both the number of blocks and inodes available for each user.

The system will allow the user to exceed the soft limit during a certain **grace** period. After the grace period has expired the soft limit will be enforced as a hard limit.

5. START enforcing quotas:

```
quotaon -a
```

Users can query the quota status with **quota**. The system administrator can generate reports with **repquota** or **quotastats**.

6. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. When LILO is installed and further changes are made to **lilo.conf** it is not necessary to re-run **/sbin/lilo** _____

2. When GRUB is installed and further changes are made to **grub.conf** it is not necessary to re-run **grub-install** _____

3. If **/root** is beyond the 1024 cylinder limit then a Linux system may not boot _____

4. Quotas may only be used on an entire partition _____

Glossary

Term	Description
partition	an independent section of a hard drive which can either be used to directly store data or can be further divided into (logical) partitions. Since all partitions mounted and manipulated by the operating system have a separate filesystem and are considered as independent devices, a partition is also sometime called a 'device' or even a 'filesystem'
primary (partition)	partition described by one of the original partition record kept in the partition table. Only four such records are available, therefore disks can only have four primary partitions
extended (partition)	a primary partition whose partition record contains a linked list of partition records making it possible to create further partitions called 'logical partitions' as opposed to primary
logical (partition)	a partition contained in an extended partition
MBR	the first sector (512 bytes) of a hard drive which contains the boot loader and the disks partition tables
1024 cylinder limit	when using CHS addressing old BIOS systems would use 10 bits to read the number of cylinders ,8 bits for the heads and 6 bits for the sectors. This would allow access to disks with a maximum size of (2^10)*(2^8)*(2^6-1)*512 bytes which corresponds to 8.5 GB (metric). When running **/sbin/lilo** addresses are given in CHS form (unless lba32 or linear is used) for the BIOS to read. If the second stage boot loader **/boot/boot.b** is 'further' than 1024 cylinders away from the MBR then the system will not boot
bootloader	code stored in the first 512 bytes of a disk which is read by the BIOS and used to start an operating system
GRUB	Grand Unified Bootloader
LILO	Linux boot Loader
quotas	mechanism available per device to set restrictions on the amount of inodes and data blocks per user or group
grace	time limit for files and data that have exceeded their soft limit

Files

File	Description
/boot/boot.b	The LILO second stage bootloader
grub.conf	configuration file for GRUB
/etc/fstab	**fstab(5)** – The file fstab contains descriptive information about the various file systems. fstab is only read by programs, and not written; it is the duty of the system administrator to properly create and maintain this file
/etc/lilo.conf	configuration file read by the bootloader installation mapper **/sbin/lilo**
/proc/devices	devices found on the system and their associated major number
aquota.user	database file stored on the root of the device where user quotas are inforced

Commands

Command	Description or *apropos*
/sbin/lilo	**lilo(8)** – install boot loader
edquota	**edquota(8)** – edit user quotas
fdisk	**fdisk(8)** – partition table manipulator for Linux
grub-install	**grub-install(8)** - install GRUB on your drive
mount	**mount(8)** – mount a file system
quotas	**quota(1)** – display disk usage and limits
quotacheck	**quotacheck(8)** – scan a filesystem for disk usage, create, check and repair quota (database) files
quotaon	**quotaon(8) / quotaoff** – turn filesystem quotas on and off
quotastats	**quotastats(8)** – program to query quota statistics
repquota	**repquota(8)** – summarize quotas for a filesystem
tune2fs	**tune2fs(8)** – adjust tunable filesystem parameters on second extended filesystems
usrquota,grp quota	(not a command) option set in **/etc/fstab** to enable quotas on a device
umount	**umount(8)** – unmount file systems

Exercises

1. Create 1 new partition on the **/dev/hda** device using **fdisk**.

```
fdisk /dev/hda
```

HINT: To create a new partition type **n**. The partition type defaults to **83** (Linux)
To write the new partition table type **w**.
The partition table needs to be read: REBOOT the computer !

2. Make a new filesystem (format) on one of the partitions:

```
mkfs <device>
```

3. **(i)** Make a directory called **data in the root directory.**

```
mkdir /data
```

(ii) Edit **/etc/fstab** and allocate the mount point **/data** to this new resource

```
<device>    /data ext2  defaults    0 2
```

4. Force **mount** to read **/etc/fstab**:

```
mount -a
```

If this doesn't work check that each entry is correct in the **fstab** and make sure that the directory **/data** exists (**2 (i)**)

5. Follow the steps in this chapter to enforce quotas on this device.

After step **(2)** run the **mount** command and look at the output. Which option from **/etc/fstab** can be seen showing that quotas are availabe on the device? _____

After step **(3)** which file is created in the **/data** directory? _____

Before testing quotas for with non-root users, add read-write permissions on **/data**

```
chmod o+rw /data
```

In extreme cases it may be easier to reboot and let the init scripts build the **aquota.user** (or **aquota.group**) file. If nothing is showing with the **quotas**, **repquota**, or **quotastats** tools make sure you have read-write access for everyone on **/data** [chmod a+rw /data]

6. (OPTIONAL) The instructor computer has a NFS share. Find out which directory is shared and edit **/etc/fstab** to mount this share on **/mnt/nfs**. Use the **noauto** option fot the share not to mount at boot time.

7. SWAPPING bootloaders

a. Uninstall LILO from the MBR (or the floppy)
```
lilo -u
```
b. Modify the **grub.conf** sample on p. 28 to reflect your system
c. Install GRUB on the floppy with **grub-install /dev/fd0**

The Linux Filesystem

Prerequisites

Experience with the Linux installation process (also see the section **Installation** on p.1)

Goals

Introduce the base directories and concepts from the File System Hierarchy (FHS)
Make a filesystem of any type (e.g EXT2or EXT3) on a partition
Monitor free space per device or directory
Understand the UNIX-like file and directory permissions

Contents

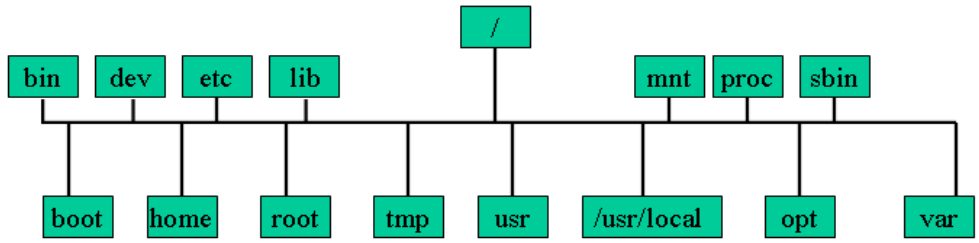
1. The Filesystem Structure

A filesystem is similar to a tree structure. The root of the tree is represented at the top and the leaves below.

As mentioned earlier, once partitions have been created each partition must be given a *mount point*. This is typically done at installation time. To help us understand where things are kept, let us look at the Linux file system hierarchy.

The top of a Linux file system hierarchy starts at root (/). This is similar to C:\ under DOS except that C:\ is also the first device, whereas the root directory can be mounted anywhere.

Figure 1: The base directories



The base directories

Directories that can be mount points for separate devices

The base directories are the first subdirectories under the root directory. These are installed by an *rpm* package usually called *filesystem*.

```
rpm -ql filesystem
```

During the booting process the kernel first mounts the root (/) partition. In order to mount and check any further partitions and filesystems a certain number of programs such as **fsck**, **insmod** or **mount** must be available.

☞ The directories **/dev**, ***/bin***, ***/sbin***, ***/etc*** and ***/lib*** must be subdirectories of root (/) and not mounted on separate partitions. Added to this, there must be an empty directory **/proc** on the root device used by the kernel to report information about the operating system (e.g processes, memory statistics, etc ...)

Base directories:

- **/bin and /sbin**
  Contain binaries needed to boot up the system and essential commands.

- **/dev**
  Location for device or special files

- **/etc**
  Host specific configuration files

- **/lib**
  Shared libraries for binaries in **/bin** and **/sbin**. Also contains kernel modules

- **/mnt/ or /media (Suse)**
  Mount point for external filesystems

- **/proc**
  Kernel information. Read-only except for **/proc/sys/**

- **/boot**
  Contains the Linux kernel, the system maps and the “second stage” bootloaders.

- **/home (optional)**
  The directories for users. Initially contains the contents from **/etc/skel/**

- **/root (optional)**
  The directory for user root

- **/tmp**
  Temporary files

- **/usr**
  User Specific Resource. Mainly static and shareable content

- **/usr/local or /opt (optional)**
  Add-on software applications. Can also contain shared libraries for add-on software.

- **/var/www, /var/ftp/ or /srv (Suse)**
  Location for HTML pages and anonymous FTP directories.

- **/var**
  Variable data, such as spools and logs. Contains both shareable (eg. /var/spool/mail) and non-shareable (eg. /var/log/) subdirectories.

2. Formatting and File System Consistency

In order to organise data on a disk partition one needs to create a file system. At installation time you will be asked which type of file system must be used.

Many file system types are supported. The **ext2** file system type is the default and is also known as “Linux Native”. In some more recent installers, ext3 is the default. This is really only an ext2 filesystem with a journal patched on top.

A different file system type must be used for SWAP. The file system for Swap is of type **swap** and cannot be anything else.

The Second Extended File System

Lets take a closer look at the **ext2** (second extended) file system. The **ext2** consists of blocks of size 1024 bytes =1 KB (default). Without entering into too much detail, there are three types of blocks:

● Superblocks:

Repeated every 8193 blocks. Contains information about block-size, free inodes, last mounted time, etc …

● Inodes:

Contains pointers to data blocks. The first 12 blocks of data are directly accessed. If the data exceeds 12KB, then indirect inodes act as relays.
Each inode is 256 bytes and contains the user, group, permissions and time stamp of the associated data.

● Data Blocks:

These are either files or directories and contain the actual data.

Formatting tools

The file systems supported by the kernel allow one to read from a pre-formatted disk. To create these file systems while running a Linux system one also needs to install the associated formatting tools.

The formatting tool for **ext2** is **mkfs.ext2** or **mke2fs**. Similarly the formatting tool for the **xfs** file system type from Silicon Graphics will be **mkfs.xfs** and may have to be installed separately.

The **mkfs** tool acts as a front for all these file system types. The syntax is:

mkfs –t <fstype> <DEVICE>

Notice that the **ext3** is an **ext2** file system type on which a journaling system has been added (see the exercises for details).

Example 1: Making a jfs filesystem

```
mkfs -t jfs /dev/hda12
```

Example 2: Making a ext2 filesystem

```
mke2fs /dev/hda11    [or mkfs -t ext2 /dev/hda11]
```

File System Consistency

If the file system is damaged or corrupt, then the **fsck** utility should be run against the partition (the minimum requirement is that the file system be unmounted or mounted read-only).

fsck acts as a front that automatically detects the file system type of a partition. Then as with **mkfs**, the tools **fsck.ext2**, **fsck.ext3** will be called accordingly. Since EXT2 is the main filesystem type for Linux there is a **e2fsck** command that only handles this filesystem type.

You can explicitly specify a file system type with the following syntax:

fsck -t <fstype> <device>

Example: Checking a *reiserfs* filesystem on the /dev/sdb10 device:

```
fsck -t reiserfs /dev/sdb10
fsck.reiserfs /dev/sdb10
```

File System Debug Commands

The **debugfs** and **dumpe2fs** are seldom used but can be useful in providing low level information about an ext2 or ext3 filesystem.

debugfs [-b blocksize] [-s superblock] [-f cmd_file] [-R request] [-V] [[-w] [-c] [-i] [device]]

The debugfs program is an interactive file system debugger. It can be used to examine and change the state of an ext2/3 file system.

Once in the debugfs shell, internal commands can then be used to change directory, examine inode data, remove files, create links, dump the ext3 journal logs etc. While this is a very powerful command, it should be used with caution, generally only after the fsck command has failed to make any headway.

dumpe2fs [-bfhixV] [-ob superblock] [-oB blocksize] device

dumpe2fs prints the super block and block group information for the filesystem present on *device*.

```
dumpe2fs /dev/hda1

dumpe2fs 1.35 (28-Feb-2004)
Filesystem volume name:   /boot1
Last mounted on:          <not available>
Filesystem UUID:          d741042c-3eaf-49ee-94c1-7dd8c5459764
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index
filetype needs_recovery sparse_super
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              25584
Block count:              102280
Reserved block count:     5114
Free blocks:              80564
Free inodes:              25537
First block:              1
Block size:               1024
Fragment size:            1024
Reserved GDT blocks:      256
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1968
Inode blocks per group:   246
Filesystem created:       Sat May  7 10:40:51 2005
Last mount time:          Sun May 29 04:08:01 2005
Last write time:          Sun May 29 04:08:01 2005
Mount count:              10
Maximum mount count:      -1
Last checked:             Sat May  7 10:40:51 2005
Check interval:           0 (<none>)
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Journal inode:            8
Default directory hash:   tea
Directory Hash Seed:      50108791-6a0a-41ff-9608-0485c993eaf9
Journal backup:           inode blocks


Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
  Block bitmap at 259 (+258), Inode bitmap at 260 (+259)
  Inode table at 261-506 (+260)
  0 free blocks, 1942 free inodes, 2 directories
  Free blocks:
  Free inodes: 27-1968

[....]
```

3. Monitoring Disk Usage

Using **mount** and **df**:

Both these tools work on a device level, as opposed to a directory level. The **mount** and **umount** tools maintain the list of mounted filesystems in **/etc/mtab**.

Typing **mount** with no options will show all filesystems currently mounted. The output is similar to **/etc/mtab**. Notice that the kernel also keeps track of mounted filesystems in **/proc/mounts**.

In addition to showing all mounted devices the **df** tool will also show *Used* and *Available* disk space. By default this is given in blocks of 1K.

```
df -h
```

➔

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda9             289M  254M   20M  93% /
/dev/hda2              23M  7.5M   14M  35% /boot
none                   62M     0   61M   0% /dev/shm
/dev/hda5             1.4G  181M  1.1G  13% /share
/dev/hda7             787M   79M  669M  11% /tmp
/dev/hda3             4.3G  3.4G  813M  81% /usr
/dev/hda6             787M  121M  627M  17% /var
//192.168.123.2/share   12G  8.8G  3.7G  71% /mnt/smb
```

Using **du**:

This tool will display *disk usage*. This is done on a per directory basis. Notice that **du** cannot display available space since this information is only available at a device level.

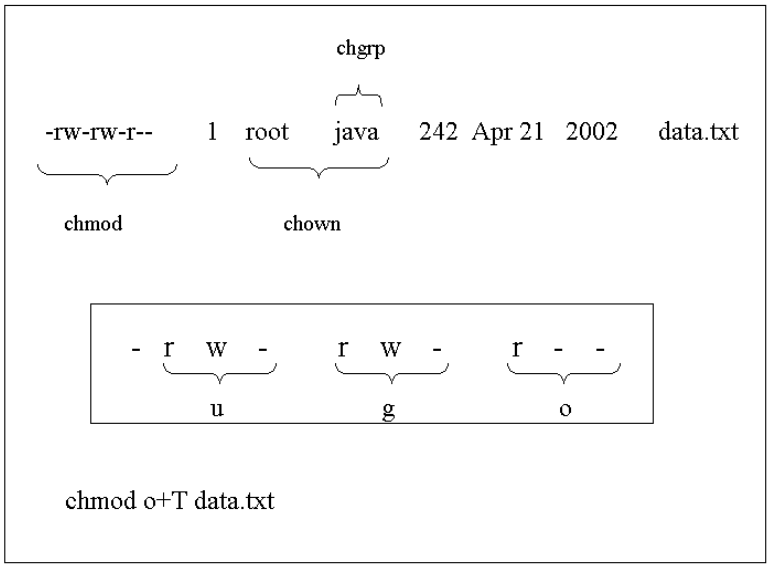
The following command will list the current usage of the **/etc** directory in human readable units (using the **-h** switch) and will only print the grand total (using the **-s** switch):

```
du -sh /etc
```

➔

```
62M     /etc/
```

4. File Permissions and Attributes



Changing permissions and owners

From the previous figure we see that permissions can be acted upon with **chmod**. There are 3 categories of ownership for each file and directory:

u: a valid user with an entry in /etc/passwd
g: a valid group with an entry in /etc/group
o: other

Example:

-rw-rw-r-- 1 jade sales 24880 Oct 25 17:28 libcgic.a

Changing Permissions with **chmod**:

```
chmod g=r,o-r libcgic.a
chmod g+w  libcgic.a
```

Changing user and group with **chown** and **chgrp** :

```
chown  root  libcgic.a
chgrp  apache libcgic.a
```

NOTICE

A useful option for **chmod**, **chown** and **chgrp** is **–R** which recursively changes ownership and permissions through all files and directories indicated.

Symbolic and octal notation

Permissions can be read=r, write=w and execute=x. The octal values of these permissions are listed in the next table.

Octal and symbolic permissions.

Symolic	octal	binary
read	4	' 100'
write	2	' 010'
execute	1	' 001'

Permissions apply to the user, the group and to others. An item has a set of 3 grouped permissions for each of these categories.

How to read a 755 or -rwxr-xr-x permission

user	group	other
rwx 4+2+1=**7**	r_x 4+1=**5**	r_x 4+1=**5**

The standard permission

UNIX system create files and directories with standard permissions as follows:

Standard permission for:

Files	666	-rw-rw-rw-
Directories	777	-rwxrwxrwx

Umask

Every user has a defined **umask** that alters the standard permissions. The **umask** has an octal value and is subtracted(*) from the octal *standard permissions* to give the files permission (this permission doesn't have a name and could be called the file's *effective* permission).

(*) While subtraction works in most cases, it should be noted that technically the standard permissions and the umask are combined as follows:

Final Permissions = Standard Permissions (logical AND) (NOT)Umask

On systems where users belong to separate groups, the umask can have a value of 002.
For systems which place all users in the *users* group, the umask is likely to be 022.

SUID permissions

An executable can be assigned a special permission which will always make it run as the owner of this file. This permission is called SUID meaning 'set user ID'. It has a symbolic value **s** or a numerical value **4000**.

Administrative tools may have the SUID bit set in order to allow non-root users to change system files.

For example the **passwd** command can be run by any user and will interactively change his or her current password. This password will be saved to **/etc/passwd** or **/etc/shadow**. However both these files belong to user root with typical permissions of 644 and 600 respectively.

This problem has been solved by setting the SUID bit on **passwd** hence forcing it to run as user root with the correct permissions to modify **/etc/passwd** or **/etc/shadow**.

The SUID on **passwd**

```
ls -l $(which passwd)
-r-s--x--x    1 root     root        18992 Jun  6  2003 /usr/bin/passwd
```

NOTICE

The SUID bit is shown in symbolic form in the command above. It is possible to get more information about a file using **stat** as well as seeing the octal representation of the permissions as follows:

```
stat /usr/bin/passwd
File: `/usr/bin/passwd'
Size: 18992     Blocks: 40    IO Block: 4096    regular file
Device: 305h/773d    Inode: 356680      Links: 1
Access: (4511/-r-s--x--x)  Uid: ( 0/ root)    Gid: ( 0/ root)
```

WARNING! WARNING! WARNING!

The SUID permission is often associated with security issues. Here is an example that illustrates this.

1. A user would like to read user root' s mail. For this he changes the environmental variable as follows:

```
export MAIL=/var/spool/mail/root
```

2. The user then uses the command **mail**, hoping to see something!

```
mail
/var/spool/mail/root: Permission denied
```

So far it doesn't work. This would be too easy!

But if root can be convinced to set the SUID bit on **mail** the previous commands would allow any user to read anybody's mail (including root).

The next examples are dangerous. Why?

```
chmod 4755 /bin/cat
chmod u+s /bin/grep
```

SGID permissions

The SGID is a permission similar to SUID that is set for group members. The symbolic value is **s** and the octal value of **2000**.

Setting SGID on a directory changes the group ownership used for files subsequently created in that directory to the directories group ownership. No need to use `newgrp` to change the effective group of the process prior to file creation (see exercise p.55) .

Examples:

```
chmod 2755 /home/data
chmod g+s /bin/wc
```

The sticky bit

The sticky bit permission with value **1000** has the following effect:

- Applied to a directory it prevents users from deleting files unless they are the owner (ideal for directories shared by a group)
- Applied to a file this used to cause the file or executable to be loaded into memory and caused later access or execution to be faster. The symbolic value for an executable file is **t** while for a non executable file this is **T**. As file system caching is more generic and faster, file sticky bits tend not be supported any more.

Examples:

```
chmod 1666 /data/store.txt
chmod o+t /bin/bash
```

File Attributes

Alongside the standard permissions there is another system that can be used to change the way a file can be used. File Attributes do not show up in the 'ls' command. The '**lsattr**' command must be used instead. The '**chattr**' command is used to set and drop these attributes.

The following attributes are available. Please note the case.

'A' When a file with the 'A' attribute set is accessed, its atime (access time) record is not modified. This avoids a certain amount of disk I/O, typically for temporary files. Do be aware that some tools, such as tmpwatch, rely on the atime record to determine if the file has been used recently. If the atime record is not being updated the file's status might be misinterpreted.

'a' A file with the 'a' attribute set can only be open in append mode for writing. Only the superuser or a process possessing the CAP_LINUX_IMMUTABLE capability can set or clear this attribute. This is probably most effectively used on system log files, to prevent intruders removing evidence of their passage. Do be aware that in order for the intruder to have any chance of editing these log files, they need to have root access. With root access they could remove the 'a' attribute, make the edits and then re-establish the 'a' attribute.

'c' A file with the 'c' attribute set is automatically compressed on the disk by the kernel. A read from this file returns uncompressed data. A write to this file compresses data before storing them on the disk. NB Sadly, while the attribute is set on the file and is displayed by the lsattr command, it is not yet honoured by the ext2 or ext3 filesystem kernel drivers.

'D' When a directory with the 'D' attribute set is modified, the changes are written synchronously on the disk; this is equivalent to the 'dirsync' mount option applied to a subset of the files. When this attribute is in operation against a directory, the following operations are synchronous within that directory: create, link, unlink, symlink, mkdir, rmdir, mknod and rename.

'd' A file with the 'd' attribute set is not candidate for backup when the **dump** (8) program is run.

'i' A file with the 'i' (immutable) attribute cannot be modified: it cannot be deleted or renamed, no link can be created to this file and no data can be written to the file. Only the superuser or a process possessing the CAP_LINUX_IMMUTABLE capability can set or clear this attribute.

'j' A file with the 'j' attribute has all of its data written to the ext3 journal before being written to the file itself, if the filesystem is mounted with the "data=ordered" or "data=writeback" options. When the filesystem is mounted with the "data=journal" option all file data is already journalled and this attribute has no effect. Only the superuser or a process possessing the CAP_SYS_RESOURCE capability can set or clear this attribute.

's' When a file with the 's' attribute set is deleted, its blocks are zeroed and written back to the disk. NB As with the 'c' attribute, this attribute is honoured by everything except the kernel filesystem driver.

'S' When a file with the 'S' attribute set is modified, the changes are written synchronously on the disk; this is equivalent to the 'sync' mount option applied to a subset of the files. It is most often used for the 'cooked files' used by database programs to hold their data. When used in this way the addition of two different caching systems together is avoided. The caching system of the database, which is optimised for that systems use of data, is allowed to write direct to disk.

'T' The 'T' attribute is only usable when using the 2.6.x kernel. The 'T' attribute is designed to indicate the top of directory hierarchies, this is designed for use by the Orlov block allocator. The newer file allocation policies of the ext2 and ext3 filesystems place subdirectories closer together allowing faster use of a directory tree if that directory tree was created with a 2.6 kernel.

't' A file with the 't' attribute will not have a partial block fragment at the end of the file merged with other files (for those filesystems which support tail-merging). This is necessary for applications such as LILO which read the filesystems directly, and which don't understand tail-merged files. Note: As of this writing, the ext2 or ext3 filesystems do not (yet, except in very experimental patches) support tail-merging.

'u' When a file with the 'u' attribute set is deleted, its contents are saved. This allows the user to ask for its undeletion. This is another attribute that is supported by everything except the kernel itself.

Example:

```
# lsattr testfile
------------- testfile

# chattr +i testfile
# lsattr testfile
----i-------- testfile
# rm -f testfile
rm: cannot remove `testfile': Operation not permitted

# chattr -i testfile
# rm -f testfile
# ls testfile
ls: testfile: No such file or directory
```

5. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. The **/usr** directory must always be on the root device since it contains essential tools needed at boot time _____

2. If a user's home directory is anywhere else than in the **/home** directory then some operations may not work _____

3. Making a filesystem on a partition always deletes all the data on that partition _____

Glossary

Term	Description
base directories	subdirectories that are directly under the root directory
data block	block used to store data on a filesystem. Data blocks are referenced by inodes
essential root (/) subdirectories	term used in this manual to identify directories that must be present at boot time once the root filesystem has been mounted
ext2	the second extended filesytem type was adopted as the generic Linux filesystem
ext3	the third extended filesystem supports the same features as ext2 with an added journalling system
file permissions	attributes stored inside a file's inode giving simple read,write and execute permissions to the file owner (u), the group (g) and any other (o) user as well as more advanced permissions such as SUID, SGID and the 'sticky bit'
filesystem hierarchy standard (FHS)	"This standard consists of a set of requirements and guidelines for file and directory placement under UNIX-like operating systems. The guidelines are intended to support interoperability of applications, system administration tools, development tools, and scripts as well as greater uniformity of documentation for these systems" – see http://www.pathname.com/fhs/
inode	block used to store information about files, directories and symbolic links. Information includes the location of the file's data block, file permissions, time stamps and file type (e.g directory, file or symlink)
Orlov block allocator	Scheme that increases the performance of an EXT2/EXT3 filesystem. It is only available for 2.6 kernels. It was ported from BSD and improved by Alexander Viro, Andrew Morton, and Ted Ts'o. The original author is Grigory Orlov. This scheme can be switched on and of using the **chattr** command under Linux
superblock	is read when the filesystem is mounted, contains information such as the block-size used for the current filesystem (default 1024), the number of free inodes, the mount count and maximum mount count (used to determine if a full filesystem check should be performed)

Files

File	Description
/etc/mtab	file used by mount to keep track of currently mounted devices
/proc/mounts	file used by the kernel to keep track of currently mounted devices

Commands

Command	Description (apropos)
chattr	changes file attributes on a ext2/3 filesystem
chgrp	chgrp - change the group ownership of a file
chmod	chmod – change file access permission
chown	chown(1) – change the user and group ownership of a file
df	df - report filesystem disk space usage
du	du - estimate file space usage
e2fsck	e2fsck - check and repair a Linux second extended file system
fsck	check and repair any file system
lsattr	lists the file attributes on a ext2/3 filesystem
mke2fs	mke2fs – create an ext2/3 filesystem
mkfs	mkfs – build a Linux file system
mount	mount(1) – All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree. Conversely, the umount(8) command will detach it again
umask	tool used to set the file-creation mask mode – see **help umask**

Filesystem

1. Create 2 new partitions (larger than 50M) on the **/dev/hda** device using **fdisk**.
   HINT: To create a new partition type **n**. The partition type defaults to **83** (Linux)
   To write the new partition table type **w**.
   The partition table needs to be read: REBOOT the computer !

2. Format the first partition using the **ext2** filesystem type and the second with **reiserfs**.
   HINT: The **mkfs** tool is a front for **mkfs.ext2** or **mkfs.reiserfs**, etc. The syntax is
   ```
   mkfs -t <fstype> <device>
   ```

3. Make directories in **/mnt** and mount the new partitions

   ```
   mkdir /mnt/ext2
   mkdir /mnt/reiserfs
   ```

4. Check the status of your system:

   Use **mount** to verify which devices are mounted. The permissions set in **fstab** are visible too.

Use **df** to check the total number of blocks used. The **–k** option will convert the number of blocks in kilobytes (the default block size for **ext2**)

Run **fsck** on one of the newly created filesystems. The **fsck** utility is a front for **fsck.ext2**, **fsck.ext3**, **fsck.reiserfs**, etc. The syntax is:

```
fsck  <device>
```

5. Going further: Changing from **ext2** to **ext3** :
Update the device mounted on **/mnt/ext2** to **ext3** with tune2fs. This will add a journal to the existing filsystem. Make sure to make the relevant change for the filesystem type in **/etc/fstab**

```
tune2fs -j /dev/hdaN
```

At this stage the system has added a **journal** to the **/dev/hda10** partition, making it an **ext3** formated partition. This process is non-destructive and reversible. If you mount an **ext3** as an **ext2** filesystem, the **.journal** file will be erased. You can add it again with **tune2fs** …

File permissions

1. Login as a user (non root). Create a file using **touch** and verify that it has the effective permission 664.

2. Change the **umask** to 027. If you create a new file what is it's effective permission? _____

Where is the value of **umask** set? Depening the systems this can be **/etc/profile** or **/etc/bashrc**

3. Add 2 users to your system.

```
useradd user1
useradd user2
```

Add passords with `passwd user1` and `passwd user2`

4. Create a group called **sales**.

```
groupadd sales
```

5. Add the users to the group **sales**

```
gpasswd -a user1 sales
gpasswd -a user2 sales
```

6. Create a directory **/news** owned by the group **sales** and read-writable for this group.

```
mkdir -m 770 /news ; chown .sales /news
```

7. Set the GID to the **/news** directory.

```
chmod g+s /news
```

What are the symbolic permissions (eg. -rwxr_xr_x) on **/news**? [use **`ls -ld /news`**] _____

Verify that a group member doesn't need to type "`newgrp sales`" in order to create files with the right permissions. Can members of the group **sales** modify any files in this directory?

8. Add the *sticky-bit* permission on the **/news** directory. Verify that only user-owners can modify the files in the that directory. What are the permissions like on **/news**? _____

10. As root set SUID root **xeyes**. Login as a non root user. Check that this binary runs with UI root.

```
chmod u+s `which xeyes`
```

Log in as another user and run xeyes. Then do:

```
ps aux | grep xeyes
```

(the binary should be running as root)

The Command Line

Prerequisites

None

Goals

Introduce the **bash** shell and basic concepts such as interactively starting an executable
Distinguish variables defined as local or global (exported)
Manipulate data streams using pipes and other redirection operators
Understand meta-characters used for "file globbing"

Contents

Overview

A basic way to interact with a computer system is to use the command line. The shell interprets the instructions typed in at the keyboard. The shell prompt (ending with $ or # for user root) indicates that it is ready for user input.

The shell is also a programming environment which can be used to perform automated tasks. Shell programs are called scripts.

Most Common shells	
The Bourne shell	/bin/sh
The Bourne again shell	/bin/bash
The Korn shell	/bin/ksh
The C shell	/bin/csh
Tom's C shell	/bin/tcsh

Since the bash shell is one of the most widely used shells in the Linux world the LPI concentrates mainly on this shell.

1. The interactive shell

Shell commands are often of the form
`command [options] {arguments}.`

- **Printing text to the screen**

The the bash shell uses the **echo** command to print text to the screen.

```
echo "this is a short line"
```

- **Full/Relative path**

The shell interprets the first ¨word¨ of any string given on the command line as a command. If the string is a **full or relative path** to an executable then the executable is started. If the first word has no ¨/¨ characters, then the shell will scan directories defined in the PATH variable and attempt to run the first command matching the string.

For example if the PATH variable only contains the directories **/bin** and **/usr/bin** then the string **xeyes** won't be found since it is stored in **/usr/X11R6/bin/xeyes** so the full path needs to be run

```
/usr/X11R6/bin/xeyes
```

An alternative to typing the full path to an executable is to use a **relative** path. For example, if the user is in the directory where the **xeyes** program is stored then one can type

```
./xeyes
```

2. Variables

Shell variables are similar to variables used in any computing language. Variable names are limited to alphanumeric characters. For example CREDIT=300 simply assigns the value 300 to the variable named CREDIT.

1. initialise a variable:	Variable-Name=value (no spaces !!)
2. reference a variable:	$Variable-Name

```
CREDIT=300
echo $CREDIT
```

The value of a variable can be removed with the **unset** command.

Export, Set and Env:

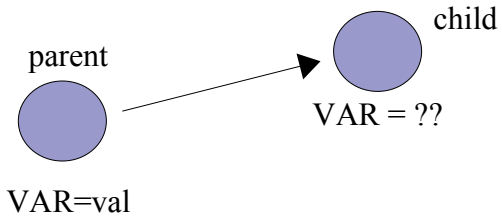
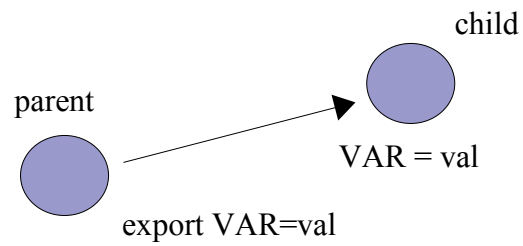
There are two types of variable: local and exported.

Local variables will be accessible only to the current shell. On the other hand, exported variables are accessible by both the shell and any child process started from that shell.

The commands **set** and **env** are used to list defined variables

The set and env commands	
set	Lists all variables
env	Lists all exported variables

A global variable is global in the sense that any child process can reference it.

LOCAL	***EXPORTED***
	

Example: Make the CREDIT variable a global variable. Test whether it's listed with **set** or **env**.

```
export CREDIT
env | grep CREDIT
```

Start a new shell (child process) and verify that CREDIT is accessible. Can one start any shell and be sure that CREDIT is still declared ?

List of common predefined variables

PREDEFINED VARIABLES	**MEANING**
DISPLAY	Used by X to identify where to run a client application
HISTFILE	Path to the users .bash_history file
HOME	The path to the user's home
LOGNAME	The name used by the user to log in
PATH	List of directories searched by the shell for programs to be executed when a command is entered without a path.
PWD	The current working directory
SHELL	The shell used (bash in most Linux distributions)
TERM	The current terminal emulation

Special variables

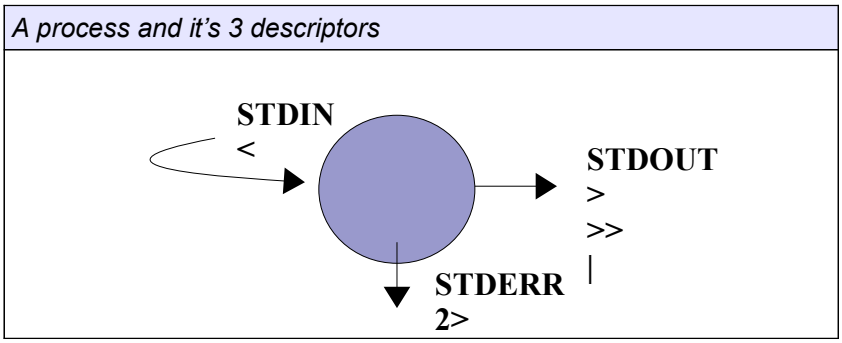
The next few variables are related to process management.

$!	represents the PID value of the last child process
$$	represents the PID of the running shell
$?	is 0 if the last command was executed successfully and 1 otherwise

3. Input, Output, Redirection

UNIX processes normally open three standard file descriptors which enable it to process input and output. These standard descriptors can be redefined for any given process. In most cases the **stdin** descriptor is the keyboard, and the two output descriptors, **stdout** and **stderr**, is the screen.

A process and it's 3 descriptors



Numerical values for stdin, stderr and stdout	
stdin	0
stdout	1
stderr	2

stdout redirection

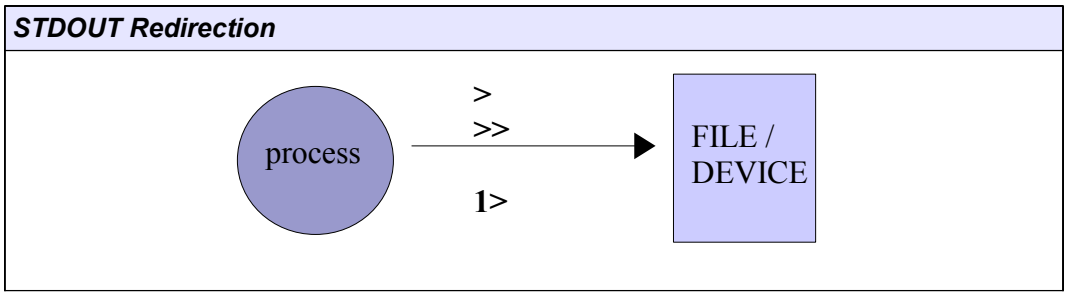
program > file

The data flows from left to right.

```
fdisk -l > partions.txt
```

This will run the **fdisk** utility and output the result to the *partitions.txt* file. No output is visible. Also notice that the shell will read this line from the right. As a result, the *partitions.txt* file will be created first if it doesn't exist and overwritten if the '**>**' operator is used.

The '**>>**' operator will append standard output to a file.



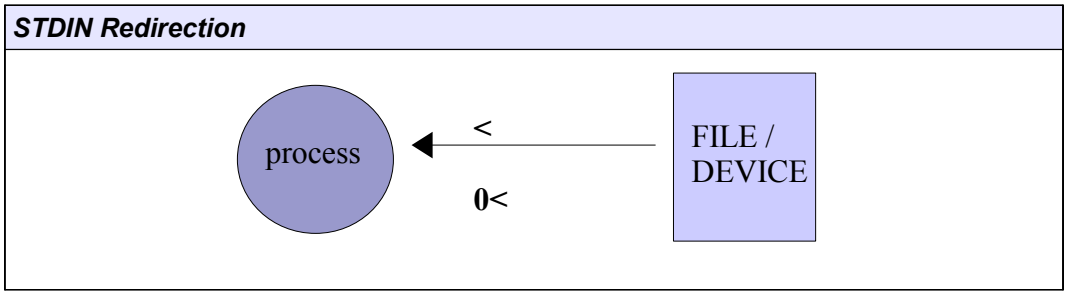
stdin redirection

program < file

In this case data flows from right to left. The '**<**' operator is only used for **stdin** and cannot be used for **stdout**.

If the file *instuctions* contains on each line the letters *p*, *m*, and *q* then the next example would cause **fdisk** to print the partition table of */dev/hda*, print the utility's help screen and finally quit:

```
fdisk /dev/hda  < instructions
```

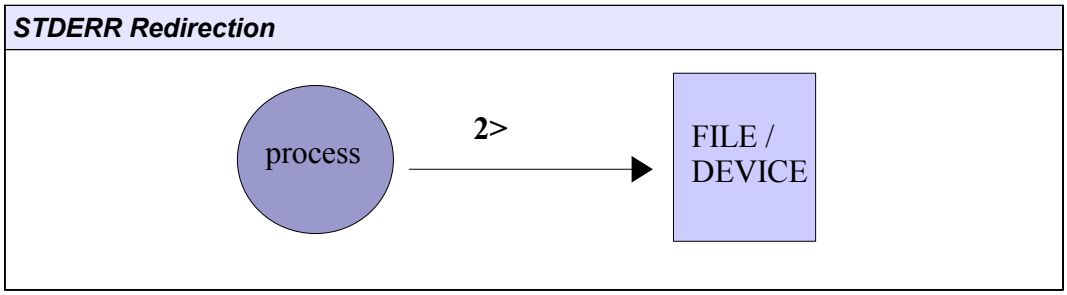


stderr redirection

program 2> errorfile

stdin, stdout and stderr are represented by 0, 1 and 2 respectively. This allows one to select the **stderr** stream:

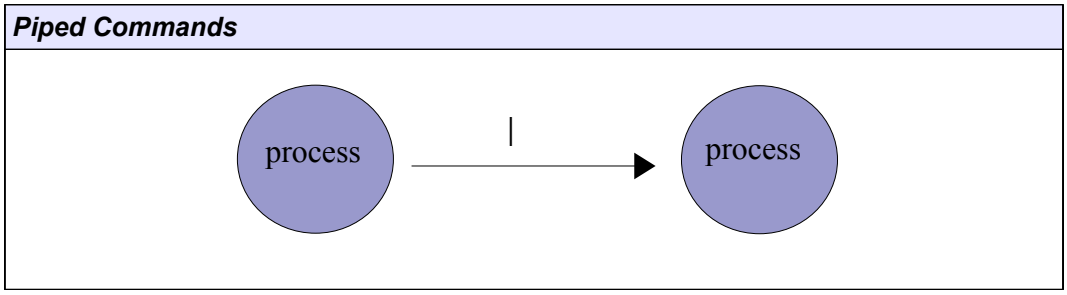
```
find / 2> /dev/null
```



piped commands

program1* | *program2

Pipes are represented by the "|" symbol. The data stream goes from the left to the right. The next figure illustrates how the **stdout** for one process is redirected to the **stdin** for another process.




```
cat /var/log/messages | less
```

NB Multiple output redirects are parsed from right to left, so the following commands are not equivalent.

```
Do-command  2>&1  >logfile
Do-command  >logfile  2>&1
```

The tee Command

command **|** **tee** *FILENAME*

This command is used after a pipe and takes a filename as an argument. The standard output from the previous command is then sent to the file given as an argument but **tee** also lets the stream through to **stdout**. The **stdout** has been duplicated in this way.

4. Metacharacters and Quotes

Metacharacters are characters that have special meaning for the shell. They are mainly used for *file globbing*, that is to match several files or directory names using a minimum of letters.

The input (<), output (>) and pipe (|) characters are also special characters as well as the dollar ($) sign used for variables. We will not list them here but note that these characters are seldom used to name regular files.

Wildcards

● The ***** wildcard can replace any number of characters.

```
ls  /usr/bin/b*          lists all programs starting with a 'b'
```

● The **?** wildcard replaces any one character.

```
ls  /usr/bin/?b*         lists all programs having a 'b' as the second letter
```

● **[ ]** is used to define a range of value.

```
ls  a[0-9]       lists all files starting with an 'a' and have a digit in second position. Also
ls  [!Aa]*       lists all files that don't start with an 'a' or an 'A'
```

● **{***string1,string2***};** although not just a file naming wildcard, it can be used to match the names of existing files.

```
ls  index.{htm,html}
```

Quotes and escape codes

The special meaning of metacharacters can be cancelled by *escape characters*, which are also metacharacters.

The backslash (\) is called the **escape character and** cancels the meaning of all metacharacters forcing the shell to interpret them literally.

The single quotes (' ') cancel the meaning of all metacharacters except the backslash.

The double quotes (" ") are the weakest quotes but cancel most of the special meaning of the enclosed characters except the pipe (|), the backslash (****) and a variable ($var).

The back tick

Back quotes `` will execute a command enclosed. The next example defines the variable TIME using the **date** command.

```
TIME="Today's date is `date +%a:%d:%b`"
echo $TIME
Today's date is Sun:15:Jul
```

Another way of executing commands (similar to the back ticks) is to use **$()**. This will execute the enclosed command and treat it as a variable.

```
TIME=$(date)
```

5. The Command History

To view the list of previously typed commands you can use the **bash** built-in command **history**.

```
history
1       ls
2       grep   500 /etc/passwd
```

This has listed all the cached commands as well as the commands save in **~/.bash_history**. When a user exits the shell cached commands are saved to **~/.bash_history**.

You can recall commands by using the Up-arrow and Down-arrow on your keyboard. There are also emacs key bindings that enable you to execute and even edit these lines.

Emacs Key Bindings for Editing the Command History	
Ctrl+P	Previous line (same as Up-arrow)
Ctrl+n	Next line (same as Down-arrow)
Ctrl+b	Go back one character on the line (same as Left-Arrow)
Ctrl+f	Go forward one character on the line (Same as Right-Arrow)
Ctrl+a	Go to the beginning of the line (Same as <Home>)
Ctrl+e	Go to the end of the line (Same as <End>)

The bang **!** key can be used to rerun a command.

Example

!x	executes the latest command in the history list starting with an 'x'
!2	runs command number 2 from the **history** output
!-2	runs the command before last
!!	runs the last command
^string1^string2	run previous command and replace string1 by string2

6. Other Commands

Aliases

You can create aliases for commands needing many arguments. The format to create an alias is

```
alias myprog='command [options]{arguments}'
```

By typing **alias** alone at the command line you will get a list of currently defined aliases.

Command completion

By pressing **TAB,** the shell will complete the commands you have started typing in.

<< is a redirection for EOF

For example

```
cat << stop
```

will accept standard input until the keyword 'stop' is entered.

Compound commands

command1; command2; command3	The three commands are run in sequence regardless of the success of the previous command
command1 && command2 && command3	Each command will execute only if the previous exit code is 0 (success)
command1 || comand2 || command3	The next command will execute only if the previous exit code is not 0 (failure)

The "exec' command

This command is not a binary but rather is part of the shell. It is used to start other commands. Ordinarily if a command is executed, a sub-process is started. If the exec command is used to initiate the new program, it reoccupies the process used to start it. It replaces the current shell (in a script or the interactive shell).

When the new command terminates, control is not passed back to the calling shell, but returns to the process that called the shell used to make the exec call.

```
echo $$
414

$ bash
$ echo $$
455

$ echo hello
hello
$ echo $$
455

$ exec echo hello
hello
$ echo $$
414
```

The above shows control falling back to the second shell (process 455) after a straight forward echo and the first shell (process 414) using an exec.

Manpages and the whatis database

The manpages are organised in sections	
NAME	the name of the item followed by a short one line description.
SYNOPSYS	the syntax for the command
DESCRIPTION	a longer description
OPTIONS	a review of all possible options and their function
FILES	files that are related to the current item (configuration files etc)
SEE ALSO	other manpages related to the current topic

These are the main sections one can expect to find in a manpage.

The **whatis** database stores the NAME section of all the manpages on the system. This is done through a daily **cron**. The **whatis** database has the following two entries:

```
name(key)  -  one line description
```

The syntax for **whatis** is:
`whatis <string>`

The output is the full NAME section of the manpages where *string* matched *named(key)*

One can also use the **man** command to query the whatis database. The syntax is

`man -k <string>`

This command is similar to **apropos**. Unlike whatis this will query both the “name” and the “one line description” entries of the database. If the string matches a word in any of these fields the above query will return the full NAME section.

Example: (the matching string has been highlighted)

```
whatis lilo
lilo                    (8)   - install boot loader
lilo.conf [lilo]        (5)   - configuration file for lilo
```

```
man -k  lilo
grubby            (8)   - command line tool for configuring grub, lilo, and
elilo
lilo              (8)   - install boot loader
lilo.conf [lilo]  (5)   - configuration file for lilo
```

The FHS recommends manpages to be kept in **/usr/share/man**. However additional locations can be searched using the MANPATH environment variable set in **/etc/man.config**. Each directory is further divided into subdirectories corresponding to manpage sections.

Manpage Sections	
Section 1	Information on executables
Section 2	System calls, e.g mkdir(2)
Section 3	Library calls, e.g stdio(3)
Section 4	Devices (files in /dev)
Section 5	Configuration files and formats
Section 6	Games
Section 7	Macro packages
Section 8	Administration commands
Section 9	Kernel routines

To access a specific section *N* one has to enter:

man *N* command

Examples:

```
man mkdir
man 2 mkdir
```

```
man crontab
man 5 crontab
```

7. Exercise and Summary

Review Questions (answers p.150)

Yes or No

1. If the PATH variable isn't properly set then programs can be started only if users type in the executable's full or relative path _____

2. The STDOUT from a process can be piped into a file _____

3. Once a data stream has gone through a pipe that stream is generally no longer visible on STDOUT _____

4. All the commands entered at the shell are stored in a mysql database _____

Glossary

Term	Description
compound commands	several commands given in a single line at the shell using delimiters. Depending on the delimiter the shell will execute the commands differently (p.64)
metacharacters	character that is not interpreted literally by the shell but has added meaning
command substitution	use the output of a command as a variable. This is done by enclosing the command in back ticks `` or round brackets $() For example `ls /home/$(whoami)` will list the current user's home directory
file globbing	term used when handling multiple file names using wild cards. The name comes from the **glob** sub-program in old UNIX shells used to expand wild cards given on the command line
redirection and pipes	operations that manipulate data streams and the file descriptors of a process. A redirection involves a process and a file, whereas a pipe will involve only processes
stderr, stdin, stdout	names of the file descriptors available for any process to stream error messages, read input streams and write output (non-error) streams
wild cards	the following `*`, `?` , "`{}`" or "`[ ]`" metacharacters used to match more than one character when working on the command line

Commands

Command	Description (or *apropos*)
alias	set an alias for a single or a sequence of commands (see **man builtins** or **help alias**)
echo	print text to STDOUT
env	list variables that have been exported (see **man builtins** or **help env**)
exec	Shell built-in used to execute new programs but instead of starting a new sub-process it replaces the calling process
export	export the value of a variable to the environment of subsequently executed commands (see **man builtins** or **help export**)
history	display the command history list with line numbers (see **man builtins** or **help history**)
tee	**tee(1)** – read from standard input and write to (both) standard output and files
set	list all variables in the environment of the current process (see also **man builtins** and **help set**)

Exercises

WARNING: You will need the **uuencode** and **uudecode** commands in the exercises. These commands are provided by the **sharutils** package.

Stdin-stdout-stderr

Type the next commands and represent the sequence of execution (if possible) using diagrams similar to the ones used in this chapter.

```
ls /etc ; df > /tmp/out.1
(ls /etc ; df) > /tmp/out.2

find /etc -type f  2> /dev/null | sort

tr [a-z] [A-Z] < /etc/passwd | sort > /tmp/passwd.tmp

cat /tmp/passwd.tmp | tr [A-Z] [a-z]
```

Command Line

1. List all files in /usr/X11R6/bin that don't start with an x

```
ls /usr/X11R6/bin/[!x]*
```

2. The command **xterm** has the following options:

-bg <color> set background
-fg <color> set foreground
-e <command> execute 'command' in terminal

Set a new alias such that the **su** command opens a new color xterm and prompts for a root password.

```
alias su="xterm -bg orange -fg brown -e su - &"
```

Where would you store this alias on the system? _____

3. You can encode files using **uuencode**. The encoded file is redirected to stdout.
For example: *uuencode /bin/bash super-shell > uufile* encodes **/bin/bash** and will produce a file called **super-shell** when running **uudecode** against the *uufile*

- Mail the uuencoded /bin/bash to a local user (for this you can either use **uuencode** and a pipe **|** , or save the uuencoded output to a file *uufile* and use STDIN redirection **<**).
- Split the uuencoded file into 5 files:

```
uuencode /bin/bash super-shell > uufile
split –b 150000 uufile  base-name.
```

This will create files called `base-name.aa, base-name.ab,` etc

To get a uuencoded file with all the original data (unsplit) do

```
cat base-name.* > uufile.new
```

Finally uudecode the file and check it still works.

```
uudecode   uufile.new
```

This should create a binary file called **super-shell**

3. Which tool finds the full path to a binary by scanning the PATH variable? _____

Variables

1. Do the following

Assign the value ‘virus’ to the variable ALERT.

```
ALERT=virus
```

Verify that it is defined using the **set** command:

```
set |grep ALERT
```

Is ALERT listed when using **env** instead of **set**?

Next type ‘bash’. Can you access the ALERT variable?

```
bash
echo $ALERT
```

NOTE the value of ALERT: _____ (is it blank?)

Type exit (or ^D) to return to your original session.

Use the **export** command to make ALERT a global variable.

```
export ALERT
```

Verify that it is a global (env) variable

```
env | grep ALERT
```

(v) Start a new **bash** shell and make sure that ALERT is defined in the new shell:

```
bash
echo $ALERT
```

In this new shell, redefine the variable ALERT

```
export ALERT=green
```

Exit that shell. What is the value of ALERT in the original shell? _____

2. At the command prompt type the following lines:

```
CREDIT01=300;CREDIT02=400
for VAR in CREDIT01 CREDIT02;do echo $VAR;done
```

Notice that the variable VAR is referenced with $VAR.

(i) Rerun this command.

(ii) Rerun this command replacing CREDIT01 by $CREDIT01

3. Using appropriate quotes change your PS1 variable to include the full path to your working directory. (Hint: the value of PS1 is [\u@ \W]\$, you only need to replace the \W by a \w)

```
PS1='[\u@\h \w ]\$ '
```

What does PS2 look like? _____

File Management

Prerequisites

The Command Line (see p.56)
Understand the EXT2 file system (see p.42)

Goals

Effectively move around the filesystem to create, delete and find files or directories
Distinguish between hard and symbolic links

Contents

1. Moving around the filesystem

Absolute and relative paths

A directory or a file can be accessed by giving its full pathname, starting at the root (/) or its relative path, starting from the current directory.

Absolute path: independent of the user's current directory
starts with /
Relative path: depends on where the user is
doesn't start with /

As in any structured filesystem there are a number of utilities that can help you navigate through the system. The next two commands are built-in commands.

pwd: Gives your actual position as an absolute path.
cd: The 'change directory' command

2. Finding Files and Directories

We will describe the **find**, **which**, **whereis** and **locate** utilities.

- **find**

Syntax:
`find <DIRECTORY> <CRITERIA> [-exec <COMMAND> {} \;]`

The *DIRECTORY* argument tells **find** where to start searching and *CRITERIA* can be the name of a file or directory we are looking for.

Examples:

```
find /usr/X11R6/bin -name  ¨x*¨.
find / -user 502
```

Matching lines are listed to standard out. This output can be acted upon. For example delete the file, or change the permission. The **find** tool has the build-in option **–exec** which allows you to do that. For example, remove all files belonging to user 502:

```
find / -type f -user 502 -exec rm -f {} \;
```

- **xargs**

This tool is often thought of as a companion tool to **find**. In fact **xargs** will process each line of standard output as an *argument* for another tool. We could use **xargs** to delete all files belonging to a user with:

```
find / -type f -user 502 | xargs rm -f
```

Certain commands such as **rm** cannot deal with too long arguments. It is sometimes necessary to delete all files in a directory with

```
ls |xargs rm -f
```

Common criteria switches for **find**	
-type	specify the type of file
-name	name of the file
-user	user owner
-atime, ctime, mtime	access, creation and modified times (multiples of 24 hrs)
-amin, cmin, mmin	access, creation and modified times (multiples of 1 min)
-newer *FILE*	files newer than *FILE*

- **locate**

Syntax:
`locate <STRING>`

When using **locate** all files and directories that match the *expression* are listed.

```
locate X11R
```

The search is much faster. In fact **locate** queries the **/var/lib/slocate/slocate.db** database. This database is kept up to date via a daily cron job which runs **updatedb**.

When running **updatedb** from the command line the **/etc/updatedb.conf** file is read to determine pruned files systems (e.g NFS) and directories (e.g /tmp)

- **which**

Syntax:
`which string`

This tool will return the full path to the file called **string** by scanning the directories defined in the user's PATH variable only. As a result **which** is only used to find commands.

- **whereis**

Syntax
whereis string

This tool will return the full path to source or binaries as well as documentation files matching **string** by scanning the PATH variable as well as a number of well known locations

Getting the most from ls

Most common options for ls	
-I	show inode
-h	print human readable sizes
-n	list UIDs and GIDs
-p	append descriptor (/=@) to list
-R	recursively display content of directories
-S	sort by file size
-t	sort by modification time (similar to -c)
-u	show last access time

3. Handling directories

Making a directory with mkdir:
When making a directory you can set the permission mode with the **-m** option. Another useful option is **-p** which creates all subdirectories automatically as needed.

Example:

```
mkdir -p docs/programs/versions
```

Removing directories:
To remove a directory use either **rmdir** or **rm -r**. If you are root you may have to specify **-f** to force the deletion of all files.

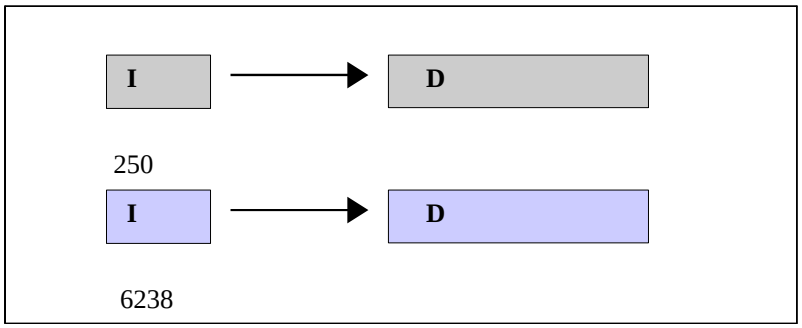
Notice: `rm -rf /dir1/*` removes all files and subdirectories leaving dir1 empty
`rm -rf /dir1/` removes all files and subdirectories including dir1

4. Using cp and mv

cp
Syntax:
`cp [options] file1 file2`
`cp [options] files directory`

It is important to notice that **cp** *file1 file2* makes a new copy of *file1* and leaves *file1* unchanged.

Fig: file1 with inode 250 is copied to file2, duplicating the data to a new data area and creating a new inode 6238 for file2



You can also copy several files to a directory, using a list or wildcards. The following table lists the most used options.

Most common options for cp	
-d	do not follow symbolic link (when used with -R)
-f	force
-I	interactive, prompt before overwrite
-p	preserve file attributes

-R	recursively copy directories

Note: `cp -r /dir/* /dir2/` will copy all files and subdirectories omitting mydir
`cp -r /mydir/ /dir2/` will copy all files and subdirectories including mydir

mv
Syntax:
```
mv [options] oldname newname
mv [options] source destination
mv [options] source directory
```

The **mv** command can both *move* and *rename* files and directories. If *oldname* is a file and *newname* is a directory then the file *oldname* is moved to that directory.

If the source and destination are on the same filesystem, then the file isn't copied but the inode information is updated to specify the new location. Most common options are **-f** force overwrite and **-i** query interactively.

5. Hard Links and Symbolic Links

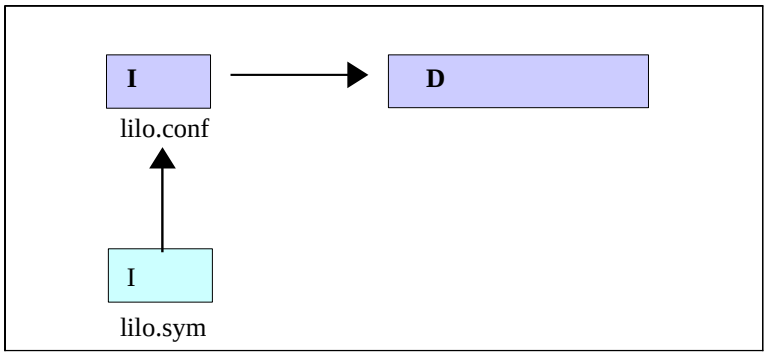
Symbolic links

A soft link to a file or a directory creates a new inode that points to the same data area:

```
ln -s lilo.conf lilo.sym
```

This is the listing for these files. Notice that the reference count is **1** for both files.
```
-rw-------   1  root  root  223 Nov  9 09:06   lilo.conf
lrwxrwxrwx   1  root  root  9 Nov  9 09:06  lilo.sym -> lilo.conf
```

Fig2: A soft link to a file



Soft links can be created across filesystems.

Hard Links

A hard link is an additional name for the same inode and as such the reference count of the file increases by one for every new hard link.

```
ln lilo.conf lilo.link
```

In the listing notice that the reference count is **2** and that both files have the same size. In fact they are identical.

```
-rw-------  2 root  root  223  Nov  9 09:06 lilo.conf
-rw------- 2 root  root  223  Nov  9 09:06 lilo.link
```

Hard links can only be created within the same filesystem.

7. Touching and dd-ing

touch

Another way of creating or modifying a file is to use **touch**.

Syntax: touch {options} *file(s)*

If *file* doesn't exist it is created. You can also change the access time of a file using the **-a** option, **-m** changes the modification time and **-r** is used to apply the time attributes of another file.

Example:

```
touch file1.txt file2.txt          creates new files
touch myfile -r /etc/lilo.conf     myfile gets the time attributes of lilo.conf
```

☝ To create a file called ***–errors*** use the – option:

```
touch -- -errors
```

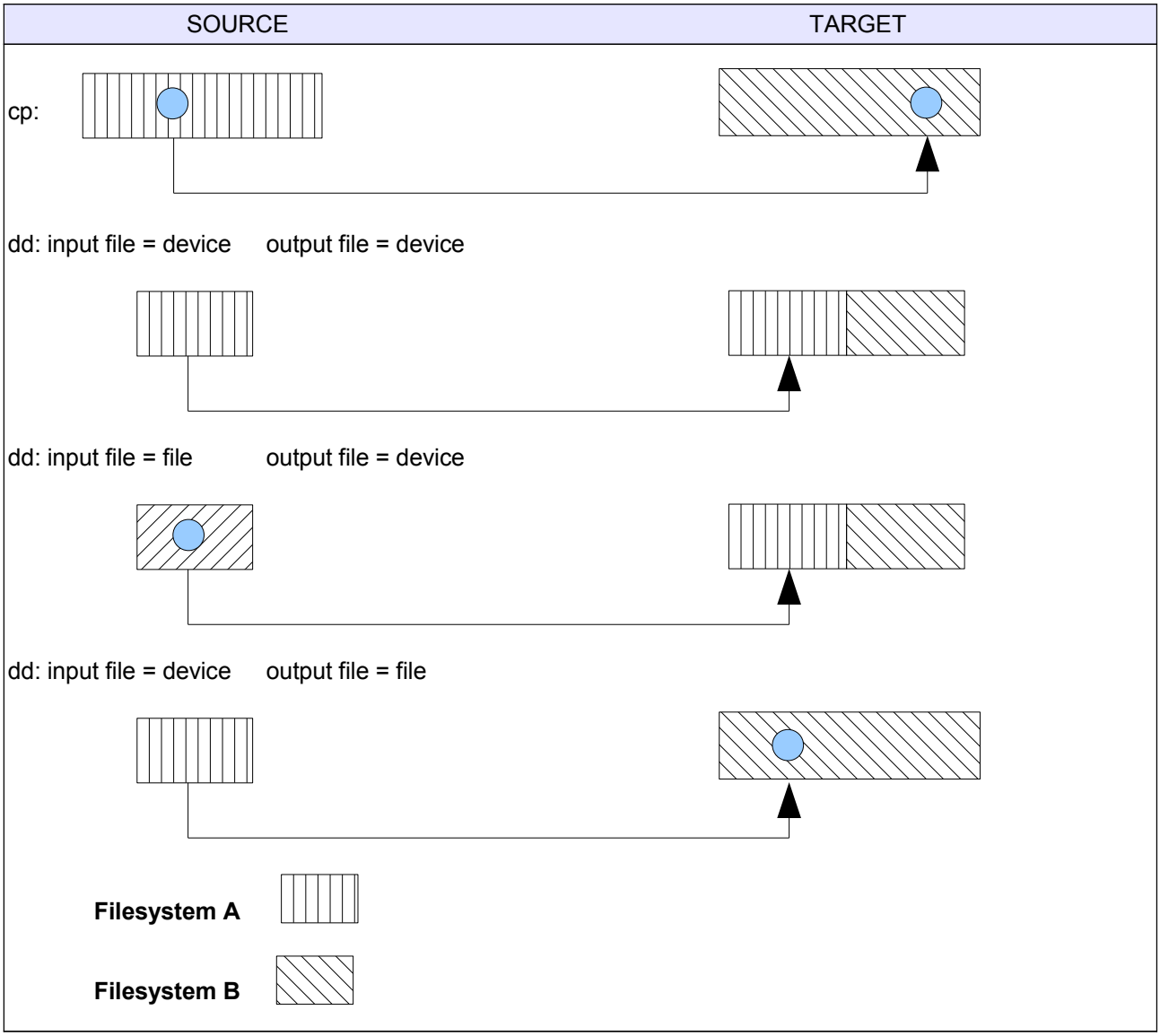
dd

This command copies a file with a changeable I/O block size. It can also be used to perform conversions (similar to **tr**). Main options are **if=** (input file) **of=** (output file) **conv=** (conversion)
The conversion switch can be: lcase ucase ascii

Example:

```
dd if=/mnt/cdrom/images/boot.img of=/dev/fd0
```

Notice that unlike **cp** the **dd** tool will copy portions of a device and preserve the underlying filesystem. On the other hand **cp** only deals with the data and will transfer it from one filesystem to another:



8. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. The **cd –** command will take you back to a previous directory? _____

2. Typing **cd ~** (in the bash shell) is the shortest command that will take you to your home directory ? _____

3. One can make two new directories /dir1/dir2 using **mkdir** without any options _____

4. The command **updatedb** will update the **locate** database _____

5. The syntax to create a symbolic link called FILE-LINK pointing to the file FILE is
```
ln -s FILE-LINK   FILE
```
_____

6. The commands **cd /etc** and **cd ./etc** are always equivalent _____

Files

File	Description
/etc/updatedb.conf	configuration file for the **updatedb** tool
/var/lib/slocate/slocate.db	the locate (or slocate 'secure locate') database

Commands

Command	Description (apropos)
cd	change current directory – see `help cd`
cp	**cp(1)** – copy files and directories
dd	copy and convert files. Often used to copy the content of a disk device to another device or file
find	**find(1)** – search for files in a directory hierarchy
ln	**ln(1)** – make links between files
locate	command used to search files and directories in the locate database
ls	**ls(1)** – list directory contents
mkdir	**mkdir(1)** – make directories
mv	**mv(1)** – move (rename) files
pwd	**pwd(1)** – print name of current/working directory
rm	**rm(1)** – remove files or directories
touch	create new empty file or change file timestamps
updatedb	command used to update the locate database
whereis	**whereis(1)** – locate the binary, source, and manual page files for a command
which	**which(1)** – shows the full path of (shell) commands

Exercises

File Navigation

1. Make a new directory in **/tmp** called **/etc**.

```
mkdir /tmp/etc
```

2. In **/tmp/etc/** create a file called *newfile* (use touch, cat or vi).

3. Go to the root directory (cd /).

4. Test which of the following commands will show the content of *newfile* ?

```
cat etc/newfile
cat /etc/newfile
cat tmp/etc/newfile
cat /tmp/etc/newfile
```

5. Remove the **/tmp/etc** directory with **rmdir**. Do step 1 again then remove **/tmp/etc** with **rm**

Making space on the filesystem

In order to create more space on the device containing the directory **/usr/share/doc** we need to find a spare device with enough space and copy the contents of **/usr/share/doc** to that device. Then we create space by deleting the **/usr/share/doc** directory and creating a symbolic link from **/usr/share/doc** to the new location.

6. Make a directory called **/spare** on which we will mount a spare devices (one of the partitions created in the previous exercises should be suitable)

```
mkdir /spare
mount <device> /spare
```

7. Test with **df -h /spare** and **du -hs /usr/share/doc** that the device is large enough to contain all of the existing data.

8. Next, copy the contents of **/usr/share/doc** to **/spare/**

```
cp -a /usr/share/doc /spare
```

9. Make sure the data has all been copied across then edit **/etc/fstab** to make that device available at boot time.

10. Delete **/usr/share/doc** and create a symbolic link pointing from **/usr/share/doc** to **/spare/doc**

```
ln -s /spare/doc /usr/share/doc
```

11. (optional) Do the same with **/home**. Any extra problems?

Finding Files on the System

12. Copy the file */etc/lilo.conf* to */etc/lilo.conf.bak*
    (i) Use **find** to find this new file
    (ii) Use **locate** to find */etc/lilo.conf.bak.*
    (iii) Update the locate database and retry (ii)

Backup strategy (first step)

Find all files in your home directory that have been modified in the past 24 hours.

`find /home -mtime -1 |tee list1 |wc --lines` (-1 means less than one day)

We will introduce archiving tools in LPI 102, but the output of the find command can be piped directly into **cpio**.

Process Management

Prerequisites

The Command Line (p.56)

Goals

Find the process ID (or PID) of a running process using different tools
Use **kill** and **killall** effectively with the appropriate signal
Manage jobs from the command line in the foreground or the background

Contents

1. Viewing running processes

Processes have a unique Process ID the **PID**. This number can be used to modify a process' priority or to stop it.

A process is any *running* executable. If process_2 has been spawned by process_1, it is called a *child* process. The spawning process_1 is called the *parent* process.

The process family tree

The **pstree** command gives a good illustration of *parent* and *child* process hierarchy.

Figure 1: Part of the pstree output

```
bash(1046)---xinit(1085)-+-X(1086)
                         `-xfwm(1094)-+-xfce(1100)---xterm(1111)---bash(1113)-+-pstree(1180)
                                      |                                       |-soffice.bin(1139)---soffice.bin(1152)-+
                                                                                                                      -soffice.bin(1153)
                                      |                                       |                                       |-soffice.bin(1154)
                                      |                                       |                                       |-soffice.bin(1155)
                                      |                                       |                                       |-soffice.bin(1156)
                                      |                                       |                                       `-soffice.bin(1157)
                                      |                                       `-xclock(1138)
                                      |-xfgnome(1109)
                                      |-xfpager(1108)
                                      |-xfsound(1107)
                                      `-xscreensaver(1098)
```

In the above figure all the process' PIDs are shown; these are clearly incremental. The most common used options are **-p** to display PIDs and **-h** to highlight a users processes only.

Finding running processes

A more direct way to determine which processes are running is to use **ps**. Most users have a set combination of options which work for most situations.

Here are three such options:

ps ux	all processes run by the user
ps T	processes run under the current terminal by the user
ps aux	all processes on the system

It is recommended you read the **ps manpage** and choose your own best options!

***ps** accommodates UNIX-style and BSD-style arguments*
usage: ps -[Unix98 options] ps [BSD-style options] ps --[GNU-style long options] ps –help for a command summary

Summary of options
-a show all processes for the current user linked to a tty (*except* the session leader) **-e** or **-A** show all processes **-f** gives the PPID (Parent Process ID) and the STIME (Start Time) **-l** is similar to **-f** and displays a long list **a** show all processes linked to a tty, including other users x show all processes without a controlling tty as well

Continuously updating process information

The **top** utility will update information on processes at an adjustable rate.

While **top** is running you can type **h** for a list of commands. The space bar will update information instantly.

You can also use **top** to change a process' priority as we shall see in the next section.

2. Modifying Processes

Stopping processes

The **kill** command can be used to send *signals* to processes. There are 63 signals available. The default signal terminates a process and is called SIGTERM with value 15.

kill
Syntax
kill SIGNAL process_PID

Every process can choose whether or not to catch a signal except for the SIGKILL which is dealt with by the kernel. Most daemons redifine the SIGHUP to mean “re-read configuration file”.

Most Common Signals

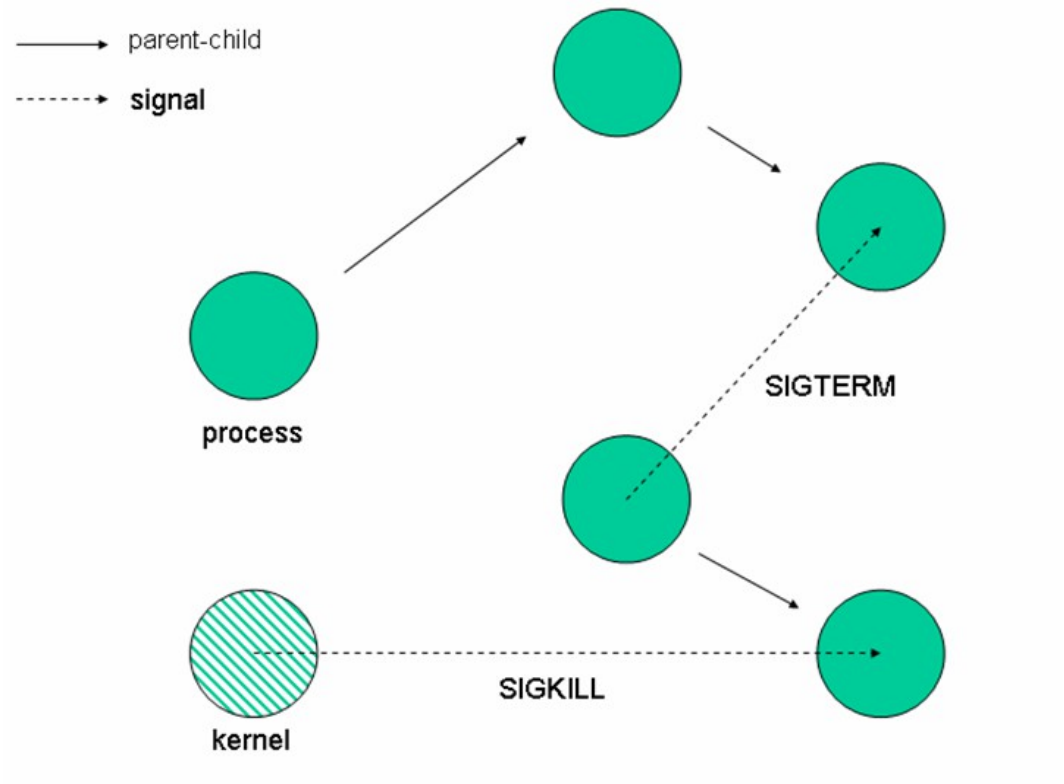
1 or SIGHUP hangup or disconnect the process
2 or SIGINT same as Ctrl+C interrupt
3 or SIGQUIT quit
9 or SIGKILL kill the process through a kernel call
15 or SIGTERM terminate a process 'nicely'. This is the DEFAULT signal.

One can also stop processes without knowing the process' PID using **killall**.

killall

Syntax
`killall SIGNAL process_NAME`

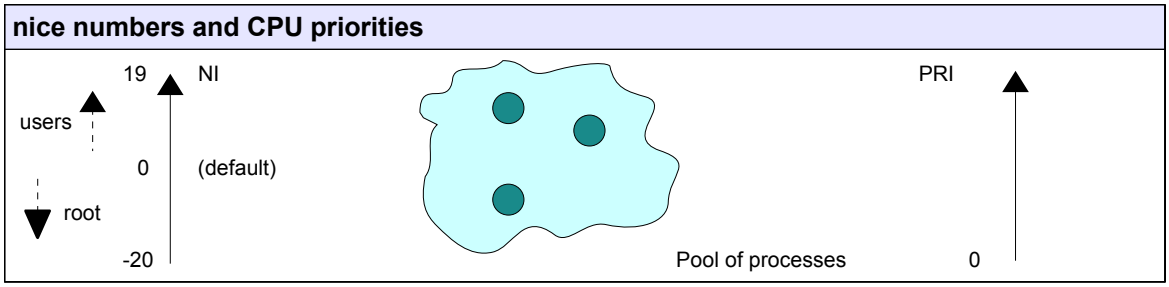
Fig1: Interprocess signaling



Process priority and nice numbers

Nice numbers (`NI`) alter the CPU priority and are used to balance the CPU load in a multiuser environment. Each process is started with a default nice number of **0**. Nice numbers range from **19** [lowest] to **-20** [highest].

Only root can decrease the nice number of a process. Since all processes start with a default nice number of zero as a consequence negative nice numbers can only be set by root!



To modify a process' priority that is already running use **renice**. To set a process' priority use **nice**.

Syntax

```
nice -<NI>  <process>
renice  <+/-NI>  -p <PID>
```

Notice that **renice** works with PIDs and handles lists of processes at a time. A useful option to **renice** is the **-u** option which affects all processes run by a user.

Set nice number 1 for processes 234 and 765:

```
renice +1  -p 234 765
```

Set nice number -5 for **xclock**:

```
nice  --5  xclock
```

3. Processes and the shell

background and forground processes

After you have started a process from the shell you automatically leave the shell interpreter. You will notice that no commands will respond. The reason for this is that it is possible to run programs in the *foreground* **fg** or in the *background* **bg** of a shell.

When a program is running in the foreground it is possible to recover the shell prompt but only by interrupting the program for while. The interruption signal is **Ctrl Z**.

Stopping and starting jobs

A process started from a shell is also called a *job*. Once the job receives the **^Z** signal it is stopped and the shell prompt is recovered. To restart the program in the background simple type: **bg**.

Example

```
[mike localhost  /bin]$xclock
                                        xclock running in forground, shell prompt lost
[1]+  Stopped                 xclock    xclock received ^Z signal
[mike localhost  /bin]$bg               shell prompt recovered, issue the bg command
[1]+ xclock &                           xclock is running in the background
[mike localhost  /bin]$
```

Notice the [1]+ symbol above. The integer is the process' *job number,* which it can be referred to as.
The '+' sign indicates the last modified process. A '-' sign would indicate the second last modified process.

One can start a process in the background by appending a **&** to the command.

```
xclock&
[1] 6213
```

Listing jobs

The **jobs** utility lists all running processes started from the current shell. The *job number*, the job's state (running/stopped), as well as the two last modified processes, will be listed.

Output for jobs	
[1]- Stopped	xclock
[2] Running	xman &
[3]+ Stopped	xload

The job number

One can conveniently stop and start a selection of jobs using the *job number*. This is achieved with the **fg** command.

Calling job 2 to the foreground and killing job 1

fg 2 or kill –9 %1

fg %2 or

fg %?xma

Avoiding HUP with nohup

Finally there is a program called **nohup** which acts as a parent process independently from the user's session. When a user logs off, the system sends a HUP to all processes owned by that process group. For example, to avoid this HUP signal a script called **bigbang** which attempts to calculate the age of the Universe should be started like this:

```
nohup bigbang &
```

4. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. Running **kill** against a process will always attempt to kill the given process _____
2. The commands '`kill $(pidof xeyes` ' and '`killall xeyes` ' are equivalent _____
3. A program started with a pending '&' will run in the background _____
4. A process's nice number is the same as its CPU priority _____

Glossary

Term	Description
background process	a process started on the shell with `$command &` Unlike a foreground process the shell doesn't need to wait for the process to terminate before running another command
foreground process	a process started on the shell with `$command` Once the process is started the shell has to wait for it to terminate before it can run another command
orphaned process	a process whose parent process has terminated. An orphaned process is then 'adopted' by **init**
PID	a number associated with a process
zombie process	a process that has exited but is still considered by the parent process as present until the next `wait()` system call. The `wait()` system call performed by the parent process should refresh the status of the child process as terminated if the process has exited. A zombie process usually doesn't last for long. However due to bugs some zombie processes can last longer taking up system resources even though the process itself has physically exited!

Commands

Command	Description (apropos)
bg	resume a suspended job in the background
Ctrl+Z	keyboard combination used to suspend the current foreground process
fg	send a job in the foreground (making it the current process)
jobs	list of processes started from the current shell
kill	send a specified signal to a process using PIDs
killall	send a specified signal to a process using process names
nice	starts a process with a modified scheduling priority

nohup	**nohup(1)** – run a command immune to hangups, with output to a non-tty
ps	**ps(1)** – gives a snapshot of the current processes. If you want a repetitive update of this status, use **top**
pstree	prints current processes in a hierarchical tree rooted (by default) at **init**
renice	modified scheduling priority of a running process
top	**top(1)** – display top CPU processes

Exercises

1. Check the current nice value of your running x-terminal. Change this value using **top** or **renice**.

2. What is the equivalent signal of a **^Z** sent to a process? (List all signals with **kill –l**)

3. Which signal is redefined for most daemons and forces the configuration file to be reread?

4. What is the default signal sent to a process, using **kill** or **killall**?

5. Which signal is directly handled by the kernel and cannot be redefined?

6. Make sure you log into a virtual terminal (tty1 to tty6) before doing this. We want to run a script that will continue to run once we logout using the **nohup** parent process.

In the **/tmp** directory create a file called *print-out* with the following content:

```
count=0
while (true) do
    echo this is iteration number $count
    let count+=1
done
```

We first do the following (without using **nohup**) :

```
cd /tmp
./print-out &
exit
```

You may not see the command line when typing **exit** but this should log you out. When you log back in check that *print-out* is no longer running

```
ps ux |  grep  print-out
```

Next start the command with

```
nohup /tmp/print-out &
exit
```

Log back in and test these commands

```
ps ux |grep print-out
tail -f ~/nohup.out
Ctrl+C
killall print-out
ps ux|grep print-out
tail -f ~/nohup.out
```

Text Processing

Prerequisites

The Command Line (p.56)

Goals

Effectively manipulate files and data streams to alter the content as required (e.g sort or format)
Improve command line skills by memorising and understanding simple text tools

Contents

1. cat the Swiss Army Knife

- **cat the editor**

The **cat** utility can be used as a rudimentary text editor.

```
cat  > short-message
we are curious
to meet
penguins in Prague
Crtl+D
```

Notice the use of Ctrl+D. This command is used for ending interactive input.

- **cat the reader**

More commonly **cat** is used only to flush text to *stdout*. Most common options are

-n number each line of output
-b number only non-blank output lines
-A show carriage return

Example

```
cat  /etc/resolv.conf
search    mydomain.org
nameserver   127.0.0.1
```

- **tac reads back-to-front**

This command is the same as **cat** except that the text is read from the last line to the first.

```
tac  short-message
penguins in Prague
to meet
we are curious
```

2. Simple tools

● using head or tail

The utilities **head** and **tail** are often used to analyse logfiles. By default they output 10 lines of text. Here are the main usages.

List 20 first lines of **/var/log/messages**:

```
head -n 20 /var/log/messages
head -20  /var/log/messages
```

List 20 last lines of **/etc/aliases**:

```
tail -20  /etc/aliases
```

The **tail** utility has an added option that allows one to list the end of a text starting at a given line.

List text starting at line 25 in **/var/log/messages**:

```
tail +25 /etc/log/messages
```

Exercise: If a text has 90 lines, how would you use **tail** and **head** to list lines 50 to 65? Is there only one way to do this ?

Finally **tail** can continuously read a file using the **-f** option. This is most useful when you are expecting a file to be modified in real time.

● counting lines, words and bytes

The **wc** utility counts the number of *bytes*, *words*, and *lines* in files. Several options allow you to control **wc**'s output.

Options for **wc**

-l	count number of lines
-w	count number of words
-c or -m	count number of bytes or characters

Remarks:
With no argument **wc** will count what is typed in *stdin*.

- **numbering lines**

The **nl** utility has the same output as **cat -b**.

Number all lines including blanks

```
nl -ba /etc/lilo.conf
```

Number only lines with text

```
nl -bt  /etc/lilo.conf
```

- **replacing tabs with spaces**

The **expand** command is used to replace TABs with spaces. One can also use **unexpand** for the reverse operations.

- **viewing binary files**

There are a number of tools available for this. The most common ones are **od** (octal dump) and **hexdump**.

- **splitting files**

The **split** tool can split a file into smaller files using criteria such as size or number of lines. For example we can spilt */etc/passwd* into smaller files containing 5 lines each

```
split -l 5 /etc/passwd
```

This will create files called *xaa*, *xab*, *xac*, *xad* ... each file contains at least 5 lines. It is possible to give a more meaningful prefix name for the files (other than '*x*') such as '*pass-5.*' on the command line

```
split -l 5 /etc/passwd passwd-5
```

This has created files identical to the ones above (*aa*, *xab*, *xac*, *xad* ...) but the names are now *passwd-5aa*, *passwd-5ab*, *passwd-5ac*, *passwd-5ad* ...

- **Erasing consecutive duplicate lines**

The **uniq** tool will send to STDOUT only one version of consecutive identical lines. Consider the following example:

```
uniq > /tmp/UNIQUE
line 1
line 2
line 2
line 3
line 3
line 3
line 1
^D
```

The file */tmp/UNIQUE* has the following content:

```
cat /tmp/UNIQUE
line 1
line 2
line 3
line 1
```

NOTICE

From the example above we see that when using **uniq** non consecutive identical lines are still printed to STDOUT. What is the content of */tmp/UNIQUE* if we first send the STDIN through **sort** (see p.95) as follows:

```
sort | uniq > /tmp/UNIQUE
```

3. Manipulating text

The following tools modify text layouts.

- **choosing fields and characters with cut**

The **cut** utilility can extract a range of characters or fields from each line of a text.

The **–c** option is used to manipulate characters.

Syntax:

```
cut –c {range1,range2}
```

Example

```
cut –c5-10,15- /etc/password
```

The example above outputs characters 5 to 10 and 15 to end of line for each line in /etc/password.

One can specify the field delimiter (a space, a commas etc ...) of a file as well as the fields to output. These options are set with the **–d** and **–f** flags respectively.

Syntax:

```
cut -d {delimiter}  -f {fields}
```

Example

```
cut -d: -f 1,7 --output-delimiter=" " /etc/passwd
```

This outputs fields 1st and 7th of /etc/passwd delimited with a space. The default *output-delimiter* is the same as the original input delimiter. The **--output-delimiter** option allows you to change this.

- **joining and pasting text**

The easiest utility is **paste,** which concatenates two files next to each other.

Syntax:

```
paste   text1   text2
```

With **join** you can further specify which fields you are considering.

Syntax:

```
join  -j1 {field_num}  -j2{field_num}  text1  text2                    or
join   -1 {field_num}  -2{field_num}  text1  text2
```

Text is sent to stdout only if the specified fields match. Comparison is done one line at a time and as soon as no match is made the process is stopped even if more matches exist at the end of the file.

- **sorting output**

By default, **sort** will arrange a text in alphabetical order. To perform a numerical sort use the **-n** option.

- **formatting output with fmt and pr**

You can modify the number of characters per line of output using **fmt**. By default **fmt** will concatenate lines and output 75 character lines.

fmt options

-w number of characters per line

-s split long lines but do not refill

-u place one space between each word and two spaces at the end of a sentence

Long files can be paginated to fit a given size of paper with the **pr** utility. One can control the page length (default is 66 lines) and page width (default 72 characters) as well as the number of columns.

When outputting text to multiple columns each column will be evenly truncated across the defined page width. This means that characters are dropped unless the original text is edited to avoid this.

- **translating characters**

The **tr** utility translates one set of characters into another.

Example changing uppercase letters into lowercase

```
tr 'A-B' 'a-b'  < file.txt
```

Replacing delimiters in **/etc/passwd**:

```
tr  ':'  ' ' < /etc/passwd
```

Notice: **tr** has only **two arguments**! The file is not an argument.

4. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. The commands '`cat FILE` ' and '`cat < FILE` ' will both display the contents of `FILE` _____

2. The command '`last FILE` ' will display the 10 last lines of FILE _____

3. When altering lines from a file using **cut** those changes are made on the STDOUT only _____

4. When running **uniq** against a file consecutive identical lines are deleted in the file _____

Commands

Command	Description (apropos)
cat	**cat(1)** – concatenate files and print on the standard output
cut	**cut(1)** – remove sections from each line of files
expand	**expand(1)** – convert tabs to spaces
fmt	**fmt(1)** – simple optimal text formatter
head	**head(1)** – output the first part of files
join	**join(1)** – join lines of two files on a common field
nl	**nl(1)** – number lines of files
od	**od(1)** – dump files in octal and other formats
paste	**paste(1)** – merge lines of files
sort	**sort(1)** – sort lines of text files
split	**split(1)** – split a file into pieces
tac	**tac(1)** – concatenate and print files in reverse
tail	**tail(1)** – output the last part of files
tr	**tr(1)** – translate or delete characters
unexpand	**unexpand(1)** – convert spaces to tabs
uniq	**uniq(1)** – remove duplicate lines from a sorted file
wc	**wc(1)** – print the number of bytes, words, and lines in files

Exercises

1. Use **cat** to enter text into a file called *message*.

```
cat >> message
line 1
^D
```

Do the same but use the keyword STOP instead of the predefined eof control (^D).

```
cat >> message << STOP
line 2
STOP
```

Next, append text to *message* using **echo**.

```
echo line 3 >> message
```

2. Create a file called *index* with two fields *REFERENCE* and *TITLE* separated by a space.
e.g `001 Using_Linux`

Create a second file *pricing* with two fields *REFERENCE* and *PRICE* separated by a space
e.g `001 9.99`
Use **join** to display the reference, title and prices fields.

3. Using **tr** replace all colons by semi-colons in */etc/passwd*.
Do the same using **cut**.

4. Use **head** and **tail** to list lines 70 to 85 of */var/log/messages.*

5. Use the **cut** utility together with **grep** and **ifconfig** to printout only the IP address of the first network interface eth0.

6. In **/tmp** make a directory called *files*

```
mkdir /tmp/files
```

Create 50 files in that directory:

```
#!/bin/bash
count=0
while [ $count -lt 50 ]; do
touch /tmp/files/$count.txt
let count+=1
done
```

We want to change all the **txt** extensions to **dat** extentions. For this we need to type the following on the command line:

```
for FILES in $(ls *.txt)
do
FILENAME=$(echo $FILES| cut -d. -f1)
mv  $FILES $FILENAME.dat
done
```

Software Installation

Prerequisites

The Command Line (p.56)

Goals

Understand the use of a Makefile when compiling large projects from source
Manipulate source archives effectively and run the appropriate build commands
Fix problems related to shared (or dynamic) libraries
Use the RPM package manager to query, add, remove, update or verify software

Contents

1. Introduction

We begin with a short code example. Although we don't need an advanced understanding of the C language, these examples can help trouble shoot common situations.

The main.c file:

```
#include<stdlib.h>

int main(){
   Hello();
}
```

The Hello.c file:

```
#include<stdio.h>

void Hello(){
     printf("Hi ! \n");
}
```

Notice that the `main.c` is incomplete in the sense that the Hello() function is undefined. In the same way `Hello.c` doesn't have a "main" declaration. So these files are interdependent. One can however compile **object** files (.o) which are like non-executable binary files which can be used to 'build' an application.

Compiling the object files:

```
gcc -c main.c
gcc -c Hello.c
```

This will generate two files `main.o` and `Hello.o` which can now be used to build the application **app**.

*Compiling **app***:

```
gcc -o app main.o Hello.o
```

The **-o** option simply specifies a name for the compiled code. If no name is specified the compiled output is called **a.out** by default.

All these steps can be automated using a **Makefile**. Here is a minimal Makefile which would compile the **app** executable.

Makefile

```
SHELL = /bin/sh
CC = /usr/bin/gcc
app: main.o Hello.o
     $(CC) -o app main.o Hello.o
main.o: main.c
     $(CC) -c main.c
Hello.o: Hello.c
     $(CC) -c Hello.c
```

2. Static and Shared Libraries

Functions that will often be used are archived as libraries. During compilation these libraries can be *linked* to the code which uses the library function calls. The library can either be *statically* or *dynamically* linked to the code.

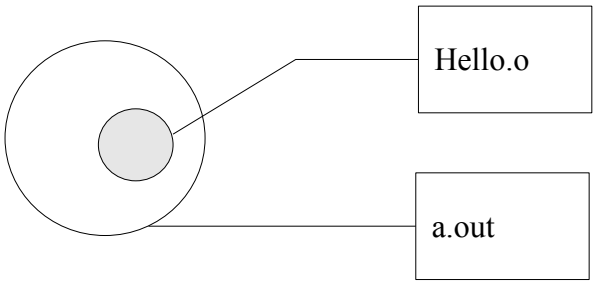
The **`gcc`** compiler can link libraries in a variety a ways (many options). However by default it will link files that are given on the commandline that don't have a **`.c`** extention (only the **`.c`** files are treated as code).

Listing 1: Linking by default

```
gcc main.c Hello.o
```

This will produce an **`a.out`** executable with the Hello.o object statically linked to it.

Illustration of a statically linked application (a.out):



● Static libraries

Static libraries are archived **`.o`** files. These archives are created with the **`ar`** tool and have a **`.a`** extention.

Listing2: adding an object file to an archive:

```
ar rcs libfoo.a file1.o file2.o
```

● **Dynamic/Shared Libraries**

A shared library is a library that will be loaded by the program when it is executed. One also says that the library is *dynamically loaded*.

Listing 3: Creating a shared library:

```
gcc -c -fPIC Hello.c      creates the object file
gcc -shared -W1,soname,libfoo.so.1 -o libfoo.so.1.0 Hello.o
```

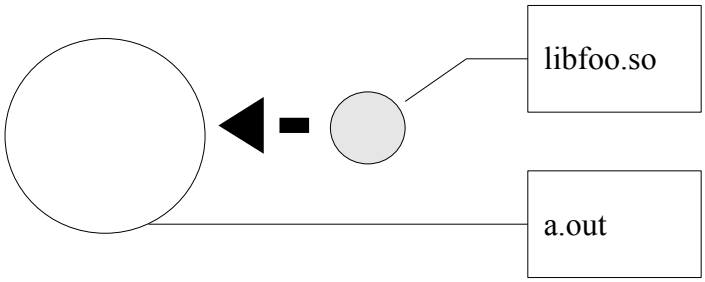
The `-fPIC` flag enables the Position Independent Code generation.

Listing4: Compiling with a shared library:

```
gcc main.c libfoo.so.1.0
```

This will produce an **a.out** executable. However if you try to run this it will complain with the error message listed below.

Illustration of a dynamically linked application (a.out):



The process of attaching a dynamic library at run-time is called linking and is handled by the ld.so library. How does the linker know where to find libfoo.so?

Shared library not found error:

```
./a.out: error while loading shared libraries: libfoo.so.1.0: cannot open
shared object file: No such file or directory
```

This error illustrates the case where the linker could not find the dynamic library libfoo.so.1.0. In the next section we will see what can be done to fix this problem.

● **Shared Library naming and dynamic loading**

We will use the above example to understand how Linux libraries are maintained.

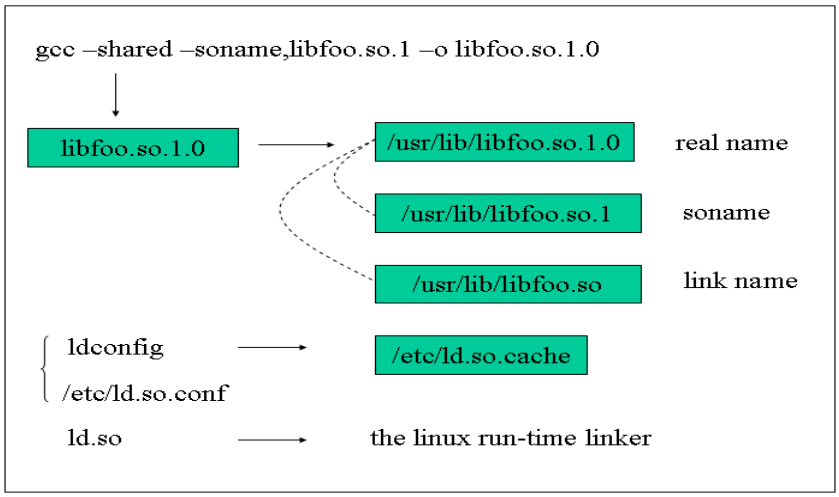


Figure 1: The Shared Library Names

To find out which shared libraries an executable needs at execution time the **`ldd`** tool is used.

Example:

```
ldd a.out
libfoo.so.1.0 => not found
libc.so.6 => /lib/libc.so.6 (0x40028000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Notice that *`libfoo.so.1.0`* is not found. This is because the **`a.out`** needs to dynamically load this library and the dynamic linker **`ld.so`** is not aware of this new library.

In fact the linker uses a database called the *ldcache* containing entries of the form:

soname =>/path/to/library

The content of the ld-cache can be viewed with the following command:

```
ldconfig -p
libaudiofile.so.0 (libc6) => /usr/lib/libaudiofile.so.0
libaudiofile.so (libc6) => /usr/lib/libaudiofile.so
libaudio.so.2 (libc6) => /usr/X11R6/lib/libaudio.so.2
libattr.so (libc6) => /usr/lib/libattr.so     ..........
```

The ld-cache is generated at boot time by the same **ldconfig** tool. By default **ldconfig** will scan the directories **/lib** and **/usr/lib** to build the ld-cache.

If libraries are installed in different locations (e.g **/usr/local/lib**, **/opt/lib** or **/usr/X11R6/lib**) these directories need to be listed in **/etc/ld.so.conf** allowing **ldconfig** to take these directories into consideration when building the cache.

What happens when an application is started?

The application will ask the linker for the dynamic libraries it needs using a soname, the linker will then query the ld-cache and associate this name with the full path to the actual library. Once the full path is known the linker can link the library to the application.

What happens if the ld-cache doesn't contain the full path to the library?

In general the application will fail to start and will print an error message saying " `cannot open shared object file: No such file or directory` ". But one can also define a global variable called LD_LIBRARY_PATH and assign to this variable the name of the directory containing the library.

Knowing this we can now fix the problem with our application above using one of the tw0 methods below:

1. If the binary needs to be temporary tested define the LD_LIBRARY_PATH variable as follows:

```
export LD_LIBRARY_PATH=$(pwd)
```

2. If you are root and would like the library to be available for all then copy the **`libfoo.so.1.0`** file to **`/usr/local/lib/`** and run **`ldconfig`** to update the ld cache.

The GNU specification advises libraries to be stored in **`/usr/local/lib`**. These guidelines are followed by developers and most tarballed code will install libraries in that directory and the binaries in **`/usr/local/bin`**. Installing and removing this code from the system would be done by ¨make install¨ and ¨make uninstall¨.

The FHS (Filesystem Hierarchy Standard) recommends libraries be kept in **`/usr/lib/`** and associated binaries in **`/usr/bin/`**. This convention standard is adhered to by Linux distributions. In effect mature and stable code is stored in **`/usr/`** rather than **`/usr/local/`** and the two standards do not lead to any contradictions. Installing and removing this code code would be done using the rpm command.

NOTICE
With certain distributions the /usr/local/lib/ directory is not scanned by **`ldconfig`**. It is simply a matter of adding this directory to **`/etc/ld.so.conf`** and ... reboot?

3. Source Distribution Installation

Open source projects are often distributed as tarballs (i.e compressed tarred archives). Many development environments (glade, kdevelop…) generate the files that help facilitate compiling and installation of a project.

Uncompressed Archives

Uncompressed archives have a **.tar** extension. For example if a project has been developed in a directory called **my-projetc-v.1/** then the following command would archive this directory with all its files and subdirectories:

```
tar c my-project-v.1/ > my-project-v.1.tar
```

or

```
tar cf  my-project-v.1.tar my-project-v.1/
```

Since most projects are very large and are available for download from the Internet they are rarely uncompressed.

Compression

The three compression tools commonly used are **compress** (old), **gzip** and **bzip2**. Unlike the windows **zip** these compressions can only be applied to files. But since an archive is a file that contains all the data needed to recover directories, these compressions are suitable for archives. A compressed archive is then called a tarball.

compression tool	de-compression tool	cat decompression	file extension
compress	**uncompress**	**zcat**	.Z
gzip	**gunzip**	**zcat**	.gz
bzip2	**bunzip2**	**bzcat**	.bz2

Examples

```
compress -v FILE1
FILE1:  -- replaced with FILE1.Z Compression: 40.29%
```

```
gzip  -v FILE2
FILE2:   53.4% -- replaced with FILE2.gz
```

```
bzip2 -v FILE3
FILE3:    2.326:1,  3.439 bits/byte, 57.01% saved, 605504 in, 260320 out.
```

NOTICE
1. When compressing a file, the original file name is appended a .**Z**,.**gz** or .**bz2** **2.** Compression tools listed above only work on files and not on directories **3.** Only one file at a time can be compressed (no wild cards!)

The **zcat** and **bzcat** tools can be used to decompress files, however the decompressed file will be sent to STDOUT so it is necessary to use a file redirection:

```
zcat FILE1.Z > FILE1
```

Archives and Compression

compression tool	tar switch	archive extension
compress	Z	.tar.Z or .tgZ
gzip	z	.tar.gz or .tgz
bzip2	j	.tar.bz2

The table above introduces the **tar** options Z,z and j which call the appropriate compression tools when needed.

The next two examples are equivalent:

```
tar cf  my-project-v.1.tar my-project-v.1/
```

```
bzip2  my-project-v.1.tar
```

```
tar cjf  my-project-v.1.tar.bz2 my-project-v.1/
```

Working with tarballs

We know how to create archives. All we need is an overview of the main **tar** switches.

tar operations	Create	Extract	Test
minimal switches	c or cf	xf	tf
optional switches	v,Z,z,j	v,Z,z,j	v,Z,z,j

Examples (extractions)

```
tar xvjf myproject-v.1.tar.bz2
```

```
tar xzf some-other-project-v.2.0.tar.gz
```

Examples (tests)

```
tar tjf myproject-v.1.tar.bz2
```

```
tar tzf some-other-project-v.2.0.tar.gz
```

Alternative Examples (using **zcat** and **bzcat**)

```
bzcat myproject-v.1.tar.bz2 | tar xf -
```

```
zcat some-other-project-v.2.0.tar.gz | tar tf -
```

Common Files

Once a project has been extracted you can expect to find the following files:

configure: This is a script which determines what architecture is being used. It also checks that the required compiler, libraries an headers are present. This information is then stored in files called `Makefile`

The safest way to run the script is to use '`./configure`'.

You can also decide where the project will be installed using the `--prefix` option. The default installation directory for most projects is **/usr/local**. If you want to install the compiled project in your home directory you should type:

```
./configure -prefix=$HOME
```

Makefile: This acts like a configuration file for the **make** utility. The main information provided is:

- The name of the compiler and compiling options
- The path to the shared libraries and header files
- Mapping between code files (.c) and object files (.o)

Compiling the project

If the files above are present then there is a good chance that you will successfully 'port' the program to your computer. Here are the routine steps:

```
./configure
make
make install
```

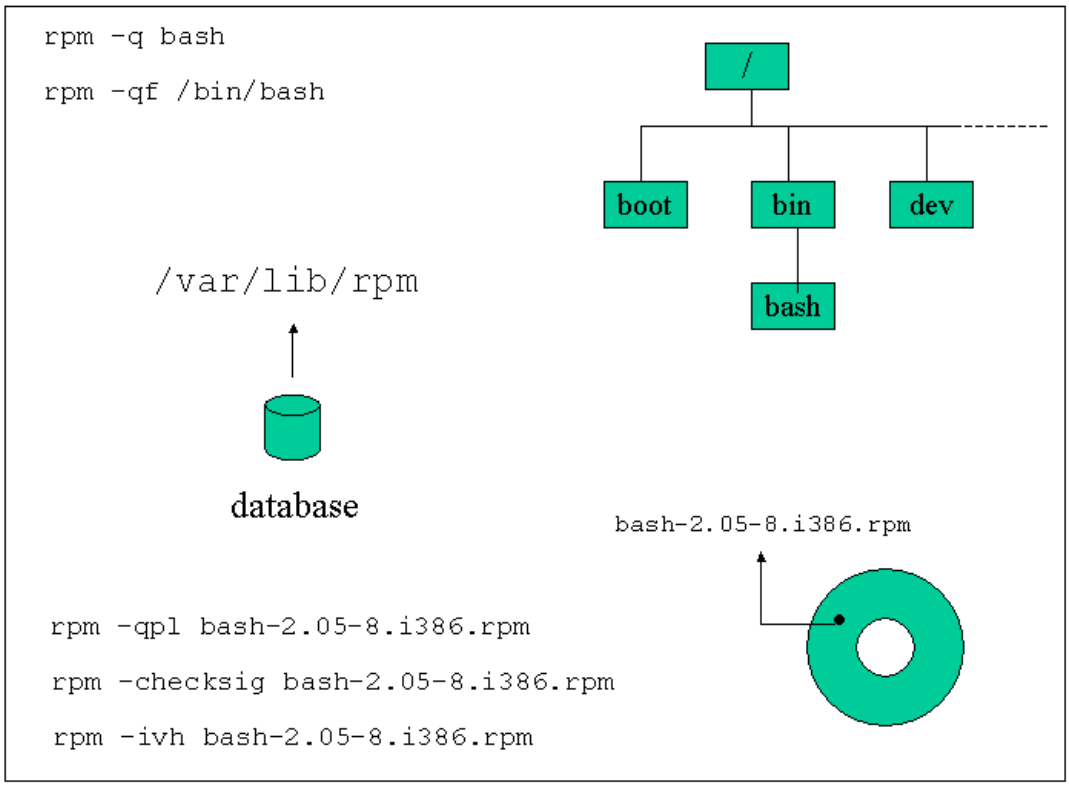
It is strongly recommended to run **./configure** and **make** as a non root user.
`make install` must be run as root only if the installation directories are write protected (/usr/ or /usr/local).

There are many options to the **./configure** script. To customise your installation you could type

```
./configure --help
```

4. The RedHat Package Manager RPM

Most Linux distributions manage software using some form of package management to perform tasks such as installations, updates and queries. The most popular package types are Debian and RPM. We only cover RPM in this manual.



The Functions of a Package Manager

Package naming

There is no strict convention but most rpm package names are formed as follows:

name-version-release.architecture.rpm

The architecture name can either indicate which computer architecture the enclosed binaries are made for (e.g i386, ppc, ia64, noarch) or it can indicate that the package contains the source code (src).

Major and minor modes

Some short name options are similar but perform different actions depending on their position on the command line. A distinction is made between the first option and other options.

The first option given to **rpm** is in major mode. For example in `rpm -iv A.rpm` the option 'i' is a major option and will cause package *A* to be installed.

Similarly an option that is not in first position is in minor mode. For example in `rpm -qpi A.rpm` the option 'i' is a minor mode and will get information from the package *A* such as the author and the licence type.

These are the major mode options for **rpm**.

Short	Long	Description
-i	–install	Installs the package
-U	–update	Updates or installs a package
-F	--freshen	Updates only installed package
-V	--verify	file size, MD5, permissions, type ...
-q	--query	Queries installed/uninstalled packages, and files
-e	–erase	Uninstall package

These are the minor mode options for **rpm**.

Short	Description
a	applies to all installed packages
c	together with **q** lists **c**onfiguration files
d	together with **q** lists **d**ocumentation files
f	together with **q** queries which package installed a given file
h	adds hashes while processing
i	together with **q** lists **i**nformation about a package
l	together with **q** lists all files and directories in a package
p	together with **q** specifies that the query is performed on the package file
v	verbose

Query modes

Three query types: uninstalled packages, installed packages and files

Query Type	Option
Package file	**-qp**
Installed package	**-q**
File	**-qf**

An extra option will allow you to get information on all installed files **–l**, documentation **–d** configuration files **–c**, etc ...

We consider for example the package **routed-0.17.i386.rpm**. We can query this package and list its contents before installation with the **l** option as follows:

```
rpm -qpl routed-0.17.i386.rpm
```

Once this package is install we can query the installed package with:

```
rpm -ql routed-0.17      or
rpm -ql routed
```

Finally if we want to find out which package installed the file **/usr/sbin/routed** the rpm database can be queried with:

```
rpm -qf /usr/sbin/routed
```

Special Options

--nodeps	install a package regardless of dependencies
--force	force an upgrade
--test	doesn't actually install or upgrade, just prints to stdout
--requires PACKAGE	together with **q** lists capabilities required by a package
--whatrequires CAPABILITY	together with **q** lists packages which require the capability

Package Signatures

You can check the signature of each package that is distributed as part of a project. For example to load the keys of all the developers involved with the Fedora project do the following (just once):

```
rpm –-import /usr/share/rhn/RPM-GPG-KEY-fedora
```

You can now download any package from an FTP site which mirrors the project's RPMs. For example we downloaded **zlib-1.2.1.1-2.1.i386.rpm** from ftp.mirror.ac.uk in the Fedora subdirectory. We next check the authenticity of the file:

```
rpm --checksig /home/adrian/zlib-1.2.1.1-2.1.i386.rpm
/home/adrian/zlib-1.2.1.1-2.1.i386.rpm: (sha1) dsa sha1 md5 gpg OK
```

Package Integrity

The next command checks the integrity of the package **bash**:

```
rpm –V bash
```

This returns nothing. We next do the following as user root:

```
chown bin /bin/bash
```

```
chmod 775 /bin/bash
```

If we check the integrity of **bash** again this time we get:

```
rpm –V bash
.M...U..   /bin/bash
```

The package manager has compared the current status of all files which are part of the **bash** package with the known original state of these files stored in a database. The changes made to **/bin/bash** have been identified.

It is possible to verify the integrity of all packages installed on the system by adding the '**a**' (`--all`) option after '**V**' (`--verify`)

The `–verify` option performs a number of tests on each file; when a test is positive a number of characters (listed below) are used to identify the errors:

Returned character	Error description
.	the test was successful
?	the test couldn't be performed
S	file size has changed
M	permission mode or file type has changed
5	the file's MD5 sum has changed
D	device major/minor number miss-match
L	broken symbolic link
U	the user owner of the file has changed
G	the group owner of the file has changed
T	the mtime (modified time) has changed

Going Further: Building RPM packages (*not for LPI exam purpose*)

NOTICE:
This is additional information, this paragraph is not an LPI 101 objective. When doing this section you may encounter problems with the *–-rebuild* option, this is due to the fact that the new versions of RPM use **rpmbuild** instead of **rpm** when rebuilding packages.

The source code for many RPM packages is also available as an RPM package and will be used to build a binary package. The naming convention is:

`name-version-release.src.rpm`

These packages contain at least two files, the tarball with the code and a spec file. The spec file contains instructions to patch, compile and build the RPM package. If the code needs to be patched before compilation then the patches are included in the source package.

There are three different ways to build a RPM package. We will assume that we have a package called `name-version-release.src.rpm`.

*For these methods to work you first need to install the **rpm-build** package*

Method 1:

Install the RPM source package with:

```
rpm –ivh name-version-release.src.rpm
```

This will copy files to the following directories:
`/usr/src/redhat/SPECS`
`/usr/src/redhat/SOURCES`

In the `/usr/src/redhat/SPECS` directory there is now a file called **name.spec** (where 'name' is the name of the package). To start building the compiled package, that is **name-version-release.i386.rpm**, we type in the following command:

```
rpm –ba name.spec
```

This will start a series of scripts. The tarball in `/usr/src/redhat/SOURCES` will be unpacked in `/usr/src/redhat/BUILD`.

If the compilation succeeds then the built binary package will be saved in `/usr/src/redhat/RPMS/`. There are different subdirectories corresponding to various CPU models/generations. If the compilation didn't involve specific features from these chips then the package will be saved in the `noarch` directory.

Method 2:

This method triggers the same chain of events as the previous one but is started with the following single command:

```
rpm --rebuild  name-version-release.src.rpm
```

Method 3:

In some cases developers will distribute a tarball together with a *spec* file. If the tarball is called `name-version-release.tar.gz` you can search for a .spec file with the following:

```
tar tzvf name-version-release.tar.gz | grep .spec
```

If the tarball has a spec file then you can build an RPM package by typing:

```
rpm –bt name-version-release.tar.gz
```

5. Debian Package Management

Systems using Debian based variants of Linux don't use the rpm package management system, but rather the Debian Package Management system. The Debian system is more rigorous and configurable than the rpm system, but for historical reasons is less widely used.

The approach used by the the Debian system is very similar to that used by the rpm system. The equivalent command to 'rpm' in a Debian system is '**dpkg**'.

Package Naming

Similarly to RPM-based system, Debian packages come in files whose names are formed as follows:

name_version-release_architecture.deb

The release number indicates which Debian release of the version of the software the package contains, while the architecture name specifies the computer architecture (i386, sparc, all).

dpkg

dpkg is a medium-level tool to install, build, remove and manage Debian packages. Other front-end packages are more commonly used to contol **dpkg,** including the **apt** tools and others such as **dselect**. **dpkg** itself is controlled via command line parameters, which consist of an action and zero or more options. The action parameter tells dpkg what to do and options control the behaviour of the action in some way.

dpkg maintains some usable information about available packages. The information is divided in three classes: **states**, **selection states** and **flags**.

Package States

State	Description
installed	The package is unpacked and configured OK.
half-installed	The installation of the package has been started, but not completed for some reason.
not-installed	The package is not installed on your system.
unpacked	The package is unpacked, but not configured.
half-configured	The package is unpacked and configuration has been started, but not yet completed for some reason.
config-files	Only the configuration files of the package exist on the system.

Package Flags

Flag	Description
hold	A package marked to be on **hold** is not handled by **dpkg**, unless forced to do that with option –**force-hold**.
reinst-required	A package marked **reinst-required** is broken and requires reinstallation. These packages cannot be removed, unless forced with option **--force-reinstreq**.

Actions

The heart of dpkg operation is the command line parameters specifying the action which should be performed. While there are a large number of these, the following table summarises the main actions you are likely to require on any regular basis.

Action	Description
-l	Prints a list of the packages installed on the system, or matching a pattern if any is given. The first three characters on each line show the state, selection state, and flags of the package
-s	Shows the status and information about particular installed package(s)
-I	Show information about a package in a **.deb** file
-L	List the files included in a package
-S	Show the package which includes the file specified
-i	Install (or upgrade) and configure a package from a **.deb** file
--unpack	Unpack (only) a package in a **.deb** file
--configure	Configure an unpacked package. With **-a** (or **--pending)** configures all packages requiring configuration
-r	Remove a package (but leave its configuration files)
-P	Purge – remove a package along with its configuration files
--get-selections	Get a list of package selections from a system (to stdout)
--set-selections	Set the list of package selections for a system (from stdin)

Options

All options can be specified both on the commandline and in the **dpkg** configuration file */etc/dpkg/dpkg.cfg*. Each line in the configuration file is either an option (exactly the same as the commandline option but without leading dashes) or a comment (if it starts with a **#**).

Option	Description
-force-*thing*	Forces **dpkg** to perform an action which it would normally not take (for example, to ignore dependency information - **--force-depends**, or to downgrade a package with –**force-downgrade**)
--refuse-*thing*	Refuse to do something which **dpkg** would normally automatically do
--ignore-depends	Ignore dependency checking for a package
--no-act	Show what **dpkg** would do, but don't do it (also: **--simulate)**
-R	Recurse through directories (using with **-i** or **--unpack**)

Files

dpkg uses a number of files in its operation, including ***/etc/dpkg/dpkg.cfg*** which contains default configuration settings.

Lists of available packages along with their statuses are held in the files ***/var/lib/dpkg/available*** and ***/var/lib/dpkg/status***.

A **.deb** file, along with the files making up a packages programs, libraries and configuration, will also include a number of control files which allow the execution of scripts before and after installation and removal, along with lists of files and configuration files. These can be found in the ***/var/lib/dpkg/info*** directory once the packages are installed.

Use of dpkg

To install a package from a .deb file, you could use dpkg as follows:

```
dpkg -i hello_2.1.1-4_i386.deb        OR
dpkg --unpack hello_2.1.1-4_i386.deb
dpkg --configure hello
```

To remove the hello package along with its configuration, you could use:

```
dpkg -P hello
```

While:

```
dpkg -r hello
```

would remove only the package, leaving its configuration files installed.

The get a list of all the packages installed on the system, use the command:

```
dpkg -l
```

Note that when dealing with a package file, the filename is given, while when dealing with an installed package, the package name only is given.

APT

The **dpkg** tool is fine for installing individual packages with no dependencies, but when installing a number of packages which may have dependencies, the APT tool is generally used instead.

APT is one of the strengths of dpkg, and provides an easy way of installing and updating a system. It is controlled by two files:

File	Description
/etc/apt/apt.conf	Contains general configuration options for APT, such as which release of Debian to install, whether/which proxy settings to use, etc
/etc/apt/sources	Lists sources of Debian files, which may be on CDs, or on the network

In general, to use APT you must first configure the sources it is to used. This can be done (if you are using CDs) by using the command:

```
apt-setup
```

which asks the user to choose which mirror to download from, and tests it, or if you are using CDs, using:

```
apt-cdrom
```

which allows individual CDROMs to be scanned for packages.

Once APT knows where the Debian packages are located, two command line tools are used for package management: **apt-cache** and **apt-get**.

apt-cache

apt-cache allows manipulation of the APT package cache (which is stored in files in ***/var/cache/apt***). An action normally follows apt-cache on the command line, and common options include:

Action	Description
search	Search all the available package descriptions for the string given, and print a short description of the matching package
show	Shows a full description of the package specified

apt-get

While **apt-cache** is useful for finding out information about available packages, **apt-get** allows updating of package information, retrieval, installation and removal of packages, and even upgrading of an entire Debian distribution. apt-get expects an action to be provided on the command line, and the most common are listed below:

Action	Description
update	Update the list of packages from the sources in /etc/apt/sources.list
install *package*	Install the package(s) specified, along with any dependencies
upgrade	Upgrade any packages which have newer versions available
dist-upgrade	Upgrade entire distribution to the latest release (best to read the release notes first!)
remove	Remove the package(s) specified

Use of APT

The two main uses of APT are for updating the system (for example if security-related updates have become available). This is normally done using the two commands:

```
apt-get update
```

```
apt-get upgrade
```

The other main use of APT is to install required packages. This normally involves the following commands:

```
apt-get update            #update list of packages
```

```
apt-cache search frob      #find packages relating to frobbing
```

```
apt-cache show frobnicate  #show information regarding a particular
package
```

```
apt-get install frobnicate #install frobnicate package and its
dependencies
```

6. The Alien Tool

The **alien** tool will change Debian packages into RedHat ones and vice versa. One can download it at: http://kitenet.net/programs/

Convert a debian package to an rpm:

```
alien -to-rpm  package.deb
```

Convert an rpm package to debian:

```
alien -to-debian package.rpm
```

7. Exercises and Summary

Review Questions (answers p.150)

Yes or No

1. When building a project from source one has to compile sections of the code in the correct order using **gcc** on the command line _____

2. The program **make** will only build a project if it is started in a directory containing the appropriate Makefile _____

3. Pre-compiled binary packages and source code packages are two types of RPM packages _____

4. Once shared libraries have been installed from source it is recommended to run **ldconfig** _____

5. The **ldconfig** tool is used to update the ld-cache _____

6. Program installed from source can be queried using a package manager _____

7. The APT tools can install packages and resolve all dependencies _____

Glossary

Term	Description
build	term used when compiling a project from source, usually started by typing **make**
compile	translate programming instructions written in a high-level language into machine readable code. The output of a compilation is called the object code
dynamic library	a library that can be loaded at the same time as the execution of a program
shared library	a library intended to be used by more than one executable. Shared libraries are often dynamic libraries and have a **.so** extension (shared object)
static library	a library that is copied into the executable during the compilation. Static libraries have **.a** extensions (archive)
high-level language	a programming language readable by humans used to write source code
linker	**1.** program used during the compilation process to assemble objects generated by the compiler into an executable – see **ld(1)** **2.** program that dynamically loads shared libraries needed by an executable at runtime – see **ld.so(8)**
object code	the output of a compilation. Object code is either an executable or may be linked to another object code to form an executable
source code	programming instructions written in a high-level language that need to be compiled with a compiler or interpreted with an interpreter
tarball	a compressed **tar** archive

Files

File	Description
/etc/ld.so.conf	configuration file for **ldconfig**
Makefile	file read by the **make** utility when building a project
/etc/rpmrc	used by **rpm** and **rpmbuild** (see LPI 201), this file contains information such as the system's architecture or the path to macros and utilities used when handling packages. This file is often located in the **/usr/lib/rpm/** directory
/usr/lib/rpm/	directory containing all the macros needed when handling packages
/var/lib/rpm/	directory where databases for the package manager (RPM) are kept

Commands

Command	Description
alien	**alien(1)** – Convert or install an alien binary package. It converts between Red Hat rpm, Debian deb, Stampede slp, Slackware tgz, and Solaris pkg file formats. If you want to use a package from another Linux distribution than the one installed on your system you can use alien to convert it to your preferred package format and install it
APT tools	Tools used to perform advanced operations on Debian packages located on a CD or a server
configure	script often included with a project source code used to create makefiles. It attempts to determine information such as the system's CPU type or installed components needed to build the project(compiler, header files or libraries).
dpkg	tool used to manipulate packages in the DEBIAN format
LD_LIBRARY_PATH	environment variable containing the search path to shared libraries used by the linker (ld.so)
ldconfig	program that builds the 'ldcache' used by the linker to find shared libraries, with the **-p** flag it will print the current content of the cache
ldd	**ldd(1)** – prints the shared libraries required by each program or shared library specified on the command line
make	**info make** – The `make' utility automatically determines which pieces of a large program need to be recompiled, and issues commands to recompile them.
rpm	tool used to manipulate packages in the RPM format

Exercises

In the following examples download a source RPM file (e.g. bash-2.05-8.src.rpm for RedHat 7.2) from www.rpmfind.net.

1. Installing as a tarball.

- Extract the contents of the RPM package without compiling anything with:

```
rpm –ivh bash-2.05-8.src.rpm
```

- In the `/usr/src/redhat/SOURCES` directory, unpack the tarball with:

```
tar xvzf bash-2.05-8.tar.gz
```

- Optional (recommended!): The patches can be applied. Depending on which directory you are in the syntax will vary.

From `/usr/src/redhat/SOURCES`:

```
patch -p0 -b < file.patch
```

From `/usr/src/redhat/SOURCES/bash-2.05-8`

```
patch -p1 -b < file.patch
```

- We will choose to install the files in a temporary root directory, for example **/tmp/project-test**/. On a production system the usual location should be `/usr/local`. We create this directory:

```
mkdir /tmp/project-test
```

- Finally follow the usual compilation steps.

```
./configure -prefix=/tmp/project-test
 make
 make install
```

You can now list the content on **/tmp/project-test**.

2. (Going further – not required for LPI101) We next rebuild the project into an RPM package.

```
rpm --rebuild  package.src.rpm
```

The compiled binary package should be in `/usr/src/redhat/RPMS`

- Check the package's contents with the **–qpl** option
- Install the package(s), and run queries on the installed package
- Uninstall the package

3. Configure **/etc/apt/sources** using **apt-setup**. Use the APT tools and **dpkg** to query/install/update available packages.

Advanced Text Manipulation

Prerequisites

The Command Line (p.56)
Text Processing (p.91)

Goals

Distinguish expressions used for file globbing (metacharacters) and regular expressions
Use the **grep** tools effectively
Understand simple **sed** commands

Contents

Advanced Text Manipulation

Overview

Finding a word or multiple words in a text is achieved using **grep**, **fgrep** or **egrep**. The keywords used during a search are a combination of letters called *regular expressions*. Regular expressions are recognised by many other applications such as **sed**, and **vi**.

1. Regular Expressions

Traditional Regular Expressions (regex)

A regular expression is a sequence of characters (or atoms) used to match a pattern. Characters are either constants (treated literally) or metacharacters.

Table1: Main metacharacters

Characters	Search Match
\<KEY	Words beginning with 'KEY'
WORD\>	Words ending with 'WORD'
^	Beginning of a line
$	End of a line
[Range **]**	Range of ASCII characters enclosed
[^c **]**	Not the character 'c'
\[	Interpret character '[' literally
"ca*t"	Strings containing 'c' followed by no 'a' or any number of the letter 'a' followed by a 't'
" . "	Match any single character

Extended regex: The main eregex's are: +,?,() and |

Table2: List of main eregex

Characters	Search Match
"A1**|**A2**|**A3**"**	Strings containing 'A1' or 'A2' or 'A3'
"ca**+**t"	Strings containing a 'ca' followed by any number of the letter 'a' followed by a 't'
"ca**?**t"	Strings containing 'c' followed by no 'a' or exactly one 'a' followed by a 't'
"ca*t"	Strings containing 'c' followed by no 'a' or any number of the letter 'a' followed by a 't'

2. The grep family

basic grep

The **grep** utility supports regular expressions *regex* such as those listed in *Table1*.

egrep

The **egrep** tool supports extended regular expressions *eregex* such as those listed in *Table2*.

fgrep

Fgrep stands for *fast grep* and **fgrep** interprets strings literally (no regex or eregex support)

3. Working with grep

Syntax for grep:
grep PATTERN FILE

grep	*Main Options*
-c	count the number of lines matching PATTERN
-f	obtain PATTERN from a file
-i	ignore case sensitivity
-n	indicate the input file's line number
-v	output all line except those containing PATTERN
-w	match exact PATTERN

For example list all non blank lines in /etc/lilo.conf:

```
grep –v "^$" /etc/lilo.conf
```

4. egrep and fgrep

The **fgrep** utility does not recognise the special meaning of the regular expressions. For example

```
fgrep 'cat*' FILE
```

will only match words containing 'cat*'. The main improvement came from **fgrep**'s ability to search from a list of keywords entered line by line in a file, say LIST. The syntax would be

```
fgrep –f LIST FILE
```

The **egrep** utility will handle any modern regular expressions. It can also search for several keywords if they are entered at the commandline, separated by pipes. For example;

```
egrep "linux|^image"   /etc/lilo.conf
```

5. The Stream Editor - sed

At this point the stream editor makes its appearance! It is an old type of tool and originally the only one available under UNIX to manipulate text.

The **sed** utility is most often used to search and replace patterns in text. It supports most regular expressions.

5.1 Beginning sed

*Syntax for **sed***

```
sed   [options] ´command'   [INPUTFILE]
```

The input file is optional since **sed** also works on *file redirections* and *pipes*.
Here are a few examples assuming we are working on a file called MODIF.

Delete all commented lines:

```
sed '/^#/ d ' MODIF
```

Notice that the search pattern is between the double slashs //.

Substitute /dev/hda1 by /dev/sdb3:

```
sed 's/\/dev\/hda1/\/dev\/sdb3/g'   MODIF
```

The **s** in the command stands for 'substitute'. The **g** stands for "globally" and forces the substitution to take place throughout each line.

If the line contains the keyword KEY then substitute ':' with ';' globally:

```
sed ' /KEY/ s/:/;/g'   MODIF
```

5.2 More Advanced sed

You can issue **several commands** each starting with **–e** at the command line. For example, (1) delete all blanks then (2) substitute 'OLD' by 'NEW' in the file MODIF

```
sed -e '/^$/ d'   -e 's/OLD/NEW/g'   MODIF
```

These commands can also be written to a file, say COMMANDS. Then each line is interpreted as a new command to execute (no quotes are needed).

An example COMMANDS file
`1 s/old/new/`
`/keyword/ s/old/new/g`
`23,25 d`

The syntax to use this *COMMANDS* file is:

```
sed -f COMMANDS MODIF
```

This is much more compact than a very long commandline !

*Summary of options for **sed***

Commandline flags
-e Execute the following command
-f Read commands from a file
-n Do not printout unedited lines

sed commands
d Delete an entire line
r Read a file and append to output
s Substitute
w Write output to a file

6. Exercises and Summary

Review Questions (answers p.151)

Yes or No

1. The extended regular expression '`nucle?ar`' will match `nuclear` and `nuclar`

   ____

2. The regular expression '`baza*r`' will match `bazaar` and `bazar` but not `bazor` ____

3. The extended regular expression '`nucle+ar`' will only match `nuclear`

   ____

4. The regular expression '`baza*`' will match `bazaar, bazar` and `bazor`

   ____

Commands

Command	Description or *apropos*
egrep	print lines containing matching patterns using extended regular expressions
fgrep	print lines containing matching patterns using literal stings
grep	print lines containing matching patterns using regular expressions
sed	**sed(1)** – stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline)

Exercises

1. Create a new file called FILE containing the lines:

```
Using grep,
fgrep and
egrep
to grep for 99% of the cats
% these are two
% commented lines
```

- Use **grep** to output only uncommented lines.

- Find all lines containing ‘grep’ exactly. (Not ‘egrep’ nor ‘fgrep’.Use **-w** to match the word)

- Find lines containing words starting with an ‘a’

2. Regular expressions. Append the following lines to the previous file:

```
ct
cat
caats
caaatss
ca+t
ca*t
ca?t
```

crate
carts

- Investigate the outcome of the following using **grep**, **egrep** and **fgrep**:

```
grep 'ca+t' FILE
grep 'ca?t' FILE
grep 'ca.t' FILE
grep 'caa*t' FILE
grep 'ca*r.' FILE
```

3. Use **sed** to do the following changes in FILE
(use a COMMAND file, then do everything on the commandline)

- in the first line substitute '`grep,`' with '`soap`'
- delete '`fgrep`' in the second line
- substitute '`egrep`' with ' `water`'
- in the fourth line replace '`grep for`' with '`wash`'

Save the result to a file using the **w** option

Using vi

Prerequisites none

Goals

Understand the three operating modes of **vi**
Introduce most common editing commands
Recognise the use of regular expressions and sed-like commands

Contents

In most Linux distributions **vi** is the text editor of choice. It is considered an essential admin tool such as **grep** or **cat** and is found therefore in the **/bin** directory.

1. vi Modes

In order to perform complex operations such as copy/paste **vi** can operate in different modes.

● Command mode

This is the editing and navigation mode. Commands are often just a letter. For example use **j** to jump to the next line.
As a rule of thumb if you want to perform an operation several times you can precede the command by a number. For example **10j** will jump 10 lines.

In some situations the arrow keys on the keyboard are not mapped properly, it is still possible to navigate using the commands **h j k l** with the following effect:

● Last Line (or column) Mode

You enter this mode from the command mode by typing a colon**.** The column will appear at the bottom left corner of the screen. In this mode you can perform a simple search operation, save, quit or run a shell command.

● Insert Mode

The easiest way to enter this mode while in command mode is to use **i** or **a**. This mode is the most intuitive and is mainly used to interactively enter text into a document.

☞ The Esc key will exit the *insert mode* and return to *command mode*

2. Text Items

Items such as words and paragraphs are defined in *command mode* to allow editing commands to be applied to text documents without using a mouse.

Word, sentences and paragraphs

e resp. **b**	Move to the **e**nd/**b**egining of the current word
(resp. **)**	Move to the begining/end of the current sentence
{ resp.**}**	Move to the begining/end of the current paragraph
w	Similar to **e** but includes the space after the word

Beginning and End

^	Beginning of line
$	End of line
1G	Beginning of file
G	End of file

All these text items can be used to navigate through the text one word (**w**) or paragraph (**}**)at a time, go to the beginning of a line (**^**) the end of the file (**G**) etc. One can also use these text items to execute commands such as deleting and copying.

3. Inserting Text

When in command mode typing **i** will allow you to enter text in the document interactively. As with all other features in **vi** there are many other ways of doing this. The table below lists all possible inserting modes.

Insert commands

a	Append text with cursor on the last letter of the line
A	Append text with cursor after last letter at the end of the line
i	Insert text at the current position
o	Insert text on a new line below
O	Insert text on a new line above
s	Delete the current letter and insert
S	Delete current line and insert

A very useful option when modifying a document is to delete a section of text you wish to replace just before entering insert mode. This is done by the *change* **c** command. As the other commands in this section **c** will put you into INSERT mode but you can specify which portion of the text needs to be deleted before. For example:

c$

will delete all the text from the current cursor position to the end of the line.

Another command used to replace a single character (nothing else!) is **r**. First choose which character needs to be replaced and put the cursor on this character. Next press **r** followed by a new character. The new character will replace the old one. This command will leave the editor in COMMAND and not INSERT mode!

4. Cut and Paste

If you want to delete a single character while in command mode you would use **x** and **dd** would delete the current line. One can then paste the deleted item with the command **p**.

Remark: Nearly all **vi** commands can be repeated by specifying a number in front of the command. You can also apply the command to a text item (such as word., sentence, paragraph ...) by placing the entity after the command.

Examples:
Delete a word:

```
dw
```

Delete text from here to the end of the current line
```
d$
```

Delete text from here to the end of the current paragraph
```
d}
```

One can simultaneously delete an item and switch to insert mode with the **c** command. As usual you can use this command together with a text item such as **w** or **{**.

5. Copy Paste

The copy action in **vi** is the command **y** (for yank, the letter **c** was already taken for *change*), and the paste action is still **p**.

If an entire line is yanked the pasted text will be inserted on the next line below the cursor.

The text selection is made with the familiar text items **w**, **l**, **}**, **$** etc ... There are a few exceptions such as the last example.

Examples:

Copy the text from here to the end of the current line
```
y$
```

Copy the entire current line
```
yy
```

Copy 3 lines
```
3yy
```

☞ The latest deleted item is always buffered and can be pasted with the **p** command. This is equivalent to
a cut-and-paste operation.

6. Search and Replace

Since searching involves pattern matching we find ourselves once again dealing with regular expressions (regex). As many UNIX text manipulation tools such as **grep** or **sed**, **vi** recognises regular expressions too.

To perform a search one must be in COMMAND mode. The **/** (forward slash) command searches forward and the **?** command searches backwards.

One can also perform search and replace operations. The syntax is similar to **sed**.

Example:
Downward search for words beginning with ‘comp’ in all the text
```
/\<comp
```

Upward search for lines starting with the letter z
```
?^z
```

Search in the whole text for the keyword ‘VAR’ and replace it by ‘var’
```
:% s/VAR/var
```

7. Undo and Redo

At this stage is is worth mentioning that one can always undo changes! This must be done in COMMAND mode with the **u** command (works as long as one hasn't yet saved the file). The redo command is **^R**.

8. Running a Shell Command

While in LASTLINE mode everything following an exclamation mark **!** is interpreted as a shell command.

For example while editing lilo.conf or grub.conf you may need to find out the name of the root device. This can be done with:

```
:!df /
```

9. Save and Quit

The command for saving is **:w**. By default the complete document is saved. In some case **vi** will refuse to save changes made to a document because of insufficient rights. In such cases ione can attempt to force a write with **:w!**

One can also specify an alternative name for the file. Portions of the text can be saved to another file while other files can be read and pasted in the current document. Here are the examples which illustrate this.

Examples:
Save the current document as 'newfile'
```
:w newfile
```

Save lines 15 to 24 in a file called 'extract'
```
:w 15,24 extract
```

Read from file 'extract'. The text will be pasted at the cursor
```
:r extract
```

Warning: In the *column mode* context we have the following
- **.** is the current line
- **$** is the end of the document

The following are different ways available to quit **vi**:

:wq	save and quit
:q!	quit but do not save changes
:x	exit and save when changes exist
:quit	same as **:q**
:exit or **:e**	same as **:x**
ZZ	same as **:x**

10. Exercises and Summary

Review Questions (answers p.151)

Yes or No

1. The command 'l' (lower case L) will place the cursor one position to the left _____
2. The command 3dd will delete three lines _____
3. The command 3wd will delete three words _____
4. The command :qw will write and exit _____

Commands

vi action	Description
^,$	beginning and end of line
1G,G	beginning and end of document
b,e	beginning and end of word
(,)	beginning and end of sentence
{,}	beginning and end of paragraph
w,W	word and word including hyphens and punctuations
h,j,k,l	left, up, down, right navigation commands
:!	call a shell command
:quit,:q	quit
:quit!,:q!	force quit, discard changes
:wq	write and quit
:exit,:x,:e,ZZ	exit (saves changes if needed)
/,?	search forwards or backwards
a,A,i,o,O,s,S	start insert mode
c	start insert mode while changing an item
r	replace a single character in command mode
d,dd	delete an item or delete entire line
x	delete a single character
y,yy	yank item or entire line
p	paste content of buffer
u,^R	undo, redo

Exercises

*As root cp **/var/log/messages to /tmp**. Using **vi**'s search and replace utility make each line begin with **print "** and end with **";***

Type "u" to undo all the changes

*Copy **/etc/lilo.conf** to **/tmp**, edit this file and try to copy/paste **yy/p** and cut/paste with **dd/p***

*Investigate the outcome of **:x**, **ZZ**, **:quit**, **:wq**, and **:q!** (which ones save and which one don't?)*

*Investigate the outcome for the various inserting modes: **A**, **a**, **O**, **o**, **S** and **s***

*Optional: If you have time the **vim-enhanced** package installs a program called **vimtutor** which takes you through most common **vi** options.*

The X Environment

Prerequisites

None

Goals

Understand the different components of the "Xwindows" environment
Recognise the function of each configuration file (no editing is expected)
Run applications on any display
Understand the function of a display manager including XDMCP

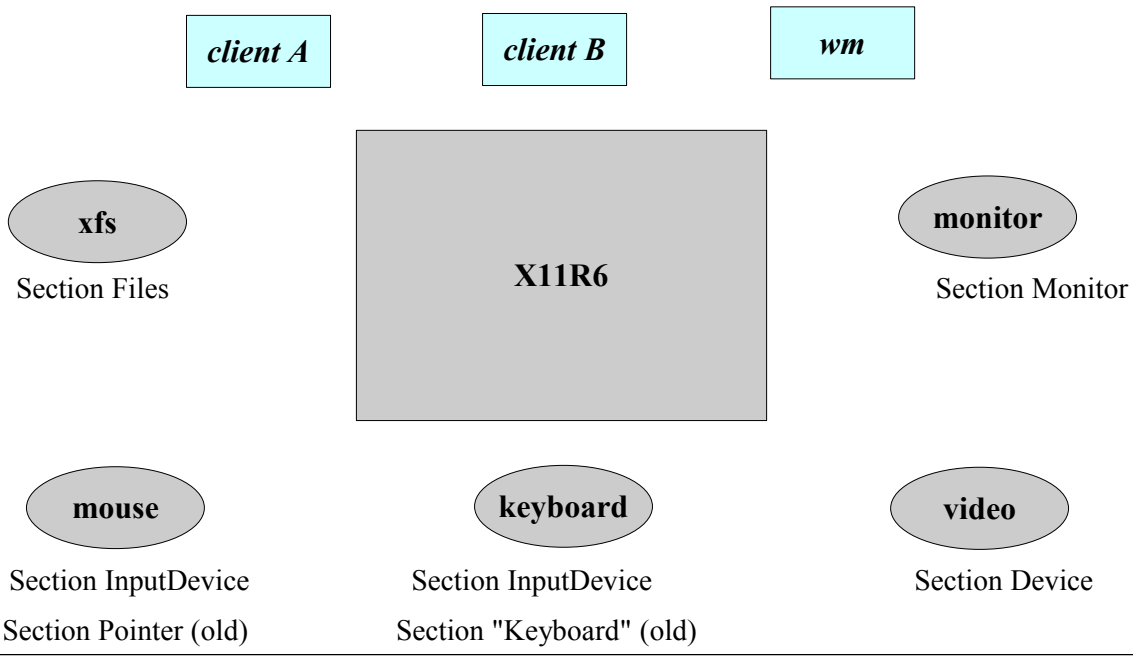
Contents

1. Introduction

The X Windows system was developed as the display component of Project Athena at the Massachusetts Institute of Technology. It is the graphical environment for UNIX. The X Window system for Linux is based on the freely distributable port of X Window version 11 release 6 (Commonly referred to as X11R6).

This freely distributable port is commonly known as **xfree86** for the 80386/80486 and Pentium processor families. Since its initial port, Xfree86 has been ported to other computing platforms, including System V/386 and 386BSD.

X11R6 Components and Configuration Sections



The above diagram shows the components of the X11R6 server. The “Section” names refer to configuration sections in the **XF86Config** configuration file (covered in the next section).

The two clients depicted on top of the server are so-called *x-applications* (e.g xclock or xterm). The window manager is also a client. Window managers add “windowing” facilities around the other x-application clients, allowing functionalities such as window dragging, focus, iconification, etc.

NOTICE:
The X11R6 server is independent from the clients that run on top. Clients are configured using specific configuration files or global files usually called **Xdefaults** or **Xresources**. The X server configuration file will only configure components such as the font server and font directories, mouse, keyboard, monitor resolution and color depth.

2. Configuring X11R6

Two of the configuration utilities provided with the Xfree86 software are the **XF86Setup** and **xf86config** scripts. Other vendors have specific utilities such as:

Xconfigurator, **redhat-config-xfree86** (RedHat)
XFdrake (Mandrake)
sax (Suse)

Once the server has been configured one can change the horizontal and vertical settings for the monitor with **xvidtune**.

All the above mentioned configuration utilities will create and edit the **XF86Config** configuration file. This file is read at start up by the X Server and determines its behaviour. This file is typically found in the /etc/X11 directory, and this is its' full path: **/etc/X11/XF86Config**.

There are 11 configuration sections in the config file, they are listed below:

ServerFlags
Module
InputDevice
Device
VideoAdapter
Monitor
Modes
Screen
ServerLayout
DRI
Vendor

NOTICE:
The obsolete section names *Keyboard* and *Pointer* are still recognised for compatability reasons, the new section name is now *InputDevice*

One of the first sections is the Section "Files". The `FontPath` keyword tells whether to get fonts from a local directory or from a font server. The `RgbPath` keyword is used to indicate the full path to rgb text file used to map color names to RGB notation:

```
Section "Files"
      FontPath "/path/to/fonts/dir/"
      FontPath "trans/hostname:port"
      RgbPath   "/path/to/rgb"
EndSection
```

Where trans is the transport type **unix**, hostname is the fully qualified domain name of the font server, and port is the port to connect to, usually port 7100.

Example:

```
      FontPath "unix/:7100"   # Local Font Server
```

```
      FontPath "unix/myfontserver.mydomain.com:7100"
```

Below is a sample **XF86Config** file:

```
Section "Files"
   RgbPath    "/usr/X11R6/lib/X11/rgb"
   FontPath
"/usr/X11R6/lib/X11/fonts/misc:unscaled,/usr/X11R6/lib/X11/fonts/75dpi:unsc
aled,/usr/X11R6/lib/X11/fonts/100dpi:unscaled,/usr/X11R6/lib/X11/fonts/misc
/"
EndSection

Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "keyboard"
EndSection

Section "InputDevice"
        Identifier  "Mouse0"
        Driver      "mouse"
        Option      "Protocol" "IMPS/2"
        Option      "Device" "/dev/psaux"
        Option      "ZAxisMapping" "4 5"
EndSection

Section "Monitor"
   Identifier      "Primary Monitor"
   VendorName      "Unknown"
   ModelName       "Unknown"
   HorizSync       31.5-37.9
   VertRefresh     55-90
   Modeline  "800x600"    40.00 800 840 968 1056 600 601 605 628 +hsync
+vsync
EndSection

Section "Device"
   Identifier      "Primary Card"
   VendorName      "Unknown"
   BoardName       "None"
   VideoRam        2048
EndSection

Section "Screen"
   Driver          "Accel"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 24
   BlankTime       0
   SuspendTime     0
   OffTime         0

   SubSection "Display"
      Depth        24
      Modes        "800x600"
   EndSubSection
   SubSection "Display"
```

```
        Depth           32
        Modes           "800x600"
```

3. Controlling X clients

Setting Fonts and Colours

X clients are configured using the **.Xresources** or **.Xdefaults** file. These file are kept in the users home directory. It is not automatically created by default, as system-wide defaults are also available for each program.

Below is an extract from a **.Xresources**:

```
xterm_color*background: Black
xterm_color*foreground: Wheat
xterm_color*cursorColor: Orchid
xterm_color*reverseVideo: false
xterm_color*scrollBar: true
xterm_color*saveLines: 5000
xterm_color*reverseWrap: true
xterm_color*font: fixed
xterm_color.geometry: 80x25+20+20
xterm_color*fullCursor: true
xterm_color*scrollTtyOutput: off
xterm_color*scrollKey: on
term_color*VT100.Translations: #override\n\
      <KeyPress>Prior : scroll-back(1,page)\n\
      <KeyPress>Next : scroll-forw(1,page)
xterm_color*titleBar: false
```

Each of these directives is a system default directive that describes how a client will be displayed. Each line consists of the client name followed by an asterisk and the X Window parameter. Through a carefully configured .Xresources file the user can define the way a client will look each time it is started.

The DISPLAY Variable

When an x-application (or X client) is started it needs to know which X server to run on. An X server is referred to as a display. For example the first X server you start (using **startx** for example) is called **:0** the second would be called **:1** and so on. The first X server (or display) running on the host 192.168.1.99 is called **192.168.1.99:0**

Most native X clients such as **xterm** or **xclock** have a `-display` switch which can be used to set the display. But the easiest method is to set the environment variable called DISPLAY!

The next two commands are equivalent:

```
xclock -display 192.168.1.99:0
```

```
DISPLAY=192.168.1.99:0 xclock
```

However the X server on the host 192.168.1.99 will not allow this x-application to run. The user that started the X server on the remote host (192.168.1.99) needs to run the **xhost** command. This tool can selectively add or remove hosts to an access control list.

Example: Allow remote x-applications from host 192.168.1.7 to run on local server

```
xhost + 192.168.1.7
     192.168.1.7 being added to access control list
```

NOTICE

The **xhost** mechanism must be used in conjunction with **xauth** (not part of the LPI objectives). For a remote x-client from 192.168.1.7 to run on our local server we still need to run the following on the local host:

```
xauth extract - $DISPLAY | ssh 192.168.1.7  xauth merge -
```

(Assuming that the user names are the same and that the hostname contained in $DISPLAY can be resolved)

4. Starting X

An X session can be started using 2 methods:

Method 1: From the command line, after logging in onto a virtual terminal the user launches the X Server using a script called **startx**

Method 2: A Display Manager is running prompting the user with a graphical login, this is available for a specific runlevel (on RedHat type distributions this is **runlevel 5)**.

1. From the Command Line

The **startx** script starts **xinit**. The **xinit** script has two main arguments (a) the X server and (b) the **xinitrc** script. The **xinitrc** script will source (read) the files **Xresourses** (controlling the x-applications) and the **Xclients** (choosing a window manager). So we can trace the startup sequence as follows:

startx --> xinit --> X -> xinitrc -> Xclients

2. Using a Display Manager

A display manager will automatically be started if the system is running in a given runlevel (e.g runlevel 5). We first describe the login process, the next section covers more advanced functionalities of a Display Manager. The login process follows the following steps:.

x**dm --> xlogin --> Xsession -->** (optionally) **Xclients** or **~/.Xclients**

Different versions of display managers as well as different Linux distributions may use slightly different steps. In general however, note that **startx** uses **xinit** whereas **xdm** uses **Xsession**.

CUSTOMISING
Each user can further customise their environment by using a **.xinitrc** file. This file will be merged into the system **xinitrc**. The **switdesk** tool allows users to define a custom **.Xclients** file

5. The Display Manager

There are three main display managers, xdm (generic), gdm (GNOME) and kdm (KDE). According to the LPI objectives the configuration files are in the following directories:

/etc/X11/xdm/
/etc/X11/gdm/
/etc/X11/kdm/ (see LPI objectives at the end of this manual p.162)

However, most often the configuration files for **kdm** are in **/usr/share/config/kdm** (also see below).

KDM

This display manager is installed with the KDE desktop environment. It is based on the generic **xdm** display manager and shares many common configuration files. These configuration files for are in **/usr/share/config/kdm**. The file that controls most functionalities is **kdmrc**.

The path to the **kdm** binary is **/usr/bin/kdm**

KDM Configuration files:

kdmrc **Xaccess** (same as xdm) **Xservers** (same as xdm) **Xsession** (same as xdm) **Xsetup** **Xstartup**

GDM

This display manager is distributed with the GNOME desktop environment. The main configuration file is **gdm.conf**

The path to the **gdm** binary is **/usr/bin/gdm**

GDM Configuration Files (**/etc/X11/gdm**)**:**

Sessions/ **gdm.conf**

XDM

The **xdm** display manager is part of the Xfree86 application. The main configuration file is **xdm-config**

The path to the **xdm** binary is **/usr/bin/xdm**

XDM Configuration Files:

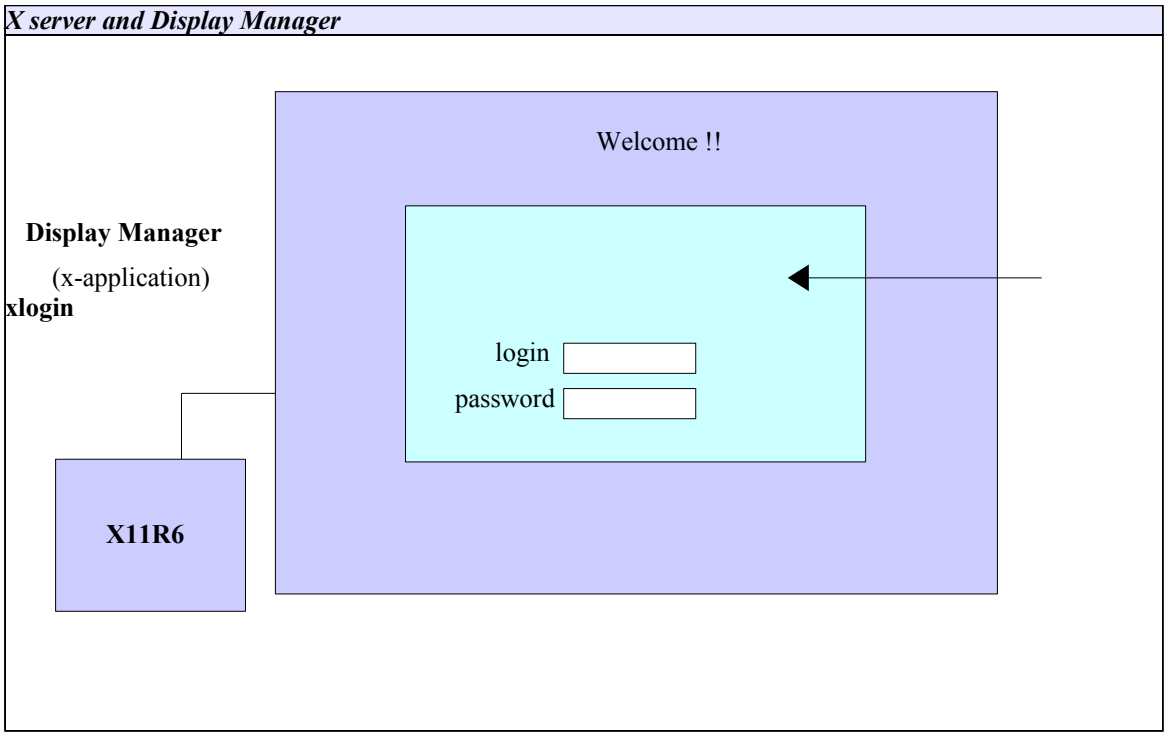
Xaccess **Xresources** **Xsession** **xdm-config** **Xservers** **_**

We will look at the **xdm** configuration files in more detail later in this section.

Display Managers are used mainly in run level 5:

Set default runlevel in **/etc/inittab**
`id:5:initdefault:`

Display managers allow local users to log onto the system using the graphical interface. They can also be used to provide a graphical login interface over the network. For this they use a protocol called **XDMCP** or X Display Manager Control Protocol. By default XDMCP is disabled (we will enable XDMCP as an exercise).



- **Configuration Files**

/etc/X11/xdm/Xrescources

Since the Display Manager is also an x-application, the fonts, the background colors and **xlogin** can be configured with the **Xresourses** file in **/etc/X11/xdm/**. When using **gdm**, the **/etc/X11/gdm/Init/Default** script will source **Xresources**.

/etc/X11/xdm/Xservers

This file simply maps the name of a display with an X server. For example display **:0** is understood to be the local X server. Remember that X always runs on the first free **/dev/tty.**

/etc/X11/xdm/xdm-config

This is the main configuration file for **xdm**. It is also used to enable XDMCP (see exercises)

/etc/X11/xdm/Xaccess

This file is used to enable XDMCP, allowing remote hosts to directly connect to the local server (using **-query**) or query about other display

The Xaccess file

```
# $XConsortium: Xaccess,v 1.5 91/08/26 11:52:51 rws Exp $
#
# Access control file for XDMCP connections
# To control Direct and Broadcast access:
#
#       pattern
#
# To control Indirect queries:
#
#       pattern         list of hostnames and/or macros ...
#
# To use the chooser:
#
#       pattern         CHOOSER BROADCAST
#
# or
#
#       pattern         CHOOSER list of hostnames and/or macros ...
#
# To define macros:
#
#        %name          list of hosts ...
#
# The first form tells xdm which displays to respond to itself.
# The second form tells xdm to forward indirect queries from hosts matching
# the specified pattern to the indicated list of hosts.
# The third form tells xdm to handle indirect queries using the chooser;
# the chooser is directed to send its own queries out via the broadcast
# address and display the results on the terminal.
# The fourth form is similar to the third, except instead of using the
# broadcast address, it sends DirectQuerys to each of the hosts in the list
#
# In all cases, xdm uses the first entry which matches the terminal;
# for IndirectQuery messages only entries with right hand sides can
# match, for Direct and Broadcast Query messages, only entries without
# right hand sides can match.
#

*                                       #any host can get a login window

#
# To hardwire a specific terminal to a specific host, you can
# leave the terminal sending indirect queries to this host, and
# use an entry of the form:
#

#terminal-a     host-a

# The nicest way to run the chooser is to just ask it to broadcast
# requests to the network - that way new hosts show up automatically.
# Sometimes, however, the chooser can't figure out how to broadcast,
# so this may not work in all environments.
#

*               CHOOSER BROADCAST       #any indirect host can get a chooser

# If you'd prefer to configure the set of hosts each terminal sees,
# then just uncomment these lines (and comment the CHOOSER line above)
# and edit the %hostlist line as appropriate
```

```
#
#%hostlist      host-a host-b
#*              CHOOSER %hostlist       #
```

The Xservers file

```
# $XConsortium: Xserv.ws.cpp,v 1.3 93/09/28 14:30:30 gildea Exp $
#
#
# $XFree86: xc/programs/xdm/config/Xserv.ws.cpp,v 1.1.1.1.12.2 1998/10/04 15:23:14 hohndel Exp $
#
# Xservers file, workstation prototype
#
# This file should contain an entry to start the server on the
# local display; if you have more than one display (not screen),
# you can add entries to the list (one per line).  If you also
# have some X terminals connected which do not support XDMCP,
# you can add them here as well.  Each X terminal line should
# look like:
#       XTerminalName:0 foreign
#
:0 local /usr/X11R6/bin/X
```

Since the Display Manager is also an *x-application t*he **Xrescources** file is similar to the **.Xrescources** file except that it controls how the login screen is displayed .

Sample Xrescources file

```
! $XConsortium: Xresources /main/8 1996/11/11 09:24:46 swick $
xlogin*borderWidth: 3
xlogin*greeting: CLIENTHOST
xlogin*namePrompt: login:\040
xlogin*fail: Login incorrect
#ifdef COLOR
xlogin*greetColor: CadetBlue
xlogin*failColor: red
*Foreground: black
*Background: #fffff0
#else
xlogin*Foreground: black
xlogin*Background: white
#endif

XConsole.text.geometry:       480x130
XConsole.verbose:     true
XConsole*iconic:      true
XConsole*font:        fixed
```

Sample xdm-config file

```
! $XFree86: xc/programs/xdm/config/xdm-conf.cpp,v 1.1.1.2.4.2 1999/10/12 18:33:29 hohndel Exp $
!
DisplayManager.servers:               /etc/X11/xdm/Xservers
DisplayManager.accessFile:    /etc/X11/xdm/Xaccess
! All displays should use authorization, but we cannot be sure
! X terminals will be configured that way, so by default
! use authorization only for local displays :0, :1, etc.
DisplayManager._0.authorize:  true
DisplayManager._1.authorize:  true
!
DisplayManager*resources:     /etc/X11/xdm/Xresources
DisplayManager*session:               /etc/X11/xdm/Xsession
```

```
DisplayManager*authComplain:  false
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:   0
```

6. Troubleshooting X Clients

Occasionally X Clients wont terminate properly leaving *zombie* processes. A zombie process in one whose parent processes has terminated, and cannot clear references to the child process. When a child process' parent exits leaving the child process still running, this is usually visible by running ps which will reveal the child process being owned by PID 1 (init). These processes should be killed because they may be using CPU resources. Killing such a process requires the user to be the user who owns the process, or root. It might be necessary to use the –9 option to actually kill these processes.

7. Choosing a Window Manager

The area that is commonly referred to as the desktop is also known in the X Window world as the screen. It covers the entire area of your monitor display. The root window is the background of your screen, typically used to display a colour or picture. The window manager provides an interface between the user and the X server. It is virtually impossible to use X without a window manager, because it provides the title bar and the familiar buttons with which you manipulate the display.

Information on available window managers is available from the Window Managers website at http://www.PliG.org/xwinman. Many of the Linux versions of these window managers are available at ftp://metalab.unc.edu/pub/Linux/X11/window-managers.

In addition to the different window managers there are also various desktop environments, among which the most common are KDE, GNOME and XFCE.

Below is a brief list of integrated window managers:

Enlightenment
fvwm
icewm
amiWM
mlvwm
dfm
olwm
olvwm
mwm
Window Maker

9. Exercises and Summary

Review Questions (answers p.151)

Yes or No

1. The configuration file for a window manager is **XF86Config** ____

2. An **x-client** can be configured to run on any X server accessible on a network ____

3. A display manager is a program that manages pixels when using a windowing environment ____

4. A user running an X server can disable access control using **xhost** ____

5. The XDMCP protocol is used by display managers to display a graphical login on remote computers ____

6. If a system is not running a display manager one usually starts a windowing environment with the **xinit** tool ____

Glossary

Term	Definition
DISPLAY	Shell environmental variable used to notify x-applications which display (in this case, which X server) to run on
XDMCP	a protocol that provides a uniform mechanism for an autonomous display to request login service from a remote host
run level	a software configuration of the system which allows only a selected group of processes to exist – see **init(8)**
x-client or x-application	In this manual the terms are used to describe applications such as **xterm** or **xclock** that run on an X server
desktop environment	a suite of applications (including a window manager) originally designed to use the entire surface of the screen as a 'desktop'. Integrated applications often include features such as short cuts for copy pasting, drag and drop, etc. Example desktops are **XFCE**, **GNOME** and **KDE**
display manager	an x-application which runs at a specific run level (often run level 5) displaying a graphical login interface. Display managers also handle the XDMCP protocol. Most common display managers are **xdm** (generic X11R6), **gdm** (part of the GNOME desktop suite) and **kdm** (part of the KDE desktop suite)
window manager	a special x-application which can move, resize or iconify windows. In some cases window managers also provide a task bar as well as drop down menus for quick launch. Examples are **twm**, **fluxbox**, **icewm** etc ...
session manager	provides a way to save a window session once the user logs out

X11 Configuration File

File	Description
XF86Config	main configuration file for the X11R6 server

Display Manager Configuration Files

System File	Description
gdm.conf	main configuration file for **gdm**
kdmrc	main configuration file for **kdm**
Xaccess	one of the files used to allow/disallow XDMCP access
xdm-config	main configuration file for **xdm**
Xrescources	customisation file used by display managers
Xservers	configures how many display managers to start on a system. The file associates a display (default :0) to a server (usually X11R6)
Xsession	script used by display managers to start a specific windowing environment
Xclients	script used by Xsession as well as **xdm** to start a system wide windowing environment

User Customisations

File	Description
~/.Xresources ~/.Xdefaults	files used to control the way x-applications start (e.g position, font size, colours, etc)
~/.xinitrc	when using startx users can specify which window manager or desktop environment to start
~/.Xclients	when using a display manager users can specify which window manager or desktop environment to start

Commands

Command	Description
gdm p.141	**gdm(1)** - GDM is a replacement for XDM, the X Display Manager. Unlike its competitors (X3DM, KDM, WDM) GDM was written from scratch and does not contain any original XDM / X Consortium code. GDM runs and manages the X servers for both local and remote logins (using XDMCP)
kdm p.141	a display manager provided by the KDE desktop, it has many configuration files in common with **xdm**
startx p.140	a script that calls **xinit** and starts the X Window System, that is an X server together with a window manager or a desktop

xauth p.140	(not an LPI objective) **xauth(1)** - The *xauth* program is used to edit and display the authorization information used in connecting to the X server. Normally *xauth* is not used to create the authority file entry in the first place; *xdm* does that – see also **pam_xauth(8)** for further reading
xdm p.141	the X Display Manager made available as part of the X server
xf86config p.137	program that interactively creates a XF86Config file
XF86Setup p.137	graphical program that interactively creates or modifies an XF86Config file
xhost p.140	**xhost(1)** - The *xhost* program is used to add and delete host names or user names to the list allowed to make connections to the X server. In the case of hosts, this provides a rudimentary form of privacy control and security. It is only sufficient for a workstation (single user) environment,
xinit p.140	**xinit(1)** - The *xinit* program is used to start the X Window System server and a first client program (usually a window manager) on systems that cannot start X directly from */etc/init* or in environments that use multiple window systems. When this first client exits, *xinit* will kill the X server and then terminate
xinitrc p.141	script launched by **xinit** containing a list of clients to be started. Additional clients (such as terminals, clocks, etc) should be started in the background. The first client however (usually the window manager) should be started in the forground in order to prevent **xinitrc** from exiting – see **xinit(1)**
xvidtune p.137	**xvidtune(1)** - Without any options it presents the user with various buttons and sliders that can be used to interactively adjust existing video modes. It will also print the settings in a format suitable for inclusion in an XF86Config file

Exercises

Before starting make sure you are running in runlevel **3**.

```
init 3
```

1. Log into a virtual terminal (e.g Alt+F1)

2. As root save the existing configuration file **/etc/X11/XF86Config** and try out the various configuration tools:
Redhat: Xconfigurator, redhat-config-xfree86 (8.0)
Mandrake : XFdrake
Suse: sax

XF86Setup
xf86config
X (this is the X11 server itself, use the **-configure** flag)

3. Start the X server by typing **X**. This will start X11R6 alone with no window managers. Return to a virtual terminal (e.g Ctrl+Alt+F2) and get the command line back. Then do the following:

```
export DISPLAY=localhost:0
xterm&
```

Go back into X by typing Ctrl+Alt+F7 (if you haven't changed the defaults in /etc/inittab...). You should have an xterminal running. Next type in this terminal:

```
twm&
```

What has happened? Can you kill **twm** without killing X? Go back to a virtual terminal (e.g Ctrl+Alt+F2) and type:

```
X :1
```

Log into another virtual terminal (e.g tty3) and type:

```
export DISPLAY=:1; xterm&
```

You now have two X servers running on screen **0** and **1**. How do you switch from one to another?

4. Setting up XDMCP

For this to work make sure the line containing an '*****' is uncommented in **/etc/X11/xdm/Xaccess**.

If you are using **xdm** or **kdm** comment out the line in **xdm-config** as follows

```
!DisplayManager.requestPort:     0
```

This line is originally uncommented and allows only local login requests on screen **0** (more secure).

If you are using **gdm** then you will also need to edit **gdm.conf** and put
`enable=true`

This will turn off the default security settings for **gdm**.

If your IP is 1.2.3.4 then users on your network can start an X session with:

```
X -query 1.2.3.4 :1
```

or

```
X -indirect 1.2.3.4 :1
```

Answers to Revision Questions

Installation (p.8)
1.No – the rawrite tool runs under DOS. The tool used to copy image files onto a block device is **dd**
2.Yes – any device that can hold data (i.e not an extended partition) can be used as the root device

Hardware Suppport (p.24)
1. No – the root filesystem can be on any disk type
2. No – the kernel module only handles the USB controller. The kernel will notify events to a USB user agent which then uses user-maps to configure the particular device

Managing Devices (p.36)
1. No – you need to re-run **/sbin/lilo** every time you change something in **/etc/lilo.conf**
2. Yes – GRUB only need to be installed once
3. No – the 1024 cylinder limit affects the second stage bootloader
4. Yes – one cannot set quotas on a directory

The Linux Filesystem (p.52)
1. No – programs stored in **/usr/** are not essential for booting up the system
2. No – a user's home directory can be anywhere on the system
3. Yes

The Command Line (p.67)
1. Yes
2. No – STDOUT can be *redirected* to a file
3. Yes
4. No – commands are generally stored in a file called .bash_history in the user's home directory

File Management (p.78)
1. Yes
2. No – **cd** with no argument will also go to the user's home directory
3. No – you must use 'mkdir -p' to make directories which contain subdirectories
4. Yes
5. No – the correct command is `'ln -s FILE FILE-LINK'`
6. No – paths beginning with a forward slash are *absolute* paths

Process Management (p.87)
1. No – **kill** will send a signal, some signals may simply pause a process, others force daemons to re-read their configuration file. However the default signal (15 or SIGTERM) will attempt to terminate a process.
2. Yes
3. Yes
4. No

Text Processing (p.97)
1. Yes
2. No – use **tail**
3. Yes – this is the case with all the text tools
4. No – all the text tools print the altered data to STDOUT and don't alter the original file

Software Installation (p.118)
1. No – one uses **make**
2. Yes – a makefile is unique for each project. The **make** tool will only read a makefile in the current directory (i.e the directory the **make** command is started from)
3. Yes
4. Yes
5. Yes

6. No
7. Yes

Advanced Text Manipulation (p.126)
1. Yes – the "e?" expression reads 'match 0 or 1 e'
2. Yes – the "a*" expression reads ' match 0 or any a'
3. Yes – the "e+" expressiom reads 'match at least 1 or any e'
4. Yes – notice that the "a*" is at the end of the expression. This means that 'baz' followed by anything will also be matched

Using vi (p.133)
1. No – look at the position of **hjkl** on the keyboard. The **l** key is on the right and will cause the cursor to move to the right
2. Yes
3. No – **3w** will move the cursor three words forward, then **d** is an incomplete command. Deleting three words can be done with **d3w**
4. No – the commands **:qw** will be read from left to right. So quit (**q**) must always be last

The X Environment (p.146)
1. No – the **XF86Config** file only configures the X server. Window managers often use directories stored in user's home directories
2. Yes – this is true although firewalls can prevent **x-applications** to connect to remote X servers
3. No – a display manager handles login access on a display (local X server or remote X server using XDMCP)
4.Yes – **n**ote that on most of the recent Linux distributions **xhost** must be used in conjunction with **xauth**
5. Yes
6. No – one usually uses **startx**

LPI 101 Objectives

http://www.lpi.org/en/obj_101.html
Thu Nov 4 04:45:58 2004

Exam 101: Detailed Objectives

This is a required exam for LPI certification Level 1. It covers basic system administration skills that are common across all distributions of Linux.

Each objective is assigned a weighting value. The weights range roughly from 1 to 10, and indicate the relative importance of each objective. Objectives with higher weights will be covered in the exam with more questions.

Topic 101: Hardware & Architecture

* **1.101.1 Configure Fundamental BIOS Settings**
  Modified: 2003-March-17
  Maintainer: Kara Pritchard
  Weight: 1

Description: Candidates should be able to configure fundamental system hardware by making the correct settings in the system BIOS. This objective includes a proper understanding of BIOS configuration issues such as the use of LBA on IDE hard disks larger than 1024 cylinders, enabling or disabling integrated peripherals, as well as configuring systems with (or without) external peripherals such as keyboards. It also includes the correct setting for IRQ, DMA and I/O addresses for all BIOS administrated ports and settings for error handling (p. 11).

Key files, terms, and utilities include:

/proc/ioports (p. 11)
/proc/interrupts (p. 11)
/proc/dma (p. 11)
/proc/pci (p. 11)

* **1.101.3 Configure Modem and Sound cards**
  Modified: 2003-March-17
  Maintainer: Kara Pritchard
  Weight: 1

Description: Ensure devices meet compatibility requirements (particularly that the modem is NOT a win-modem), verify that both the modem and sound card are using unique and correct IRQ's, I/O, and DMA addresses, if the sound card is PnP install and run sndconfig (p.23)and isapnp (p.12), configure modem for outbound dial-up, configure modem for outbound PPP | SLIP | CSLIP connection, set serial port for 115.2 Kbps (p.18)

* **1.101.4 Setup SCSI Devices**
  Modified: 2003-March-17
  Maintainer: Kara Pritchard
  Weight: 1

Description: Candidates should be able to configure SCSI devices using the SCSI BIOS as well as the necessary Linux tools. They also should be able to differentiate between the various types of SCSI. This objective includes manipulating the SCSI BIOS to detect used and available SCSI IDs and setting the correct ID number for different devices especially the boot device. It also includes managing the settings in the computer's BIOS to determine the desired boot sequence if both SCSI and IDE drives are used.

Key files, terms, and utilities include:

SCSI ID(p.14)
/proc/scsi/(p.15)
scsi_info(p.15)

* **1.101.5 Setup different PC expansion cards**
Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to configure various cards for the various expansion slots. They should know the differences between ISA and PCI cards with respect to configuration issues. This objective includes the correct settings of IRQs, DMAs and I/O Ports of the cards, especially to avoid conflicts between devices. It also includes using isapnp if the card is an ISA PnP device.
Key files, terms, and utilities include:

/proc/dma
/proc/interrupts
/proc/ioports
/proc/pci
pnpdump(8) (p.12)
isapnp(8) (p.12)
lspci(8) (p.11)

* **1.101.6 Configure Communication Devices**
Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 1

Description: Candidates should be able to install and configure different internal and external communication devices like modems, ISDN adapters (p.20), and DSL switches (p.20). This objective includes verification of compatibility requirements (especially important if that modem is a winmodem [p.20]), necessary hardware settings for internal devices (IRQs, DMAs, I/O ports), and loading and configuring suitable device drivers. It also includes communication device and interface configuration requirements, such as the right serial port for 115.2 Kbps (p.18), and the correct modem settings for outbound PPP connection(s) (p.18).
Key files, terms, and utilities include:

/proc/dma
/proc/interrupts
/proc/ioports
setserial(8)

* **1.101.7 Configure USB devices**
Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 1

Description: Candidates should be able to activate USB support, use and configure different USB devices. This objective includes the correct selection of the USB chipset and the corresponding module. It also includes the knowledge of the basic architecture of the layer model of USB as well as the different modules used in the different layers.

Key files, terms, and utilities include:

lspci(8)

usb-uhci.o(p.13)
usb-ohci.o(p.13)
/etc/usbmgr/(p.Error: Reference source not found14
usbmodules
/etc/hotplug(p.13)

Topic 102: Linux Installation & Package Management

* 1.102.1 Design hard disk layout

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidates should be able to design a disk partitioning scheme for a Linux system. This objective includes allocating filesystems or swap space to separate partitions or disks, and tailoring the design to the intended use of the system. It also includes placing /boot on a partition that conforms with the BIOS' requirements for booting.

Key files, terms, and utilities include:

/ (root) filesystem (p.5 and p.40)
/var filesystem (p.41)
/home filesystem (p.41)
swap space (p.6)
mount points (p.5)
partitions (p.5, p.28)
cylinder 1024 (p.32 and p.36)

* 1.102.2 Install a boot manager

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 1

Description: Candidate should be able to select, install, and configure a boot manager. This objective includes providing alternative boot locations and backup boot options (for example, using a boot floppy).

Key files, terms, and utilities include:

/etc/lilo.conf (p.32)
/boot/grub/grub.conf (p.32)
lilo (p.32)
grub-install (p.32)
MBR (p.31)
superblock (p.42)
first stage boot loader (p.31)

* 1.102.3 Make and install programs from source

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidates should be able to build and install an executable program from source. This objective includes being able to unpack a file of sources. Candidates should be able to make simple customizations to the Makefile, for example changing paths or adding extra include directories.
Key files, terms, and utilities include:

gunzip
gzip (p.105)
bzip2 (p.105)
tar (p.106)
configure (p.107)
make (p.107)

* 1.102.4 Manage shared libraries

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to determine the shared libraries that executable programs depend on and install them when necessary. Candidates should be able to state where system libraries are kept.

Key files, terms, and utilities include:

ldd (p.103)
ldconfig (p.104)
/etc/ld.so.conf (p.104)
LD_LIBRARY_PATH (p.104)

* 1.102.5 Use Debian package management

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 8

Description: Candidates should be able to perform package management skills using the Debian package manager. This objective includes being able to use command-line and interactive tools to install, upgrade, or uninstall packages, as well as find packages containing specific files or software (such packages might or might not be installed). This objective also includes being able to obtain package information like version, content, dependencies, package integrity and installation status (whether or not the package is installed).

Key files, terms, and utilities include:
unpack (p.114, p.114, p.115)
configure (p.114, p.115)
/etc/dpkg/dpkg.cfg (p.114)
/var/lib/dpkg/* (p.115)
/etc/apt/apt.conf (p.116)
/etc/apt/sources.list (p.116)
dpkg (p.113, .115)
dselect (p.113)

dpkg-reconfigure
apt-get (p.116)
alien (p.117, p.119)

* 1.102.6 Use Red Hat Package Manager (RPM)

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 8

Description: Candidates should be able to perform package management under Linux distributions that use RPMs for package distribution. This objective includes being able to install, re-install, upgrade, and remove packages, as well as obtain status and version information on packages. This objective also includes obtaining package information such as version, status, dependencies, integrity, and signatures. Candidates should be able to determine what files a package provides, as well as find which package a specific file comes from.
Key files, terms, and utilities include:

/etc/rpmrc (p.119)
/usr/lib/rpm/* (p.119)
rpm (p.108, p.119)
grep

Topic 103: GNU & Unix Commands

* 1.103.1 Work on the command line

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidates should be able to Interact with shells and commands using the command line. This includes typing valid commands and command sequences, defining, referencing and exporting environment variables, using command history and editing facilities, invoking commands in the path and outside the path, using command substitution, applying commands recursively through a directory tree and using man to find out about commands.
Key files, terms, and utilities include:

.
bash
echo (p.57)
env (p.58)
exec (p.64)
export (p.58)
man (p.66, p.65)
pwd (p.72)
set (p.58)
unset (p.58)
~/.bash_history (p.63)
~/.profile

* 1.103.2 Process text streams using filters

Modified: 2003-March-17

Maintainer: Kara Pritchard
Weight: 6

Description: Candidates should should be able to apply filters to text streams. Tasks include sending text files and output streams through text utility filters to modify the output, and using standard UNIX commands found in the GNU textutils package.

Key files, terms, and utilities include:

cat (p.91)
cut (p.94)
expand (p.93)
fmt (p.95)
head (p.92)
join (p.95)
nl (p.93)
od (p.93)
paste (p.95)
pr (p.96)
sed (p.124)
sort (p.95)
split (p.93)
tac (p.91)
tail (p.92)
tr (p.96)
unexpand (p.93)
uniq (p.94)
wc (p.92)

* 1.103.3 Perform basic file management

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to use the basic UNIX commands to copy, move, and remove files and directories. Tasks include advanced file management operations such as copying multiple files recursively, removing directories recursively, and moving files that meet a wildcard pattern. This includes using simple and advanced wildcard specifications to refer to files, as well as using find to locate and act on files based on type, size, or time.

Key files, terms, and utilities include:

cp (p.74)
find (p.72)
mkdir (p.74)
mv (p.75)
ls (p.73)
rm (p.74)
rmdir(p.74)
touch (p.76)
file globbing (p.67)

* 1.103.4 Use streams, pipes, and redirects

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidates should be able to redirect streams and connect them in order to efficiently process textual data. Tasks include redirecting standard input, standard output, and standard error, piping the output of one command to the input of another command, using the output of one command as arguments to another command and sending output to both stdout and a file.
Key files, terms, and utilities include:

tee (p.62)
xargs (p.72)
< (p.60)
<< (p.64)
>(p.60)
>>(p.60)
| (p.61)
` ` (p.63)

* 1.103.5 Create, monitor, and kill processes

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidates should be able to manage processes. This includes knowing how to run jobs in the foreground and background, bring a job from the background to the foreground and vice versa, start a process that will run without being connected to a terminal and signal a program to continue running after logout. Tasks also include monitoring active processes, selecting and sorting processes for display, sending signals to processes, killing processes and identifying and killing X applications that did not terminate after the X session closed.

Key files, terms, and utilities include:

& (p.86)
bg (p.85)
fg (p.85)
jobs (p.86)
kill (p.83)
nohup (p.86)
ps (p.82)
top (p.83)

* 1.103.6 Modify process execution priorities

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should should be able to manage process execution priorities. Tasks include running a program with higher or lower priority, determining the priority of a process and changing the priority of a running process.

Key files, terms, and utilities include:

nice (p.84)
ps (p.82)
renice (p.85)
top (p.83)

* 1.103.7 Search text files using regular expressions

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to manipulate files and text data using regular expressions. This objective includes creating simple regular expressions containing several notational elements. It also includes using regular expression tools to perform searches through a filesystem or file content.
Key files, terms, and utilities include:

grep (p.122)
regexp ()
sed (p.124)

* 1.103.8 Perform basic file editing operations using vi

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 1

Description: Candidates should be able to edit text files using vi. This objective includes vi navigation, basic vi nodes, inserting, editing, deleting, copying, and finding text.
Key files, terms, and utilities include:

vi (p.129)
/ (p.131), ? (p.Error: Reference source not found131)
h,j,k,l (p.129)
G, H, L
i (p.130), c (p130Error: Reference source not found), d, dd (p.130), p (p.Error: Reference source not found130), o (p.Error: Reference source not found130), a (p.130)
ZZ (p.132), :w! (p.132), :q! (p.132), :e! (p.132)
:! (p.132)

Topic 104: Devices, Linux Filesystems, Filesystem Hierarchy Standard

* 1.104.1 Create partitions and filesystems

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to configure disk partitions and then create filesystems on media such as hard disks. This objective includes using various mkfs commands to set up partitions to various filesystems, including ext2, ext3, reiserfs, vfat, and xfs.
Key files, terms, and utilities include:

fdisk (p.29)
mkfs (p.42)

* 1.104.2 Maintain the integrity of filesystems

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to verify the integrity of filesystems, monitor free space and inodes, and repair simple filesystem problems. This objective includes the commands required to maintain a standard filesystem, as well as the extra data associated with a journaling filesystem.
Key files, terms, and utilities include:

du (p.45)
df (p.45)
fsck (p.43)
e2fsck (p.43)
mke2fs (p.42)
debugfs (p.43)
dumpe2fs (p.43)
tune2fs (p.34)

* 1.104.3 Control mounting and unmounting filesystems

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to configure the mounting of a filesystem. This objective includes the ability to manually mount and unmount filesystems, configure filesystem mounting on bootup, and configure user mountable removeable filesystems such as tape drives, floppies, and CDs.
Key files, terms, and utilities include:

/etc/fstab (p.33)
mount (p.33)
umount (p.34)

* 1.104.4 Managing disk quota

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidates should be able to manage disk quotas for users. This objective includes setting up a disk quota for a filesystem, editing, checking, and generating user quota reports.
Key files, terms, and utilities include:

quota (p.35)
edquota (p.35)
repquota (p.35)
quotaon (p.35)

* 1.104.5 Use file permissions to control access to files

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidates should be able to control file access through permissions. This objective includes access permissions on regular and special files as well as directories. Also included are access modes such as suid, sgid, and the sticky bit, the use of the group field to grant file access to workgroups, the immutable flag, and the default file creation mode.
Key files, terms, and utilities include:

chmod (p.46)
umask (p.47)
chattr (p.49)

* 1.104.6 Manage file ownership

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 1

Description: Candidates should be able to control user and group ownership of files. This objective includes the ability to change the user and group owner of a file as well as the default group owner for new files.
Key files, terms, and utilities include:

chmod (p. 46)
chown (p.46)
chgrp (p.46)

* 1.104.7 Create and change hard and symbolic links

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 1

Description: Candidates should be able to create and manage hard and symbolic links to a file. This objective includes the ability to create and identify links, copy files through links, and use linked files to support system administration tasks.
Key files, terms, and utilities include:

ln (p.75)

* 1.104.8 Find system files and place files in the correct location

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidates should be thouroughly familiar with the Filesystem Hierarchy Standard, including typical file locations and directory classifications. This objective includes the ability to find files and commands on a Linux system.
Key files, terms, and utilities include:

find (p.72)
locate (p.73)
slocate (p.73)
updatedb (p.73)

whereis (p.73)
which (p.73)
/etc/updatedb.conf (p.73)

Topic 110: The X Window System

* 1.110.1 Install & Configure XFree86

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 5

Description: Candidate should be able to configure and install X and an X font server. This objective includes verifying that the video card and monitor are supported by an X server, as well as customizing and tuning X for the videocard and monitor. It also includes installing an X font server, installing fonts, and configuring X to use the font server (may require a manual edit of /etc/X11/XF86Config in the "Files" section).
Key files, terms, and utilities include:

XF86Setup (p.137)
xf86config (p.137)
xvidtune (p.137)
/etc/X11/XF86Config (p.Error: Reference source not found)
.Xresources (p.139)

* 1.110.2 Setup a display manager

Modified: 2003-March-17
Maintainer: Kara Pritchard
Weight: 3

Description: Candidate should be able setup and customize a Display manager. This objective includes turning the display manager on or off and changing the display manager greeting. This objective includes changing default bitplanes for the display manager. It also includes configuring display managers for use by X-stations. This objective covers the display managers XDM (X Display Manger), GDM (Gnome Display Manager) and KDM (KDE Display Manager).
Key files, terms, and utilities include:

/etc/inittab
/etc/X11/xdm/* (p.141)
/etc/X11/kdm/* (p.141)
/etc/X11/gdm/* (p.141)

* 1.110.4 Install & Customize a Window Manager Environment

Modified: 2003-March-17

Maintainer: Kara Pritchard
Weight: 5

Description: Candidate should be able to customize a system-wide desktop environment and/or window manager, to demonstrate an understanding of customization procedures for window manager menus and/or desktop panel menus. This objective includes selecting and configuring the desired x-terminal (xterm, rxvt, aterm etc.), verifying and resolving library dependency issues for X applications, exporting X-display to a client workstation.
Key files, terms, and utilities include:

.xinitrc (p.141)
.Xdefaults (p.139)
xhost (p.140)
DISPLAY environment variable (p.139)

http://www.lpi.org/en/obj_101.html
Copyright © 1999, 2000, 2001, 2002, 2003 Linux Professional Institute. All rights reserved.

GNU FDL License Agreement

Copyright (c) 2004 LinuxIT.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with the Invariant Sections being History, Acknowledgements, with the
Front-Cover Texts being “released under the GFDL by LinuxIT”.

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- **O.** Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

INDEX

Index

Index

Index