Study Guide for

Linux System Administration 1

Lab work for LPI 101 (RPM)

released under the GFDL by LinuxIT

GNU FDL License Agreement

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 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.

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 userfriendly 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.

Audience

This course is designed as a 3 to 4 days practical course preparing for the LPI 101 exam. It is recommended that candidates have at least one year experience doing Linux administration professionally. However for those who are ready for a challenge the training is designed to provide as much insight and examples as possible to help non specialists understand the basic concepts and command sets which form the core of Linux computing.

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 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 installation exercises suggest a network installation (prepare floppies + installation server).

The exercises in the device and filesystem sections 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
```

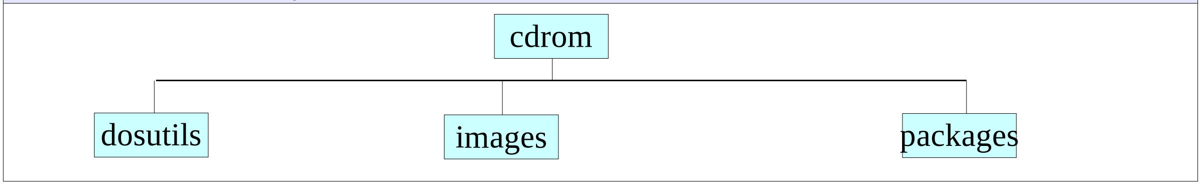
Installation

Rather than discuss a step by step installation we will introduce in this module the installation CD, the different installation methods and the “rescue mode”.

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**

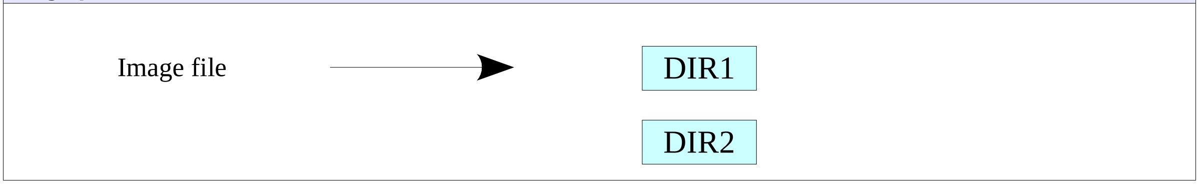
images: This directory contains various “images”. These are special flat files often containing directory structures. An initial ramdisk (initrd) is an example of an image file. There are different types of images necessary to:

- 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/fdo
```

Installation

The first part of the installation is text based and will allow you to setup the network parameters needed. The rest of the installation can be done via FTP, NFS or HTTP. Protocols that allow a full mount (NFS) allow the install to be done in graphical mode, while file retrieval protocols (FTP HTTP) allow only text mode.

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.

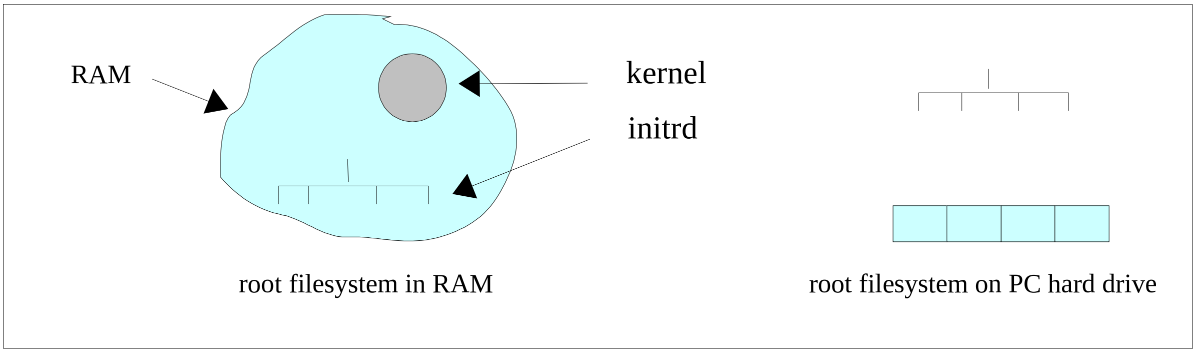
The Linux operating system runs entirely in RAM. The aim is to access the root filesystem on the PC hard drive. Most rescue disks can determine this automatically. Assuming the root filesystem was found on the first logical partiton of the computer's first IDE disk (/dev/hda5), the rescue disk script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

Rescue mode



Getting started

<u>Old Method</u>:

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

<u>New Method:</u>

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

```
chroot /mnt/sysimage
```

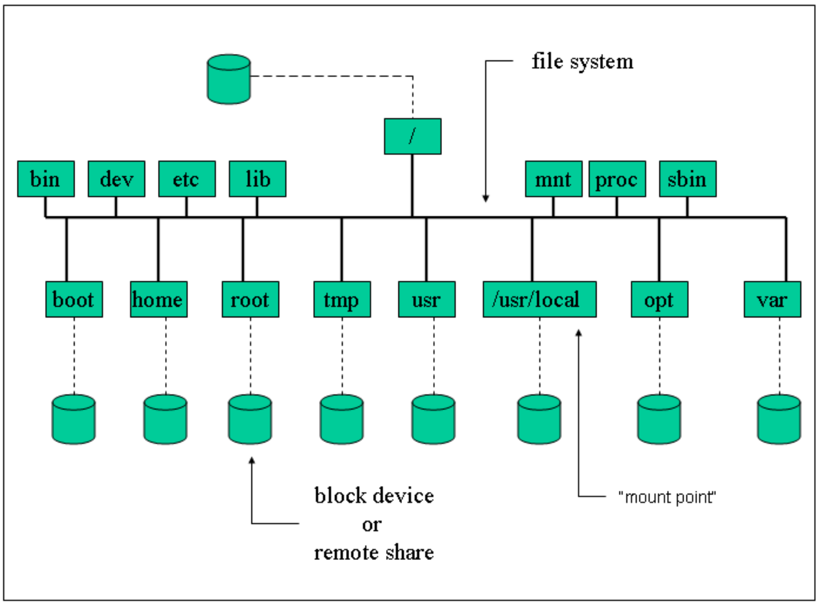
5. Partitioning Schemes

The figure below shows a possible partitioning scheme. The File System layout is a tree of directories and subdirectories. The physical resources with the data are **mounted** at specific locations on the file system called **mount points**.

The root of the tree structure is called **root** and is represented by a forward slash "**/**". At boot time, the boot loader is told which device to mount at root. The leaves in this tree structure are subdirectories.

During installation you will partition the hard drive and assign a size and a mount point for each partition.

Fig 2: Mount points on the file system



6. Easy Dual Booting

(This section is not for exam purposes).

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 Windows 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 and the second half must be large enough to hold a Linux installation.

Notice: The average amount of space needed for a recent Linux distribution is 1GB.

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

1. Do a network installation using the ready prepared bootnet.img floppy disk.

(i) Choose “Custom System” installation

(ii) Partition the disk 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)

(iii) Install LILO on /dev/hda2 or /dev/hda3. 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)

“Network Support”
“Classic X Window System”
“X Window System”
“Software Development” [This is important, we will need this to compile packages later]

(v) Don’t create a bootable floppy

2. Rescue the system:

(i) Reboot with the bootnet.img floppy disk (or the installation CDROM of you have it). This time type

```
linux rescue
```

at the prompt.

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

(iii) 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

1. Memory Support

The system’s RAM is first detected by the BIOS. All types of RAM (EDO, DRAM and SDRAM) are recognised by the Linux kernel. There can be problems with old hardware when the BIOS cannot detect 64MB of RAM or more. In this case one needs to passe parameters to the kernel at boot time.

When using LILO insert the following into **/etc/lilo.conf**:

```
append=”mem=<amount of ram>M”
```

Remember to run **/sbin/lilo**.

If you are using GRUB add the following to **/etc/grub.conf** on the line beginning with *kernel* :

```
kernel      vmlinuz     mem=<amount of ram>M
```

2. 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. This displays the kernel messages logged at boot time. 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	

?M**anual 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 § 3

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
```

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)

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`

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.

Each device is assigned a unique **SCSI ID** 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.

Booting SCSI disks

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

5. Network cards

?The Network Interface

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
      eth0: Lite-On 82cl68 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

?The Modem device

We will only consider serial modems. The following table shows the equivalence between DOS COM ports and Linux serial devices.

Table 1: Serial port equivalence DOS-Linux

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

Most Linux distributions have hardware browser tools (GUIs) which can detect modems. But 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*

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

A symbolic link called **/dev/modem** pointing to used serial portcan be 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.

?Dialup Configuration (*The LPI101 objectives only cover hardware detection and not configuration*)

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

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>
```

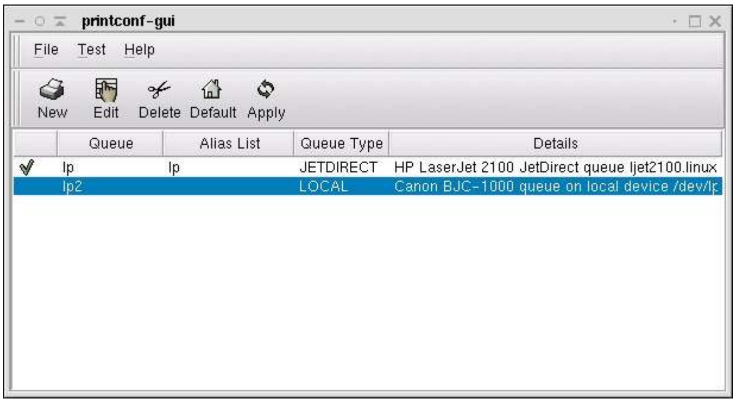
A quick way to get started is to replace *Defaults* with the name of your provider say WorldISP, fill in the Usernam/Password entries and type the following:

```
wvdial WorldISP
```

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.



	Queue	Alias List	Queue Type	Details
✓	lp	lp	JETDIRECT	HP LaserJet 2100 JetDirect queue ljet2100.linux
	lp2		LOCAL	Canon BJC-1000 queue on local device /dev/lp

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. 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 in present?
- Verify that there are as many '*bus* ' entries in **/proc/pci**. Does this file give as much information as the commands above?

4. USB tools:
- Use **lsmod** and **lsusb** to determine which type of 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.

Managing Devices

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.

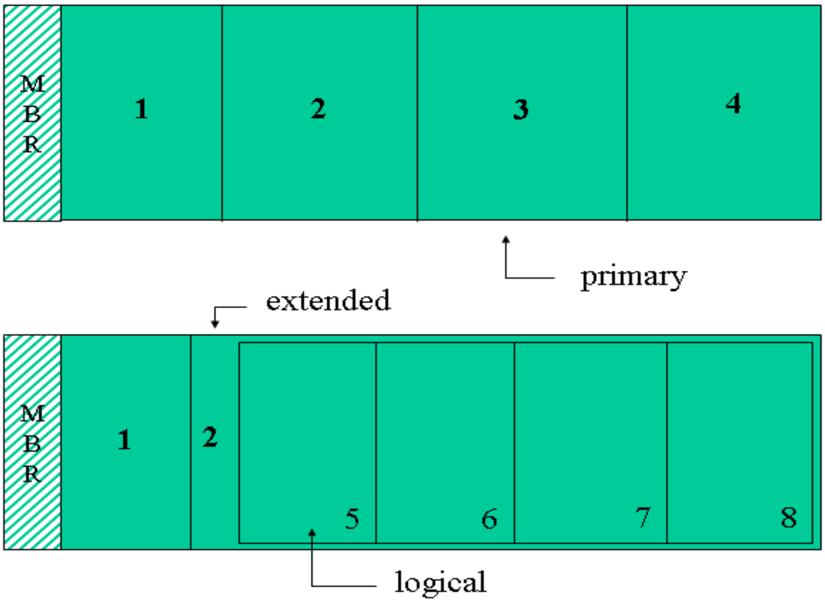
Disk Partitions:

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:

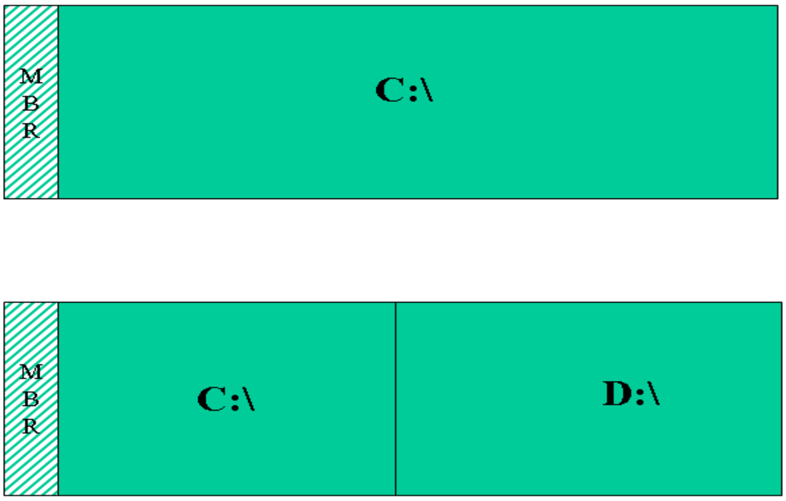
1. Before installation: (not for exam purpose)

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)

During the installation process the Linux partition is partitioned again. Why do Linux systems require further partitioning? To answer this question we first define mount points.

*Defining a **mount point**: (also see figure page5)*

*One has the choice to associate a piece of hardware (or resource) to a directory. For example the root directory "**/**" which is more or less like the C:\ drive for DOS could correspond to the /dev/hda2 partition, and the subdirectory **/boot** could correspond to the partition /dev/hda3.*

*"/dev/hda3 is said to be **mounted** on **/**boot". The directory on which a block device is mounted is then called a **mount point**.*

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

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 userfriendly tool that does both these steps at once:

diskdrake (Mandrake)
DiskDruid (RedHat)

The very early success of RedHat over other projects such as Debian was the introduction of intuitive installation tools such as **DiskDruid**.

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

<u>3. On a Running System</u>:

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 envolved:

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.

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

GRUB keywords (used 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

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
```

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
users	the device can be read and unmounted by all users
user	the device can 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**

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** 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

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

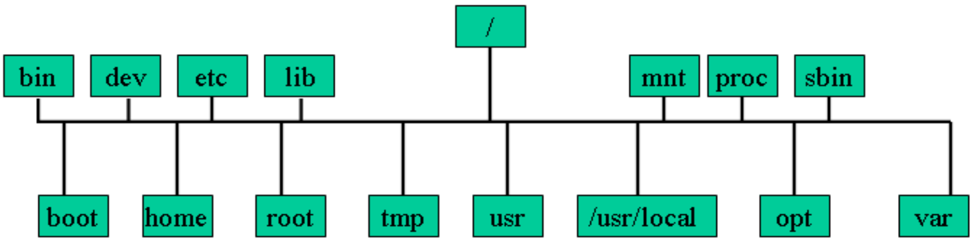
1. The Filesystem Structure

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

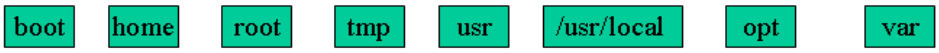
As mentioned earlier, once partitions have been created each partition must be given a *mount point*. This is typically done at installation time. To help us understand where things are kept, let us look at the Linux file system hierarchy.

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.

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>
```

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 Consistancy

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 named accordingly.

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
```

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/mount**.

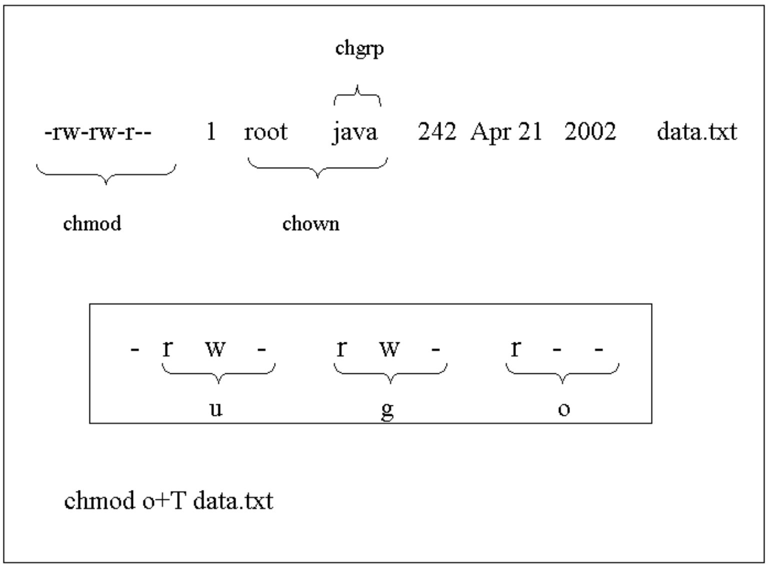
In addition to showing all mounted devices the **df** tool will also show *Used* and *Available* disk space. By default this is given in blocks of 1K.

```
            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.

4. File Permissions



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:

The symbolic values for the ***owner*** *fields:*

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:

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

Changing user and group:

```
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.

Table 2: Octal and symbolic permissions.

Symolic	octal
read	4
write	2
execute	1

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

Table 3: 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

It is possible for root to give users permission to execute programs they would usually be unable to. This permission is the SUID permission with a symbolic value **s** or a numerical value **4000**.
For example root can write a C shell script that executes a program and set the SUID of the script with chmod 4777 script or chmod u+s script. (NB Bourne and Bash scripts do not honour SUID bits set on the script files.)

Examples:

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

SGID permissions

The SGID is a similar permission 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.

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
```

5. Exercises

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** :
Notice that there are no tools to create **ext3** formated partitions. In fact the **ext3** format is the same as the **ext2** format with a journal added. These are the steps:

```
mke2fs /dev/hda10
tune2fs –j /dev/hda10
```

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** …

<u>File permissions</u>

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

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, thenthe 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
```

Export, Set and Env:

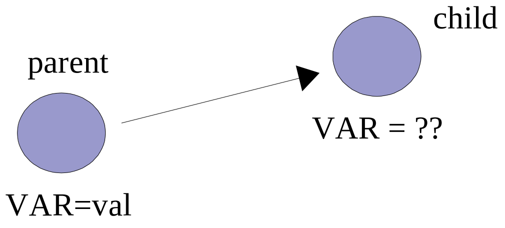
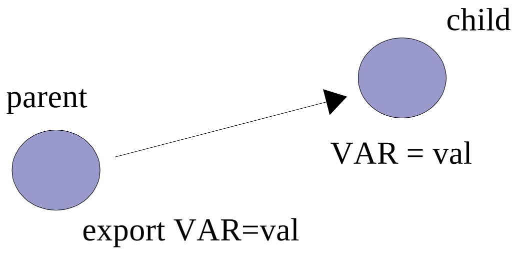
There are two types of variable: local and exported.

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 ?

Table 1.2 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

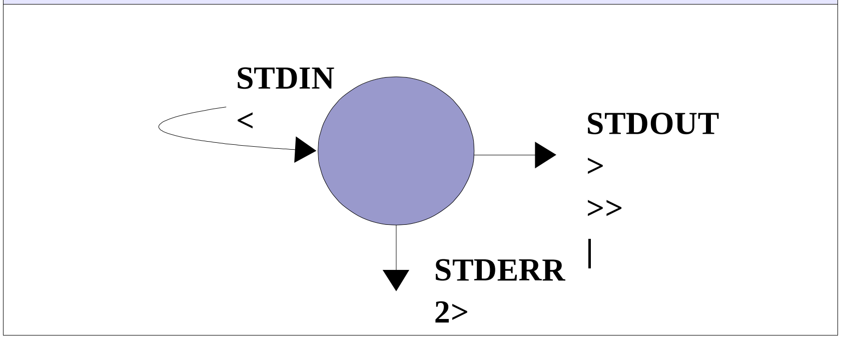
The next few variables are related to process management.

$!   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

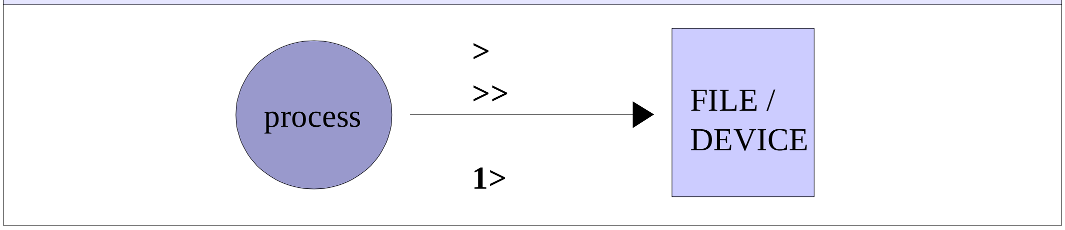
The data flows from left to right.

```
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 output to a file.

STDOUT Redirection



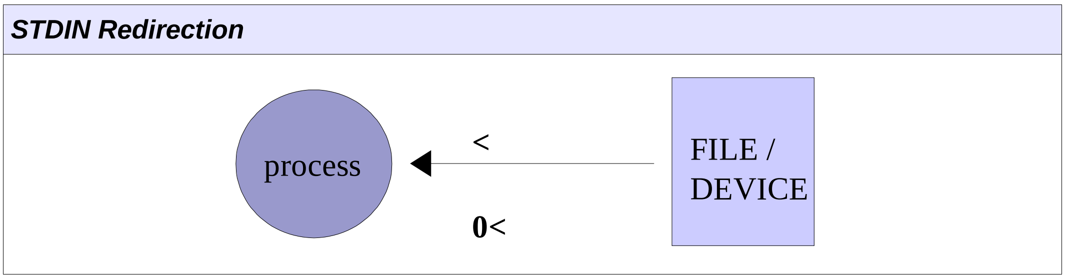
stdin redirection

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
```

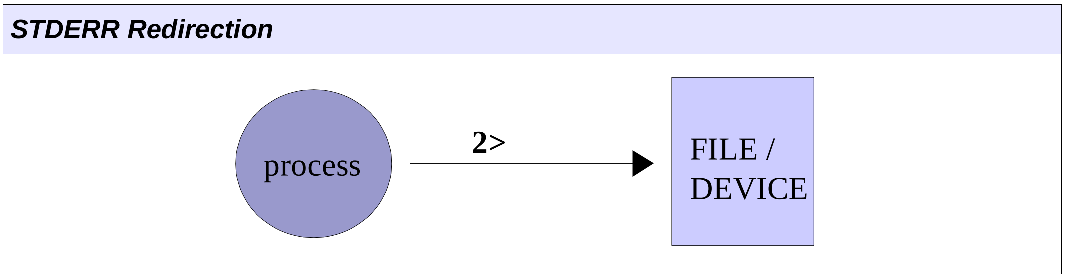


stderr redirection

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
```

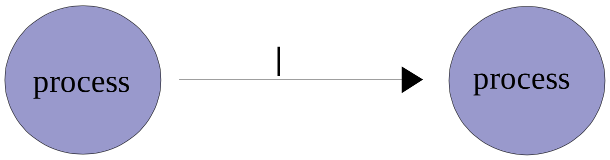


piped commands

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.

Piped Commands



```
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
```

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
```

Ranges

?[ ] 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
```

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 <End>)
Ctrl+e	Go to the end of the line (Same as <Home>)

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}'
```

Command completion

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

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

<< is a redirection for EOF

For example

```
cat << stop
```

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

7. Exercise

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

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** 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.

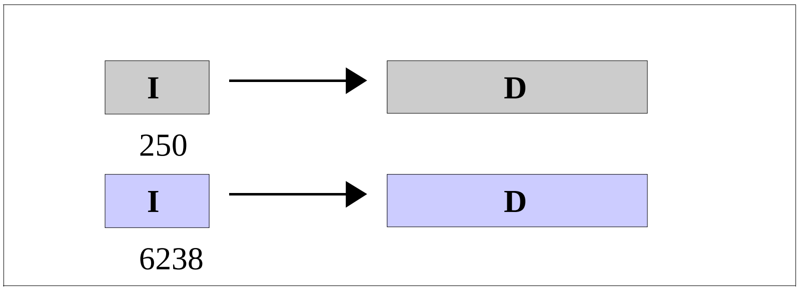
Notice: `rm –rf /dir1/*` removes all files and subdirectories leaving dir1 empty
`rm –rf /dir1/` removes all files and subdirectories including dir1

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

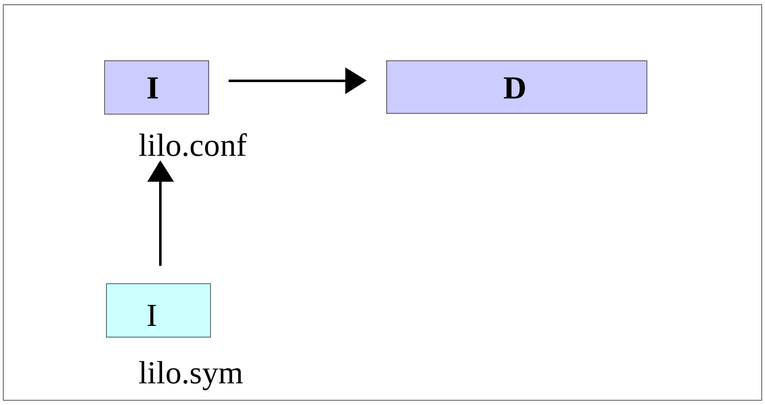
A soft link to a file or a directory creates a new inode that points to the same data area:

```
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
```

8. Exercises

File Navigation

Make a new directory in **/tmp** called **/bin**.

```
mkdir /tmp/bin
```

In **/tmp/bin/** create a file called *newfile* (use touch, cat or vi).
Go to the root directory (cd /). View the content of *newfile* from there.
Which is the shortest command which will take you back to **/tmp/bin** ?
Which is the shortest command which will take you to your home directory ?
Is the PWD variable local or global ?

Creating and deleting directories

Which is the quickest way to make two new directories /dir1/dir2 ?
Remove the directories with **rmdir** then 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 create by deleting the **/usr/share/doc** directory and creating a symbolic link point from **/usr/share/doc** to the new location.

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

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

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

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

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

Make sure the data has all been copied across then edit **/etc/fstab** to make that device available at boot time.
Delete **/usr/share/doc** and create a symbolic link pointing from **/usr/share/doc** to **/spare/doc**

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

Do the same with **/home**. Any extra problems?

Finding Files on the System

Copy the file */etc/lilo.conf* to */etc/lilo.conf.bak*

1. Use **find** to find this new file
2. Use **locate** to find */etc/lilo.conf.bak.* (How do you update the **slocate** database ?)

Backup strategy (first step)

Find all files in your home directory that have been modified today.

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

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

Process Management

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.

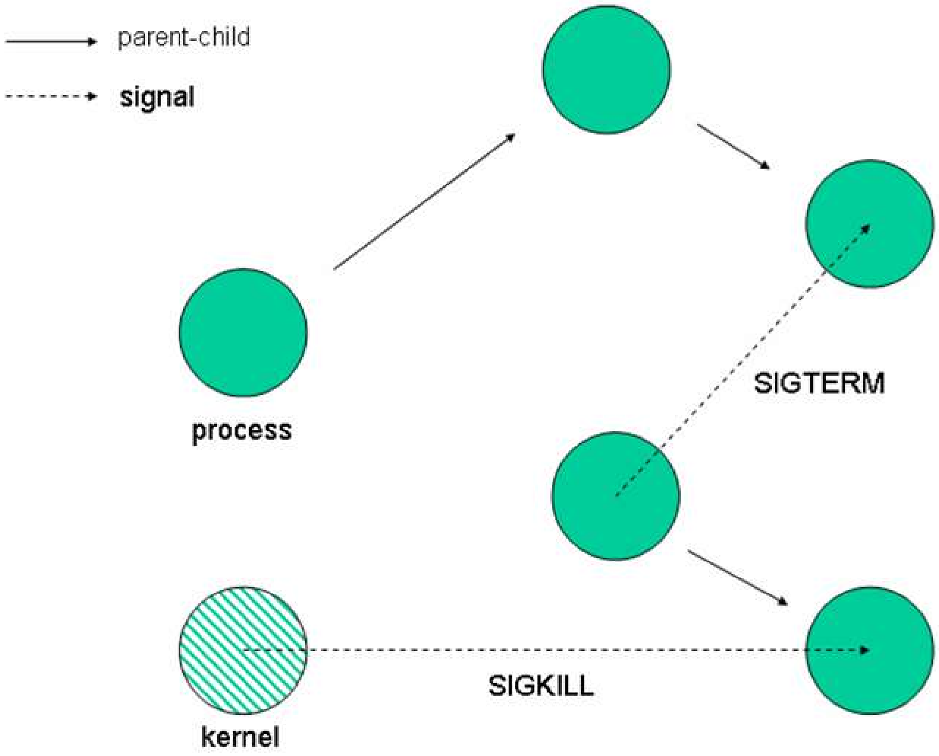
One can also stop processes without knowing the process' PID using **killall**.

killall

Syntax

```
killall SIGNAL process_NAME
```

Fig1: Interprocess signaling

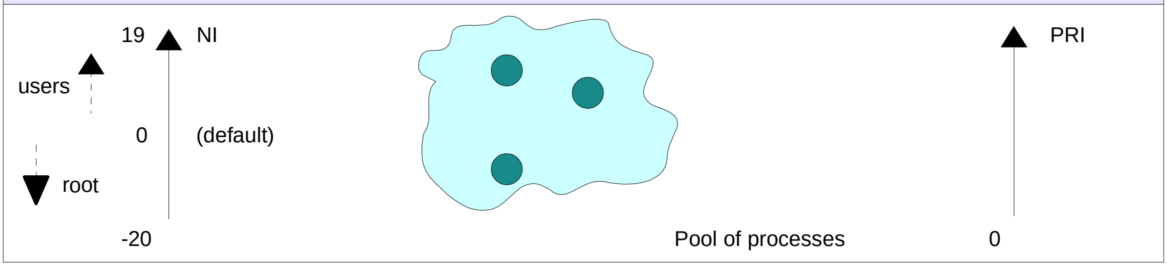


Process priority and nice numbers

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!

nice numbers and CPU priorities



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.

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

You should run X before starting these 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:

```
#!/bin/bash

 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

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/resolve.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 characters or 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**.

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

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

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

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

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

Often used functions 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.

?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 -Wl,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.

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
```

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.

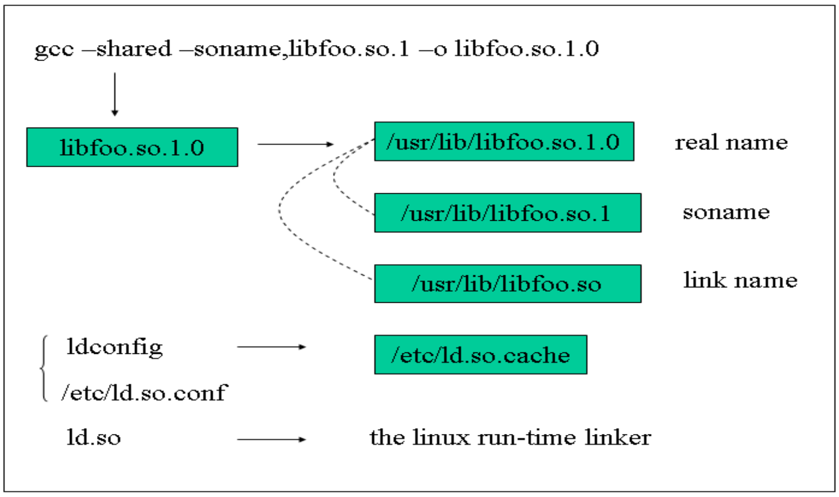


Figure 1: The Shared Library Names

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.

To fix this problem one can do one of the following:

1. If the binary needs to be temporary tested define the LD_LIBRARY_PATH variable as follows:

```
          export LD_LIBRARY_PATH=$(pwd)
```

2. copy the **libfoo.so.1.0** to **/usr/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.

Common Files

configure: This is a script which determines what architecture is being used. It also checks that the required compiler and libraries are present. The safest way to run the script is to use '`./configure`'.

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)

Compilation

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:

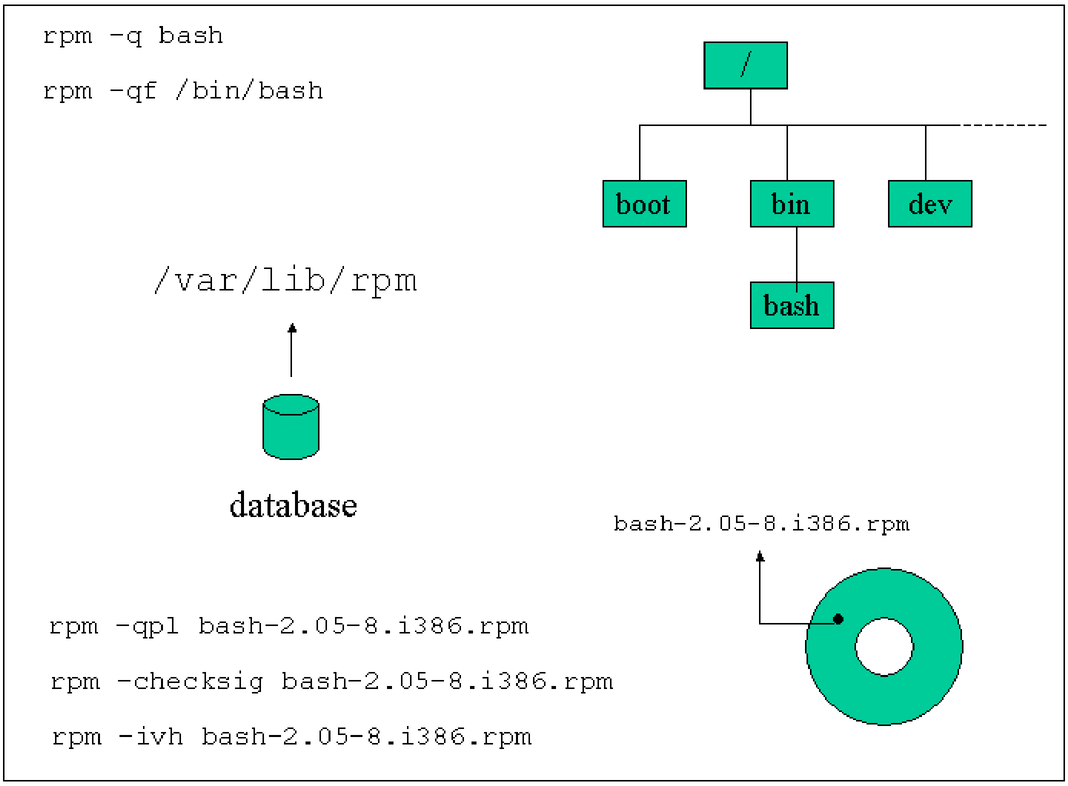
```
./configure
make
make install
```

`make install` must be run as root if the install directory is /usr/ or /usr/local

There are many options to the **./configure** script. To customise your installation you could type **./configure –help**.

An easy option is `--prefix` which allows you to specify the root directory for the installation. This is usually set to **/usr/local/** by default.

4. The RedHat Package Manager RPM



```
rpm -q bash
rpm -qf /bin/bash
rpm -qpl bash-2.05-8.i386.rpm
rpm -checksig bash-2.05-8.i386.rpm
rpm -ivh bash-2.05-8.i386.rpm
```

Fig 1: Package Manager Functions

Package naming

Rpm packages generally follow the following naming convention:

name-version-release.architecture.rpm

Major modes

These are the major modes 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

Minior mode

Short	Description
a	applies to all installed packages
c	together with **q** lists **c**onfiguration files
d	together with **q** lists **d**ocomentation files
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

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 as 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
```

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 ...

Special options

--nodeps	this allows to install without regard to dependencies
--force	force an upgrade
--test	doesn’t actually install or upgrade, just prints to stdout
--requires	show package requirement

Going Further: Building RPM packages (*not for LPI exam purpous*)

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. The Alien Tool

This tool will change Debian packages into RedHat ones and vice versa. You can download it at: http://kitenet.net/programs/

6. 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
```

- Finally follow the usual compilation steps:

```
./configure
  make
```

If you are sure you want to install this package then `make install` but remember that this will not install the software using the package manager.

2. Rebuilding with the RPM package manager.

```
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

Advanced Text Manipulation

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

Table1:List of main regex's

Characters	Search Match
x (or any character)	Strings containing an 'x'
\<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
"cat*"	Strings containing 'ca' or 'cat' plus anything
"."	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'
"cat**+**"	Strings containing at least cat plus anything
"cat**?**"	Strings containing 'ca' or 'cat' plus anything

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 literaly (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

Command options
d Delete an entire line
r Read a file and append to output
s Substitute
w Write output to a file

6. 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:

```
ca
cat
cats
catss
cat+
cat*
cat?
car
carriage
```

- Investigate the outcome of the following using **grep**, **egrep** and **fgrep**:

```
grep ‘cat+’ FILE
grep ‘cat?’ FILE
grep ‘cat.’ FILE
grep ‘cat*’ 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

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.

?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

4. Deleting Text

If you want to delete a single character while in command mode you would use **x** and **dd** would delete the current line.

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.

Table4:Words and Characters

w	single word
l	single character

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 Pasting

The copy action in **vi** is the command **y** (for yank), and the paste action is **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. Searching

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 colon 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:

Search for words beginning with ‘comp’ in all the text

```
/\<comp
```

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. Undoing

At this stage is is worth mentioning that one can always undo changes (while in command mode) with the **u** command, and this as long as one hasn't saved the file yet.

8. Saving

The command for saving is **w**. By default the complete document is saved. 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

9. 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.*

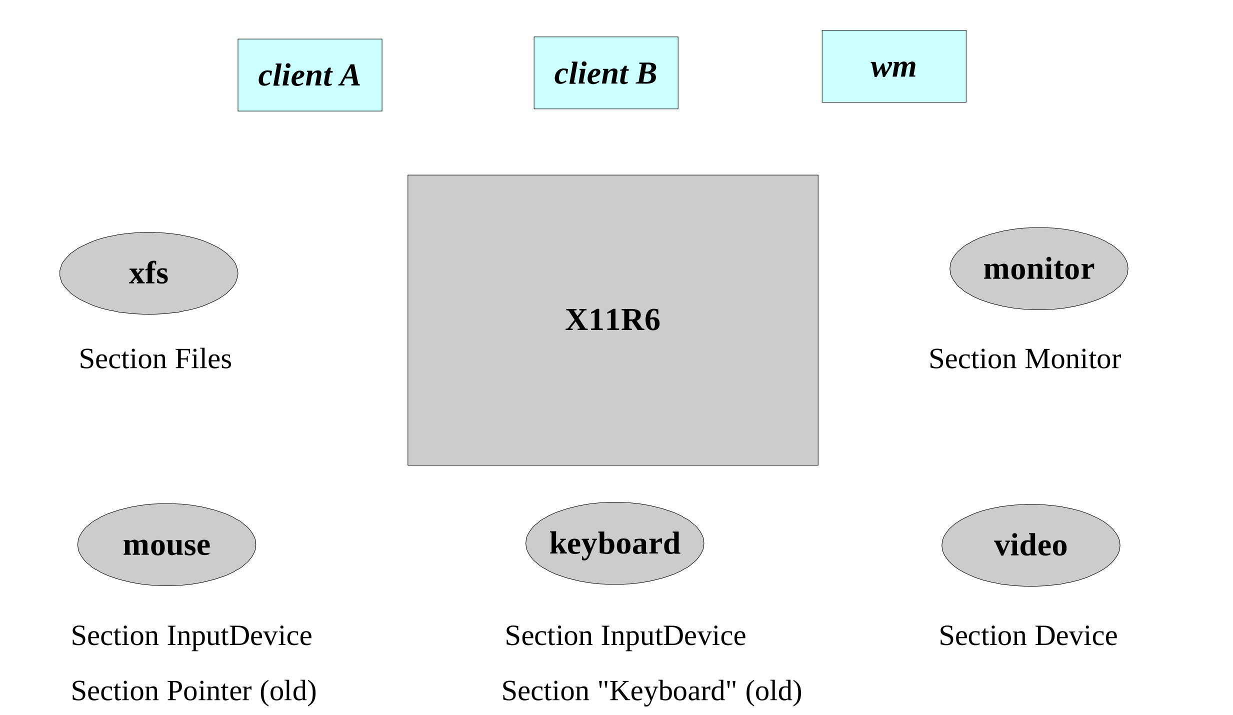
The X Environment

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)

- ***The XF86Config File***

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

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.

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, in **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 symbolise the startup sequence as follows:

startx --> xinit --> X -> xinitrc -> Xclient (wm/desktop)

2. Using a Display Manager

We will first describe the login. The next section covers all the functionalities of the Display Manager.

x**dm --> xlogin --> Xsession --> Xclient**

5. The Display Manager

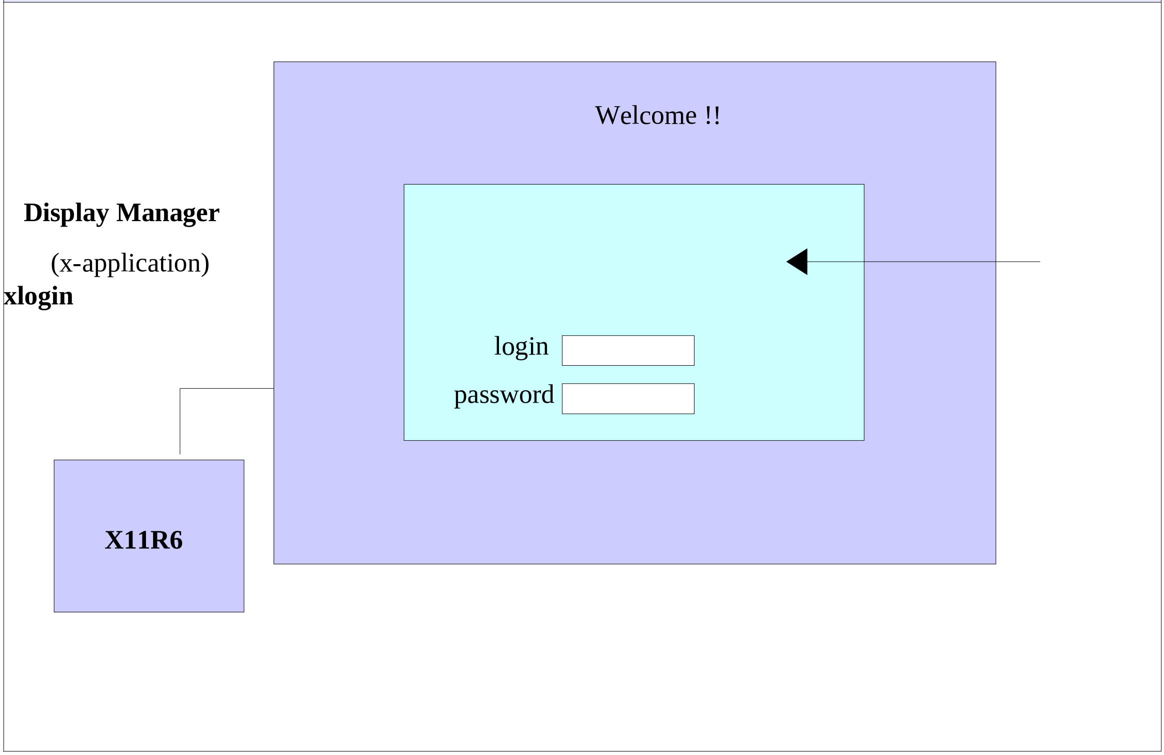
There are three main display managers, xdm (generic), gdm (GNOME) and kdm (KDE). According to the LPI objectives the configuration file are in the following directories:

/etc/X11/xdm/
/etc/X11/gdm/
/etc/X11/kdm/

However **kdm** no longer follows this convention. So we will take a closer look at **xdm** and **gdm**.

Display Managers are used mainly in run level 5 to allow local users to log onto the system using the graphical interface. However display managers can also be used to provide a graphical login interface over the network. To do 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).

X server and Display Manager



- **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 various window managers there are also various desktop environments, among which the most common are KDE and GNOME.

The most common window managers are listed below:

fvwm
icewm
amiWM
mlvwm
dfm
olwm
olvwm
mwm
Window Maker
AfterStep
Enlightenment

9. 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
```