Linux 101 Examination

Modular Training Notes

Leading Edge Business Solutions

This manual was written for Leading Edge Business Solutions
http://www.ledge.co.za/ as part of their Linux training programme.

101-letter.odm, 2 February 2006

LPIC objectives

Table of Contents

this page unintentionally left blank
oops!

1 Foreword

A is for **awk**, which runs like a snail, and
B is for **biff**, which reads all your mail.
C is for **cc**, as hackers recall, while
D is for **dd**, the command that does all.
E is for **emacs**, which rebinds your keys, and
F is for **fsck**, which rebuilds your trees.
G is for **grep**, a clever detective, while
H is for **halt**, which may seem defective.
I is for **indent**, which rarely amuses, and
J is for **join**, which nobody uses.
K is for **kill**, which makes you the boss, while
L is for **lex**, which is missing from DOS.
M is for **more**, from which **less** was begot, and
N is for **nice**, which it really is not.
O is for **od**, which prints out things nice, while
P is for **passwd**, which reads in strings twice.
Q is for **quota**, a Berkeley-type fable, and
R is for **ranlib**, for sorting **ar** table.
S is for **spell**, which attempts to belittle, while
T is for **true**, which does very little.
U is for **uniq**, which is used after **sort**, and
V is for **vi**, which is hard to abort.
W is for **whoami**, which tells you your name, while
X is, well, **X**, of dubious fame.
Y is for **yes**, which makes an impression, and
Z is for **zcat**, which handles compression.

– *THE ABC'S OF UNIX*
/usr/share/games/fortune/songs-poems

This course material is based on the objectives for the Linux Professionals Institute's LPI 101 examination (specifically, release 2 of the objectives). The course is intended to provide you with the basic skills required for operating and administering Linux systems.

At every good training course the student should come away with some paper in his hand, to file in the company filing cabinet. A really excellent course will include some knowledge and practical ability in the student's head as well. We hope to achieve at least the first with these notes. The second is up to the instructor.

Goal of this course

This course aims to equip you with the knowledge to be able to pass the LPI 101 examination

(release 2). We hope that in the course of doing this course you will acquire the skills that go with an understanding of how Linux works.

Target audience

This course is aimed at ...

- People who wish to write the LPIC 101 exam, as part of the LPIC Level 1 certification.
- Technically inclined people who wish to become familiar with Linux, particularly with a view to administer the system.

Prerequisites for taking this course

People wishing to take this course will probably fit the following profile

- You should have basic skills for using Linux, including command line usage. A minimum of 3 months of experience is recommended. Completion of an introductory Linux course is an alternative to this experience.
- You are a system administrator or hold a similar technical position (or you would like a job like that).
- You are interested in technical things and the fascinating little details that make your computer behave strangely.
- You want to know how things work – specifically how Linux works, and are willing to spend some time finding out.
- You have practical administrative experience with other computer systems.
- You already have some practical familiarity with using Linux. You have probably installed Linux and have used it without gaining a complete understanding of many functions.

We recommend that this course be followed by professional people who have completed their secondary education, and possibly an additional qualification. It is preferable that you already hold a position in which you can use Linux on a day to day basis.

Flow of instruction

Each section in the notes is structured as an independent entity. Each section covers a single LPIC topic. Each section is structured as follows:

- LPIC objectives
- Introductory material
- Detailed material
- Review material (quiz questions and assignments).

Some of the sections are more demanding than others, and the certification does not weight all of the sections equally.

There are a number of ways to approach this material:

- Over a number of weeks, as a self-study course with some classroom time. This approach is recommended for novices.
- Over two weeks, covering all the sections in some detail.
- Over a single week, as an instructor-led course. This approach crams a lot of material into

a small time, and is recommended for experienced Linux users as a pre-exam cram.

The following order of study is recommended, especially if you have not previously studied the material (i.e. for first-time users). This is not the order in which the material appears in the manuals.

1. Topic 103 GNU & Unix Commands
2. Topic 101 Hardware & Architecture
3. Topic 102 Linux Installation & Package Management
4. Topic 104 Devices, Linux Filesystems, Filesystem Hierarchy Standard
5. Topic 110 X

If you do the material over two weeks, the material can be used in the following arrangement:

101 Part A – Linux shell	**101 Part B – Installation and administration**
1.103.1 - Work on the command line 1.103.2 - Process text streams using filters 1.103.3 - Perform basic file management 1.103.4 - Use streams, pipes, and redirects 1.103.5 - Create, monitor, and kill processes 1.103.6 - Modify process execution priorities 1.103.7 - Regular expressions 1.103.8 - Perform basic file editing operations using vi 1.104.5 - Use file permissions to control access to files 1.104.6 - Manage file ownership 1.104.7 - Create and change hard and symbolic links 1.104.8 - Find system files and place files in the correct location	1.101.1 - Configure Fundamental BIOS Settings 1.101.3 - Configure modem and sound cards 1.101.4 - Setup SCSI Devices 1.101.5 - Setup different PC expansion cards 1.101.6 - Configure Communication Devices 1.101.7 - USB hardware 1.102.1 - Design hard disk layout 1.102.2 - Install a boot manager 1.102.3 - Make and install programs from source 1.102.4 - Manage shared libraries 1.102.5 - Use Debian package management 1.102.6 - Use Red Hat Package Manager (RPM) 1.104.1 - Create partitions and filesystems 1.104.2 - Maintain the integrity of filesystems 1.104.3 - Control mounting and unmounting filesystems 1.104.4 - Managing disk quota 1.110.1 - Install & Configure XFree86 1.110.2 - Setup a display manager 1.110.4 - Install & Customize a Window Manager Environment

What you need for this course – part time over 8 weeks

You will need the following in order to complete this course.

- A dedicated computer to work on outside of course contact time. As part of the course, the existing data on this computer will most likely be destroyed. If you do not have an appropriate computer, you should consider buying a laptop, or at least a new hard disk for

an existing computer.

- Committed time for eight working weeks:
  - Lecture, tutorial and review time: 2 hours per week (excluding travel time).
  - Self-study and practice time: minimum of 2 hours per day, Monday to Friday.

What you need for this course – instructor led over 1 week

You will need the following in order to complete this course.

- Committed time for 1 working week – lecture, tutorial and review time: 6 hours per day

The training venue should have a suitable computer.

After the course, it is recommended that you review the material before writing the certification examination.

Typographic conventions

Command names and example of command are printed in **boldface**. So for example, **ls -la** is used for printing a list of files in the current directory, and **pwd** prints the current working directory.

Syntax explanations are shown like this.

```
ls [directory-name]
```

In this particular case, it means that you can tell **ls** to list a particular directory.

Interactive command sessions are shown in a block

```
# This is an interactive session
# What was typed is shown in boldface.
foo:~ $ su - jack
Password:
[jack@foo jack]$ ls
[jack@foo jack]$ ls -a
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .gtkrc  .kde
      .xauthwEl4ka
[jack@foo jack]$ pwd
/home/jack
```

The student is encouraged to try these example commands on his[1] computer, as the results may differ from one system to the next. Often the output shown is incomplete, and a valuable learning experience awaits the person bold enough to type the bold text.

1.1 About these notes

These notes have been written with the LPI's objectives and criteria for approved training materials in mind. We have designed them to be modular, so that a course following LPI objectives can easily be built up from a selection of topics.

Printed copies of this and other manuals can be purchased from Leading Edge Business Solutions (Pty) Ltd – see www.ledge.co.za. We offer training courses based on this material. The contact address for queries related to these notes is lpinotes@ledge.co.za.

1 And when we say “his”, we mean “her” if the student happens to be female.

1.2 Revisions and bugs

Gentle reader, we hope that these notes provide a wonderful learning experience for you. In this process we trust that you will be kind enough to point out to us the typos, stylistic faults and gross errors in the text. If you make changes to these notes, or produce them in an alternative format, we would appreciate it if you would send us a copy of your revisions.

Known bugs

OpenOffice.org suffers from a confusion of its bullets and numbering system which affects this document. The sub-document is correctly numbered and bulletted, but this does not reflect in the master document. If you know how to fix this, please do let us know.

1.3 Copyright notice

1.4 GNU Free Documentation License

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.

2 BIOS Settings

> *valerie kernel: mtrr: your CPUs had inconsistent variable MTRR settings*
> *valerie kernel: mtrr: probably your BIOS does not setup all CPUs*
> It indicates your bios authors can't read standards. Thats a quite normal state of affairs, so common that the kernel cleans up after them
>
> *Alan Cox on linux-kernel*

This chapter is on the configuration of the BIOS. Although this chapter is mainly information, you will need to know how to start up your Linux system and execute commands and read files using **less** in order to make progress.

LPIC topic 1.101.1 — Configure Fundamental BIOS Settings [1]

Weight: 1

Objective:

Candidates should be able to configure fundamental system hardware by making the correct settings in the system BIOS. This objective includes a proper understanding of BIOS configuration issues such as the use of LBA on IDE hard disks larger than 1024 cylinders, enabling or disabling integrated peripherals, as well as configuring systems with (or without) external peripherals such as keyboards. It also includes the correct setting for IRQ's, DMA and I/O addresses for all BIOS administrated ports and settings for error handling.

Key files, terms, and utilities include:

/proc/ioports	Linux's view of used I/O ports
/proc/interrupts	Linux's view of used interrupts
/proc/dma	Linux's view of assigned DMA channels
/proc/pci	PCI bus information

2.1 BIOS architecture

If you open up your computer's case, you will see a number of components. You may be able to identify some of these:

- A CPU. For IBM PC clones, this has been compatible with the 8088 processor.
- A ROM chip containing the BIOS code
- A CMOS memory chip for BIOS settings (Complementary Metal-Oxide Semiconductor).
- A battery which makes sure that the CMOS settings are not lost when the power goes off.

The BIOS is the **B**asic **I**nput/**O**utput **S**ystem of your computer. In the good old days, this was intended to be used as the lowest level of the operating system, and was used in that way by MS-DOS. Modern operating systems implement the basic I/O functionality themselves.

In recent times, the BIOS has been relegated to a more limited set of functions:

- Power on self test. (P.O.S.T.) This includes testing the memory (although the test is not exhaustive).

- Putting the hardware in a sane an predictable state (e.g. setting the display card's video mode and timing parameters to use text mode; setting up power management; initialising hard disks)
- Chipset-specific configuration for your machine (e.g. configuring the PCI bus and built-in peripherals)
- Loading the operating system from the disk (from floppy disk, IDE disk, SCSI, network ...)
- Providing basic I/O (keyboard, floppy, hard disk, CDROM).

Modern BIOS programs include an interactive program that can be used to change the configuration settings, i.e. to change the contents. This is what most people refer to as “the BIOS” (e.g. “Enter the BIOS and tell it to autodetect the hard disks”).

2.2 Changing BIOS configuration

The method of entering the BIOS setup program varies from one BIOS manufacturer to the next. Various computer vendors use different methods. You should be able to enter the BIOS by pressing one of the following key combinations either before, during or just after the P.O.S.T. (and usually after):

Del, F2, Ctrl+Alt+Esc, Ctrl+Alt+S, Ctrl+Alt+Ins, Ctrl+Alt+Del (although this will reboot on most PC's), F1, F3, F10, Fn+F1 (laptop), Esc.

This necessarily requires that a keyboard be connected to the computer. If you are setting up a computer that does not have a keyboard, then you will need to plug one in during configuration, but you will have to make sure that the system works without a keyboard later.

2.3 IDE disks

IDE (Integrated/Intelligent Drive Electronics) disks are the default fixed data storage media for most PC's. IDE disks are usually described in terms of their size, ranging from small (20Mb) to large (80Gb[2]).

The BIOS is involved in the usage of IDE disks for three reasons:

- The BIOS must load the essential parts of the operating system from the disk during booting.
- The BIOS must provide correct information about the installed media to the operating system. This information is the number of cylinders, heads and sectors on the disk. In the past these used to correspond to the number of spinning platters, the number of magnetic read-write heads and the density of the magnetic media. Although most hard disks have one read-write head, the number reported depends on the size of the disk. You don't need to worry about it actually.

2.3.1 Disk geometry and addressing

In order to do the important task of loading the operating system, the operating system loader has to request data from exact locations on the disk. The method of defining exactly where the data is has changed over time.

2 Well, when we wrote this, a 80Gb disk was large. It might not be large anymore.

Disk geometry refers to the logical dimensions of the disk – how many cylinders, heads and sectors the disk controller can access. These values once corresponded with physical reality but this is no longer the case. Instead they refer to the 3 sets of coordinates accepted by the IDE interface for specifying a data sector.

Interrupt 13h

The BIOS function to read a sector of data from disk is accessed via a software interrupt, interrupt 13h (hexadecimal). To read from the disk using interrupt 13h, you specify the exact head, cylinder and sector numbers. The limits on these are shown in the table.

Parameter	***Bits***	***Maximum***
Cylinder	10	1024
Head	8	256
Sector	6	64
Total	24	16.7 million

The location of the data is specified with 24 bits, which means that you can access a maximum of 16.7 million sectors. With a sector size of 512 bytes, you can access a hard disk of up to 8.4 GB.

There is a problem, however. It is unusual that a disk will have the number of heads and cylinders that happens to match the limits imposed by the design of interrupt 13h. The first fix for this problem was to employ various translation modes which shuffle bits between the head and sector addresses. Doing this meant that it was only necessary to change the BIOS – not to change the operating systems. However, disk sizes soon passed 8.4 GB.

To handle disks larger than 8.4 GB an extended version of the interrupt 13h functions allows the location of data to to be specified using 64 bits, meaning that it should work until disk sizes hit 9.4 x 10^{21} bytes, which probably won't happen this year. This is called Logical Block Addressing (LBA mode). When using LBA mode, cylinder addresses are a single number, and there is no such thing as cylinders, sectors and heads, except for compatibility with older operating systems.

"Normal" mode (CHS)

In this mode, the location of data on the disk is described in terms of the Cylinder, Head and Sector[3] at which the data resides.

The BIOS interface for reading from the disk is via a software interrupt, interrupt 13h (13 hexadecimal). The limits for the cylinder, head and sector numbers do not correspond with the limits for the IDE drive interface, although they were adequate for older disk drives.

Interface	***Cylinders***	***Heads***	***Sectors***
IDE/ATA	65536	16	256

3 Hard disks may have a number of electromagnetic read-write heads. Each *head* can move across the rotating disk platter to a *cylinder* and wait for a given *sector* to fly past.

Interface	*Cylinders*	*Heads*	*Sectors*
BIOS Int 13h	1024	256	64
Normal mode compromise	1024	16	64

In Normal mode, the cylinder, head and sector numbers requested when interrupt 13h is called are used as-is.

One of the results of this arrangement is that Normal mode cannot handle disks larger than 504Mb (that is 1024 cylinders, 16 heads and 64 sectors).

"Large" mode

If the disk size is between 504 Mb and 8Gb, a horrible hack exists to make the disk addressable, without requiring changes to the software. You may notice that the "Heads" parameter of the interrupt 13h service has 4 bits of extra capacity that will never be used with an IDE disk. Large mode shifts one, two or four bits from the cylinder specification to the head specification. Shifting 4 bits means that you can point to 16 times as much data. It's a horrible hack, but it means you can have disks as large as 8Gb.

Because the translation mechanism is not easily predictable, the BIOS may not access the sectors you intend when the cylinder you ask for is above 1024.

The disk also looks rather different to what it would in "Normal" mode.

2.3.2 Logical Block Addressing (LBA)

When a disk is in LBA mode, each location on the disk is specified with a single number, up to 24 bits long. Not every BIOS supports LBA addressing. If the BIOS does not support LBA addressing, it is generally risky for it to access data which is after cylinder 1024.

There is a complication on this too. The BIOS is responsible for setting up the correct translation between LBA addresses and CHS addresses. Not every BIOS does this predictably – especially after cylinder 1024 when the translation hacks start coming into play.

To play it safe, many Linux distributions ensure that they create a boot partition which the BIOS will always be able to read, regardless of whether it supports LBA mode or not. This partition contains the system files for booting up.

Summary of limits

The various modes of accessing a disk are subject to the following limitations.

Method	*Disk size limit*	*Cylinders*	*Bits*
Normal mode (CHS addressing)	504Mb	1024	24
Interrupt 13h	8.4Gb	1024	24
Large mode (ECHS addressing)	8.4Gb	16384	24
ATA interface	128Gb	-	28
Interrupt 13h extended	9.4×10^{21} bytes	-	32

Method	*Disk size limit*	*Cylinders*	*Bits*
LBA mode	9.4 x 10^{21} bytes	-	32

A kernel patch for the 2.4.18 kernel allows it to address disks with up to 48 bits.

2.3.3 Data transfer

PIO modes

The original for reading and writing an IDE disk is called programmed IO. The choice of PIO mode determines the speed at which the CPU reads data from the IDE interface. Mode 0 is the slowest, and mode 4 is the fastest.

DMA modes

DMA is the technique where the IDE interface writes data to memory by *direct memory access*. This requires little intervention by the CPU, which can spend its time doing more productive work. Newer hard disks and motherboards do support this. However, if you connect a new hard disk into an older motherboard, you may have to manually specify that you do not wish to use DMA to avoid triggering hardware bugs.

If a BIOS supports setting the DMA mode, Mode 0 is the slowest and Mode 5 is the fastest (and possibly most error-prone).

32 bit transfer

IDE interfaces always use 16 bit transfer. The 32 bit transfer mode setting that you will find in some BIOS configuration programs refers to transfer on the PCI bus, rather than to the actual interface. Enabling this causes a small performance improvement.

Block mode

Block mode involves transferring up to 32 blocks of data between the IDE interface and memory without monitoring by the CPU. On some older motherboards does not work correctly, but the BIOS still provides an option to enable it.

2.4 Integrated peripherals

Motherboard manufacturers include a selection of useful peripherals in their hardware.

Display adapter

The BIOS will generally offer two settings related to an on-board display adapter.

- Enable / disable – if you have a display card which you prefer to the on-board one, you will want to choose to use it.
- Memory to allocate – on-board display adapters will use the top 4MB to 16MB (or more) of your system memory. You can set this amount in the BIOS settings.

Network card

If there is an on-board network card, you may want to enable or disable it.

Printer and communications port

Usually a built-in parallel port will allow you to choose the IO address and interrupt to be used for the port.

Port	*LPT1*	*LPT2*	*LPT3*	*COM1*	*COM2*	*COM3*	*COM4*
IO address	0x378	0x278	0x3BC	0x3F8	0x2F8	0x3E8	0x2E8
Interrupt	7	5	7	4	3	4	3
Linux device	/dev/lp0	/dev/lp1	/dev/lp2	/dev/ttyS0	/dev/ttyS1	/dev/ttyS2	/dev/ttyS3

2.5 IRQ, DMA, I/O addresses

- IRQ's – interrupt requests (this is used for signaling the CPU that data is available).
- DMA channels – some devices write directly to memory. Each device that does this must use its own DMA channel.
- I/O addresses – each device needs to be addressable by the CPU. The I/O address is used for input and output to the device.

The architecture of the PC demands that only a single device use any of the following at a time. Since each of these parameters has to be unique, conflicts arise when they are duplicated between devices.

In terms of configuration to resolve conflicts, you can do the following:

- Change the configuration of built-in devices
- Change jumpers on plug-in expansion cards.
- Configure a device by running software
- Not use the device, or not use the feature of the device.

The BIOS automatically handles resource allocation for the following:

- PCI devices. Devices on a PCI bus *can* share interrupts with each other.
- ISA Plug and play devices – the ISA plug and play protocol is designed to allow the BIOS to allocate resources to a device and configure it to use them. It is usually better to let Linux configure ISA Plug and Play devices.

2.6 Error handling

The selection of error handling by the BIOS can determine whether your computer will boot up or not under certain conditions. One of the peripherals the BIOS checks and initialises is the keyboard. If the keyboard is not present, the BIOS helpfully says “Keyboard error, Press F1 to continue”.

It is better to tell the BIOS to ignore all errors – especially in a system that will be run without a keyboard and a screen.

2.7 Power management*

The BIOS may include a number of power management options. It is customary for these to be poorly implemented, and to cause problems with Linux systems – for example, the BIOS may decide to power down the hard disk for some reason. If you have problems with random crashes, these are often related to power management bugs.

2.8 Linux view of the BIOS

When Linux is running, there are a number of commands to determine what the BIOS has been doing to your hardware.

2.8.1 /proc/{ioports,interrupts,dma,pci}

The **/proc** file system contains a number of virtual files which show the state of your hardware.

/proc/ioports

This shows the port numbers which are in use by devices for which a kernel module is loaded. If two devices use the same IO ports, then it is unlikely that they will work together. One of them needs to be reconfigured – either by changing jumper settings, plug and play settings, or reconfiguring internal peripherals.

```
foo:~ $ cat /proc/ioports
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
0376-0376 : ide1
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
0cf8-0cff : PCI conf1
1000-10ff : VIA Technologies, Inc. VT82C686 AC97 Audio Controller
  1000-10ff : via82cxxx_audio
1400-141f : VIA Technologies, Inc. UHCI USB
  1400-141f : usb-uhci
1420-142f : VIA Technologies, Inc. Bus Master IDE
  1420-1427 : ide0
  1428-142f : ide1
1430-1433 : VIA Technologies, Inc. VT82C686 AC97 Audio Controller
  1430-1433 : via82cxxx_audio
1434-1437 : VIA Technologies, Inc. VT82C686 AC97 Audio Controller
  1434-1437 : via82cxxx_audio
4000-40ff : PCI CardBus #02
```

```
4400-44ff : PCI CardBus #02
4800-48ff : PCI CardBus #06
4c00-4cff : PCI CardBus #06
6800-687f : VIA Technologies, Inc. VT82C686 [Apollo Super ACPI]
8080-808f : VIA Technologies, Inc. VT82C686 [Apollo Super ACPI]
```

The I/O port ranges are listed in hexadecimal. Some of the devices are integrated into the motherboard (such as the DMA controller). The kernel tracks their port numbers so that it can prevent unexpected interference in their operation (e.g. by user space programs).

/proc/interrupts

This file shows the interrupt assignments which are in use by devices supported by the kernel and the loaded modules. If two devices use the same interrupt this is not necessarily a problem. Devices on a PCI bus will often share interrupts, and this causes no problems. ISA devices cannot easily share interrupts, since the ISA bus uses edge triggering, while sharing interrupts requires level triggering.

```
foo:~ $ cat /proc/interrupts
           CPU0
  0:     560557          XT-PIC  timer
  1:      11392          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  5:          0          XT-PIC  usb-uhci, via82cxxx
  8:          1          XT-PIC  rtc
 10:        139          XT-PIC  O2 Micro, Inc. 6832, O2 Micro, Inc.
      6832 (#2)
 12:     114626          XT-PIC  PS/2 Mouse
 14:      37059          XT-PIC  ide0
 15:          1          XT-PIC  ide1
NMI:          0
ERR:          0
```

The example above is for a single CPU machine. On this particular machine, the USB controller and the VIA sound controll.er share interrupt 5.

/proc/dma

DMA channels enable fast data transfer which does not involve the CPU.

```
foo:~ $ cat /proc/dma
 4: cascade
```

That machine does not have any notable DMA assignments. Its problems are clearly caused by other factors.

2.8.2 lspci

Older versions of the Linux kernel used **/proc/pci** to list the devices plugged into the PCI bus. Curent versions use **lspci**.

```
foo:~ $ /sbin/lspci
00:00.0 Host bridge: VIA Technologies, Inc. VT8501 [Apollo MVP4] (rev 03)
00:01.0 PCI bridge: VIA Technologies, Inc. VT8501 [Apollo MVP4 AGP]
00:07.0 ISA bridge: VIA Technologies, Inc. VT82C686 [Apollo Super South] (rev 19)
00:07.1 IDE interface: VIA Technologies, Inc. VT82C586B PIPC Bus Master IDE (rev 06)
00:07.2 USB Controller: VIA Technologies, Inc. USB (rev 0a)
```

```
00:07.4 Non-VGA unclassified device: VIA Technologies, Inc. VT82C686 [Apollo Super
       ACPI] (rev 20)
00:07.5 Multimedia audio controller: VIA Technologies, Inc. VT82C686 AC97 Audio
       Controller (rev 21)
00:0a.0 CardBus bridge: O2 Micro, Inc. OZ6832/6833 Cardbus Controller (rev 34)
00:0a.1 CardBus bridge: O2 Micro, Inc. OZ6832/6833 Cardbus Controller (rev 34)
01:00.0 VGA compatible controller: Trident Microsystems CyberBlade/i7d (rev 5c)
```

2.8.3 boot messages and dmesg*

The messages displayed during boot up correspond to the kernel initialising the devices installed in the system. The messages are usually appended to **/var/log/messages**, and can be displayed with **dmesg** after the system has booted up.

2.9 Review

Quiz questions

1. You have a system with an ISA bus and an internal modem requiring IRQ 5. The BIOS assigns IRQ5 to the on-board parallel port. What problem will result (if any). How do you make the modem and parallel port work simultaneously?
2. What is the significance of hard disk cylinder 1024 to the BIOS?
3. When is it better to use LBA mode?
4. What are the error handling settings supported by some BIOSes
5. Which “files” in **/proc** show the Linux view of the BIOS?
6. Which commands list PCI and ISA peripherals?
7. Why is **dmesg** significant for diagnosing BIOS problems?

Assignment

1. Set up your BIOS so that your system will boot up even if the keyboard and screen are absent. Test whether the system does actually boot up.
2. Configure your BIOS so that your system will not boot up at all, and then fix the problem. What did you change? Are there other settings that can cause problems?
3. Make a list of the IRQ's which are available on your PC.

Answers to quiz questions

1. You can either disable the parallel port in the “integrated peripherals” section, or you can change its settings so that it does not use IRQ 5. You also need to reserve IRQ 5 for the ISA bus.
2. Explained in the text
3. Always
4. Halt on keyboard, etc.
5. pci, dma, interrupts
6. lspci and lspnp (for newer kernels)
7. **dmesg** shows kernel messages, and the kernel talks to the BIOS as it boots up.

3 Modems and sound cards

> If you put your supper dish to your ear you can hear the sounds of a restaurant.
>
> *Snoopy*

This chapter deals with the configuration of modem and sound card hardware. A separate chapter examines the topic of PPP connections (although it is included in the objectives for this topic). The ISA PnP standard is discussed more fully in the chapter on PC Cards.

To follow the examples in this chapter, you should install the following software:

- setserial
- uucp (for the **cu** program)
- ISA PnP tools

LPIC topic 1.101.3 — Configure modem and sound cards [1]

Weight: 1

Objective:

Ensure devices meet compatibility requirements (particularly that the modem is NOT a win-modem), verify that both the modem and sound card are using unique and correct IRQs, I/O, and DMA addresses, (if the sound card is PnP) install and run sndconfig and isapnp, configure modem for outbound dial-up, configure modem for outbound PPP | SLIP | CSLIP connection, and set serial port for 115.2 Kbps.

Key files, terms, and utilities include:

Linux compatibility (including winmodems)
Hardware resource assignment (IRQs, I/O, DMA)
PnP sound cards
Serial port configuration
PPP outbound connections
SLIP outbound connections
CSLIP outbound connections

3.1 Modem compatibility and winmodems

Modem stands for **mod**ulator / **dem**odulator, which means that a modem converts binary information to analogue signals (modulation) and also decodes analogue signals into binary (demodulation). The binary signals are given by the computer, and the analogue signals fly over the telephone lines.

Not all modems are created equal. Most modems work with Linux. Just a few don't.

3.1.1 External modems

In the good old days when modems ran at 9600 baud (bits per second), you would plug in your modem into your computer's RS232 serial port and talk to it with the Hayes AT commands.

If you have an external modem connected to **/dev/ttyS0** and you have **uucp** (or **minicom**) installed you can talk to it like this:

```
foo:~ $ cu -l /dev/ttyS0
atz   AT command to reset the modem
OK    The modem responds
ati4  AT information request
56000 BPS External Modem D56RSN-W1              It's that kind of
      modem
OK
atdt0800011111                                  Phone someone
CONNECT 57600                                   They're there!
Red Hat Linux release 8.0 (Psyche)              And we can log in :)
Kernel 2.4.18-19.8.0 on an i586
login:
```

This same interface is supported by almost all external modems, including ISDN terminal adapters which pretend to be analogue modems. (An ISDN terminal adapter is not a "modem", since it does digital to digital translation rather than digital to analogue modulation / demodulation).

3.1.2 Internal modems

There is generally no configuration necessary for the first and second serial ports on a PC (**/dev/ttyS0** and **/dev/ttyS1** in Linux, or COM1 and COM2 in the DOS world). There is unfortunately no standard for additional serial ports, so these usually have to be configured manually.

Internal modems are just like external modems, except that the RS232 port is not outside the computer, but inside. In order to use an internal modem (and to use additional serial ports), you need to tell Linux where the RS232 port is, using the **setserial** command.

Here are the parameters that **setserial** deals with:

```
[root@sarge /root]# setserial /dev/ttyS0 -a
/dev/ttyS0, Line 0, UART: 16550A, Port: 0x03f8, IRQ: 4
        Baud_base: 115200, close_delay: 50, divisor: 0
        closing_wait: 3000
        Flags: spd_normal skip_test
```

If you have an internal modem with a known port and IRQ setting, you would set the port and IRQ like this.

```
[root@sarge /root]# /bin/setserial -v /dev/ttyS2 irq 7
/dev/ttyS2, UART: 16450, Port: 0x03e8, IRQ: 7
```

The "standard MS-DOS" port associations are these:

```
/dev/ttyS0 (COM1), port 0x3f8, irq 4
/dev/ttyS1 (COM2), port 0x2f8, irq 3
/dev/ttyS2 (COM3), port 0x3e8, irq 4
/dev/ttyS3 (COM4), port 0x2e8, irq 3
```

You will notice that there are two interrupts shared between four ports. This means that the serial ports will only work correctly if level-triggered interrupts are used, or if additional interrupt lines are configured. IRQ 5 is a good choice, since the Linux parallel port driver uses polling by default rather than using an interrupt.

3.1.3 Winmodems[4]

It can be useful for an internal modem to run at the fastest possible speed, 115200 bps. Some older applications do not request a connection at speeds higher than 38400 bps. When the application requests 38400 bps, the actual baud rate will be set to 115200 if the **spd_vhi** flag is given.

```
foobar:~# /bin/setserial -v /dev/ttyS2 irq 4 spd_vhi
/dev/ttyS2, UART: 16550A, Port: 0x03f8, IRQ: 4, Flags: spd_vhi
```

There are a number of devices which are called Winmodems – including host-based, HCF-, HSP-, HSF-, controllerless, host-controlled, and soft modems. Winmodems rely on vendor *software* to act as true modems. While software for some versions of Microsoft **Win**dows is available, drivers for Linux are not always. Many USB modems are Winmodems.

Linux support for Winmodems is coordinated by www.linmodems.org. Winmodems –

- Require relatively high CPU speeds to operate reliably (better than 400MHz) (although this is not problematic on modern PC's)
- May require closed source software to operate. As a result you may be tied in to a specific kernel version.
- You will generally have to load a kernel module to make a Winmodem work. This may require some research and recompiling the Linux kernel.

3.2 Sound cards

Installing an ISA sound card is fun, since you seldom have any assistance in selecting the correct parameters, and you have to specify them correctly for the device to work.

3.2.1 ISA cards ... I/O, IRQ and DMA values

When installing new hardware, you need to choose values for I/O ports, IRQ(s) and DMA channels that do not conflict with existing addresses. You configure the card to use the values you choose using jumpers, **isapnp** or the BIOS.

1. Find out what values of I/O, IRQ and DMA are possible. You do this by reading the documentation, or by examining the card for jumpers, etc.
2. Configure all other devices.
3. See which values are free for use by your card by checking **/proc/ioports**, **/proc/interrupts** and **/proc/dma**. If there are no appropriate values, adjust the configuration of other devices in the system.
4. Configure the card to use the values you choose.
5. Configure the kernel module to use the same values.
6. Test it

3.3 PnP sound cards

sndconfig is the console-based RedHat sound configuration tool. It automatically sets up ISA

4 Winmodem is a trademark of US Robotics.

PnP sound cards by configuring them to their default settings as given by **pnpdump**. It is not included on a default installation.

Kernel 2.2

For kernel 2.2 and before, you use isapnptools. **pnpdump** dumps all the possible combinations of the devices. You then edit the file it produces, and it becomes your **/etc/isapnp.conf** file.

```
# /sbin/pnpdump > /etc/isapnp.conf
```

The resulting file contains a number of commented out lines. After removing the comments, the interesting part of the file looks something like this:

```
# Card 1: (serial identifier 48 00 1b e3 d6 e4 00 8c 0e)
# Vendor Id CTL00e4, Serial Number 1827799, checksum 0x48.
# Version 1.0, Vendor version 1.0
# ANSI string -->Creative SB AWE64 PnP<--
# Vendor defined tag: 73 02 45 20
(CONFIGURE CTL00e4/1827799 (LD 0
#      ANSI string -->Audio<--
(INT 0 (IRQ 9(MODE +E)))
(DMA 0 (CHANNEL 3))
(DMA 1 (CHANNEL 7))
(IO 0 (SIZE 16) (BASE 0x0220))
(IO 1 (SIZE 2) (BASE 0x0330))
(IO 2 (SIZE 4) (BASE 0x0388))
(NAME "CTL00e4/1827798[0]{Audio }")
# End dependent functions
(ACT Y)
))
```

The corresponding line for modprobe specifies the parameters to load the sound card module.

```
modprobe sb io=0x220, irq=9, dma=3, dma16=7, mpu_io=0x330,
      midi=0x388
```

The following lines are required in **/etc/modules.conf**:

```
options sb io=0x220, irq=9, dma=3, dma16=7, mpu_io=0x330, midi=0x388
```

Once **/etc/isapnp.conf** exists, you can configure the card(s) to the chosen settings by running **isapnp**, and then load the sound module.

```
# /sbin/isapnp /etc/isapnp.conf
# modprobe sb
```

This is the electronic equivalent of physically changing the jumpers on the sound card to select IO ports, interrupts and DMA channels.

Kernel 2.4

Kernel 2.4 introduces a kernel module **isa-pnp** which handles the configuration of ISA PNP devices automagically. If the module you are using has not been updated to use isa-pnp configuration, you will have to configure it with **pnpdump** and **isapnp**.

3.4 Review

Quiz questions

1. What are the differences between PPP, SLIP and CSLIP? (This is part of the LPIC objectives...)
2. What is the aim of the plug 'n play standard (PnP)?
3. What do you need to do to establish which IRQ, IO Port and DMA settings a PnP compatible card uses?
4. Which program sets up an ISA PnP sound card?
5. Which modems are generally compatible with Linux?
6. Why is it necessary to set the serial port speed to 115200 baud?

Assignment

1. Use the Hayes AT commands to dial a remote modem.
2. See if you can find a computer with an ISA bus, and a sound card for it. Make the sound card play sounds using either cat /dev/urandom > /dev/dsp or **sox**.

Answers to quiz questions

1. PPP is newer, and supports a number of options and dynamically configures to the lowest common denominator. SLIP requires matching configurations at both ends. CSLIP is compressed SLIP.
2. Zero configuration of hardware.
3. Run **pnpdump**
4. **isapnp** with an appropriate configuration file
5. External modems
6. You need to be able to get data from the modem as fast as it is providing it.

4 SCSI devices

> SCSI, n. *[Small Computer System Interface]* A bus-independent standard for system-level interfacing between a computer and intelligent devices. Typically annotated in literature with 'sexy' (/sek'see/), 'sissy' (/sis'ee/), and 'scuzzy' (/skuh'zee/) as pronunciation guides – the last being the overwhelmingly predominant form, much to the dismay of the designers and their marketing people. One can usually assume that a person who pronounces it /S-C-S-I/ is clueless.
>
> *– Jargon File 4.2.0*

It will help you to have a system with SCSI devices for this chapter. You can set up IDE-SCSI by adding **hdc=ide-scsi** or similar to the kernel command line for your CDROM device.

LPIC topic 1.101.4 — Setup SCSI Devices [1]

Weight: 1

Objective:

Candidates should be able to configure SCSI devices using the SCSI BIOS as well as the necessary Linux tools. They also should be able to differentiate between the various types of SCSI. This objective includes manipulating the SCSI BIOS to detect used and available SCSI IDs and setting the correct ID number for different devices particularly the boot device. It also includes managing the settings in the computer's BIOS to determine the desired boot sequence if both SCSI and IDE drives are used.

Key files, terms, and utilities include:

SCSI ID	Each SCSI device has it's own number (usually from 1 to 7)
/proc/scsi/	Linux shows SCSI information here
scsi_info	What you find in /proc/scsi/scsi

4.1 SCSI Architecture

There is a bewildering range of hardware that uses the SCSI protocol, running at speeds from 5MBps to 320 Mbps. Fortunately understanding SCSI cabling is not part of the LPI objectives, so we wish you well and may your SCSI bus always be correctly terminated (on both ends).

By design, devices connected to a SCSI bus can talk to each other. When you set up a SCSI bus, your SCSI host adapter is one of those devices, and each hard disk, cdrom, tape, or scanner is another. In order to use SCSI in Linux the kernel must support the SCSI host adapter. For SCSI disks, tapes and CD-ROMs, no additional software is required, since the protocols for these devices are part of the SCSI protocol.

ID's and jumpers

Each device on a SCSI chain must have a unique SCSI ID. This must be set manually, i.e. by changing jumpers or switches[5].

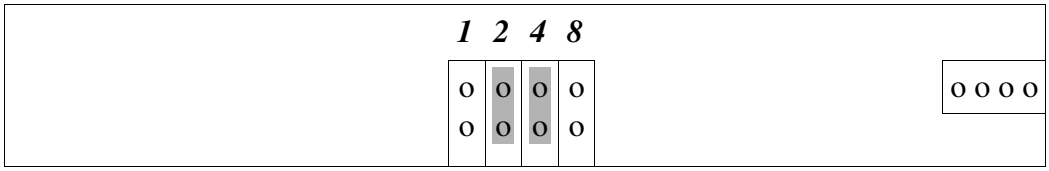


Illustration 1 Jumper settings for device ID=6 (2+4)

It is important not to have two devices on the SCSI chain with the same SCSI ID. The host adapter also needs a device ID. You can use the SCSI BIOS or the Linux tools to find out which SCSI ID's are taken.

4.2 The Linux view of SCSI

4.2.1 Kernel modules

In order to communicate with your SCSI disks, you need to load the appropriate kernel modules, or compile support for your SCSI host adapter into the kernel. If your root file system is on a SCSI disk, you will need to modify **/etc/modules.conf** to contain a line specifying your SCSI host adapter's driver before you run **mkinitrd**.

```
# /etc/modules.conf
 alias scsi_hostadapter "aic7xxx"
```

And then run **mkinitrd**.

```
# cd /boot
# mkinitrd /boot/initrd-2.4.19.img 2.4.19
```

lilo.conf (or **grub.conf**) must specify that this particular initial root disk is loaded on startup:

```
# /etc/lilo.conf
image=/boot/vmlinuz-2.4.19
    label=linux
    initrd=/boot/initrd-2.4.19
    read-only
    root=/dev/sda7
title Boot from SCSI
        root (hd0,6)
        kernel /boot/vmlinuz-2.4.19 ro root=LABEL=/
        initrd /boot/initrd-2.4.19
```

Note that **grub** does not use the Linux disk naming scheme – **hd0** means the “first” hard disk, and not particularly an IDE hard disk.

5 An exception to this is SCA disks which set their own SCSI ID by black magic. The SCA connector includes power, SCSI ID, and the SCSI bus all in one connector. SCA drives are used for RAID and "hot swap" situations.

4.2.2 /proc/scsi

In the directory **/proc/scsi** you will find useful information about the SCSI subsystem.

A very useful entry is **/proc/scsi/scsi** which lists the attached SCSI devices.

```
# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: IBM      Model: DGHS09U          Rev: 03E0
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 06 Lun: 00
  Vendor: PIONEER  Model: CD-ROM DR-U06S   Rev: 1.04
  Type:   CD-ROM                           ANSI SCSI revision: 02
```

Each SCSI adapter (or at least its driver) has an entry in **/proc** such as **/proc/scsi/aic7xxx/0**. The driver name can be something like **aic7xxx** or **BusLogic**. The digit is the bus number, and is relevant if you have more than one SCSI bus in your system.

```
# cat /proc/scsi/aic7xxx/0

Adaptec AIC7xxx driver version: 5.1.19/3.2.4
Compile Options:
  TCQ Enabled By Default : Disabled
  AIC7XXX_PROC_STATS     : Disabled
  AIC7XXX_RESET_DELAY    : 5
Adapter Configuration:
           SCSI Adapter: Adaptec AHA-294X Ultra SCSI host adapter
                         Ultra Wide Controller
    PCI MMAPed I/O Base: 0xeb001000
 Adapter SEEPROM Config: SEEPROM found and used.
      Adaptec SCSI BIOS: Enabled
                    IRQ: 10
                   SCBs: Active 0, Max Active 2,
                         Allocated 15, HW 16, Page 255
             Interrupts: 160328
      BIOS Control Word: 0x18b6
   Adapter Control Word: 0x005b
   Extended Translation: Enabled
Disconnect Enable Flags: 0xffff
     Ultra Enable Flags: 0x0001
 Tag Queue Enable Flags: 0x0000
Ordered Queue Tag Flags: 0x0000
Default Tag Queue Depth: 8
 Tagged Queue By Device array for aic7xxx host instance 0:
  {255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255}
 Actual queue depth per device for aic7xxx host instance 0:
  {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}
Statistics:
(scsi0:0:0:0)
  Device using Wide/Sync transfers at 40.0 MByte/sec, offset 8
  Transinfo settings: current(12/8/1/0), goal(12/8/1/0),
      user(12/15/1/0)
  Total transfers 160151 (74577 reads and 85574 writes)
(scsi0:0:6:0)
  Device using Narrow/Sync transfers at 5.0 MByte/sec, offset 15
```

```
Transinfo settings: current(50/15/0/0), goal(50/15/0/0),
    user(50/15/0/0)
Total transfers 0 (0 reads and 0 writes)
```

The contents of the file depend on the lower level driver. For some drivers, parameters can be set by writing to this file.

4.2.3 Devices

An example of a Linux SCSI device name is **/dev/sda**, which is the first SCSI hard disk. Linux SCSI supports SCSI devices in addition to disks.

Name	*Meaning*	*Usage*
sd	SCSI disk	Used with fdisk, mount and (to some extent hdparm)
sr *or* scd	SCSI CD ROM	Audio and data CD ROMs
st	SCSI tape	Used with tar, cpio and friends, and mt (magnetic tape) for control
sg	SCSI generic character interface	Used by scanners (SANE), CD writers (cdrecord, cdrdao) and for reading audio CD's digitally (cdda2wav, cdparanoia)

4.2.4 SCSI naming

To address a particular SCSI device, the kernel supplies these details to the SCSI adapter.

- host – SCSI adapter number – The SCSI adapter is a host on the SCSI bus. The numbers start at 0 and are allocated arbitarily.
- bus – Channel number – A single SCSI adapter may have a number of SCSI buses attached.
- target – id number – Which remote host on the SCSI bus is being addressed.
- lun – Logical Unit Number – Which of the devices attached to the remote host is being addressed.

Sometimes bus 1, target 3 and lun 40 is written as “1,3,0”. Unless you are using **devfs** (device filesystem), you won't have to pay too much attention to all that.

4.3 Booting off a SCSI disk

You need to convince your motherboard BIOS to boot of the SCSI disk, and you need to tell the SCSI BIOS which particular SCSI disk to boot off:

1. If SCSI and non-SCSI drives are installed, the non-SCSI disk drive is the default boot device. This must be changed with the BIOS configuration program, so that the SCSI drive is the default boot device. On at least some BIOS programs you set “BIOS bootorder” to “SCSI, C, A”.
2. In the SCSI BIOS configuration program, you mst set the Boot SCSI ID and LUN to the ID and LUN of the SCSI disk you wish to boot. The SCSI ID of SCSI disks is normally set with jumpers or switches on the drive.

If you have an Adaptec SCSI card, you will see a prompt during the BIOS boot sequence, something like the following:

```
Press <Ctrl><A> for SCSISelect(TM) Utility!
```

SCSI BIOS configuration Screen:

```
SCSI Bus Interface Definitions
Host Adapter SCSI ID:           7
SCSI Parity Checking:           Enabled
Host Adapter SCSI Termination:  Automatic

Boot Device Options
Boot SCSI ID:                   0
Boot LUN Number:                0
```

4.4 Review

Quiz questions

1. What are the advantages of using SCSI peripherals?
2. What types of SCSI architecture are available?
3. Why are SCSI disks more expensive than IDE disks?
4. What is the purpose of the SCSI ID?
5. How does one set the SCSI ID of a device?
6. How does one configure booting off a SCSI disk if an IDE disk is present?
7. Where does Linux provide general information about the SCSI systems present in your computer?

Assignment

1. Configure your BIOS to boot Linux from a chosen SCSI disk, without relying on a floppy or IDE disk.
2. Find out which SCSI bus format is the most popular in your country.
3. Find out what **ide-scsi** is, and why it is used.

Answers to quiz questions

1. SCSI removes load from the CPU, as it is a well-designed protocol. As a result, SCSI works faster.
2. Far too many.
3. They're better.
4. The ID uniquely identifies each device (or controller) on the SCSI chain.
5. Manually – usually by changing jumpers, but occasionally via software.
6. Set the SCSI boot device to the disk, and set the BIOS to boot off the SCSI disk. Install a boot loader on the SCSI disk.
7. /proc/scsi/scsi

5 PC cards

> *Is there an API or other means to determine what video card, namely the chipset, that the user has installed on his machine?*
>
> On a modern X86 machine use the PCI/AGP bus data. On a PS/2 use the MCA bus data. On nubus use the nubus probe data. On old style ISA bus PCs don a large pointy hat and spend several years reading arcane and forbidden scrolls
>
> *– Alan Cox on hardware probing*

For this chapter you should know how to log in as root and execute commands. It will help you to have a system with unusual ISA and PCI cards to experiment with.

LPIC topic 1.101.5 — Setup different PC expansion cards [3]

Weight: 3

Objective:

Candidates should be able to configure various cards for the various expansion slots. They should know the differences between ISA and PCI cards with respect to configuration issues. This objective includes the correct settings of IRQs, DMAs and I/O ports of the cards, especially to avoid conflicts between devices. It also includes using isapnp if the card is an ISA PnP device.

Key files, terms, and utilities include:

/proc/dma	Direct memory addressing channels assigned to devices
/proc/interrupts	Hardware interrupts assigned to devices
/proc/ioports	Input/Output ports assigned to devices
/proc/pci	PCI bus information, as shown by **lspci**
pnpdump(8)	Dump possible settings for ISA Plug and play
isapnp(8)	Tell ISA Plug and play devices what ports to use
lspci(8)	Show PCI bus information

5.1 Bus architecture

A bus is physically a set of parallel conductors that connect the components of a computer[6]. The components of the computer use the bus to communicate by sending electrical signals to each other. The components that are connected by the bus include the CPU, the system memory (RAM and ROM) and peripherals (e.g. disk interface, display adapter, interrupt controller).

5.2 Bus resources

The following bus concepts are used in both the ISA bus and PCI bus.

6 A bus is also a long motor vehicle for carrying passengers, usually along a fixed route. The fixed route part is the reason that the term is used in computers.

- I/O addresses – to specify a specific component on a bus, the CPU uses its I/O address. This means that systems on the bus cannot share I/O addresses.
- Memory addresses – to read from or write to a particular location in memory the CPU hardware uses a memory address. The ISA bus separates the concept of memory addresses from I/O addresses, although there is not a huge difference between these.
- Interrupt requests (IRQs) – when a peripheral requires CPU attention, it issues an interrupt request. The CPU, in response to the request, stops what it is doing, and speaks to the relevant peripheral (usually to retrieve data from it, or to send more data to it). Busses generally have a number of interrupt request lines, which enables the kernel to determine which peripheral caused the interrupt. If two peripherals share an interrupt request line, it is possible that they will trigger an interrupt request *simultaneously*. The kernel must respond to *both* devices for proper operation. Shared interrupts work when interrupts are Level triggered (e.g. on the PCI bus) and not possible when interrupts are edge triggered (e.g. on the ISA bus).
- DMA channels – Direct Memory Access – Using a DMA channel allows a peripheral to copy data directly into or out of the computer's memory. This means that the CPU is not intimately involved with the data transfer once it is in motion. Use of a DMA channel allows a device to perform direct memory access. It is generally not possible to share DMA channels between peripherals.

5.3 Bus conflict resolution

The Linux kernel is responsible for managing the assignment of IRQs (interrupt request lines), I/O ports and DMA channels. The kernel will not assign a resource to two drivers, unless it is shareable (i.e. a level triggered interrupt on a PCI bus).

A number of resource conflicts can occur on the bus.

- I/O address conflict – either one or both of the devices will fail to work. Even if one of the devices is unconfigured, and no kernel module has been loaded for it, it may respond to its I/O address and cause problems.
- Shared IRQs – sharing IRQ's with level triggered interrupts *can* cause conflicts if the kernel only services one of the devices, and not both. This is not likely with newer kernels, but it can happen. Sharing IRQ's with edge triggered interrupts will cause one or both of the devices to fail. If a kernel module for a given device is not loaded, the device will never generate interrupts, and will coexist with other devices.
- DMA channels cannot be shared. The kernel will refuse to assign a DMA channel to two devices. The device for which a module is loaded first will work.

To resolve conflicts for resources, you need to reconfigure the devices so that they do not conflict:

- Really old cards (and some new ones) are configured by changing jumpers on the card.
- **isapnp** can change the resource assignments for plug and play cards. Some PC BIOS's contain this facility too.
- Often the BIOS includes facilities for changing the interrupts and memory locations of built-in devices.

- IRQ conflicts between devices on the PCI bus and the ISA bus can be resolved by telling the BIOS to reserve certain IRQ lines for the ISA bus. I/O ports do not conflict, since the PCI bus uses an independent range of port addresses.

5.4 PCI card configuration

The Peripheral Component Interconnect (PCI) bus includes resource assignment as part of its specification. To make a PCI card work, you merely have to load the correct kernel module. To determine which kernel module will work, you can consult the output of the **lspci** command.

The module is then loaded using **modprobe modulename**. If there are options required for the modules, you enter them in **/etc/modules.conf** and run **depmod -a**.

lspci – list PCI devices

Each device on the PCI bus is plugged into a specific slot, and identifies itself by manufacturer and device number:

```
bar:~ # /sbin/lspci -n
00:00.0 Class 0600: 1106:0305 (rev 03)
00:01.0 Class 0604: 1106:8305
00:07.0 Class 0601: 1106:0686 (rev 40)
00:07.1 Class 0101: 1106:0571 (rev 06)
00:07.2 Class 0c03: 1106:3038 (rev 1a)
00:07.3 Class 0c03: 1106:3038 (rev 1a)
00:07.4 Class 0680: 1106:3057 (rev 40)
00:07.5 Class 0401: 1106:3058 (rev 50)
00:09.0 Class 0200: 1282:9102 (rev 31)
01:00.0 Class 0300: 10de:002d (rev 15)
```

Happily, **lspci** can display the names of the devices by cross referencing against **/usr/share/pci.ids** which contains a list of vendors and their products. Even if the vendor ID is not listed, **lspci** will tell you what type of device it is.

To list the components on the PCI bus, you can look at the file **/proc/pci** or use the **lspci** command.

```
foo@bar:~> /sbin/lspci
00:00.0 Host bridge: VIA Technologies, Inc. VT8363/8365
      [KT133/KM133] (rev 03)
00:01.0 PCI bridge: VIA Technologies, Inc. VT8363/8365 [KT133/KM133
      AGP]
00:07.0 ISA bridge: VIA Technologies, Inc. VT82C686 [Apollo Super
      South] (rev 40)
00:07.1 IDE interface: VIA Technologies, Inc. VT82C586B PIPC Bus
      Master IDE (rev 06)
00:07.2 USB Controller: VIA Technologies, Inc. USB (rev 1a)
00:07.3 USB Controller: VIA Technologies, Inc. USB (rev 1a)
00:07.4 Bridge: VIA Technologies, Inc. VT82C686 [Apollo Super ACPI]
      (rev 40)
00:07.5 Multimedia audio controller: VIA Technologies, Inc. VT82C686
      AC97 Audio Controller (rev 50)
00:09.0 Ethernet controller: Davicom Semiconductor, Inc. Ethernet
```

```
      100/10 MBit (rev 31)
01:00.0 VGA compatible controller: nVidia Corporation RIVA TNT2
      Model 64 (rev 15)
```

lspci can show more information. Here's **lspci** displaying information for a specific slot:

```
nobar:~ # /sbin/lspci -s 00:09 -v
00:09.0 Ethernet controller: Davicom Semiconductor, Inc. Ethernet
      100/10 MBit (rev 31)
        Subsystem: Unknown device 3030:5032
        Flags: bus master, medium devsel, latency 32, IRQ 11
        I/O ports at e800 [size=256]
        Memory at dd000000 (32-bit, non-prefetchable) [size=256]
        Expansion ROM at <unassigned> [disabled] [size=256K]
        Capabilities: [50] Power Management version 1
```

This is quite similar to the information in **/proc/pci** (the listing below is a partial listing).

```
bar:~ # cat /proc/pci
PCI devices found:
      ###################### lotsa stuff ######################
  Bus  0, device   9, function  0:
    Class 0200: PCI device 1282:9102 (rev 49).
      IRQ 11.
      Master Capable.  Latency=32.  Min Gnt=20.Max Lat=40.
      I/O at 0xe800 [0xe8ff].
      Non-prefetchable 32 bit memory at 0xdd000000 [0xdd0000ff].
  Bus  1, device   0, function  0:
    Class 0300: PCI device 10de:002d (rev 21).
      IRQ 10.
      Master Capable.  Latency=32.  Min Gnt=5.Max Lat=1.
      Non-prefetchable 32 bit memory at 0xda000000 [0xdaffffff].
      Prefetchable 32 bit memory at 0xd8000000 [0xd9ffffff].
```

5.5 ISA card configuration

To change the settings that an ISA card is using generally requires changing the jumpers on the card. The section following this one explains how to change the settings on an ISA card which supports ISA Plug'n Play. With a few exceptions, the kernel modules supporting ISA cards have to be configured with specific parameters. Most drivers are capable of detecting interrupt values, so the critical values to assign are the I/O addresses.

The best way to assign values is to read the relevant documentation and set the jumpers according to the free resources on your PC. However, you can sometimes guess. Here's an example of loading the kernel module for a NE-2000 Ethernet Adapter. The NE2000 driver requires the 8390 driver:

```
bar:~ # insmod 8390.o
bar:~ # insmod ne.o
ne.o: init_module: No such device or address
Hint: insmod errors can be caused by incorrect module parameters,
      including invalid IO or IRQ parameters.
      You may find more information in syslog or the output from
      dmesg
```

Okay, so that didn't work – let's guess an IO address ...

```
bar:~ # insmod ne.o io=0x220
ne.o: init_module: No such device or address
Hint: insmod errors can be caused by incorrect module parameters,
      including invalid IO or IRQ parameters.
      You may find more information in syslog or the output from
      dmesg
```

So we learn that the network card is not at address **0x220**. To try every conceivable address, we can do something like this

```
bar:~ # for ((IO=0x200;IO<0x380;IO+=0x10)) ; do IOX=`printf '%x' $IO
      `; insmod ne io=$IOX ; done
bar:~ # lsmod
```

5.6 ISA PnP devices

A “Plug and Play” (PnP) device is very similar to a card which is configured with jumpers – except that the jumpers are missing. ISA PnP devices are automagically configured by the BIOS or the operating system.

Before the ISA PnP standard was devised, many hardware manufacturers configured their hardware's I/O addresses using special MS-DOS based utility programs. The **isapnp** program replaces this functionality.

To configure a PnP device:

- Use **pnpdump > /etc/isapnp.conf** to determine possible configuration settings for the device.
- Edit **/etc/isapnp.conf** and remove comments (optionally change the parameters from the defaults)
- Run **isapnp /etc/isapnp.conf** to set the parameters the device actually uses.
- Load a kernel module for the device.

The part of the file **/etc/isapnp.conf** which needs to be edited looks something like this:

```
(CONFIGURE TCM5094/614027740 (LD 0
#     Compatible device id PNP80f7
#     IRQ 3, 5, 7, 9, 10, 11, 12 or 15.
#         High true, edge sensitive interrupt (by default)
# (INT 0 (IRQ 3 (MODE +E)))
#     Logical device decodes 16 bit IO address lines
#         Minimum IO base address 0x0210
#         Maximum IO base address 0x03e0
#         IO base alignment 16 bytes
#         Number of IO addresses required: 16
# (IO 0 (SIZE 16) (BASE 0x0210) (CHECK))
 (NAME "TCM5094/614027740[0]{3Com 3C509B EtherLink III}")
# (ACT Y)
))
# End tag... Checksum 0x00 (OK)
```

And you change it by removing selected comments.

```
(CONFIGURE TCM5094/614027740 (LD 0
#     Compatible device id PNP80f7
#     IRQ 3, 5, 7, 9, 10, 11, 12 or 15.
```

```
#         High true, edge sensitive interrupt (by default)
(INT 0 (IRQ 3 (MODE +E)))
#     Logical device decodes 16 bit IO address lines
#         Minimum IO base address 0x0210
#         Maximum IO base address 0x03e0
#         IO base alignment 16 bytes
#         Number of IO addresses required: 16
(IO 0 (SIZE 16) (BASE 0x0210) (CHECK))
 (NAME "TCM5094/614027740[0]{3Com 3C509B EtherLink III}")
(ACT Y)
))
# End tag... Checksum 0x00 (OK)
```

Newer kernels include a module named **isa-pnp** which automatically configures ISA PnP cards. You can configure misbehaving and unsupported cards manually by writing to **/proc/isapnp**, although using **pnpdump** and **isapnp** may prove simpler.

5.7 Kernel interface commands

The following kernel interface files in **/proc** display the assigned bus resources:

/proc/interrupts

For each CPU **/proc/interrupts** shows the interrupts which are in use by the kernel, and the number of times each interrupt has triggered.

```
[root@onecpu /proc]# cat /proc/interrupts
           CPU0
  0:     810743          XT-PIC  timer
  1:      33722          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  5:          0          XT-PIC  usb-uhci, via82cxxx
  8:          1          XT-PIC  rtc
 10:      16270          XT-PIC  O2 Micro, Inc. 6832, O2 Micro, Inc.
     6832 (#2), eth0
 12:      61166          XT-PIC  PS/2 Mouse
 14:      73642          XT-PIC  ide0
 15:          1          XT-PIC  ide1
NMI:          0
ERR:          0
```

SMP processors display more interesting information, such as the allocation of interrupts to the processors.

```
[root@threecpio /proc]# cat /proc/interrupts
           CPU0       CPU1
  0:    1245694    1389471 IO-APIC-edge timer
  1 :       157        121 IO-APIC-edge keyboard
  2:          0          0 XT-PIC cascade
  8:          0          1 IO-APIC-edge rtc
 10:      10685      10958 IO-APIC-level usb-uhci, e100
 14:      36926      50216 IO-APIC-edge ide0
 15:          1          1 IO-APIC-edge ide1
NMI:          0          0
LOC:    2635076    2635074
```

```
ERR:          0
MIS:          0
```

/proc/ioports and iomem

The I/O ports which are in use by kernel drivers are displayed by the **ioports** interface

```
baa:~ # cat /proc/ioports
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
01f0-01f7 : ide0
0210-021f : 3c509 PnP
0233-0233 : isapnp read
0240-025f : eth0
02f8-02ff : serial(auto)
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
0a79-0a79 : isapnp write
0cf8-0cff : PCI conf1
d400-d40f : PCI device 1039:5513
```

In this particular configuration, it will not work well to add an additional card requiring the address **0x210**, since that is in use (by a 3c509 network adapter).

The **iomem** interface is provided from kernel 2.4. It shows 32 bit memory-mapped I/O allocations:

```
baa:~ # cat /proc/iomem
00000000-0009ffff : System RAM
000a0000-000bffff : Video RAM area
000c0000-000c7fff : Video ROM
000f0000-000fffff : System ROM
00100000-01ffffff : System RAM
  00100000-0025b584 : Kernel code
  0025b585-002c0eab : Kernel data
f4000000-f7ffffff : PCI device 5333:8901
fffe0000-ffffffff : reserved
```

The PCI device in the above listing is a VGA display adapter. Memory mapped devices include network adapters, PCMCIA, ACPI support and video ROM.

/proc/dma

The **dma** interface shows the DMA assignments in use for the ISA bus.

```
baa:~ # cat /proc/dma
 1: SoundBlaster8
 4: cascade
 5: SoundBlaster16
```

5.8 Review

Quiz questions

The answers to these questions appear in this chapter.

1. What is the purpose of an interrupt?
2. What is the purpose of an I/O address?
3. What is the purpose of a DMA channel?
4. What happens when there is a resource conflict between two devices?
5. Under what circumstances can two devices share the same IRQ value?

Assignment

1. Experiment with **lpsci** and **pnpdump** (perhaps **lspnp**) on your PC. Write an explanation of how the output from the commands corresponds with the contents of **/proc/interrupts**, **/proc/dma** and **/proc/ioports**.
2. Obtain an expansion card (such as a network card) and install it in your computer. Explain what effect merely plugging in the card has on the BIOS, and on the output of various diagnostic commands. Load an appropriate kernel module for the card, and write an explanation of the effect this has on the output of the diagnostic commands.

Answers to quiz questions

1. An interrupt indicates that a peripheral requires CPU attention – i.e. reading or writing data.
2. An IO address is used on a bus to identify a particular peripheral or a component of a peripheral.
3. A DMA channel is used for coordinating data transfers between peripherals and memory. This frees the CPU for more productive work.
4. Either one or both of the devices does not work. It is possible that one of the devices may function unreliably.
5. On a bus which uses level triggered interrupts it is possible to share interrupt values. On the ISA bus and the EISA bus interrupt sharing is not possible. On the PCI bus, interrupt sharing is the norm.

6 Device configuration

> Some of my readers ask me what a “Serial Port” is.
> The answer is: I don't know.
> Is it some kind of wine you have with breakfast?
>
> – *the fortune cookie database (no attribution)*

ISDN, DSL and PPP are all ways of getting connected from where you are to where you want to be. Note that some of the objectives in this section are handled in the chapter on modem and sound card hardware (namely the usage of **setserial** and modem compatibility requirements). By the end of this chapter, you should know enough to set up a communications controller, but not necessarily to how to use it for networking.

LPIC topic 1.101.6 — Configure Communication Devices [1]

Weight: 1

Objective:

Candidates should be able to install and configure internal and external communication devices such as modems, ISDN adapters, and DSL switches. This objective includes verification of compatibility requirements (especially important if that modem is a winmodem), necessary hardware settings for internal devices (IRQs, DMAs, I/O ports), and loading and configuring suitable device drivers. It also includes communication device and interface configuration requirements, such as the correct serial port for 115.2 Kbps, and the correct modem settings for outbound PPP connection(s).

Key files, terms, and utilities include:

/proc/dma	Direct memory accessing channels in use
/proc/interrupts	Interrupts in use
/proc/ioports	I/O ports in use
setserial(8)	Configure serial port access for an internal modem

6.1 PPP connections

Networks like ethernet, frame relay and ATM connect a number of computers together. A computer connected to the network can speak to a number of other computers simultaneously over the same connection.

Point to point connections connect two computers to each other. Point to point connections are commonly used for connecting to an Internet service provider. People pay their Internet service providers money so that they can join a well-connected network.

There are a number of methods of doing this:

- Analogue modem – we phone our ISP on a standard phone, and our modem talks to their modem, so that our computer can talk to their computer.
- ISDN modem – we phone our ISP on a fancy digital phone, and our ISDN terminal adapter talks to their ISDN terminal adapter so that our computers can talk to each other.

- DSL – the telephone company installs a special line that links directly to our ISP's DSL concentrator via the local telephone exchange(s). Our computer talks to a fancy DSL modem which talks to our ISP's expensive equipment, which does black magic to connect us to the network. ADSL is the same, just cheaper (for the phone company, that is).

6.2 Types of modem

The basic function of a modem is to provide a digital text communication channel in two directions. The point to point protocol (PPP) uses the modem connection for networking protocols. Two machines communicating via PPP each run **pppd** which sends information to the kernel for networking processing.

6.2.1 PPP on analogue modems

One of the properties of PPP is that it is a peer to peer protocol. Either or both sides of the connection can request authentication. Generally, when negotiating an outbound link the person dialing out will not require authentication, and the server which answers the call will require authentication.

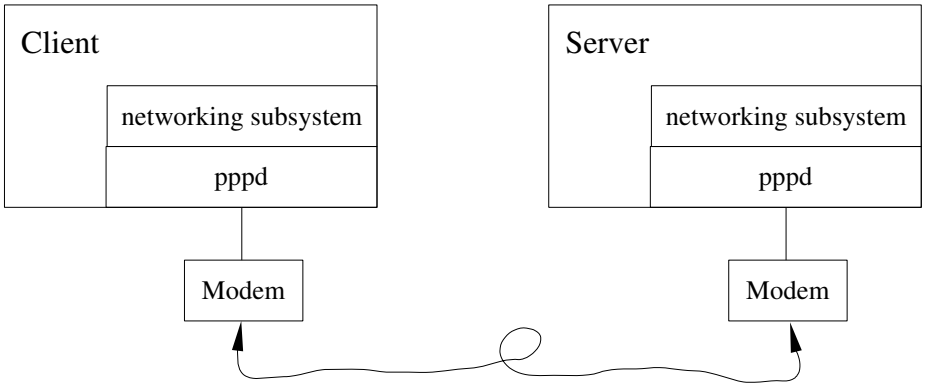


Illustration 2: pppd runs on both sides of a point to point link, creating a peer to peer network link

In order to dial up to an internet service provider (ISP), you will need to find an appropriate means to supply the following data:

- Modem to use (e.g. /dev/ttyS0) (Remember that if this is a winmodem, you will have to load a working kernel module ... which may not be possible. Also remember that for an internal modem you will have to first configure it with **setserial**.).
- Telephone number to dial (e.g. 555 5555)
- User name that the ISP expects (e.g. “user42”).
- Password for that user (e.g. “bigsecret”).

There are a large variety of programs that can be used to set up the PPP parameters – kppp, wvdial. A number of distributions provide an interface activation script, and **ifup ppp0** will activate the link.

6.2.2 pppd call myisp

To activate your ISP link using **pppd** directly:

/etc/ppp/peers/myisp – this file is consulted when you run **pppd call myisp**:

```
ttyS0 19200 crtscts
connect "chat /etc/ppp/chat-myisp"
user user42
```

The chat script **/etc/ppp/chat-myisp** provides the modem initialisation string and details how to dial the ISP's number.

```
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
ABORT "BUSY"
ABORT "Username/Password Incorrect"
"" "at"
OK "at&d0&c1"
OK "atdt55555555"
CONNECT
```

To supply the user name and password, you need an entry in either **/etc/ppp/chap-secrets** or **/etc/ppp/pap-secrets**, depending on the authentication method your ISP uses. You can put the entry in both if you choose.

```
"user42"   *       "bigsecret"
```

When you run **pppd call myisp**, the output is added to **/var/log/messages** (dependant on the configuration of your **syslogd**).

6.3 ISDN adapters

Internal ISDN adapters

The kernel module that supports most internal ISDN terminal adapters is the **hisax** module, which supports the **hisax** ISDN chipset manufactured by Siemens.

External ISDN terminal adapters

There are a number of external ISDN terminal adapters available which emulate the functions of an analogue modem. To connect one of these, you *usually* have to specify the correct initialisation string. This may be one of the following (but consult the .INF file on the Windows driver disk if in doubt):

```
ATB30                   # I've seen it work on a ... junk modem ...
AT*PPP=1                # At least one US Robotics requires this
```

6.4 DSL

DSL uses a technique called PPP over Ethernet, or **pppoe**. For DSL to work, you need a properly configured ethernet card, and then you run

```
pppd pty 'pppoe [pppoe_options]' [pppd_options]
```

Fortunately, there's usually a script to do this

6.5 Diagnostic tools

The following tools are useful for diagnostics, even if they are covered in more detail in the following course.

ifconfig

ifconfig shows the currently configured interfaces. If one of the interfaces is **ppp0** or similar, then the ppp connection is active. If **ppp0** is absent, the connection is inactive.

route -n

When a PPP connection is active, there will be a route to the remote side of the PPP connection. Usually there will be a route to the network on the other side. It is a common PPP error to have a default route configured to your local network when starting PPP. In this situation, PPP does not override the default route with its own.

ping

To diagnose a PPP connection using **ping**, the procedure is:

1. Ping the loopback interface (**ping 127.0.0.1**)
2. Ping the local side of the PPP connection.
3. Ping the remote side of the PPP connection.
4. Ping the default gateway of the PPP connection.

If one of these fails, you have a routing problem, or a problem with your PPP connection (e.g. an interfering firewall).

6.6 Review

Quiz questions

1. Which communication devices are not generally Linux compatible?
2. Why is it necessary to use **setserial** when using an internal modem for PPP?
3. What is the purpose of the baud rate setting for a modem?
4. How does Linux handle an external ISDN terminal adapter?
5. Which modules exist for using with internal ISDN adapters?
6. Which files configure **pppoe**?
7. How do you activate a DSL connection?

Assignment

1. Obtain a modem and a telephone line and then set up an actual network connection to your own ISP. Check that you can browse the web using a web browser such as **lynx** or **w3m**. Make notes about how you configured the modem.

2. The Linux **pppoe** daemon, **pppoed**[7] provides the same functionality as a DSL modem. Set up a **pppoed** server and connect to it using **pppoe** from another computer in the network. Diagnose the connection using the techniques described in this chapter. Does the server route packets for you?

Answers to quiz questions

1. Winmodems, which includes a number of USB devices.
2. To tell the kernel what the IO address and IRQ is for the modem.
3. To set the speed of communication (in bits per second).
4. It appears to be an external analogue modem.
5. hisax – and you have to tell it which particular chip it uses.
6. Usually **/etc/ppp/pppoe**, but it may be different in your distribution.
7. **pppd pty 'pppoe [pppoe_options]' [pppd_options]**

7 This question is strictly for those genius types who are too smart for their own good. You guys must learn how to hide your talents a bit so the rest of us can feel better.

7 USB hardware

> Looks nice to me but about the only way you are likely to get Linus to take in kernel debugging patches is to turn them into hex and disguise them as USB firmware ;)
>
> *– Alan Cox's guide on submitting Linux patches, today chapter #3, kernel debuggers*

For this chapter, it will be helpful to have some USB hardware to experiment with. Get a USB scanner, a USB memory stick, a USB camera or anything.

LPIC topic 1.101.7 — USB hardware [1]

Weight: 1

Objective:

Candidates should be able to activate USB support, use and configure different USB devices. This objective includes the correct selection of the USB chipset and corresponding module. It also includes the knowledge of the basic architecture of the layer model of USB as well as the different modules used in the different layers.

Key files, terms, and utilities include:

lspci(8)	list PCI devices
usb-uhci.o	UHCI chipset support kernel module
usb-ohci.o	OCHI chipset support kernel module
/etc/usbmgr/	USB hotplug information (the one)
usbmodules	Which USB modules might support connected hardware
/etc/hotplug	USB hotplug information (the other)

7.1 USB architecture

In 1994 Compaq, Intel, Microsoft and NEC designed the Universal Serial Bus (USB) specifications with the following goals:

- Ease-of-use
- PC based telephony must be possible
- Port expansion (i.e. you are not limited to the number of physical ports on your computer).

USB Version 2.0 was announced in 1999. USB is a hierarchical bus and it is controlled by one host. All communication on the bus is initiated by the host and devices cannot establish a connection to other devices. USB version 1 allows 127 devices to be connected at a time, and operates at a speed of 12Mbit/s. In practical circumstances, actual throughput cannot exceed about 8.5Mbit/s under the best conditions. USB version 2 runs at up to 480Mbit/s – approaching the speed of competing technologies.

To accommodate more devices on the bus than you have ports for, you use a USB hub – either bus powered, or self-powered. Only low power devices can be attached to a self-powered hub.

7.2 USB chipsets and drivers

There are two USB host controller specifications available. Two kernel modules support these controllers.

- **usb-ohci.o** – The Open Host Controller Interface (OHCI, by Compaq) (used by the Nvidia chipset)
- **usb-uhci.o** – The Universal Host Controller Interface (UHCI, by Intel) standard (simpler hardware, more complex software, marginally higher CPU load). (Used by the VIA chipset, and, of course, Intel.)

To determine which controller you have in a particular computer, you use **lspci**. The particular controller in the systems shown below is a UHCI controller.

```
foodbar:~ $ /sbin/lspci -v | grep -i usb
00:07.2 USB Controller: VIA Technologies, Inc. USB (rev 0a) (prog-if
      00 [UHCI])
        Subsystem: VIA Technologies, Inc. (Wrong ID) USB Controller

cashbar:~ $ /sbin/lspci -v | grep -i usb
00:02.2 USB Controller: Intel Corporation 82371AB PIIX4 USB (rev 01)
      (prog-if 00 [UHCI])
```

As USB becomes increasingly popular, you are likely to come across the following kernel modules which support specific classes of USB devices:

- **mousedev** – USB mouse or pointer
- **keybdev** – USB keyboard
- **usb-storage** – CD-ROM, hard disk, or ZIP drives
- **audio** – USB sound cards
- **joydev** – USB joystick

All of these rely on the **usbcore.o** and either **usb-uhci.o or usb-ohci.o**.

7.3 USB protocol

USB provides the following basic capabilities to the other kernel modules:

- Control messages – configuration of peripherals and small messages.
- Bulk transfers – used by scanners and hard disk adapters.
- Interrupt transfers – these are bulk transfers that are polled periodically – with an interval of 1ms to 127ms.
- Isochronous transfers – to send or receive data at high speed, but without guarantees of reliability. Audio and video devices use this kind of transfer.

The USB subsystem of the kernel provides basic USB connectivity (i.e. low level interface). This is sufficient for a device to “appear” on the USB bus, but to do something useful, the correct kernel module must be loaded.

USB devices identifies themselves on the bus by vendor, and also by USB class (e.g. mouse, keyboard, etc). The kernel makes this information available in **/proc/bus/usb/** and **lsusb** tool shows it in more readable format.

```
cheapo:~ # /sbin/lsusb
Bus 002 Device 001: ID 0000:0000
```

```
Bus 001 Device 001: ID 0000:0000
Bus 001 Device 007: ID 04a5:20b0 Acer Peripherals Inc. S2W
      3300U/4300U
```

With **-v** more information is shown (a lot more).

```
cheapo:~ #  lsusb -s 001:007 -v | head -11
Bus 001 Device 007: ID 04a5:20b0 Acer Peripherals Inc. S2W
      3300U/4300U
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               1.00
  bDeviceClass          255 Vendor Specific Class
  bDeviceSubClass       255 Vendor Specific Subclass
  bDeviceProtocol       255 Vendor Specific Protocol
  bMaxPacketSize0         8
  idVendor           0x04a5 Acer Peripherals Inc.
  idProduct          0x20b0 S2W 3300U/4300U
  bcdDevice            1.20
  iManufacturer           1 Color
  iProduct                2  FlatbedScanner 23
```

7.4 usbmgr

If your system does not use the kernel based **/sbin/hotplug**, you are probably using **usbmgr**, and your kernel is probably version 2.2. You will probably also not have **lsusb** installed.

usbmgr is user-mode daemon which loads and unloads USB kernel modules. When USB devices connect into or disconnect from a USB hub, **usbmgr** does the following, based on its configuration directory **/etc/usbmgr/**

- Loads and unloads Linux kernel modules.
- Execute a file to setup USB devices.

Here's what you find in the **/etc/usbmgr** directory:

```
gabriel:/etc/usbmgr # ls -F
class/  host  preload.conf  usbmgr.conf  vendor/
```

The **class/** directory specifies which module should be loaded for the various classes of USB device:

```
gabriel:/etc/usbmgr # find class -type f | xargs grep .
class/03/01/01/module:hid
class/03/01/01/module:keybdev
class/03/01/02/module:hid
class/03/01/02/module:mousedev
class/07/01/01/module:printer
class/07/01/02/module:printer
class/07/01/03/module:printer
class/02/module:acm
class/01/01/module:audio
class/08/04/module:scsi_mod
class/08/04/module:sd_mod
class/08/04/module:usb-storage
class/e0/01/01/module:bluetooth
```

```
class/09/module:none
```

The **vendor** directory specifies vendor and product ID's and the corresponding module that should be loaded.

```
gabriel:/etc/usbmgr # find vendor -type f | xargs grep . | tail
vendor/04bb/0904/script:network
vendor/05e9/0009/module:pegasus
vendor/05e9/0009/script:network
vendor/082d/0100/module:usb-serial
vendor/07b5/9902/module:hid
vendor/07b5/9902/module:joydev
vendor/047d/3002/module:hid
vendor/047d/3002/module:joydev
vendor/050d/0004/module:plusb
vendor/067b/0000/module:plusb
```

7.5 /sbin/hotplug

When a USB device (or a PCMCIA card, etc.) is plugged in kernel 2.4 this triggers a chain of events:

- The kernel runs the program **/sbin/hotplug**.
- **/sbin/hotplug** runs the appropriate script from **/etc/hotplug/*.agent.**
- If the device is USB, **/etc/hotplug/usb.agent** runs **usbmodules** to find out which kernel modules may be able to manage interfaces on currently plugged in USB devices.

What **/etc/hotplug/usb.agent** does is more or less the equivalent of this:

```
for module in $(usbmodules --device $DEVICE) ; do
      modprobe -s -k "$module"
done
```

The configuration files for **/etc/hotplug/usb.agent** are in **/etc/hotplug**:

- **usb.distmap**, **usb.usermap** – a list of kernel modules, and the USB id's for which they are suitable.
- **usb.handmap** – a list of kernel modules which are hot-pluggable on kernel 2.2.
- **usb.rc** – startup and shutdown script for USB.

7.6 Review

Quiz questions

1. Why are there two USB drivers?
2. What are the layers in the USB model?
3. Which files configure hotplugging?

Assignment

1. Connect a USB device on your computer, such as a mouse, a hard disk, scanner or memory stick. Write down the name and model number of the device, and identify which kernel modules are loaded when you plug the device in. If no kernel modules are loaded, verify that **hotplug** or **usbmgr** is configured and running, and whether the device is supported.

2. Configure your computer to use the device, i.e. configure X to use your mouse, or mount your USB disk, or set up **xsane** for your hard disk or **gphoto** for your camera.

Answers to quiz questions

1. There are two USB interface standards
2. The basic USB layer, and the high level device communication layer (i.e. usb-uhci, usbcore, keybdev).
3. Either **/etc/hotplug** or **/etc/usbmgr**, depending on your kernel and software versions.

8 Partitioning disks

Nine megs for the secretaries fair,
Seven megs for the hackers scarce,
Five megs for the grads in smoky lairs,
Three megs for system source
One disk to rule them all,
One disk to bind them,
One disk to hold the files
And in the darkness grind 'em.

– *fortune cookie database*

In order to keep your computer neat, disk space is divided into areas called “partitions”. Partitioning a disk makes it possible to allocate parts of the disk to various operating systems and specific purposes. For beginners, partitioning disks is the most complicated part of installing a Linux system.

LPIC topic 1.102.1 — Design hard disk layout [5]

Weight: 5

Objective:

Candidates should be able to design a disk partitioning scheme for a Linux system. This objective includes allocating filesystems or swap space to separate partitions or disks, and tailoring the design to the intended use of the system. It also includes placing /boot on a partition that conforms with the BIOS' requirements for booting.

Key files, terms, and utilities include:

/ (root) filesystem	Where everything goes
/var filesystem	Variable stuff
/home filesystem	User files
swap space	Extra memory
mount points	“Directories” where file systems appear
partitions	Slices of a hard disk
cylinder 1024	What an older BIOS flips above

8.1 Disks and partitions

The whole aim of partitioning disks is to organise data. Once you have organised the data, the kind of data you can put in a partition is ...

- A filesystem for Linux (e.g. ext2, ext3, reiser, vfat)
- A filesystem for another operating system (e.g. Windows NTFS, BSD UFS).
- Swap space (extra memory)
- Other partitions (“logical partitions” in the “extended partition”)

The partition table contains the following information for each partition:

- The partition type (i.e. which operating system does it belong to)

- The starting cylinder for the partition
- The ending cylinder for the partition

Linux naming scheme

The Linux disk naming scheme identifies each disk and each partition on each disk. IDE disk names are prefixed “hd” (for “hard disk”) and numbered “a”, “b”, “c”, “d” (and perhaps “e”, “f” and more). SCSI disks are prefixed “sd” (for “SCSI disk”). Partitions are numbered, starting at 1 and up to 32 (if you have 32 partitions, you need help).

Here are a few examples of disk and partition names:

Device name	***Meaning***
/dev/hda	devices, hard disk “a” (IDE, primary channel, master)
/dev/hdb	devices, hard disk “b” (IDE, primary channel, slave)
/dev/hdc5	devices, hard disk “a”, partition “5” (IDE, secondary channel, master)
/dev/hdd2	devices, hard disk “a”, partition “2” (IDE, secondary channel, slave)
/dev/sda	devices, SCSI disk “a”
/dev/sdd9	devices, SCSI disk “d”, partition “9”

Extended partitions

The original MS DOS partition table was limited to four partitions[8]. To work around this limitation, the current scheme allows additional “logical” partitions – accomodated in a single “extended partition”. If there are more than 4 partitions, one of the first four “partitions” will be an extended partition.

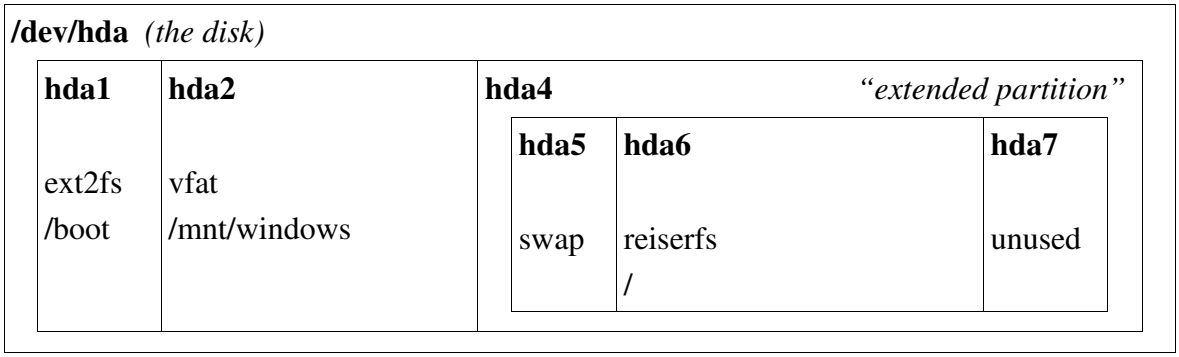


Illustration 3 Partitions and extended partitions

When you set up a new Linux system you will usually:

- Create an extended partition, to put all the actual partitions into.
- Create a new **/boot** partition, around 32Mb, and set its type to **83** (Linux).
- Create a new swap partition (your memory size times two), and set its type to **82** (Linux swap)
- Create a root partition and set its type to **83** (Linux). If you require a separate **/home**

8 Why would you ever need more than four partitions?

and **/var** partition, create these too.

Instead of starting with an extended partition, you may choose to create up to four primary partitions.

The Linux **fdisk** disk partitioning tool is used by some distributions during installation. Most distributions use either **cfdisk** (console based, visual layout) or their own graphical partitioning tool.

8.2 Design criteria

Other operating systems

For dual booting, it is generally best to install the other operating system before you install Linux. Doing this makes it difficult for the other operating system to delete your carefully installed Linux system by mistake. Also, Linux is able to handle a number of strange partitioning schemes used by other operating systems.

When you install an operating system, it is good to decide how much disk space you will require for your operations on that operating system. Once the file systems have been created, and filled with important data, it is difficult or impossible to resize them.

/boot within 1024 cylinders

Older BIOS machines struggle to access data beyond cylinder 1024. If the Linux kernel and the boot loader are stored in a partition above the 1024 cylinder point, it is possible that the BIOS may not be able to load the operating system at all. To avoid this problem, create a boot partition, around 16Mb to 32Mb large which is entirely below the 1024 cylinder point.

The filesystem for the boot partition is usually set to ext2fs. LILO can have trouble with reiserfs partitions if the *notail* option is not provided when the filesystem is created.

/var, /home and / partitions

It is possible to install an entire Linux installation in one partition. This partition is mounted at the mount point "/". It can be useful to create a partition for **/var** to limit the size to which variable data can grow (particularly spool files). If the disk becomes full, the system can become non-functional, so limiting the impact to the **/var** directory is beneficial. A fair size for **/var** is 200Mb to 500Mb, depending on the application(s) that will be used. For similar reasons **/home** is often given its own dedicated partition.

Swap space

The rule of thumb[9] for swap space is that you should configure two times as much swap space as you have RAM. Once you have filled up all your computer's memory twice, things aren't going to be working so well, and having some swap space available allows you to continue running without causing application failures.

9 Uncle Ed's rule of thumb says you should never use your thumb for a rule, because you'll either draw skew, or you'll hit it with a hammer.

8.2.1 Mounting partitions

When a partition is mounted, the files on its filesystem become part of the system. In Linux, partitions are mounted on *directories*, and the files in the partition becomes files in the directory. To mount a newly created partition, you would do this:

```
# mke2fs /dev/hda2            # create the filesystem on a partition
# mkdir /home                 # make a place to put it
# mount /dev/hda2 /home       # put it there
```

8.2.2 fstab

Once you have created partitions, you create filesystems in them using **mkfs**.

When your system boots, the filesystems listed in **/etc/fstab** are mounted automatically, and become part of the system. The contents of **/etc/fstab** show the file systems that are available on your computer.

```
/dev/hda7    /                    ext3    defaults        1 1
/dev/hda3    /home                ext3    defaults        1 2
/dev/hda6    /var                 ext3    defaults        1 1
/dev/hda2    swap                 swap    defaults        0 0
```

8.2.3 Swap space

Swap partitions are used for extra memory (virtual memory). Before you can use a partition as a swap partition, you need to write a swap signature to it, using the **mkswap** command. This procedure stops you from using that valuable letter to your mom as a swap file by mistake. The command **mkswap /dev/hda7** will create a swap file all over the partition.

If you don't have **/dev/hda7** available for destruction, you can get a feel for swap space using a file instead of a real partition.

```
# swapon -s
$ free
$ dd if=/dev/zero of=testswap bs=1k count=2000
$ /sbin/mkswap testswap
# swapon testswap
# swapon -s
$ free
# swapoff testswap
$ free
```

8.2.4 LVM overview*

LVM (Logical volume management) enables disk volume management by grouping arbitrary physical disks into virtual disk volumes. LVM provides on-line addition and removal of physical devices and dynamic disk volume re-sizing. When using LVM, instead of mounting partitions, you can mount logical volumes.

The logical volume manager allows management of storage volumes in user-defined groups, allowing the system administrator to deal with sensibly named volume groups such as "development" and "sales" rather than physical disk names such as "**sda**" and "**sdb**".

To prepare for using LVM devices, the commands are these:

```
# /sbin/vgscan
# /sbin/vgchange -ay
```

After this, the **/dev/vg** devices are available for formatting and using as filesystems.

8.3 Review

Quiz questions

1. Which **/dev/** file represents the secondary master IDE disk.
2. Name 2 advantages of using a swap partition over a swap file.
3. When is it necessary to have a separate **/boot** partition, and how should it be installed?

Assignment

Design a partitioning layout for a mail server which acts as a file server as well. One of the concerns of the system administrator is that too much incoming mail must not cause the file server system to stop functioning, and that filling up the file server should not cause the mail server to stop functioning. The disk to be used is 12Gb in size and has 1467 cylinders.

Answers to quiz questions

1. /dev/hdc
2. It is faster, and it can be used without a filesystem being mounted beforehand.
3. When you have an old or buggy BIOS. It should end before cylinder 1024.

9 Boot managers

> Marvelous! The super-user's going to boot me! What a finely tuned response to the situation!
>
> *fortune database*

In order to run Linux, it must be loaded into memory. The program that loads Linux into memory is called a boot manager.

Read up on editing files with **vi** (maybe run **vimtutor**) in preparation for this chapter.

LPIC topic 1.102.2 — Install a boot manager [1]

Weight: 1

Objective:

Candidate should be able to select, install, and configure a boot manager. This objective includes providing alternative boot locations and backup boot options (for example, using a boot floppy).

Key files, terms, and utilities include:

/etc/lilo.conf	LILO configuration file (read by **/sbin/lilo**)
/boot/grub/grub.conf	GRUB configuration file (read at boot time)
grub-install	GRUB installer
MBR	Master boot record – where you install a boot loader
superblock	What your filesystems need to know where their stuff is
first stage boot loader	The part of GRUB or LILO that goes on the MBR

9.1 Booting and boot managers

When the BIOS has finished starting up, it loads the first sector of a boot disk into memory, and runs the code it finds there. On a floppy disk, the boot sector is the first sector, and that's the whole story.

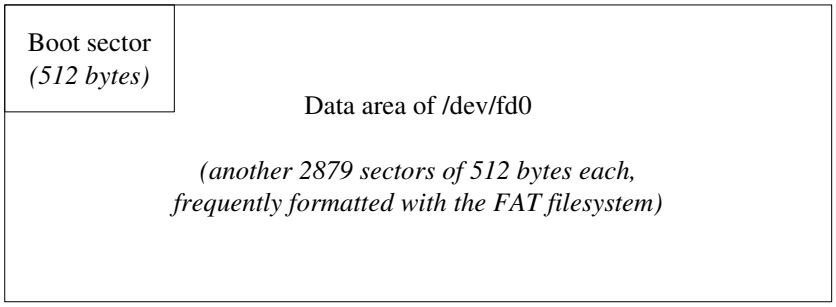


Illustration 4: Floppy disk boot sector ("logical" layout)

For hard disks, the first sector of the disk (e.g. **/dev/hda**) is the partition sector. This usually contains a program that loads the boot sector from the "active" partition. This first sector is known as the MBR (Master boot record).

Microsoft operating systems (**FDISK /MBR** in MS-DOS) install a partition sector loader which does the following:

- Check which partition is marked as “active”.
- Load the sector from the start of that partition and run it.

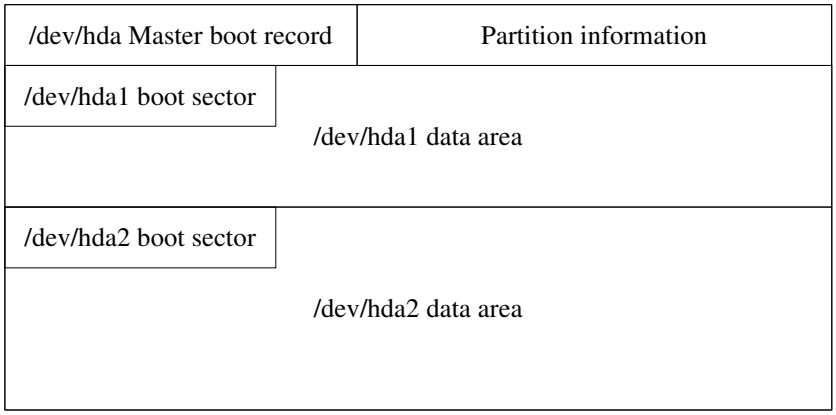


Illustration 5: Each partition on a hard disk has its own boot sector.

If you install a Linux boot loader, you will usually install it on **/dev/hda**. If you install it on **/dev/hda1** (the first sector of the first partition), then you require a partition sector loader that will load the code on your partition's boot sector.

Linux boot loaders have the following characteristics in common:

- The code stored on the boot sector or the partition sector is just the *first stageloader*. This means that it is not responsible for loading the operating system, but only for loading the boot loader into memory.
- The boot loader can be configured to load other operating systems. When loading other operating systems, the boot loader may load them into memory, or call that operating system's own boot loader.
- You can interactively enter *options* to be passed to the kernel to be loaded. Kernel options are used, among other things, to indicate where the root filesystem is (e.g. **root=/dev/hda3**), and can be used for debugging (e.g. **init=/bin/sh**).

9.2 LILO

LILO is the (almost) original **LI**nux **LO**ader. It does not depend on you using a specific filesystem, it can boot Linux kernel images from floppy disks, and it can act as a bootmanager for other operating systems.

LILO works something like this:

- You create the configuration file **/etc/lilo.conf**
- You run **/sbin/lilo**
- **/sbin/lilo** maps out the sectors on the disk that contain the data needed for booting (the kernel, the configuration options)
- **/sbin/lilo** maps out the locations of the disk that contain the data needed for booting (the kernel, the configuration options). This map is stored in **/boot/map**.

- **/sbin/lilo** installs the boot loader on the disk, and configures it with the location of **/boot/map**.

One of the important consequences of the way **LILO** works is that the location of data on the disk *may not* change after **/boot/map** has been created. This means that if you modify the configuration in **/etc/lilo.conf**, install a kernel upgrade, or rename files and directories, you need to run **/sbin/lilo** again before things work as expected.

9.2.1 /etc/lilo.conf

lilo.conf determines the configuration of the boot loader installed by **/sbin/lilo**. Here's an example of the contents of **/etc/lilo.conf**:

```
# cat /etc/lilo.conf
prompt
timeout=50
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
linear

image=/boot/vmlinuz-2.4.18-14
        label=linux
        initrd=/boot/initrd-2.4.18-14.img
        read-only
        append="root=LABEL=/"

other=/dev/hda1
        optional
        label=Windows
```

The first part of the file contains items which apply to the entire LILO configuration:

- **prompt** – this means that LILO presents the boot up menu, even if nobody presses the Shift key during booting up.
- **timeout=50** – 50 means 50 tenths of a second[10], or 5 seconds. The user has 5 seconds to choose a menu item, or to change the kernel command line.
- **default=linux** – if there is no response for the 5 seconds, the kernel image defined below with the parameter "**label=linux**" is loaded into memory.
- **boot=/dev/hda** – the first stage boot loader is installed here (as the Master boot record on the first IDE disk). If this is replaced with **/dev/fd0** then **/sbin/lilo** installs the first stage loader on the boot sector of the floppy disk. This option can be overridden by running **/sbin/lilo** as **lilo -b /dev/fd0**.
- **linear** – This causes LILO to use linear sector addresses instead of sector/head/cylinder addresses to pinpoint the location of the data files. While this is slightly more reliable with older hardware, the system may not boot if you install the disk in a different computer (with

10 Surprisingly, the decisecond is not an SI unit

different BIOS behaviour). For large disks in combination with an older BIOS, **/sbin/lilo** may generate references to parts of the disk that are inaccessible when the system is booted.

These global parameters aren't quite as exciting as the others – they could have been left out, in fact, as they are unchanged from their default settings:

- **message=/boot/message** – the contents of this file are displayed when the system boots. In more recent versions of LILO, this file contains the instructions for the graphical boot menus used by many distributions.
- **map=/boot/map** – this is the file in which the locations of the kernel, and LILO itself are stored. Generally one does not change this.
- **install=/boot/boot.b** – This file contains the data which **/sbin/lilo** installs in the boot sector.

This is the part of the file that specifies the loading of a Linux kernel:

```
image=/boot/vmlinuz-2.4.18-14
        label=linux
        initrd=/boot/initrd-2.4.18-14.img
        read-only
        append="root=LABEL=/"
```

- **image=/boot/vmlinuz-2.4.18-14** – The file **/boot/vmlinuz-2.4.18-14** contains a kernel image to be loaded into memory by LILO.
- **label=linux** – this is what you can choose during booting to load the kernel image from **/boot/vmlinuz-2.4.18-14**
- **initrd=/boot/initrd-2.4.18-14.img** – the **init**ial **r**oot **d**isk. Most distributions now distribute a standard kernel without support for vital file systems and hardware (e.g. SCSI controllers). When these systems are booted, they *initially* run using a memory disk (file system) which is loaded from the file specified. After the necessary kernel modules are loaded, booting continues as normal.
- **read-only** – This option is added to the kernel command line, and instructs the system to start up with the root file system mounted as read-only. The system initialisation scripts will check the file system while it is mounted read-only before continuing with initialisation.
- **append="root=LABEL=/"** - The text "**root=LABEL=/**" is added to the kernel command line. This tells the kernel to use the partition *labelled* as "LABEL=/" as the root partition. The regular way of doing this is to specify "**root=/dev/hda3**" in **lilo.conf**. You can append a number of parameters for the kernel command line, such as "**append="apm=off floppy=nodma init=/bin/bash"**". You can see the parameters with which your kernel was booted with **cat /proc/cmdline**.

9.2.2 lilo command options

/sbin/lilo installs the boot loader according to the configuration you specified.

Install LILO

To install LILO, you edit **/etc/lilo.conf** and run **/sbin/lilo.**

Uninstall LILO

Occasionally, you will want to return to the state of your system as it was before LILO wrecked it. Happily, LILO keeps a copy of the original sector of each device you install it on (except perhaps for your floppy disks). If you ask LILO to uninstall itself, it simply copies this sector back to the device, and the system returns to it's original state.

```
foo:~ # lilo -u
```

You can also uninstall LILO by reinstalling the boot loader of another operating system. To reinstall the default boot loader for DOS / Windows 95..ME, the command is

```
C:\> FDISK /MBR
```

LILO boot floppy

If you install LILO on a floppy disk, that floppy disk is able to boot only the computer from which it was installed. Installing on a floppy makes it possible for you to boot Linux without interferring with the operation of other operating systems:

```
foo:~ # lilo -b /dev/fd0
```

The **-b** command line option overrides the value specified for **boot=** in **/etc/lilo.conf**.

9.2.3 LILO dual boot

Linux and other operating systems can be installed in separate partitions on a single PC. When booting up, the boot manager decides which operating system to load into memory. The bootloader that makes the decision can be LILO.

In the **lilo.conf** given above, the following lines configure LILO to boot another operating system:

```
other=/dev/hda1
        optional
        label=Windows
```

Most operating systems install their own boot loader at the start of their data partition. Once this code gets running, the system will start up as designed. All LILO needs to know is which operating systems can be booted in this way.

If configuring LILO seems hard to you, you can configure **boot.ini** on a Windows NT system to load the Linux loader from its partition. If you are using Windows NT, you probably know how to do this. You must just be sure to install LILO in a Linux partition, rather than all over the master boot record.

9.3 GRUB

GRUB is the GNU **Gr**and **U**nified **B**oot loader. GRUB provides a number of functions which PC BIOS programs do not:

- Provides fully-featured command line and graphical interfaces
- Understanding partitions (both DOS and BSD style)
- Understanding most file systems
- Booting of Linux, BSD and Multiboot-compliant kernels (such as GNU Mach)

GRUB understands enough about partitions and filesystems to find its configuration file **grub.conf** without using a map file. GRUB can also boot systems without a configuration file, if the operator is sufficiently skilled.

9.3.1 /boot/grub/grub.conf

This is the configuration file for GRUB.

```
default=0
timeout=10
splashimage=(hd0,6)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.4.18-19.8.0) apm-off floppy-nodma
        root (hd0,6)
        kernel /boot/vmlinuz-2.4.18-19.8.0 ro root=LABEL=/ apm=off
      floppy=nodma
        initrd /boot/initrd-2.4.18-19.8.0.img
```

9.3.2 Creating a boot floppy

How about this?

```
foobar ~ # dd if=/boot/vmlinuz of=/dev/fd0
foobar ~ # rdev /dev/fd0 /dev/hda3
```

That will copy the kernel image to the floppy, and set the root device encoded in the kernel to **/dev/hda3**. The floppy disk is a bootable kernel image. The only problem is that many Linux distributions require a initial root disk (usually **/boot/initrd**), so this method is not universally successful.

Quiz questions

1. What must be done after editing **/etc/lilo.conf** to make changes take effect?
2. What is the difference between LILO and GRUB?
3. How would you go about booting up a Linux installation where no boot loader has been installed?

Assignment

1. Change your boot loader from LILO to GRUB. Install the boot loader on your first hard disk can create an appropriate configuration file. You should have a rescue system handy when you do this.
2. Install LILO as your first boot loader (e.g. on **/dev/hda**), and configure it to load GRUB (e.g. from **/dev/hda3**), which in turn loads Linux. This will give you experience in configuring both GRUB and LILO.

Answers to quiz questions

1. Run **lilo**
2. LILO records absolute sector addresses for the files it loads, while GRUB understand how to read various filesystems.

3. a) Provide a floppy disk with a kernel image, and supply the correct root device using **rdev**. b) Boot off a floppy containing GRUB, and enter the correct parameters to locate the kernel and root device. c) Boot off a distribution or rescue CDROM, but specify a specific root device on the kernel command line. d) Boot a rescue system, mount the installed system (e.g. **mount /dev/hda4 /mnt**), and change root to the mounted system (**chroot /mnt**) and then install a boot loader on a floppy disk or on the hard disk.

10 Installing from source code

> Life would be so much easier if we could just look at the source code.
>
> *– Dave Olson*

Source code is human readable instructions for what your computer should do with its time. When your compiler has chewed up your source code, it spits out a binary executable, which is probably what you want if you were intending to run the program.

You should read up on the following if you are not familiar with them:

- downloading files with **wget**, **ftp** or a browser
- **tar** and **gzip** for extracting source code from downloaded archives
- editing files with **vi**

LPIC topic 1.102.3 — Make and install programs from source [5]

Weight: 5

Objective:

Candidates should be able to build and install an executable program from source. This objective includes being able to unpack a file of sources. Candidates should be able to make simple customizations to the Makefile, for example changing paths or adding extra include directories.

Key files, terms, and utilities include:

gunzip	Uncompressing a file using the gzip algorithm
gzip	Compressing a file using the gzip algorithm
bzip2	Use the bzip2 algorithm to do compression
tar	Program for tape archives, and source archives
configure	What you run before make
make	What you run after configure

10.1 Unpacking source distributions

Source code distributions of packages are generally distributed in **.tar.gz** format (compressed tape archive). The tool to extract the original files is **tar**. The exact usage of the command depends on the format of the archive you are dealing with.

You can first uncompress the file then extract it with **tar**:

```
user@bar $ gunzip gnomovision-0.31.tar.gz
user@bar $ tar -xf gnomovision-0.31.tar
```

Alternatively, you can do this in one step with the **z** switch:

```
user@bar $ tar -zvxf gnomovision-0.31.tar.gz
```

If the file is bzip2 compressed, you can use **bunzip2** or the **-j** switch to **tar**.

```
user@bar $ gunzip gnomovision-0.31.tar.gz
user@bar $ tar -xf gnomovision-0.31.tar
user@bar $ tar -jvxf gnomovision-0.31.tar.bz2
```

10.2 Compiling programs

Occasionally the program you download will be a simple C program, consisting of a single executable file. In this case, you can compile it quite simply using **cc**. The **-o** switch to **cc** specifies the output file (instead of the default file name **a.out**).

```
foo:~ $ ls -la rootkit.c
-rw-rw-r--    1 jack     jack         3736 Mar 24 15:42 rootkit.c
foo:~ $ cc -o rootkit rootkit.c
foo:~ $ ls -la rootkit*
-rwxrwxr-x    1 jack     jack        15707 Mar 27 15:44 rootkit
-rw-rw-r--    1 jack     jack         3736 Mar 24 15:42 rootkit.c
```

If you can't get used to the idea of using **cc** directly, you can get **make** to do it for you:

```
foo:~ $ make rootkit
cc     rootkit.c   -o rootkit
```

make does a lot more than this, actually. If there is a Makefile present, it can figure out which files you edited, and only recompile those files.

10.3 Simple build and installation

Most source code is distributed with at least some documentation, and often with too much documentation. Usually a file named **INSTALL** or **README** is included in the tarball, which describes how to compile the program and the required software and hardware.

Often a program will be distributed with a **Makefile**. This file contains instructions on how to compile the program. **make** relies on these instructions and, in turn, runs the compiler to convert the source code to binary code.

GNU programs are usually distributed with a script named **configure**. The task of this script is to determine the correct contents for and to create the **Makefile** so that the software can be built.

One of the problems during building software is that often the required libraries and header files are missing from your system. These need to be installed. On rpm based systems, the **-devel** package usually contains the appropriate libraries. You can, of course, download and install the required libraries too.

If you download a package from the internet, you may find that the following commands produce an installed version of it for you.

```
fred@bar ~ $ tar -zvxf wget-1.5.3.tar.gz
fred@bar ~ $ cd wget-1.5.3
fred@bar ~/wget-1.5.3 $ less README
fred@bar ~/wget-1.5.3 $ ./configure
fred@bar ~/wget-1.5.3 $ make
fred@bar ~/wget-1.5.3 $ su
root@bar /home/fred/wget-1.5.3 # make install
```

10.4 ./configure options

The behaviour of GNU **autoconf** configure scripts can be customized by passing command

line options. Here are some highlights from the help of your average configure script:

```
$ ./configure --help
`configure' configures this package to adapt to many kinds of
      systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them
      as
VAR=VALUE.  See below for descriptions of some of the useful
      variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit

Installation directories:
  --prefix=PREFIX         install architecture-independent files in
      PREFIX
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in
      EPREFIX
                          [PREFIX]

By default, `make install' will install all the files in
`/usr/local/bin', `/usr/local/lib' etc.  You can specify
an installation prefix other than `/usr/local' using `--prefix',
for instance `--prefix=$HOME'.

For better control, use the options below.

Fine tuning of the installation directories:
  --bindir=DIR           user executables [EPREFIX/bin]
  --sbindir=DIR          system admin executables [EPREFIX/sbin]
  --includedir=DIR       C header files [PREFIX/include]

Optional Features:
  --disable-FEATURE       do not include FEATURE (same as --enable-
      FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --enable-debug    Turn on debugging

Some influential environment variables:
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries
      in a
              nonstandard directory <lib dir>
  CPPFLAGS    C/C++ preprocessor flags, e.g. -I<include dir> if you
      have
              headers in a nonstandard directory <include dir>

Use these variables to override the choices made by `configure' or
      to
```

```
help it to find libraries and programs with nonstandard
      names/locations.
```

The default installation prefix is **/usr/local**, so the programs end up in **/usr/local/bin**, the man pages in **/usr/local/man**, etc. You may prefer to install with a different prefix.

```
$ ./configure --prefix=/
```

If you have previously run configure, you often have to remove the cached results to get it to work correctly. (Note that the output of configure is a few pages longer than shown here).

```
$ rm config.status
$ ./configure --prefix=/
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets ${MAKE}... yes
checking for gcc... gcc
checking for stdio.h... yes
checking for off_t... yes
checking for crypt in -lcrypt... yes
```

10.5 Editing Makefiles

Source code packages for other Unix systems often work with Linux. However, occasionally there is a change in a directory name, or a file name which causes a compilation failure. The file to edit is Makefile, and you should change the offending entry.

If you are busy compiling a jetset willy clone, you may see the following text fly by:

```
$ make
gcc -O2 -march=i386 -mcpu=i686 -pipe      -I/usr/X11R6/include
      -Dlinux -D__i386__ -D_POSIX_C_SOURCE=199309L -D_POSIX_SOURCE
      -D_XOPEN_SOURCE -D_BSD_SOURCE -D_SVID_SOURCE     -DFUNCPROTO=15
      -DNARROWPROTO   -DSCOREFILE=
      \"/usr/games/lib/jetset/jetset.score\" -DLOCKFILE=
      \"/usr/games/lib/jetset/jetset.score.lock\" -DSAVEDIR=
      \"/usr/games/lib/jetset\"     -c -o data.o data.c
make: *** Deleting file `data.o'
make: *** [data.o] Interrupt
```

The file **/usr/games/lib/jetset/jetset.score.lock** is not the ideal value, and we would prefer to use **/usr/share/games/jetset**. The file to edit is *actually* the **Imakefile**, but we can do as well by editing **Makefile**, and changing this ...

```
SAVEDIR = /usr/games/lib/jetset
SCOREFILE = ${SAVEDIR}/jetset.score
LOCKFILE = ${SAVEDIR}/jetset.score.lock
```

... to this ...

```
SAVEDIR = /usr/share/games/jetset
SCOREFILE = ${SAVEDIR}/jetset.score
LOCKFILE = ${SAVEDIR}/jetset.score.lock
```

And while we are at it, we'll install in **/usr/local/bin** rather than in **/usr/games** (which doesn't exist on my system). So this ...

```
DESTDIR = /usr/games/
BINDIR = bin
```

```
MANPATH = man
MANSUFFIX = 6
```

... becomes ...

```
DESTDIR = /usr/local/
BINDIR = bin
MANPATH = man
MANSUFFIX = 6
```

After these changes the program compiles with the new settings. A side note – for this particular program, it was necessary to run **xmkmf** before running **make**, which was explained in the **INSTALL** file (quite similar to **configure** for programs using **autoconf**).

10.6 Review

Quiz questions

1. What is the sequence of commands for unpacking and installing a GNU program from source?
2. Why must **make install** be run as root? Are there exceptions to this?
3. Why should **configure** and **make** generally not be run as root?
4. What options to **configure** are used to change the installation directory?
5. How does one add an include directory to a Makefile?
6. Which file is generated by **configure**?

Assignment

Find, download a source package of your own choice, and install it on your system. If you cannot find any other download and install the game **rocksndiamonds**. You can install any libraries required by the application from the **-devel** packages that ship with your distribution. Write instructions on how to install the application from source on your system, including a list of library software you installed.

Answers to quiz questions

1. **tar -zxf file.tar.gz ; ./configure ; make ; su -c "make install"**
2. To be able to access the target directories, such as **/usr/local/bin**. Being root is not necessary if you have permissions to the destination directory (e.g. **./configure –prefix=$HOME**)
3. Errors and sabotage in the configure script and Makefile can destroy your installation.
4. **--prefix** and **--exec-prefix** (not to mention **--bindir**, **--sbindir**, **--libexecdir**, **--datadir**, **--sysconfdir**, **--sharedstatedir**, **--localstatedir**, **--libdir**, **--includedir**, **--oldincludedir**, **--infodir**, and **--mandir**)
5. Find the variable listing **-Idirectory** in the Makefile and add **-Iotherdirectory** values to it.
6. Usually **Makefile**, and often enough **config.h** (just for extra flavour).

11 Shared libraries

Weiner's Law of Libraries:
There are no answers, only cross references.

– fortune cookie database

LPIC topic 1.102.4 — Manage shared libraries [3]

Weight: 3

Objective:

Candidates should be able to determine the shared libraries that executable programs depend on and install them when necessary. Candidates should be able to state where system libraries are kept.

Key files, terms, and utilities include:

ldd	Show shared libraries used by a program
ldconfig	Regenerate ld.so.cache
/etc/ld.so.conf	Extra library directories
LD_LIBRARY_PATH	Environment variable for libraries

11.1 Purpose and structure of shared libraries

In Linux, all the really useful functions are done by the kernel. If you choose to write a program which calls the kernel directly, that's fine and well. Most programmers choose to use the kernel in pretty much the same ways, and these methods are stored in *libraries*. The memory used by these libraries are *shared* between more than one program, so they are called *shared libraries*.

When programs are loaded into memory, part of the loading process is to make sure that the relevant *shared libraries*[11] are loaded when the program loads. If the correct shared libraries are not available, the program is incomplete, and cannot be run. A library can also require other libraries.

Dependencies between libraries are normal, especially for large applications. A program may use a graphics library, which in turn uses the X11 library, which in turn uses the C library.

11.2 Using ldd

ldd prints a list of the shared libraries required by a given program or library.

```
foo:~ $ ldd /bin/bash
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40024000)
        libdl.so.2 => /lib/libdl.so.2 (0x40028000)
        libc.so.6 => /lib/libc.so.6 (0x4002b000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

The left hand column indicates what the program would like to be linked to (usually

11 In Microsoft Windows shared libraries are “dynamic link libraries” (or DLLs), and named WHATEVER.DLL.

no path name given), and the right hand column shows what the program will actually be linked to (full path) (often a symbolic link...)

```
foo:~ $ ls -lg /lib/libc.so.6
lrwxrwxrwx    1 root    14 Oct 14 10:47 /lib/libc.so.6 -> libc-
      2.2.93.so
foo:~ $ ls -lg /lib/libc-2.2.93.so
-rwxr-xr-x    1 root  1327065 Sep  6  2002 /lib/libc-2.2.93.so
```

So, **bash** asked to link to "**libc.so.6**", and it ended up linking to "**/lib/libc-2.2.93.so**".

11.3 Symbol versions

While it is very nice to check that you are loading the correct library, things in the Linux world change rather quickly, and it is quite convenient if more than one interface for a particular function is available. This means that older and newer code can continue to work, even if there are changes.

A side effect of this is that the dynamic loader can recognise libraries which are too old and don't contain all the symbols or contain incompatible implementations.

11.4 Configuring the dynamic linker

The configuration file for the dynamic linker is **/etc/ld.so.conf**. This contains a list of directories, in addition to **/lib** and **/usr/lib** which contain library files. **/sbin/ldconfig** updates **/etc/ld.so.cache**.

Applications can be made to load libraries from directories not listed in **/etc/ld.so.conf** by setting the value **LD_LIBRARY_PATH**. The variable **LD_PRELOAD** specifies libraries containing symbols that override symbols in the real libraries. **LD_PRELOAD** is seldom used except for debugging and compatibility applications (e.g. SOCKS proxy support).

11.5 Review

Quiz questions

1. Why do shared libraries include version numbers?
2. Why do programs use shared libraries, rather than including the library code in the executable file?
3. What is the purpose of **LD_PRELOAD**?
4. When is it necessary to run **ldconfig**?
5. What is **LD_LIBRARY_PATH**, and what does it configure?
6. Where are system libraries found on a Linux system, and what is the purpose of each location?

Assignment

1. Find where the **glibc** library files are store on your computer.
2. Install either Mozilla or OpenOffice.org. Find the dynamic libraries that are loaded as part

of the application and determine whether these are in the regular library path. Which environment variables are set during the startup of the application, and what is the actual executable file for the application?

3. Download and install the binary of a program dynamically linked against **libc5** (or a version of **glibc** that you do not have) and make it work. (You will have to download the libraries it links against, and make them work.)

Answers to quiz questions

1. This makes it possible to use new programs together with programs which use older libraries.
2. Including the code from shared libraries would make them bulky, and the system works better with a single copy of library code being shared between programs.
3. This variable can be set to the name of a library which overrides functions from the standard libraries.
4. It's the search path for the system libraries. It determines which directories are searched when a dynamic executable is loaded.
5. **LD_LIBRARY_PATH** is an environment variable. It determines the directories in which the dynamic loader will search for libraries.
6. **/lib** contains libraries required by the system commands in **/bin**. **/usr/lib** contains most system libraries. A complete list of additional directories is in **/etc/ld.so.conf**, largely for X and its applications.

12 Debian package management

> Software suppliers are trying to make their software packages more “user-friendly”. ... Their best approach, so far, has been to take all the old brochures, and stamp the words, "user-friendly" on the cover.
>
> *– Bill Gates, Microsoft, Inc.*

Debian (named after Debbie and Ian) is a free Linux distribution. Its package management includes the aspect of caching packages which are not installed. The cache can be queried, and dependencies can be automatically determined when installing packages. To really understand this chapter, you should install Debian – in fact, that's one of the assignments.

LPIC topic 1.102.5 — Use Debian package management [8]

Weight: 8

Objective:

Candidates should be able to perform package management skills using the Debian package manager. This objective includes being able to use command-line and interactive tools to install, upgrade, or uninstall packages, as well as find packages containing specific files or software (such packages might or might not be installed). This objective also includes being able to obtain package information like version, content, dependencies, package integrity and installation status (whether or not the package is installed).

Key files, terms, and utilities include:

unpack	**dpkg** first unpacks the files from a package
configure	What **dpkg** does after unpacking the packages
/etc/dpkg/dpkg.cfg	Configuration for **dpkg**
/var/lib/dpkg/*	**dpkg** database of installed packages
/etc/apt/apt.conf	Configuration file for **apt**
/etc/apt/sources.list	Source location for **apt**
dpkg	Debian package installer
dselect	Debian package selection tool
dpkg-reconfigure	Debian package configuration
apt-get	Debian download tool
alien	Debian package import / export

12.1 Debian and .deb

The Debian distribution uses its own form of binary packaging to distribute software. The convention for Debian package file names is

`program_version_architecture.deb`

Here's how it looks in practice. The particular FTP server in the example lists Debian Linux

i386 and Debian hurd-i386 packages.

```
ftp> cd rxvt
250 OK. Current directory is /debian/pool/main/r/rxvt
ftp> ls
227 Entering Passive Mode (196,4,160,12,175,197)
150 Accepted data connection
-rw-r--r--    1      253200 Jun 16  2001 rxvt-ml_2.6.2-2.1_i386.deb
-rw-r--r--    1      282262 May  1  2002 rxvt-ml_2.6.4-6_hurd-
      i386.deb
-rw-r--r--    1      256320 Apr  9  2002 rxvt-ml_2.6.4-6_i386.deb
-rw-r--r--    1       19500 Jun 16  2001 rxvt_2.6.2-2.1.diff.gz
-rw-r--r--    1        1262 Jun 16  2001 rxvt_2.6.2-2.1.dsc
-rw-r--r--    1      216576 Jun 16  2001 rxvt_2.6.2-2.1_i386.deb
-rw-r--r--    1       26137 Apr  9  2002 rxvt_2.6.4-6.diff.gz
-rw-r--r--    1         591 Apr  9  2002 rxvt_2.6.4-6.dsc
-rw-r--r--    1      234662 May  1  2002 rxvt_2.6.4-6_hurd-i386.deb
-rw-r--r--    1      223030 Apr  9  2002 rxvt_2.6.4-6_i386.deb
```

Debian packages contain the following:

- A description of the package (including the list of files)
- Packages or capabilities that this package requires
- The binary files to be installed
- Scripts which should be run before and after installation (and package removal).

The following **dpkg** operations are the most commonly used for installing and querying installed packages:

- **dpkg -i package.deb** – install the package file (like **rpm -i package.rpm**).
- **dpkg -r package** – remove an installed package (like **rpm -e package**).
- **dpkg -L package** – list the files installed with a package (like **rpm -ql package**).
- **dpkg -S file** – search the list of installed files for the pattern given (similar to **rpm -qf file**, but more like **rpm -qla | grep file**).

These **dpkg** options allow you finer control over when the package you install is configured.

- **dpkg --unpack package.deb** – install the package, but don't configure it. The package can be configured later with **dpkg-reconfigure**.
- **dpkg --configure package** – configure an installed package.

Reconfiguring packages

After a package has been installed it is usually configured automatically by **apt** (see further on). If the configuration is incorrect, **dpkg-reconfigure** reconfigures an installed package. **dpkg-reconfigure** is most often used for reconfiguring the X server, or the mail server.

12.2 apt

apt maintains the cache of packages that can be installed.

- **apt-cache search searchstring** – Show information about packages containing the name given

- **apt-get update** – Connect to the apt sources listed in **/etc/apt/sources.list** and determine the current versions of cached packages.
- **apt-get install package** – install a package listed in the cache. The dependencies for the package are also installed.
- **apt-get upgrade** – upgrade all out of date packages to the current versions as known from the cache.

For complete information on **apt**, read to the man pages for **apt-get** and **apt-cache**.

12.3 Review

Quiz questions

1. What makes Debian package management unique?
2. How do you use **dpkg** to do the following functions:

- install a package
- obtain package information for an installed package
- obtain package information for a package file
- remove a package
- find out which package owns a given file
- find out whether a package is installed

3. How do you use **apt-cache** to:

- find a package that is not installed?
- find the full description of a package based on its contents?

4. How do you use **apt-get** to

- install a package?
- install a package together with its dependencies?

Assignment

1. Obtain and install Debian.
2. Configure or reconfigure your X server using **dpkg-reconfigure**. You will need to find the package name of the X server using **apt-cache** first.
3. Find a great game using **apt-cache**, and install it using **apt-get**. Heroes is quite fun.
4. Use **apt-get** to check whether there are any upgrades available for your installed distribution.

Answers to quiz questions

1. It is possible to install a package together with its dependencies.
2. dpkg ...

- dpkg -i package.deb (install)
- An exercise to the reader **:**)
- dpkg -I package.deb (information)

- dpkg -r package (remove)
- dpkg -S `pwd`/filename (search)
- dpkg -s package (status)

3. apt-cache search string ; apt-cache show package
4. apt-get install package ; apt-get install package

13 RPM – Redhat package manager

> Of course it doesn't work. We've performed a software upgrade.
>
> *– BOFH excuses from the fortune cookie database*

For this chapter, it is helpful to have a RPM based Linux distribution. Examples of these are RedHat (surprisingly), Mandrake and SuSE.

LPIC topic 1.102.6 — Use Red Hat Package Manager (RPM) [8]

Weight:8

Objective

Candidates should be able to perform package management under Linux distributions that use RPMs for package distribution. This objective includes being able to install, re-install, upgrade, and remove packages, as well as obtain status and version information on packages. This objective also includes obtaining package information such as version, status, dependencies, integrity, and signatures. Candidates should be able to determine what files a package provides, as well as find which package a specific file comes from.

Key files, terms, and utilities include

/etc/rpmrc	**rpm** configuration file
/usr/lib/rpm/*	**rpm** database of installed packages
rpm	**rpm** command line tool
grep	**grep** is useful for searching output of **rpm -qa**

13.1 Purpose of RPM

Not everyone enjoys compiling source code, or has the inclination to trawl the net to find out what the latest version of each package in existence is. For people like this, there are binary packages such as used by the Redhat Package Manager (RPM), containing only the executable files, customised[12] for the distribution you are running. Distributions using RPM packaging include RedHat, SuSE and Mandrake.

When you install a RPM package, **rpm** does the following for you:

- Check that the necessary libraries and programs are installed on your system. If these are not, it will complain bitterly, and you won't get much further.
- Copy files from the package on your file system – the program binaries (which end up in "bin" directories or in **/opt**), configuration files (which typically end up in **/etc**) and some documentation (e.g. man pages)
- Run a post-installation script to make sure that the package is configured with some basic functionality (optionally).

12 RPM packages are frequently compatible between Linux distributions.

In addition to this, **rpm** keeps track of the packages that are installed, and what they contain.

13.2 RPM database

rpm keeps track of the files which are installed on your system as part of **rpm** packages. This database is stored in **/var/lib/rpm**. For each package, the following information is available:

- The name, version number and revision number of the package
- A long description of the package
- A list of the files installed by the package, together with their original timestamps and MD5 checksums.
- The dependencies required by this package, and provided by this package.
- Scripts to be run before and after installation and package removal.

13.3 RPM functions

Here's the brief rundown of rpm functions (excluding query):

Command	***Package***
Operations on rpm package files	
rpm -i package.rpm	Install a package
rpm -U package.rpm	Upgrade a package to a newer version
Operations on installed packages	
rpm -e package	Erase an installed package
rpm -V package	Verify what has changed since a package was installed

13.3.1 Querying the database and package files

The RPM query command can query either installed packages or (with the **-p** switch) packages which have not been installed. Here's the usage of the rpm query command:

rpm -q -[**lRi**] [**-p** *package.rpm* | **-f** */some/file* | *package-name*]

You tell **rpm** what you want to see:

- **-l** – list the files in the package
- **-R** – list the requirements
- **-i** – show information

There are three ways of specifying the package you want information on:

- **-p package.rpm** – a particular rpm file which you have available.
- **-f /some/file** – a file which belongs to a particular package
- **package-name** – an installed package (either the short name, or the name and the version)

Here are some specific ways to use the query function:

Query an installed package:

rpm -qi package	Query information of a package
rpm -ql package	Query file list of a package
rpm -q package	Query package name

Query an installed package:
Query a package file:
rpm -qip package.rpm Query information of a package file
rpm -qlp package.rpm Query file list of a package file
rpm -qp package.rpm Query package name for a package file
Query a file which is part of an installed rpm:
rpm -qif /some/file Query information of a package owning **/some/file**
rpm -qlf /some/file Query file list of a package owning **/some/file**
rpm -qf /some/file Query package name owning **/some/file**

Here's an example of what you see for **rpm -qi**

```
bar:/data/rpm # rpm -qip xsnow-1.42-8.i386.rpm
Name        : xsnow                 Relocations: (not relocateable)
Version     : 1.42                  Vendor: Red Hat, Inc.
Release     : 8                     Build Date: Mon 26 Aug 2002
      22:51:09
Install date: (not installed)       Build Host: daffy.perf.redhat.com
Group       : Amusements/Graphics   Source RPM: xsnow-1.42-8.src.rpm
Size        : 111829                License: MIT
Signature   : DSA/SHA1, Tue 03 Sep 2002 23:45:26 SAST, Key ID
      219180cddb42a60e
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL         : http://www.euronet.nl/~rja/Xsnow/
Summary     : An X Window System based dose of Christmas cheer.
Description :
The Xsnow toy provides a continual gentle snowfall, trees, and Santa
Claus flying his sleigh around the screen. Xsnow is only for the X
Window System, though; consoles just get coal.
```

This is what package requirements look like:

```
bar:~ $ rpm -q fping
fping-2.2b1-551
bar:~ $ rpm -qR fping
ld-linux.so.2
libc.so.6
libc.so.6(GLIBC_2.0)
libc.so.6(GLIBC_2.1)
rpmlib(PayloadIsBzip2) <= 3.0.5-1
```

13.3.2 Install and upgrading packages

When you install a package with **rpm**, it does the following:

- Checks dependencies (all the things that are required should be there)
- Checks for conflicts (is there anything in the way)
- Extracts the binary files (and the documentation and configuration files)
- Runs post-installation scripts

When installing a package here are some options you may use:

- **-i** – install
- **-v** – verbose
- **-h** – hashes indicating progress
- **--nodeps** – no dependencies (i.e. just try it, even though it will not work).

Here's how you install a package:

```
bar:/data/rpm # ls -la xsnow-1.42-8.i386.rpm
-rw-r--r--    1 root  root     36302 Nov 19 11:51 xsnow-1.42-
      8.i386.rpm
bar:/data/rpm # rpm -ivh xsnow-1.42-8.i386.rpm
Preparing...        ########################################
      [100%]
   1:xsnow          ########################################
      [100%]
```

rpm -U will install a package if it is not currently installed. If it is installed, it will be *upgraded*. **rpm -F** will upgrade a package *provided* it is already installed.

Here's an example of upgrading **sendmail** on a SuSE Linux system

```
loco:~ # rpm -q sendmail
sendmail-8.12.6-104
```

Before the downloaded update is installed, we check whether it was signed by SuSE's packaging key, and then install it with **rpm -F**.

```
loco:~ # rpm --checksig sendmail-8.12.6-109.i586.rpm
sendmail-8.12.6-109.i586.rpm: md5 gpg OK
loco:~ # rpm -Fvh sendmail-8.12.6-109.i586.rpm
sendmail
      ##################################################
Updating etc/sysconfig/sendmail...
Updating etc/sysconfig/mail...
```

13.3.3 Erase a package

When you are tired of a package, you can erase it with **rpm -**e. You can either specify the full package name with the version number, or just the package name.

```
bar:~ # rpm -q xsnow
xsnow-1.42-8
bar:~ # rpm -e xsnow-1.42-8           # rpm -e xsnow would also work
```

13.4 RPM integrity checking

The RPM database in **/var/lib/rpm** stores the checksums for each file that was installed by RPM. This allows one to verify what has changed since package installation. **rpm -V** prints out a letter for each file, depending on the changes.

```
bar:~ $ rpm -Vf /etc/motd
S.5....T c /etc/inputrc
S.5....T c /etc/printcap
S.5....T c /etc/profile
..?..... c /etc/securetty
```

rpm -V may print the following letters:

- S – file **S**ize differs
- M – **M**ode differs (includes permissions and file type)
- 5 – MD**5** sum differs
- D – **D**evice major/minor number mis-match
- L – read**L**ink(2) path mis-match
- U – **U**ser ownership differs
- G – Group ownership differs
- T – m**T**ime (modification time) differs
- ? – unable to check (e.g. permission denied)

13.5 Review

Quiz questions

1. Which command will list the files that will be installed from the package **bogus-10.1.rpm**?
2. Which commands will enable you to identify the author of **lilo** given the file **/sbin/lilo** was installed as part of an rpm package?
3. Which flag for **rpm -q** specifies that you wish to query an installed package by specifying a file installed by that package?
4. What is the configuration file for **rpm**?
5. What does the **rpm** database of installed packages contain?
6. How can **grep** be used together with **rpm**?
7. How does one check that an **rpm** file has not been tampered with since it was packaged?

Assignment

1. Find out which **rpm** package owns the file **/bin/cut**. Read the package information for this package. What other files are included in the package?
2. Verify all **rpm** packages that are installed on your system. Did you personally make all the changes that are listed?
3. Use **rpm** from the command line to install a game that is available on your distribution media. You should find a game for which a number of libraries are required. Find out what the dependencies for the package are, and install packages to satisfy the dependencies as well.
4. Download an updated RPM for a package you have installed from your distribution's FTP site. Check the signature and checksum of the package you downloaded, and upgrade the installed package to the downloaded version.

Answers to quiz questions

1. rpm -qlp bogus-10.1.rpm
2. rpm -qif /sbin/lilo
3. -f

4. /etc/rpmrc
5. Information about each package, and about each installed file.
6. rpm -qa | grep package ; rpm -ql package | grep bin ; and more
7. rpm --checksig package.rpm

14 Work on the command line

"Linux: the operating system with a CLUE...
Command Line User Environment"

– seen in a posting in comp.software.testing

LPIC topic 1.103.1 — Work on the command line [5]

Weight: 5

Objective

Candidate should be able to interact with shells and commands using the command line. This includes typing valid commands and command sequences, defining, referencing and exporting environment variables, using command history and editing facilities, invoking commands in the path and outside the path, using command substitution, applying commands recursively through a directory tree and using man to find out about commands.

Key files, terms, and utilities include

.	The current directory
bash	What you bash your commands into
echo	echo
env	Show environment variables
exec	Run and don't return
export	Add a variable to the export list
man	Manual pages
pwd	Print working directory
set	Show environment settings
unset	Clear an environment variable
~/.bash_history	The last *n* commands you typed
~/.profile	What runs when you login interactively

14.1 Command line overview

The default command line on most Linux systems is **/bin/bash** – the Bourne Again Shell (an enhanced version of the Bourne Shell **/bin/sh**). For **bash** executes programs on your system for each command that you enter on the command line.

Here are some rules to live by when typing on the command line:

- Press **Enter** a lot – If you type something, the computer will only act on it if you press Enter (the return key). Until you press **Enter** nothing happens.
- Press **Ctrl+C** – If you have a bad feeling about what the computer is doing in response to your last command, press **Ctrl+C**. This sends a *INT*errupt signal to the program that is currently running.
- Press **Tab** a lot – The bash shell used by most Linux distributions has a handy feature that will type the rest of commands and file names for you. Together with the help, you get a lot

of annoying beeps.

- Press **Ctrl+Z**, type "**bg**" – If you are running a process that takes a while, and you want to let it run, press **Ctrl+Z** to stop it, then type **bg** and press **Enter** to get it into the background.
- Press **Shift+PgUp** – You can view the last few screens of text that flew off the top of the display by pressing **Shift+PgUp** a few times (this is a terminal function, not a bash function, actually).

14.2 Command line structure

The prompt and working directory

The prompt in **bash** is programmable. The usual values tell you the name of the machine you are using (sometimes your user name too) and which directory you are in (either the last part of the directory name, or the whole path name). Here are a couple of examples of prompts which you will see in Linux systems:

```
linux:~ #
abe:/var/log #
foobar:~ $
mail:/var/lib #
[root@yabba root]#
```

If the prompt ends with **#** then you are logged in as root. If the prompt ends with **$** then you are logged in as a regular user.

An important concept on the command line is the working directory. The working directory can be changed using **cd** (change directory) and displayed using **pwd** (print working directory). On most systems, the working directory is displayed as part of the prompt.

File names, by default, are *relative* to the working directory. Commands like **ls** and **du** display information for the working directory.

```
foo:~ $ pwd
/home/joe
foo:~ $ cd /usr/share/doc
foo:/usr/share/doc $ pwd
/usr/share/doc
foo:/usr/share/doc $ cd ..
foo:/usr/share $ pwd
/usr/share
foo:/usr/share $ cd .
foo:/usr/share $ pwd
/usr/share
```

The default prompt on some systems only displays the last part of the working directory.

```
[root@foo root]# pwd
/root
[root@foo root]# cd /usr/share/doc
[root@foo doc]# pwd
/usr/share/doc
[root@foo doc]# cd ..
```

```
[root@foo share]# pwd
/usr/share
[root@foo share]# cd .
[root@foo share]# pwd
/usr/share
[root@foo share]# cd
[root@foo root]# pwd
/root
```

Commands

You can enter a command in any working directory, and the system will locate the command by looking in the predefined list of locations (the PATH).

Here are a couple of examples of commands:

```
ls
mail
cal
ls -la
date
cat /etc/motd
cp /etc/motd /tmp
```

Each command that **bash** runs may be passed a number of optional parameters (arguments). Each argument is separated from the next argument by a space (or spaces, or tabs).

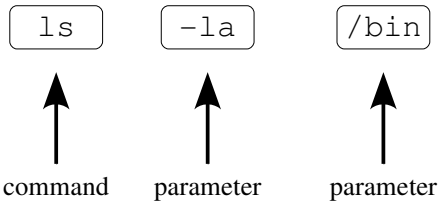


Illustration 6: Command line structure for `ls -la /bin`

In short, it's command, space, option, space, option, space.... The space after the command makes it possible for **bash** to know which part of the input is the command.

Command line options

Most commands take *switches*, which is a command line argument (option) which starts with a dash.

- **command -x** – A single letter command line option tells a command to behave slightly differently. For example, to ask ls to use the long format, the command is **ls -l**. Some of the more commonly used one-character flags are shown in the table.

Flag	*Meaning*	*Used with*
-l	Long format	**ls**
-v	Verbose	**cp**
-f	Force	**ln**, **cp**, **mv**, **rm**
-r	recursive	**grep**
-R	Recursive	**chown**, **chmod**, **ls**

- **-xyz** – For most command which accept single character flags, you can join the flags together in any order. As a result, the following are equivalent:

```
foo:~ $ ls -l -a
foo:~ $ ls -la
```

- **--option** – Many commands allow you to specify options in a more understandable format (although more keyboard intensive). A common and useful example is **--help**. Another common option is –**verbose**.

```
foo:~ $ ls -a
foo:~ $ ls --all
foo:~ $ rm -i *
foo:~ $ rm --interactive *
foo:~ $ mv --help
foo:~ $ tar --create --verbose --gzip --file backups.tar.gz /bin
foo:~ $ tar -cvzf backups.tar.gz /bin
foo:~ $ tar --list --gzip --file backups.tar.gz
foo:~ $ tar -tzf backups.tar.gz
```

- **filename** – Most commands expect you to specify the name of a file or directory to work with. These commands include **cp**, **mv** and **cd**. Remember that **../** means the previous directory, **./** means the current directory, and **~/** means your home directory.

```
foo:~ $ cp ../../etc/services ~/my-copy-of-services
```

14.3 Environment variables

The shell environment contains values called environment variables which can be displayed and changed. Many applications use environment variables, and you can also use them in your own scripts. The set command displays the current values of all environment variables. The **env** command shows a list of the exported environment variables.

To change the value of an environment variable, you can use the syntax

```
VARIABLENAME='value'
```

The following commands relate to setting environment variables.

```
foo:~ $ set | less
foo:~ $ echo PATH
foo:~ $ echo $PATH
foo:~ $ echo HOME
foo:~ $ echo $HOME
foo:~ $ echo $HOSTNAME
foo:~ $ echo $PS1
foo:~ $ PS1="# "
foo:~ $ PS1='\u: \w \$ '
foo:~ $ env | less     # search for PS1 by typing /PS1, Enter...
foo:~ $ HOME=/var
foo:~ $ cd
foo:~ $ pwd
foo:~ $ HOME=~
```

To remove a variable, you use the command **unset**.

```
foo:~ $ echo SSH_ASKPASS is $SSH_ASKPASS
SSH_ASKPASS is /usr/libexec/openssh/gnome-ssh-askpass
foo:~ $ unset SSH_ASKPASS
```

```
foo:~ $ echo SSH_ASKPASS is $SSH_ASKPASS
SSH_ASKPASS is
foo:~ $
```

The command **env** displays a list of all the **export**ed environment variables. These are the variables that are passed from the shell to applications when the applications are executed.

```
foo:~ $ env | head
KDE_MULTIHEAD=false
SSH_AGENT_PID=511
HOSTNAME=foo.ledge.co.za
TERM=xterm
SHELL=/bin/bash
XDM_MANAGED=/var/run/xdmctl/xdmctl-:0,maysd,mayfn,sched
HISTSIZE=1000
GTK_RC_FILES=/etc/gtk/gtkrc:/home/joe/.gtkrc
GS_LIB=/home/joe/.kde/share/fonts
QTDIR=/usr/lib/qt3-gcc3.4
```

The behaviour of many commands can be customised by setting (and exporting) shell variables as listed on the man pages of the application.

The **lpq** command shows the print queue for the current printer. The current printer is set using the **PRINTER** environment variable.

```
foo:~ $ echo PRINTER value is $PRINTER
PRINTER value is
foo:~ $ lpq
lp0 is ready
no entries
foo:~ $ PRINTER=lp1
foo:~ $ export PRINTER
foo:~ $ lpq
lp1 is not ready
no entries
```

Other applications which use environment variables for customising their behaviour include **less** (the pager), **ssh** (secure shell), **cvs** (concurrent versioning system), **make** (for compiling programs), **man** (system manual pages) and **startx** (uses the **WINDOWMANAGER** environment variable to choose the desktop system to use).

14.4 $PATH

When you enter a command, bash searches for the executable program in a number of directories. The places bash searches are specified in the **PATH** environment variable.

```
foo:~ $ echo $PATH
/home/user/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin
```

One of the places that bash does *not* search by default is the current directory – usually. Some distributions *do* include the current directory in the path (spot the difference).

```
linux:~ > echo $PATH
/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/home/joe/bin:
```

Because of the **PATH** environment variable, you can enter commands in any working directory:

```
linux:/usr/bin $ cd /bin
```

```
linux:/bin $ ls -la ls
-rwxr-xr-x  1 root root 90811 Apr 20 16:32 ls
linux:/bin $ cd /usr/bin
linux:/usr/bin $ ls -la ls
/bin/ls: ls: No such file or directory
```

Notice that you can use the **ls** command even when you are not in the **/bin** directory which contains the **ls** program.

14.5 Editing commands and command history

Bash has a number of command editing features which can speed up your work and give you fewer headaches than you would otherwise suffer from:

- Editing keys – while you are typing a command, you can use **Backspace** to erase what you have typed, and you can also move the cursor within the line (**left** and **right** arrows). You can jump to the beginning of the line (**Ctrl+A**) and to the end of the line (**Ctrl+E**) and cancel the entire operation (**Ctrl+C**).
- **Tab** – command and file name completion (if you press **Tab**, bash will complete the file name for you, and if you press **Tab** again, bash will show a list of all the options).
- **Up** and **Down** arrows – to show the previous command you typed in, you press the **Up** arrow. You can then edit this command with the editing keys.
- **Ctrl+R** – history search (press **Ctrl+R**, type something you typed previously, then **Enter** or **Esc**).

14.6 Command substitution $(...) and `...`

Some commands say interesting things that you want to be able to include as parameters to other commands. The command substitution operators allow you to include the output of a command in another command.

```
foo:~ $ whoami
gerald
foo:~ $ grep gerald /etc/passwd
gerald:x:500:500:G Smith:/home/gerald:/bin/bash
```

It's much easier like using substitution:

```
foo:~ $ grep `whoami` /etc/passwd
gerald:x:500:500: G Smith:/home/gerald:/bin/bash
foo:~ $ grep $( whoami ) /etc/passwd
gerald:x:500:500: G Smith:/home/gerald:/bin/bash
```

14.7 Recursive commands

Recursion in terms of Linux commands refers to applying a command to a all the files in a directory, and all the files in all the subdirectories (and subdirectories of subdirectories). Some commands, such as **ls**, **chown**, **chmod**, **cp**, **rm** and **grep** support switches of either **-R** or **-r**.

For commands that do not have a recursive mode of operation, you can combine the command with **find** to achieve your results.

14.7.1 -r switch

A common switch is **-r** which means *recursive*. Here are some commands that support this kind of recursion[13]:

- **chown -R** – change ownership
- **chmod -R** – change file mode (permissions)
- **cp -r** – copy files and subdirectories.

```
cp /tmp/alpha /tmp/beta                # copy?
cp -r /tmp/alpha /tmp/beta             # copy?
```

- **rm -r** – remove files *and* subdirectories
- **grep -r** – grep in files *and* directories.

14.7.2 cp -r

Here's how you can use **cp** to copy a directory:

```
% cp -r /etc ~
```

This will create a copy of **etc** in your home directory. Of course it's not a very good copy, since the permissions will change, and symbolic links will be updated. Often what you want is **cp -a** (archive copy) instead of **cp -r**.

14.7.3 chown and chmod

To change ownership recursively, the simplest thing to do is to use the **-R** switch:

```
mkdir -p /tmp/alpha/bravo/charlie      # make lots directories
ls -lR /tmp/alpha                      # see what we got
chown nobody /tmp/alpha                # change ownership
ls -lR /tmp/alpha                      # see what we got
chown -R nobody /tmp/alpha             # change ownership - subdirs
      too
ls -lR /tmp/alpha                      # see what we got
```

There are, of course a number of other ways to do it, not that you would want to:

```
chown -R user.group .
find . -exec chown nobody.wheel {} ";"
find . | xargs chown user.group
chown -R user.group * */* */*/*
```

Using **xargs** will fail if there are file names containing whitespace (i.e. spaces, tabs and newlines). The last example uses shell globbing and will miss all “hidden” files and only go to three direcories deep.

chmod is very similar to **chown** as far as its recursive usage is concerned:

```
chmod 700 /tmp/alpha                   # change permissions on
      directory
ls -lR /tmp/alpha                      # see what we got
chmod -R 700 /tmp/alpha                # change directory and
      subdirectories
ls -lR /tmp/alpha                      # see what we got
```

13 To understand recursion, you must first understand recursion.

The following are all equivalent:

```
chmod -R go+rX .
find . -exec chmod go+rX {} \;
find . | xargs chmod go+rX
```

14.7.4 grep -r

grep includes an option for recursively reading the contents of each file in a directory. This makes it possible to search all files in a directory and subdirectories for specific content:

```
grep "pop3" /etc              # search /etc?
grep -r "pop3" /etc           # search for "pop3" in all files in
     /etc
grep -r "127.0.0.1" /etc
```

There are other ways to do this as well:

```
find /etc -type f -exec grep "pop3" {} /dev/null ";"
```

The first example of using **grep -r** will *fail* if there is a FIFO in the **/etc** directory, unless some application helpfully closes the other side of the FIFO after **grep** opens its side. (Newer versions of **grep** avoid opening FIFO's.) Using **grep** with **find** is safer, but to force **grep** to print out the names of the files it finds, you need to specify **/dev/null** as an additional dummy file to search.

14.7.5 find for recursive commands

cat is an example of a command that does not have a recursive function. You can't **cat** all files in **/etc** with **cat -r /etc**.

We can use the recursive functionality of **find** to make commands that work on files recursive. **find ... -exec** can execute a specific command each time a file is found:

```
find /etc -type f -exec cat {} \;    # find with find, then cat each
     one
```

The output of **find** can list file names on the command line:

```
cat `find -type f /etc`              # cat whatever find says
find /etc -type f | xargs cat        # run cat with parameters from
     find
```

Using the parameter **-type f** to **find** ensures that we only consider regular files. Of course, **find** can use many other criteria to identify files, such as their name, permissions, modification date and more. **find** is discussed more fully in another chapter.

14.8 Bash session

During a login session, **bash** does a number of special things for you:

- At the beginning of the session, the file **~/.profile** is run automatically. Any special commands you place in this file are run whenever you log in.
- At the end of the session, all the commands you entered are added to the file **~/.bash_history**.

Sometimes the contents of **~/.profile** will include the **exec** command which replaces the shell

with a particular application. If you put this in a user's **~/.profile**, that user will automatically run **pine** when logging in interactively.

```
exec /usr/bin/pine          # when pine exits, the session is over
```

14.9 Man pages

Most commands have **man**ual pages which are accessed with the **man** command. The way to access these is –

```
man command
man section command
```

The following sections are used in the Linux **man** page system

- Section 1 – User Commands (important commands like **ps**, **ls** and **ln**)
- Section 2 – System Calls (for programming)
- Section 3 – Subroutines (for programming)
- Section 4 – Devices
- Section 5 – File Formats (like **passwd**, **hosts_access**, **shadow**, **inittab**)
- Section 6 – Games
- Section 7 – Miscellaneous
- Section 8 – System Administration (like **fsck**, **ping** and **reboot**)
- Section 9 – Kernel
- Section n – New

For introductory use, sections 1,5 and 8 provide the most useful information.

14.10 Review

Quiz questions

1. What command will bash attempt to run for the following example:
   ```
   % hello help me type
   ```
2. What is the command to create an environment variable called EXAMPLE and set its value to "Hello there"?
3. Why will the following code print a blank line?
   ```
   EXAMPLE="Hello there"
   echo "$example"
   ```
4. What is the effect of having a "." in the PATH environment variable, and why is this not good practice?
5. How do you set an environment variable (MYNAME) to contain the contents of a file (**/etc/HOSTNAME**)?
6. Write a command that will copy the files and subdirectories from **/etc** to **~/etcbackup**.

Assignment

1. Create a directory named **testdir** in your home directory. Copy **/usr/bin/vi** to **~/testdir/myeditor** and run it from the command line without modifying the **PATH**

environment variable. Now add the directory to your **PATH** environment variable, and verify whether you have modified the **PATH** environment variable correctly by running the command **myeditor**.

2. Use the **echo** command to show the current value of the **SHELL** environment variable. What happens if you run "**exec $SHELL**"? (Use "**echo $$**" to show your current process ID before and after running **exec**).
3. Set the variable **HELLO** to the value "Hello world". Run a sub-shell (i.e. run **bash**). Is the value of **HELLO** still set? Exit from the sub-shell and run a command which will make **HELLO** available to sub-shells.
4. Modify your **~/.profile** so that the output of the **date** command is stored in the shell variable **LOGIN_TIME**. Log in at the console and verify whether the variable was set when you logged in.
5. Read the bash man page. Pay special attention to the SHELL GRAMMAR and QUOTING sections.

Answers to quiz questions

1. bash runs the command **hello** with the first parameter "help" the second "me" and the third "type" .
2. The command is
```
EXAMPLE="Hello there"
```
3. Environment variable names are case sensitive.
4. "." is the current directory. This means that the shell will search for commands in the current directory. It is not good practice, since the current directory may include undesirable files that you do not necessarily want to run.
5. To set the variable and display its value:
```
MYNAME="`cat /etc/HOSTNAME`"
echo "$MYNAME"
```
6. This will do it ...
```
cp -a /etc ~/etcbackup
```

15 Text filters

cat > /dev/null # bash: sorry, no pets

– ~/.signature

To do bash scripting, you have to know how to filter your text.

LPIC topic 1.103.2 — Process text streams using filters [6]

Weight: 6

Objective

Candidate should be able to apply filters to text streams. Tasks include sending text files and output streams through text utility filters to modify the output, and using standard UNIX commands found in the GNU textutils package.

Key files, terms, and utilities include

cat	Show a file
cut	Cut a column out of the input
expand	Convert tabs to spaces
fmt	Do word wrapping and line wrapping
head	Show the top of the input
join	Text database join
nl	Number the lines of output
od	Octal dump (and other formats)
paste	Merge lines of files side by side
pr	Text to text conversion for printing
sed	Stream editor
sort	Sort the lines in the input
split	Create lots of little files from the input
tac	Print the input backwards
tail	Show the end of the input
tr	Translate character sets
unexpand	Convert spaces to tabs
uniq	Remove duplicates after **sort**
wc	Count words, characters and lines in the input

15.1 Introduction

Text filters are commands which receive a stream of data in, and send out a stream of modified data. The filters either select, sort, reformat, or summarise the data that they receive. Text filters are at the heart of UNIX shell scripting.

When using the text filtering commands, some of the more interesting files to play around with are these, which you are fairly sure to have on your system:

- **/etc/services** – list of TCP/IP service numbers

- **/etc/passwd** – list of users on your system
- **/etc/protocols** – list of IP based protocols
- **/etc/profile** – default login script
- **/etc/inittab** – configuration file for **init**.

Here's a brief rundown of the commands which will be discussed in this chapter.

- **cat** – con**cat**enate files (or just show a single file without alteration)
- **cut** – cut chosen text out of each line of a file and display it.
- **expand** – expand tabs into spaces
- **fmt** – reformat a text file with a consistent right margin
- **head –** show the first few (10) lines of a file
- **join** – join lines of two files on a common field
- **nl –** print the file with **n**umbered **l**ines
- **od** – **o**ctal **d**ump of a file (or hexadecimal).
- **paste** – print a number of files side by side
- **pr** – format for **pr**inting (split into pages or columns and add headers)
- **sed** – stream editor (search and replace, append, cut, delete and more)
- **sort** – sort in alphabetical order (and numerical order too)
- **split** – split a single input into multiple files
- **tac** – print the lines of a file from back to front (backwards **cat**)
- **tail** – print the last few lines of a file
- **tr** – character translation (e.g. upper case to lower case).
- **unexpand** – convert spaces to tabs (unlike **expand**).
- **uniq** – remove duplicate lines from a sorted file
- **wc** – word count (and line count, and byte count)

Other textutils programs

A more complete list of GNU textutils programs is this ...

```
$ rpm -ql textutils | grep bin | sed 's:.*/::' | sort | uniq | fmt
cat cksum comm csplit cut expand fmt fold head join md5sum nl od
      paste
pr ptx sha1sum sort split sum tac tail tr tsort unexpand uniq wc
```

The programs we are not discussing in this section are:

- **comm** – compare two files sorted line by line
- **csplit** – split a file into sections determined by context lines
- **fold** – word-wrapping (similar to **fmt -s -80**)
- **ptx** – produce a permuted index of file contents
- **cksum** – calculate a CRC checksum for files
- **md5sum** – compute and check MD5 message digest (checksum)
- **sha1sum** - compute and check SHA1 message digest (checksum)

15.2 Input and output redirection

Bash makes it possible to redirect the input and output of a command. Normally the input comes from the keyboard (and is ended by pressing **Ctrl+D**), while the output and any errors are displayed on the screen. Using redirection you can change the input of a process, its output, and the destination of the errors.

Redirection	*Effect of redirection*
command < file	Command reads input from a file
command > file	Output of command goes to file
command 2> file	Errors from the command go to a file
command >> file	Output of a command is added to a file
command > file 2>&1	Output *and* errors go to a file
command >& file	
command &> file	
command1 | command2	Output from command1 is input for command2

Work through these commands to experience redirection for yourself.

```
$ ls > file                          # pipe output to "file"
$ cat < file                         # read input from file and write
      to ..
$ ls -la | grep file                 # search output of ls using grep
$ cat /etc/passwd | sed 's/:.*//'    # filter output of cat through
      sed
$ badcommandorfilename 2> errorlog   # send errors to a file
$ grep pop3 /etc/* 2>/dev/null       # send errors to the bit bucket
$ grep pop3 /etc/* 2>&1 | less       # send errors(2) along with
      stdout(1)
$ less < errorlog                    # less gets input from file
$ less errorlog                      # less sees file name (look at
      prompt)
```

15.3 Selecting parts of a file

These are filters that print out various parts of the input they receive.

cat – concatenate

cat prints out an entire file.

```
foo:~ $ cat /etc/crontab
SHELL=/bin/sh
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin
MAILTO=root
#
# check scripts in cron.hourly, cron.daily, cron.weekly, and
      cron.monthly
#
-*/15 * * * *   root  test -x /usr/lib/cron/run-crons &&
```

```
      /usr/lib/cron/run-crons >/dev/null 2>&1
59 *  * * *     root  rm -f /var/spool/cron/lastrun/cron.hourly
14 0  * * *     root  rm -f /var/spool/cron/lastrun/cron.daily
29 0  * * 6     root  rm -f /var/spool/cron/lastrun/cron.weekly
44 0  1 * *     root  rm -f /var/spool/cron/lastrun/cron.monthly
```

cat can also join a number of files together. This works for text files and binary files.

```
foo:~/tmp $ cat /etc/passwd /etc/group /etc/protocols
foo:~/tmp $ echo hello > file1
foo:~/tmp $ echo there > file2
foo:~/tmp $ cat file1
hello
foo:~/tmp $ cat file2
there
foo:~/tmp $ cat file1 file2 file1
hello
there
hello
```

head – print out the first lines

By default head prints out the first 10 lines of a file.

```
foo:~ # head /var/log/boot.log
Apr  7 08:28:22 foo allrc: syslogd startup succeeded
Apr  7 08:28:22 foo allrc: klogd startup succeeded
Apr  7 08:28:23 foo allrc: portmap startup succeeded
Apr  7 08:27:56 foo rc.sysinit: Mounting proc filesystem:  succeeded
Apr  7 08:27:56 foo rc.sysinit: Unmounting initrd:  succeeded
Apr  7 08:27:56 foo sysctl: net.ipv4.ip_forward = 0
Apr  7 08:27:56 foo sysctl: net.ipv4.conf.default.rp_filter = 1
Apr  7 08:27:56 foo sysctl: kernel.sysrq = 0
Apr  7 08:28:26 foo lpd: execvp: No such file or directory
Apr  7 08:27:56 foo sysctl: kernel.core_uses_pid = 1
```

head can print out a specific number of lines from a file or a stream.

```
foo:~ $ ls -l / | head -n 6
foo:~ $ ls -l / | head -n 6
total 232
drwxr-xr-x    2 root     root         4096 Feb 21 15:49 bin
drwxr-xr-x    3 root     root         4096 Jan  7 10:25 boot
drwxr-xr-x    5 root     root        20480 Jan 10 11:35 data
drwxr-xr-x   21 root     root       118784 Apr  7 08:28 dev
drwxr-xr-x   64 root     root         8192 Apr  7 08:28 etc
```

One can also use **head** to extract an exact number of bytes from an input stream (rather than lines). Here's how to get a copy of the partition sector of a disk (be careful with that redirection).

```
foo:~ # head -c 512 < /dev/hda > mbr
foo:~ # ls -la mbr
-rw-r--r--    1 root     root          512 Apr  7 10:27 mbr
```

tail – show the end of a file

tail is just like **head**, but it shows the tail end of the file. Quite often it is used for viewing the

end of a log file:

```
root@foo:root # tail /var/log/messages
Apr  7 11:19:34 foo dhcpd: Wrote 9 leases to leases file.
Apr  7 11:19:34 foo dhcpd: DHCPREQUEST for 10.0.0.169 from
     00:80:ad:02:65:7c via eth0
Apr  7 11:19:35 foo dhcpd: DHCPACK on 10.0.0.169 to
     00:80:ad:02:65:7c via eth0
Apr  7 11:20:01 foo kdm[1151]: Cannot convert Internet address
     10.0.0.168 to host name
Apr  7 11:26:46 foo ipop3d[22026]: connect from 10.0.0.10
     (10.0.0.10)
Apr  7 11:26:55 foo ipop3d[22028]: connect from 10.0.0.10
     (10.0.0.10)
Apr  7 11:26:58 foo ipop3d[22035]: connect from 10.0.0.3 (10.0.0.3)
Apr  7 11:27:01 foo ipop3d[22036]: connect from 10.0.0.3 (10.0.0.3)
Apr  7 11:29:31 foo kdm[21954]: pam_unix2: session started for user
     joe, service xdm
Apr  7 11:32:41 foo sshd[22316]: Accepted publickey for root from
     10.0.0.143 port 1250 ssh2
```

tail can be used to watch a file as it grows. Run the command **tail -f /var/log/messages** on one console and then log in on another virtual console.

tail -n 20 file or **tail -20 file** will show the last 20 lines of a file. **tail -c 20 file** will show the last 20 characters of a file.

cut – pull out columns

Cut can be used to select certain columns of the input stream. Columns can be defined by either their position, or by being separated by field separators.

cut can select a certain set of data:

```
foo:~ # head /etc/passwd | cut -c 1-15
root:x:0:0:root
bin:x:1:1:bin:/
daemon:x:2:2:da
adm:x:3:4:adm:/
lp:x:4:7:lp:/va
sync:x:5:0:sync
shutdown:x:6:0:
halt:x:7:0:halt
poweroff:x:0:0:
mail:x:8:12:mai
```

cut can also select based on *fields* which are separated from each other by *field separators*. This is useful in dealing with files like **/etc/passwd** where each field is separated from the others by a colon.

```
foo:~ # head /etc/passwd | cut -d : -f 1,3-4
root:0:0
bin:1:1
daemon:2:2
adm:3:4
lp:4:7
sync:5:0
```

```
shutdown:6:0
halt:7:0
poweroff:0:0
mail:8:12
```

split – split a file into multiple parts

Split can be used to split a file into multiple parts. If you have a desire to print **/etc/services** you may want to print it on a printer which prints 66 lines per page.

```
foo:~ $ split -l 66 /etc/services
foo:~ $ wc x*
     66     341    2158 xaa
     66     353    2362 xab
     66     268    2030 xac
     66     275    2011 xad
     66     352    2441 xae
     66     322    2348 xaf
     66     367    2870 xag
     66     318    2245 xah
     39     202    1426 xai
    567    2798   19891 total
```

Split can split files into more manageable parts, e.g. for FTP uploads. The parts can be recombined with **cat**.

```
foo:~/download $ ls -la anomy-sanitizer-1.56.tar.gz
-rw-rw-r--    1 georgem  georgem    124356 Oct 22 18:37 anomy-
      sanitizer-1.56.tar.gz
foo:~/download $ split -b 32k anomy-sanitizer-1.56.tar.gz
foo:~/download $ ls -la x*
-rw-rw-r--    1 georgem  georgem     32768 Apr  7 11:48 xaa
-rw-rw-r--    1 georgem  georgem     32768 Apr  7 11:48 xab
-rw-rw-r--    1 georgem  georgem     32768 Apr  7 11:48 xac
-rw-rw-r--    1 georgem  georgem     26052 Apr  7 11:48 xad
```

Here's how to use **cat** to recombine the parts (and using **md5sum** to check whether the whole is equal to the sum of the parts).

```
foo:~/download $ split -b 32k anomy-sanitizer-1.56.tar.gz part-
foo:~/download $ ls -la part-*
-rw-rw-r--    1 georgem  georgem     32768 Apr  7 11:49 part-aa
-rw-rw-r--    1 georgem  georgem     32768 Apr  7 11:49 part-ab
-rw-rw-r--    1 georgem  georgem     32768 Apr  7 11:49 part-ac
-rw-rw-r--    1 georgem  georgem     26052 Apr  7 11:49 part-ad
foo:~/download $ cat part-* > newfile
foo:~/download $ md5sum newfile anomy-sanitizer-1.56.tar.gz
1a977bad964b0ede863272114bfc2482  newfile
1a977bad964b0ede863272114bfc2482  anomy-sanitizer-1.56.tar.gz
```

15.4 Sorting

tac – the opposite of cat

tac reverses the order of the lines in the input stream. This is quite useful for sorting backwards.

```
foo:~ $ ls / | cat | fmt
bin boot data dev etc home home.orig initrd lib lost+found misc mnt
      opt
proc root sbin suse tftpboot tmp usr var
foo:~ $ ls / | tac | fmt
var usr tmp tftpboot suse sbin root proc opt mnt misc lost+found lib
initrd home.orig home etc dev data boot bin
```

sort – sort the input

```
foo:~ $ for ((a=0;a<10;a++)) ; do echo $RANDOM hello ; done | sort
10219 hello
16397 hello
18021 hello
19353 hello
25265 hello
6923 hello
7771 hello
8340 hello
8466 hello
9117 hello
```

sort by default sorts in alphabetical order. This gives slightly strange results as shown above. To sort in numerical order, use the **-n** switch.

```
foo:~ $ for ((a=0;a<10;a++)) ; do echo $RANDOM hello ; done | sort
      -n
231 hello
1067 hello
1968 hello
4198 hello
9138 hello
15086 hello
19890 hello
20690 hello
24218 hello
25234 hello
```

uniq – discard duplicate lines

uniq is usually used with sort to discard duplicates

Here we are cutting the fourth field out of the password file (the group ID) and sorting in numerical order. **fmt** is used to make the results display on a single line.

```
foo:~ $ cat /etc/passwd | cut -d : -f 4 | sort -n | fmt
0 0 0 0 0 0 1 2 4 7 12 13 14 25 26 28 29 30 32 37 38 42 43 47 48 50
      51
69 74 77 80 89 99 100 500 501 503 504 505 506 507 509 511 512 65534
```

Here's the same command pipeline, but we are removing duplicates with **uniq** before formatting.

```
foo:~ $ cat /etc/passwd | cut -d : -f 4 | sort -n | uniq | fmt
0 1 2 4 7 12 13 14 25 26 28 29 30 32 37 38 42 43 47 48 50 51 69 74
      77
80 89 99 100 500 501 503 504 505 506 507 509 511 512 65534
```

15.5 Manipulation

tr – character set translation

tr is usually used for converting upper case to lower case. It can also do other character translations. **tr -d** can remove specific characters from a stream.

Translating from UPPER CASE to lower case. We are asking **man** to use **cat** as its pager, instead of using **less**, and opening up the man page for **man** itself.

```
foo:~ $ man -P cat man | tr A-Z a-z | less
```

Translating from lower case to UPPER CASE:

```
foo:~ $ man -P cat man | tr a-z A-Z | less
```

Convert file names to lowercase.

```
foo:/windows/C $ for FILE in * ; do
      mv "$FILE" $( echo "$FILE" | tr A-Z a-z ) ; done
```

ROT13 "encryption"[14]

```
foo:~ $ man -P cat man | tr a-zA-Z n-za-mN-ZA-M | less
```

Here we are using **tr -d** to delete the Carriage Returns (**\r**) from a file created with Windows Notepad.

```
foo:~ $ tr -d '\r' notepad.dos.txt > notepad.unix.txt
```

tr -d can be used to make normally readable text rather unusable, but it does show that vowels are not as necessary to the English language as you might have previously suspected.

```
foo:~ $ man -P cat tr | tr -d AEIOUaeiou | less
```

join – join files

join is capable of combining files based on common fields (similar to the SQL inner join, if you happen to know SQL).

```
foo:~ $ cat file1.sorted
cabbage vegetable
cat     animal
coal    mineral
piano   mineral
foo:~ $ cat file2.sorted
cabbage green   leaves
cat     white   cat-hair
piano   brown   wood
foo:~ $ join file1.sorted file2.sorted
cabbage vegetable green leaves
cat animal white cat-hair
piano mineral brown wood
```

Here we are joining **/etc/passwd** and **/etc/shadow** based on their first field (the user name). Since **/etc/passwd** and **/etc/shadow** use a colon to separate fields, it is necessary to use the **-t :** option.

```
foo:~ # join -t : /etc/passwd /etc/shadow | head
```

14 ROT13 really is encryption, even if it's spectacularly *bad* encryption.

```
root:x:0:0:root:/root:/bin/bash:$1$LHNUbu7U$oiuhqwd1oiuhqhAdiuHvA0:1
      2146:0:99999:7:::
bin:x:1:1:bin:/bin:/sbin/nologin:*:11974:0:99999:7:::
daemon:x:2:2:daemon:/sbin:/sbin/nologin:*:11974:0:99999:7:::
adm:x:3:4:adm:/var/adm:/sbin/nologin:*:11974:0:99999:7:::
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin:*:11974:0:99999:7:::
sync:x:5:0:sync:/sbin:/bin/sync:*:11974:0:99999:7:::
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown:*:11974:0:99999:7:::
halt:x:7:0:halt:/sbin:/sbin/halt:*:11974:0:99999:7:::
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin:*:11974:0:99999:7:::
```

join allows you to specify which particular field to join on, and also which particular fields should appear in the output (similar to **cut**)

The script below prints out the permissions (from **ls**) and file type (from **file**) for each file in the current directory. The 9th field from the second file (**-2**) is used for the join – this is the file name. The actual output (**-o**) consists of the first field from **.fileinfo** (**1.1**), the first field from **.listing** (**2.1**) and the second field from **.fileinfo** (**1.2**).

```
#! /bin/bash
file * >.fileinfo
ls -l * | tr -s ' ' ':' >.listing
join -t ':' -2 9 -o 1.1,2.1,1.2 .fileinfo .listing
```

nl – number lines

Ever wanted your printouts to be numbered? If so, then **nl** is the tool for you!

```
foo:/etc $ nl /etc/fstab
     1  LABEL=/       /           ext3    defaults        1 1
     2  /dev/hda3     /home       ext3    defaults        1 2
     3  none          /dev/pts    devpts  gid=5,mode=620  0 0
     4  none          /proc       proc    defaults        0 0
     5  none          /dev/shm    tmpfs   defaults        0 0
     6  /dev/hda2     swap        swap    defaults        0 0
     7  LABEL=DATA    /data       ext3    defaults        1 1
     8  /dev/cdrom    /mnt/cdrom  iso9660 user,noauto,owner,ro 0 0
```

(So as you can see in line 8, the CD ROM in this system is mounted at **/mnt/cdrom**.)

sed – stream editor

sed is a stream editor. **sed** does text transformations on an input stream. **sed** works by making only one pass over the inputs. A **sed** program consists of one or more **sed** commands which are applied to each line of the input. A command may be prefixed by an address range indicating the lines for which the command should be performed.

Here is a very abbreviated summary of the **sed** commands

- **s/PATTERN/REPLACEMENT/g** – search and replace. If you add the **g** at the end, the search and replace is applied as many times as possible to a single line.

```
foo:~ $ echo "Hi Fred, how are you" | sed 's/Fred/Joe/'
Hi Joe, how are you
foo:~ $ echo "Hi Fred, how is Fred?" | sed 's/Fred/Joe/'
Hi Joe, how is Fred?
foo:~ $ echo "Hi Fred, how is Fred?" | sed 's/Fred/Joe/g'
```

```
Hi Joe, how is Joe?
```

You can also use **i** at the end of the **s** command to make the search case-insensitive.

- **d** – delete the line. You need to select the lines as explained in the next paragraph.

```
foo:~ $ ls /bin/ | fmt -30 | nl | sed '4,15d'
     1  arch ash ash.static awk
     2  basename bash cat chgrp
     3  chmod chown chvt cp cpio csh
    16  true umount uname usleep vi
    17  vim zcat zsh
foo:~ $ ls /bin | fmt -40 | nl | sed '/e/ d'
     2  bash cat chgrp chmod chown chvt cp
    13  vi vim zcat zsh
```

The commands are most useful if you specify a range of lines to which the command applies. Here's how to specify specific lines for the **d** command:

- **/PATTERN/d** – delete all lines contains the pattern
- **4d** – delete line 4
- **4,10d** – delete lines 4 to 10
- **6,$d** – delete from line 6 to the last line

The most common way of using **sed** is to specify the editing commands as a command line argument. You can specify multiple commands separated by semicolons.

```
foo:~ $ echo "color is gray" |sed  's/color/colour/g;s/gray/grey/g'
colour is grey
```

Instead of using **s/PATTERN/REPLACEMENT/** you can use any other character in the place of the slash – e.g. a colon.

```
foo:~ $ find /usr/bin/ | sed 's:/usr/bin/:I found :' | head
I found
I found consolehelper
I found catchsegv
I found gencat
I found getconf
I found getent
I found glibcbug
I found iconv
I found ldd
I found lddlibc4
```

You can also use the line matching together with search and replace. Note that the replacement is only done on lines containing 'bash'.

```
foo:~ $ cat /etc/passwd | sed '/bash/ s/:/-RENAMED:/' | head
root-RENAMED:x:0:0:root:/root:/bin/bash
bin-RENAMED:x:1:1:bin:/bin:/bin/bash
daemon-RENAMED:x:2:2:Daemon:/sbin:/bin/bash
lp-RENAMED:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
mail:x:8:12:Mailer daemon:/var/spool/clientmqueue:/bin/false
news-RENAMED:x:9:13:News system:/etc/news:/bin/bash
```

expand – tabs to spaces

Don't like tabs? Use **expand** to make them go away!

You may notice with this example that running the output of **tr** through **expand** appears to have no effect – this is because tabs look just like spaces on the console.

```
foo:~ $ cat /etc/passwd | tr : '\t' | expand | head
root    x       0       0       root    /root   /bin/bash
bin     x       1       1       bin     /bin    /sbin/nologin
daemon  x       2       2       daemon  /sbin   /sbin/nologin
adm     x       3       4       adm     /var/adm
      /sbin/nologin
lp      x       4       7       lp      /var/spool/lpd
      /sbin/nologin
sync    x       5       0       sync    /sbin   /bin/sync
halt    x       7       0       halt    /sbin   /sbin/halt
mail    x       8       12      mail    /var/spool/mail
      /sbin/nologin
```

expand allows you to specify the size of the tab stops with **-t**. You can specify a single size for all tab stops, or you can set the tab stops at specific positions.

```
foo:~ $ cat /etc/passwd | tr : '\t' | expand -t 10,13,16,19,28,44 |
      head
root      x  0  0  root     /root           /bin/bash
bin       x  1  1  bin      /bin            /sbin/nologin
daemon    x  2  2  daemon   /sbin           /sbin/nologin
adm       x  3  4  adm      /var/adm        /sbin/nologin
lp        x  4  7  lp       /var/spool/lpd  /sbin/nologin
sync      x  5  0  sync     /sbin           /bin/sync
halt      x  7  0  halt     /sbin           /sbin/halt
mail      x  8  12 mail     /var/spool/mail /sbin/nologin
```

unexpand – spaces to tabs

If you have spaces and want tabs, you can achieve this with unexpand.

Here we are counting the number of lines, words and characters in the man page of bash. After piping the output through **unexpand** and replacing spaces with tabs the size is almost 19kb smaller, although it looks the same.

```
foo:~ $ man -P cat bash | wc
   4517   33928  300778
foo:~ $ man -P cat bash | unexpand | wc
   4517   33928  281381
```

paste – paste two files together

Using **paste** is like taking two printouts of two files and sticking the right margin of one to the left margin of the other. The glue between the files is a tab stop.

```
foo:~/tmp $ cat file2
cabbage green leaves
cat white cat-hair
piano brown wood
foo:~/tmp $ cat file1
cat animal
```

```
cabbage vegetable
piano mineral
coal mineral
foo:~/tmp $ paste file2 file1 | expand -t 22
cabbage green leaves  cat animal
cat white cat-hair    cabbage vegetable
piano brown wood      piano mineral
                      coal mineral
```

You can specify a delimiter between the files which is not a tab space with **-d**

```
foo:~/tmp $ paste -d '-' file2 file1
cabbage green leaves-cat animal
cat white cat-hair-cabbage vegetable
piano brown wood-piano mineral
-coal mineral
```

15.6 Formatting

fmt – format nicely

fmt is used to format text into a neatly word-wrapped structure.

The default right margin is 75

```
foo:~ $ ls /bin | fmt
arch ash ash.static aumix-minimal awk basename bash bash2 bsh cat
      chgrp
chmod chown cp cpio csh cut date dd df dmesg dnsdomainname doexec
domainname dumpkeys echo ed egrep env ex false fgrep gawk gettext
grep gtar gunzip gzip hostname igawk ipcalc kbd_mode kill link ln
loadkeys login ls mail mkdir mknod mktemp more mount mt mv netstat
      nice
nisdomainname pgawk ping ps pwd red rm rmdir rpm rvi rview sed
      setfont
setserial sh sleep sort stty su sync tar tcsh touch true umount
      uname
unicode_start unicode_stop unlink usleep vi vi.minimal view
      ypdomainname
zcat
```

Here we do the same thing with a different margin

```
foo:~ $ ls /bin | fmt -40 | head
arch ash ash.static aumix-minimal
awk basename bash bash2 bsh cat chgrp
chmod chown cp cpio csh cut date dd df
dmesg dnsdomainname doexec domainname
dumpkeys echo ed egrep env ex false
fgrep gawk gettext grep gtar gunzip
gzip hostname igawk ipcalc kbd_mode
kill link ln loadkeys login ls mail
mkdir mknod mktemp more mount mt mv
netstat nice nisdomainname pgawk ping
```

pr – format for printing

In the good old days, when it was traditional to send reams and reams of printouts to a line printer, it was nice to add file names as headers and to make sure you didn't print on the fold between papers. This is what **pr** does for you.

```
foo:~ $ pr /etc/host* | less
foo:~ $ ls -lR / | pr | less
```

od – octal dump (and other formats)

od, oddly enough, does not just print out in octal, but in other formats.

```
foo:/tmp $ echo "Hello World" > hello
foo:/tmp $ cat hello
Hello World
```

od's behaviour is rather odd when it is used without any arguments. It prints out the octal value of two pairs of bytes in the file. Sometimes this is useful, but usually it is not.

```
foo:/tmp $ od hello
0000000 062510 066154 020157 067527 066162 005144
0000014
```

Using the **-t** switch tells **od** to use a specific format type (the default is **od -t o2**). **od -t c** means character format. You will notice that file ends in a newline character (**\n**).

```
foo:/tmp $ od -t c hello
0000000   H   e   l   l   o       W   o   r   l   d  \n
0000014
```

od -t d1 specifies decimal format, with a width of one. The more astute among you will realise that the character encoding is ASCII.

```
foo:/tmp $ od -t d1 hello
0000000   72  101  108  108  111   32   87  111  114  108  100   10
0000014
```

You can specify more than one format

```
foo:/tmp $ od -t d1c hello
0000000   72  101  108  108  111   32   87  111  114  108  100   10
          H   e   l   l   o       W   o   r   l   d  \n
0000014
```

Here's *Hello World* in decimal, characters and hexadecimal.

```
foo:/tmp $ od -t d1cx1 hello
0000000   72  101  108  108  111   32   87  111  114  108  100   10
          H   e   l   l   o       W   o   r   l   d  \n
         48 65 6c 6c 6f 20 57 6f 72 6c 64 0a
0000014
```

wc – count words, lines and characters

For each file specified on the command line, **wc** prints out the number of words, lines and characters in the file.

```
foo:~ $ wc /etc/host*
      1       2      17 /etc/host.conf
```

```
     10      34     291 /etc/hosts
      8      36     201 /etc/hosts.allow
      5      25     196 /etc/hosts.bak
      9      65     357 /etc/hosts.deny
     33     162    1062 total
```

wc -l prints just the number of lines, **wc -w** prints only the number of words and **wc -c** prints the number of characters (bytes) in the file. These features allow **wc** to be used to count and measure the output from other commands.

```
foo:~ $ ls -l | wc -l
    211
```

Here we are using **cat** to show the contents of all the files in **/bin**, and then using **wc** to add up the size of those.

```
foo:~ $ cat /bin/* | wc -c
18682972
```

15.7 Review

Quiz questions

Which commands perform the following functions:

1. Show the first few lines of a file
2. Show the last few lines of a file
3. Select a column from a file
4. Send the entire file
5. Convert new lines to spaces
6. Convert spaces to tabs
7. Convert tabs to spaces
8. Arrange in alphabetical order
9. Remove duplicate lines
10. Rearrange lines from last to first
11. Substitute all instances of one character with another
12. Add line numbers to output
13. Search for and replace text in a stream
14. Apply a consistent right margin to text
15. Show a file in hexadecimal, octal or other formats
16. Convert a text file to postscript
17. Determine the size of a text stream

Assignment

1. Figure out something useful to do with join (other than **join -t : /etc/passwd /etc/shadow**)
2. Read the man pages for each of the commands discussed in this chapter, and make notes about options and modes of operation which have not been described.

3. **ls /etc/*.*** prints a list of files and directories containing a dot in their name. Use **sed** to remove the leading "**/etc/**" from the filenames and format the list for presentation using **fmt** and **pr**.
4. Create a tabulated list of user names, home directories and shells from **/etc/passwd**. Use **cut** to select the appropriate columns from the password file; use **tr** to translate colons ":" to tab characters; and use **expand** to format the output.
5. Concatenate all the data in the files in **/etc** using **cat**. Sort the lines alphabetically and remove duplicates. How many lines were there in **/etc** before you removed duplicates, and how many were there after you removed duplicates.
6. The command **ls -latr** prints a directory listing in reverse order, sorted by time. Use the **tail** command to show the newest 5 files from this listing. Do it again using the **tac** and **head** commands.
7. Create a file containing a copy of **/etc/services** with the lines numbered. Split the file you have created into parts of 1000 lines each named **services-aaaa, services-aaab** and so forth in your home directory. Check how many lines are in each file.

Answers to quiz questions

1. head
2. tail
3. cut
4. cat
5. fmt
6. unexpand
7. expand
8. sort
9. sort | uniq
10. tac
11. tr
12. nl
13. sed
14. fmt
15. od
16. pr doesn't – none of these commands do – mpage does it though.
17. wc -l

16 File management

Unix is all about files. Everything is a file. Because of this, it's important to be able to shunt your files around.

> Mommy, what happens to your files when you die?
>
> *– fortune cookie database*

LPIC topic 1.103.3 — Perform basic file management [3]

Weight: 3

Objective

Candidate should be able to use the basic UNIX commands to copy, move, and remove files and directories. Tasks include advanced file management operations such as copying multiple files recursively, removing directories recursively, and moving files that match a wildcard pattern. This includes using simple and advanced wildcard specifications to refer to files, as well as using find to locate files based on type, size, or time.

Key files, terms, and utilities include

cp	copy files
find	find files based on command line specifications
mkdir	make a directory
mv	move or rename files and directories
ls	list files
rm	remove files (and directories, if you ask nicely)
rmdir	remove directory
touch	touch a file to update its timestamp
file globbing	*, ?, [abc] and {abc,def} wildcards

16.1 Files, directories and ls

ls lists the contents of directories, and is one of the most frequently used commands on the command line.

```
bar@foo:~> cd /etc/ppp/
bar@foo:/etc/ppp> ls -la
total 68
drwxr-x---    5 root  dialout 4096 2003-05-12 12:36 ./
drwxr-xr-x   53 root  root    8192 2003-06-04 10:51 ../
lrwxrwxrwx    1 root  root       5 2003-03-03 11:09 ip-down -> ip-
      up*
drwxr-xr-x    2 root  root    4096 2002-09-10 19:40 ip-down.d/
-rwxr-xr-x    1 root  root    9038 2002-09-10 19:40 ip-up*
drwxr-xr-x    2 root  root    4096 2002-09-10 19:40 ip-up.d/
-rw-r--r--    1 root  root    8121 2002-09-09 22:32 options
-rw-------    1 root  root    1219 2002-09-09 22:32 pap-secrets
#type+perm link user  group   size date       time  name
```

In the output of **ls**, the meaning of the columns from left to right are:

- file type and permissions (**d** – directory, **l** – symbolic link, **r** – read, **w** – write, **x** – executable or searchable)
- number of links to the file
- user name
- group name
- size (in bytes, unless you specify **-h**).
- date, time
- file name

After the file name **ls** may print additional information about the file. "**/**" – it's a directory, "*****" – it's executable, "**-> somewhere**" - it's a symbolic link

16.2 File globbing (wildcards)

When playing card games like rummy, the joker is a "wildcard" and can take the place of any card. Similarly, in the shell, there are special characters and strings that can take the place of any characters, or of specific characters.

> **glob,n.** [Unix; common] To expand special characters in a wildcarded name, or the act of so doing (the action is also called "globbing"). The Unix conventions for filename wildcarding have become sufficiently pervasive that many hackers use some of them in written English, especially in email or news on technical topics. Those commonly encountered include the following:
> * – wildcard for any string (see also UN*X)
> ? – wildcard for any single character (generally read this way only at the beginning or in the middle of a word)
> [] – delimits a wildcard matching any of the enclosed characters
> {} – alternation of comma-separated alternatives; thus "foo{baz,qux}" would be read as "foobaz" or "fooqux".
> Historical note: The jargon usage derives from `glob', the name of a subprogram that expanded wildcards in archaic pre-Bourne versions of the Unix shell.
>
> – *the jargon file (edited)*

The shell expands the following special characters just in case they are part of a file name. Filename expansion is only performed if the special characters are in quote

~	/home/user or /root (the user's home directory)	**echo ~/.bashrc**
*****	Any sequence of characters (or no characters)	**echo /etc/*ost***
?	Any one character	**echo /etc/hos?**
[abc]	Any of a, b, or c	**echo /etc/host[s.]**
{abc,def,ghi}	Any of abc, def or ghi.	**echo /etc/hosts.{allow,deny}**

Here are some examples:

```
foo:~ $ cd /e*
```

```
foo:/etc $ echo host*
host.conf hosts hosts.allow hosts.bak hosts.deny
foo:/etc $ ls -l host*
-rw-r--r--    1 root     root           17 Jul 23  2000 host.conf
-rw-rw-r--    2 root     root          291 Mar 31 15:03 hosts
-rw-r--r--    1 root     root          201 Feb 13 21:05 hosts.allow
-rw-rw-r--    1 root     root          196 Dec  9 15:06 hosts.bak
-rw-r--r--    1 root     root          357 Jan 23 19:39 hosts.deny
foo:/etc $ grep ALL hosts.{allow,deny}
hosts.allow:ALL: friendpc
hosts.deny:ipop3d: ALL
foo:/etc $ ls [A-Z]* -lad
drwxr-xr-x    3 root     root         4096 Oct 14 11:07 CORBA
-rw-r--r--    1 root     root         2434 Sep  2  2002 DIR_COLORS
-rw-r--r--    1 root     root         2434 Sep  2  2002
      DIR_COLORS.xterm
-rw-r--r--    1 root     root        92336 Jun 23  2002 Muttrc
drwxr-xr-x   18 root     root         4096 Feb 24 19:54 X11
foo:~ $ ls -d /etc/rc.d/rc[0-6].d
/etc/rc.d/rc0.d  /etc/rc.d/rc2.d  /etc/rc.d/rc4.d  /etc/rc.d/rc6.d
/etc/rc.d/rc1.d  /etc/rc.d/rc3.d  /etc/rc.d/rc5.d
```

If finding files with shell globbing is not flexible enough, you can use the **find** command.

16.3 Directories and files

mkdir – make directory

To make a new directory, use **mkdir**. By default, the parent directory must exist, but with the **-p** switch **mkdir** will create the required parent directories too.

```
[jack@foo tmp]$ mkdir dir1
[jack@foo tmp]$ mkdir dir2 dir3
[jack@foo tmp]$ ls -la
total 20
drwxrwxr-x    5 jack     jack         4096 Apr  8 10:00 .
drwxr-xr-x   12 jack     users        4096 Apr  8 10:00 ..
drwxrwxr-x    2 jack     jack         4096 Apr  8 10:00 dir1
drwxrwxr-x    2 jack     jack         4096 Apr  8 10:00 dir2
drwxrwxr-x    2 jack     jack         4096 Apr  8 10:00 dir3
[jack@foo tmp]$ mkdir parent/sub/something
mkdir: cannot create directory `parent/sub/something': No such file
      or directory
[jack@foo tmp]$ mkdir -p parent/sub/something
[jack@foo tmp]$ find
.
./dir1
./dir2
./dir3
./parent
./parent/sub
./parent/sub/something
```

The permissions for a new directory are determined by the umask value, unless you set them manually using the **-m** switch.

umask	*Directory mode*	*Directory mode*
000	0777	drwxrwxrwx
002	0775	drwxrwxr-x
022	0755	drwxr-xr-x
027	0750	drwxr-x---

rmdir – remove directory

rmdir can be used to remove empty directories. If the directory contains any files and directories, **rmdir** will fail to remove it (although **rm -r** can).

```
[jack@foo tmp]$ find
.
./dir1
./dir2
./dir3
./parent
./parent/sub
./parent/sub/something
[jack@foo tmp]$ rmdir dir1
[jack@foo tmp]$ rmdir dir2 dir3
[jack@foo tmp]$ rmdir parent
rmdir: `parent': Directory not empty
[jack@foo tmp]$ rmdir parent/sub/something/
[jack@foo tmp]$ rmdir parent
rmdir: `parent': Directory not empty
[jack@foo tmp]$ rmdir parent/sub
[jack@foo tmp]$ rmdir parent
[jack@foo tmp]$ find
.
[jack@foo tmp]$ rmdir /etc
rmdir: `/etc': Permission denied
```

touch – update a file's time stamp

touch was originally written to update the time stamp of files. However, due to a programming error, **touch** would create an empty file if it did not exist. By the time this error was discovered, people were using **touch** to create empty files, and this is now an accepted use of **touch**.

```
[jack@foo tmp]$ mkdir dir4
[jack@foo tmp]$ cd dir4
[jack@foo dir4]$ ls -l
total 0
[jack@foo dir4]$ touch newfile
[jack@foo dir4]$ ls -l
total 0
-rw-rw-r--    1 jack     jack            0 Apr  8 10:28 newfile
```

The second time we run **touch** it updates the file time –

```
[jack@foo dir4]$ touch newfile
```

```
[jack@foo dir4]$ ls -l
total 0
-rw-rw-r--    1 jack     jack            0 Apr  8 10:29 newfile
```

touch can create more than one file at a time –

```
[jack@foo dir4]$ touch file-{one,two}-{a,b}
[jack@foo dir4]$ ls -l
total 0
-rw-rw-r--    1 jack     jack            0 Apr  8 10:33 file-one-a
-rw-rw-r--    1 jack     jack            0 Apr  8 10:33 file-one-b
-rw-rw-r--    1 jack     jack            0 Apr  8 10:33 file-two-a
-rw-rw-r--    1 jack     jack            0 Apr  8 10:33 file-two-b
-rw-rw-r--    1 jack     jack            0 Apr  8 10:29 newfile
```

rm – remove files

rm removes files, and also symbolic links and other strange file system objects.

```
[hack@foo dir4]$ cd ..
[hack@foo tmp]$ ls dir4/
file-one-a  file-one-b  file-two-a  file-two-b  newfile
[hack@foo tmp]$ rm dir4/file*a
[hack@foo tmp]$ ls dir4/
file-one-b  file-two-b  newfile
```

rm -f causes **rm** not to complain if the file you are trying to remove is not present.

```
[hack@foo tmp]$ rm dir4/file*a
rm: cannot lstat `dir4/file*a': No such file or directory
Try `rm --help' for more information.
[hack@foo tmp]$ rm -f dir4/file*a
[hack@foo tmp]$
```

rm -r can be used to remove directories *and* the files contained in them. Here we are using **-i** to specify interactive behaviour (confirmation before removal).

```
[hack@foo tmp]$ rm -ri dir4
rm: descend into directory `dir4'? y
rm: remove regular empty file `dir4/newfile'? y
rm: remove regular empty file `dir4/file-one-b'? y
rm: remove regular empty file `dir4/file-two-b'? y
rm: remove directory `dir4'? y
```

rm -rf is used to remove directories and their files without prompting.

```
[hack@foo hack]$ mkdir tmp
[hack@foo hack]$ touch tmp/one tmp/two tmp/third-file tmp/four
[hack@foo hack]$ ls tmp
four  one  third-file  two
[hack@foo hack]$ rm -rf tmp
hack@foo hack]$ ls tmp
ls: tmp: No such file or directory
```

16.4 Copying and moving

cp – copy files

cp has two modes of usage:

- Copy one or more files to a directory – e.g. **cp file1 file2 file3 /home/user/directory/**
- Copy a file and to a new file name – e.g. **cp file1 file13**

If there are more than two parameters given to **cp**, then the last parameter is always the destination directory.

Here we are copying a file (**/etc/passwd**) to a directory (**.**)

```
[jack@foo jack]$ ls
[jack@foo jack]$ cp /etc/passwd .
[jack@foo jack]$ ls
passwd
```

Here we are copying a file (**/etc/passwd**) and renaming it to **copy-of-passwd**.

```
[jack@foo jack]$ cp /etc/passwd copy-of-passwd
[jack@foo jack]$ ls
copy-of-passwd  passwd
```

cp -v is **v**erbose about the progress of the copying process. In the first example below, there is no feedback. In the second example, there's a lot of feedback.

```
[jack@foo jack]$ mkdir copybin
[jack@foo jack]$ cp /bin/* ./copybin
[jack@foo jack]$ cp -v /bin/* ./copybin
`/bin/arch' -> `./copybin/arch'
`/bin/ash' -> `./copybin/ash'
`/bin/ash.static' -> `./copybin/ash.static'
`/bin/aumix-minimal' -> `./copybin/aumix-minimal'
`/bin/awk' -> `./copybin/awk'
```

cp -i interactively asks for feedback before it does something drastic, such as overwriting an existing file.

```
[jack@foo jack]$ cp -i /bin/* ./copybin
cp: overwrite `./copybin/arch'? n
cp: overwrite `./copybin/ash'? n
cp: overwrite `./copybin/ash.static'?
```

mv – move or rename files

mv, like **cp**, has two modes of operation:

- Move one or more files to a directory – e.g. **mv file1 file2 file3 /home/user/directory/**
- Move a file and to a new file name (rename) – e.g. **mv file1 file13**

If there are more than two parameters given to **mv**, then the last parameter is always the destination directory.

All of these examples use **-v** so that **mv** is **v**erbose.

```
[jack@foo jack]$ mkdir fred
[jack@foo jack]$ mv -v fred joe
`fred' -> `joe'
[jack@foo jack]$ cp -v /etc/passwd joe/
`/etc/passwd' -> `joe/passwd'
[jack@foo jack]$ mv -v joe/passwd .
`joe/passwd' -> `./passwd'
```

mv is the command used for renaming files

```
[jack@foo jack]$ mv -v passwd ichangedthenamehaha
`passwd' -> `ichangedthenamehaha'
[jack@foo jack]$ cp /etc/services .
```

mv -i, like **cp -i**, interactively asks for confirmation if a file will be overwritten

```
[jack@foo jack]$ mv -i services ichangedthenamehaha
mv: overwrite `ichangedthenamehaha'? y
```

16.5 find

find is able to find files that

- have names that contain certain text or match a given pattern
- are links to specific files
- were last used during a given period of time
- are within a given range of size
- are of a specified type (normal file, directory, symbolic link, etc.)
- are owned by a certain user or group or have specified permissions.

(Not all of these capabilities are discussed here).

Using find

The simplest way of using **find** is to specify a list of directories for find to search. By default **find** prints the name of each file it finds.

```
[jack@tonto jack]$ find /usr/share/doc/unzip-5.50/ /etc/skel/
/usr/share/doc/unzip-5.50/
/usr/share/doc/unzip-5.50/BUGS
/usr/share/doc/unzip-5.50/INSTALL
/usr/share/doc/unzip-5.50/LICENSE
/usr/share/doc/unzip-5.50/README
/etc/skel/
/etc/skel/.kde
/etc/skel/.kde/Autostart
/etc/skel/.kde/Autostart/Autorun.desktop
/etc/skel/.kde/Autostart/.directory
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
/etc/skel/.emacs
/etc/skel/.gtkrc
```

find tests

find is usually used to search based on particular criteria. The more commonly used ones are shown in the table.

Test	*Meaning*	*Example*
-name	find based on the file's name	`find /bin /usr/bin -name 'ch*'`
-iname	case insensitive file name	`find /etc -iname '*pass*'`
-ctime	find based on the time that the file was	`find /tmp -atime +30`

Test	*Meaning*	*Example*
-mtime -atime	created, or last modified, or last accessed	`find /var/log -mtime -1` `find /home -ctime +365`
-group, -user -gid, -uid	find based on the group or user of the file	`find /tmp -user `whoami`` `find /tmp -uid `id -u``
-perm	find based on the file permissions (exactly, +including, -all)	`find /usr/bin -perm -4001` `find /etc -type f -perm +111`
-size	find files of a particular size	`find / -size +5000k`
-type	find **f**iles, **d**irectories, **p**ipes, **s**ockets and **l**inks	`find /dev /var -type p`

find -exec

When **find** finds a file, the default action is to print out the file's name. The one alternative action that is quite useful is **-exec** where **find** executes the specified program, usually passing the file name to it. **find** substitutes the name of the command in the place of the special characters **{}**. The command must end with "**;**" which must be quoted so that it is not interpreted by the shell.

```
foo:~ $ find /bin -perm -4001 -exec ls -lad {} \;
-rwsr-xr-x    1 root     root        35302 Jun 23  2002 /bin/ping
-rwsr-xr-x    1 root     root        81996 Aug 30  2002 /bin/mount
-rwsr-xr-x    1 root     root        40700 Aug 30  2002 /bin/umount
-rwsr-xr-x    1 root     root        19132 Aug 29  2002 /bin/su
```

Have you ever wondered how many things need to be fixed in the Linux kernel?

```
foo:~ $ cd /usr/src/linux
foo:/usr/src/linux $ find -name '*.c' -exec grep FIXME {} ";" | wc
      -l
   1401
```

Want to **grep** through the C files in the kernel? That **/dev/null** is there to make sure that **grep** prints out a file name together with the data from the file.

```
$ cd /usr/src/linux
$ find -name "*.c" -exec grep -w zlib_inflate {} /dev/null \;
```

16.6 Review

Quiz questions

The answers to these questions appear in this chapter.

1. What is the parameter for copy, move and remove that requests confirmation from the terminal before acting?
2. Who is the owner of a file created using touch?
3. Which parameter for copy will cause it to copy file permissions and ownership as far as possible?

4. How do you find files which have not been modified for more than 2 days?
5. How do you find executable files which have the set-gid bit set?

Assignment

1. Perform the following instructions, and make a list of the commands you used
   1. Make a directory named **stuff** for the commands below.
   2. Copy the following files to your stuff directory: **/etc/passwd**, **/etc/services**, **/etc/profile** and all files in **/etc** starting with the letter “h” or “H”.
   3. Copy the directory **/etc/sysconfig** and all of its subdirectories and files to your stuff directory.
   4. Rename the file **hosts** to the name **network peers**.
   5. Delete the files **host.conf** and **nsswitch.conf** using a wildcard (but don't delete other files ending in **.conf**).
   6. Make a list of the files in your home directory which are larger than 100kb using **find**.
   7. Remove the **sysconfig** directory you copied earlier.
2. *Ensure that you have a rescue disk handy before doing this assignment. Do not do this assignment on a system containing valuable data.*

   Install the stand-alone shell (**sash**), and review its manual page, particularly the **aliasall** command. As root, enter the command '**mv -f /***'. Now recover your system – either use a rescue disk, or run **/var/bin/sash**. You can pass the parameter **init=/var/bin/sash** to the kernel if you boot from LILO, otherwise you will need to use a rescue disk. What happened? Should you have known? Did you get some valuable experience with the **mv** command?

Answers to quiz questions

1. -i
2. The user that ran the command.
3. -a
4. find -mtime +2
5. find -type f -perm -2001

17 Redirection

> Row, row, row your bits, gently down the stream...
>
> */usr/share/games/fortune/computers*

Redirection is that property of Unix operating systems that makes them quite useful.

LPIC topic 1.103.4 — Use streams, pipes, and redirects [5]

Weight: 5

Objective

Candidate should be able to redirect streams and connect them in order to efficiently process textual data. Tasks include redirecting standard input, standard output, and standard error, piping the output of one command to the input of another command, using the output of one command as arguments to another command, and sending output to both stdout and a file.

Key files, terms, and utilities include:

tee	send output to standard output and files
xargs	expand input to command line arguments
<	redirect standard input from a file
<<	redirect standard input from text in a script
>	send standard output to a file
>>	append standard output to a file
|	pipe a programs's standard output to standard input of another
` `	capture the output of a command and use it in a script

17.1 Input and output redirection

The **bash** shell, and the other shells employed by Linux allow the user to specify what should happen to the input and output of programs that run.

The shell creates new processes as follows:

1. The shell reads the commands to be executed
2. The shell opens the files which will be needed (open(2) for files, pipe(2) for streams)
3. The shell creates the new processes which will be executed (fork(2) each process)
4. Each new process is connected to the appropriate plumbing (dup2(2) each redirection)
5. Control of the processes is passed to the programs the user asked for (execve(2) each command)

Redirections with pipes (|) are processed first, and other redirections (with **>** and **<**) are processed from left to right.

The following redirections are possible

Redirection	*Effect of redirection*
command < file	Command reads input from a file
command > file	Output of command goes to file
command 2> file	Errors from the command go to a file
command >> file	Output of a command is added to a file
command > file 2>&1 command >& file command &> file	Output *and* errors go to a file
command1 | command2	Output from command1 is input for command2

Work through these commands to experience redirection for yourself.

```
$ ls > file
$ cat < file
$ ls -la | grep file
$ cat /etc/passwd | sed 's/:.*//'
$ badcommandorfilename 2> errorlog
$ grep pop3 /etc/* 2>/dev/null
$ grep pop3 /etc/* 2>&1 | less
$ less < errorlog
$ less errorlog
```

17.2 Standard input redirection (<, <<EOF, |)

Standard input is the terminal, by default. The following redirections are possible:

No redirection. Although we are telling **wc** to examine a specific file in this example, we are not changing standard input using redirection

```
[jack@foo jack]$ wc /etc/passwd
     46      62    2010 /etc/passwd
```

Read input from the terminal:

```
[jack@foo jack]$ wc
Hello.  I typed this line.  I typed and
typed.  I wondered, as I typed, how many
words I was typing. Tis wondrous that as
I typed, bash piped.
<Ctrl+D>
      4      28     143
[jack@foo jack]$
```

Read input from a file using **<**. Notice that standard input is anonymous – **wc** does not know the name of the file which it is reading from.

```
[jack@foo jack]$ wc < /etc/passwd
     46      62    2010
```

Read input from a program using the pipe redirection. Here **cat** is piping its output to **wc**.

```
[jack@foo jack]$ cat /etc/passwd | wc
     46      62    2010
```

Read input from the script or the command line. The **<<** operator is traditionally followed by **EOF**, but any other letters can be used. When a line is found which consists only of **EOF**, that's the end of the input. This is called a "here document".

```
[jack@foo jack]$ wc << EOF
> Hello
> There are
> Four lines
> I think
> EOF
      4       7      35
```

17.3 Standard output redirection (>, >>)

Standard output is the terminal, by default. The following redirections are possible:

No redirection

```
[jack@foo jack]$ uptime
  5:09pm  up  8:26,  6 users,  load average: 0.22, 0.20, 0.20
```

Redirection to a file ("**>**" means "to"). If the file already exists, it is overwritten.

```
[jack@foo jack]$ uptime > Uptime-file
[jack@foo jack]$ ls -l Uptime-file
-rw-rw-r--    1 jack     jack           62 Apr  8 17:09 Uptime-file
[jack@foo jack]$ cat Uptime-file
  5:09pm  up  8:26,  6 users,  load average: 0.13, 0.18, 0.19
```

Redirection appending to a file ("**>>**" means "to the end of").

```
[jack@foo jack]$ uptime >> Uptime-file
[jack@foo jack]$ cat Uptime-file
  5:09pm  up  8:26,  6 users,  load average: 0.13, 0.18, 0.19
  5:10pm  up  8:27,  6 users,  load average: 0.24, 0.21, 0.20
```

Redirecting to a program. Here the output of **uptime** is being piped as the input to **wc**.

```
jack@foo jack]$ uptime | wc
      1      10      62
```

17.4 Standard error redirection (2>, 2>>, 2>&1)

By default, Linux processes produce two error streams. Regular output is sent to *stdout* (standard output, "1") and errors are sent to *stderr* (standard error, "2").

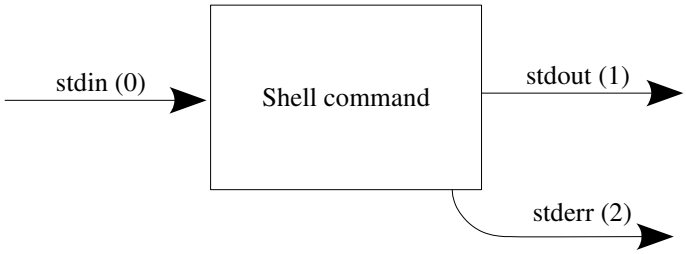


Illustration 7: shell commands receive input and produce output

The output sent to *stderr* is usually the following:

- Error messages (e.g. **command 2>/dev/null**)
- Warning messages (e.g. **command 2>> warning.log**)
- Debugging and progress indicators (e.g. **command 2>&1 | less**)

Here are the various possible error redirections.

If *stderr* is not redirected, error output is displayed on the terminal.

```
[jack@foo jack]$ tar cf sysconfig.tar /etc/sysconfig
tar: Removing leading `/' from member names
tar: /etc/sysconfig/rhn/systemid: Cannot open: Permission denied
tar: /etc/sysconfig/rhn/up2date-keyring.gpg: Cannot open: Permission
     denied
tar: /etc/sysconfig/iptables: Cannot open: Permission denied
tar: /etc/sysconfig/static-routes: Cannot open: Permission denied
tar: Error exit delayed from previous errors
```

Here we are sending the errors to a file. It is quite common to send error text to **/dev/null** – especially when you know what it says.

```
[jack@foo jack]$ tar cf sysconfig.tar /etc/sysconfig 2> /dev/null
[jack@foo jack]$ tar cf sysconfig.tar /etc/sysconfig 2> warning.log
```

We can append the errors to an existing file ...

```
[jack@foo jack]$ tar cf pam.tar /etc/pam.d/
tar: Removing leading `/' from member names
tar: /etc/pam.d/sshd: Cannot open: Permission denied
tar: Error exit delayed from previous errors
[jack@foo jack]$ tar cf pam.tar /etc/pam.d/ 2>>warning.log
[jack@foo jack]$ cat warning.log
tar: Removing leading `/' from member names
tar: /etc/sysconfig/rhn/systemid: Cannot open: Permission denied
tar: /etc/sysconfig/rhn/up2date-keyring.gpg: Cannot open: Permission
     denied
tar: /etc/sysconfig/iptables: Cannot open: Permission denied
tar: /etc/sysconfig/static-routes: Cannot open: Permission denied
tar: Error exit delayed from previous errors
tar: Removing leading `/' from member names
tar: /etc/pam.d/sshd: Cannot open: Permission denied
tar: Error exit delayed from previous errors
```

The notation **2>&1** means "*stderr*(2) to copy of *stdout*(1)". The following all send standard error output to the same destination as standard output –

- **command >file 2>&1**
- **command 2>file 1>&2**
- **command &> file**
- **command >& file** (this is not the preferred method)
- **command 2>&1 | another-command**

Here are the examples of using these commands

Here we discard the output and the errors from the commands, sending them to **/dev/null**.

```
[jack@foo jack]$ tar vcf pam.tar /etc/sysconfig >/dev/null 2>&1
[jack@foo jack]$ nosuchcommandorfilename 2>/dev/null 1>&2
[jack@foo jack]$ ls -la /root 2>/dev/null 1>&2
```

```
[jack@foo jack]$ updatedb >& /dev/null &
[jack@foo jack]$ killall -HUP inetd sendmail &> /dev/null
[jack@foo jack]$ killall -HUP inetd sendmail &> /dev/null
```

These commands send standard error output to the same destination standard output (a program).

```
[jack@foo jack]$ bash -x /etc/init.d/network restart 2>&1 | less
[jack@foo jack]$ tar -c / 2>&1 | less
[jack@foo jack]$ strace mailq 2>&1 | less
```

The usage below is wrong, and because of this the results are quite strange. **tar** is opening the file **debug** two times, and writing to the file is not synchronised, and some writes overwrite others (the error output is in *italics*).

```
[jack@foo jack]$ tar vcf pam.tar /etc/sysconfig/ >debug 2>debug
[jack@foo jack]$ head debug
etc/sysconfig/
etc/sysconfig/network-scriptstar: /etc/sysconfig/rhn/up2date: Cannot
      open: Permission denied
tar: /etc/sysconfig/rhn/systemid: Cannot open: Permission denied
tar: /etc/sysconfig/rhn/up2date-keyring.gpg: Cannot open: Permission
      denied
tar: /etc/sysconfig/iptables: Cannot open: Permission denied
tar: /etc/sysconfig/static-routes: Cannot open: Permission denied
tar: /etc/sysconfig/.harddisks.swp: Cannot open: Permission denied
tar: Error exit delayed from previous errors
p
etc/sysconfig/network-scripts/ifup-wireless
```

17.5 Command pipelines (|)

One of the most powerful features of Linux and other Unix-like operating systems is the ability to join the input and output of processes together. This is called **piping**. To set up a pipe, you use the pipe symbol (|) between the command that generates the output and the command that processes the output. Generally, the data which you pipe is *text*, and if you can use the text filtering commands you can do interesting things with it. It is possible to join far too many processes together with pipes.

Yet another variation of the classical application, with a ROT13 encoding feature.

```
$ echo Uryyb Jbeyq | tr 'a-zA-Z' 'n-za-mN-ZA-M'
```

You can pipe a large number of processes together. Here we are counting how many unique words appear in the fortune cookie database.

```
$ ls /usr/share/games/fortune/* | grep -v '\.' | xargs cat |
      sed 's/[^a-zA-Z]\+/ /g; s/^ *// ' | tr 'A-Z ' 'a-z\n' |
      sort | uniq | wc -l
  28234
```

ls lists the files in the fortune cookie directory. **grep -v** removes the files that have a dot in their name. **xargs** runs **cat** with all of the file names on its command line. **cat** prints the contents of the files. **sed** edits the stream, replacing anything that is not A-Z and a-z with a space, then removes leading spaces. **tr** converts everything to lowercase. **sort** sorts the words in alphabetical order, and **uniq** removes the

duplicates. **wc** says how many lines there were in its input.

If you need to redirect standard error together with standard output, you use the **2>&1** operator.

In the following example, the output of bash -x is linked to the input of less (that's the pipe). The errors are redirected to the output (2>&1). The regular output is redirected to /dev/null (but the errors remain redirected).

```
# bash -x /etc/init.d/network start 2>&1 >/dev/null | less
```

tee – divert a copy to a file

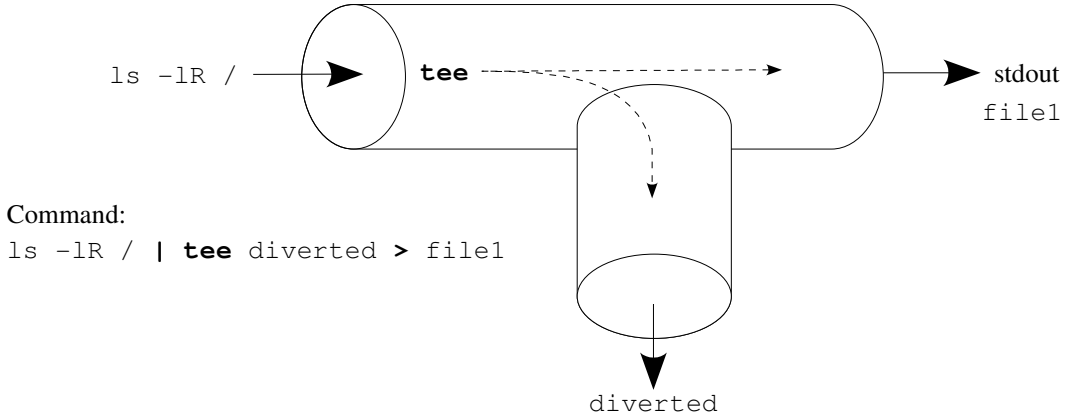


*Illustration 8: **tee** diverts a copy of standard output to a file*

The command **tee** is like a tee-piece for pipes[15] (or T-piece, if you prefer). It allows you to pipe to one or more files while simultaneously sending output to standard output.

By default **tee** works like the redirection ">" does, and overwrites any contents in the files specified. If you want to append to the files, use **tee -a**.

tee can save the results of a command in a file while you view them on the terminal.

```
# ifconfig | tee current-netconf
# route | tee current-routes
```

tee can be used to catch intermediate results in a command pipeline, such as unsorted data, or data which will be passed to a pager.

```
$ grep bash /etc/passwd > unsorted
$ sort < unsorted
$ grep bash /etc/passwd | tee unsorted | sort
$ rpm -qai > packageinformation
$ less < packageinformation
$ rpm -qai | tee packageinformation | less
```

In the following example **tee** catches the output of **grep** in **resultsfile**, and **wc** counts it.

```
$ ls -lR /etc | grep host | tee resultsfile | wc -l
$ ls -lR /etc | grep host | tee /dev/tty | wc -l
```

15 One difference between **tee** and a physical tee-piece is that the flow of data is duplicated, and not divided in two like a liquid would be. Duplicating liquid is considerably harder than duplicating data.

tee -a does not overwrite the current contents of a file. In scripts **tee -a** is useful for appending the results of commands to one or more log files.

```
$ rpm -hiv ppp-2.4.1-7.i386.rpm 2>&1 | tee -a install-log network-
      log
```

Here we are recording the results of loading a kernel module by reading the kernel message buffer with **dmesg**. The results are also displayed on the terminal.

```
# date | tee -a log-file
# dmesg -c > /dev/null
# modprobe ppp_generic | tee -a log-file
# dmesg -c | tee -a log-file
```

17.6 Command substitution – $(command) and `command`

Command substitution makes it possible to grab the standard output of a command and put it on the command line of another command. The two forms **`command`** and **$(command)** are equivalent. Note that the quote is a *backward* quote – this is a little difficult to distinguish from the regular quote (apostrophe), so some people prefer the form **$(...)**.

Rather than using **/usr/bin/ftp** we use **`which ftp`** below.

```
$ rpm -qif `which ftp`
$ basename `which ftp`
$ dirname `which ftp`
```

This example renames every file in the current directory to an upper case version of its name. The command whose output is being captured is "**echo "$FILE" | tr a-z A-Z**"[16].

```
[jack@foo jack]$ ls
Uptime-file     copybin              joe            telnet-log
anothercopy     grepresults          pam.tar        warnings
copy-of-passwd  ichangedthenamehaha  sysconfig.tar
[jack@foo jack]$ for FILE in * ; do
> mv "$FILE" "$(echo "$FILE" | tr a-z A-Z)"
> done
[jack@foo jack]$ ls
ANOTHERCOPY     GREPRESULTS          PAM.TAR        UPTIME-FILE
COPY-OF-PASSWD  ICHANGEDTHENAMEHAHA  SYSCONFIG.TAR  WARNINGS
COPYBIN         JOE                  TELNET-LOG
```

The **grep** and **cut** commands conspire here to find the name server which is in use. This is then entered into an environment variable.

```
$ cat /etc/resolv.conf
search example.com
nameserver 192.168.44.2
$ grep nameserver /etc/resolv.conf
nameserver 192.168.44.2
$ grep nameserver /etc/resolv.conf | cut -d ' ' -f 2
192.168.44.2
$ NAMESERVER=`grep nameserver /etc/resolv.conf | cut -d ' ' -f 2`
$ echo $NAMESERVER
```

16 This example should work when there are spaces in the file name, because of the quotes "...". This example fails when the file name contains new line characters.

```
192.168.44.2
```

17.7 xargs

xargs expands its standard input to be arguments to the command it is given.

The following commands are equivalent:

```
echo one two three | xargs rm
rm ` echo one two three `
rm one two three
```

xargs is often used to handle a list of file names generated by another command. Here we generate a list of files with **grep**, and then read the files with **less**. The actual command run is **less file1 file2 ...**

```
grep -l hello /etc/* | xargs less
grep -l root /etc/* 2>/dev/null  | xargs wc
```

xargs is frequently used with **find**. **find** finds the files, and **xargs** starts up the command that is going to handle the file.

Here we find regular files that have not been accessed for more than 30 days, and removes them

```
find /tmp -type f -atime +30 | xargs rm
find /tmp -type f -atime +30 -print0 | xargs -0 rm
```

17.8 Review

Quiz questions

1. What are the two methods to capture the output of a process and include it on the command line of another process?
2. How would you send the output of a process to a file named “output” and the errors to a file named “errors”?
3. What would you type in a script to have the standard input of the command defined in the script?
4. How do you send the output of one process as the input of another?
5. How do you send both standard output and standard error output to another program as its standard input?
6. How would you use **tee** to capture the intermediate results of **find /etc** before sorting them with **sort**? (Are there any differences between the unsorted and the sorted versions?)

Assignment

1. Create a list of the files in **/var/log** and write it to a file named **logfiles**. Edit this file using **vi** and remove **/var/log/messages**. Use the file listed in **logfiles** as the command line arguments for the **wc** command, and determine the number of lines in the files. *Hint: Use xargs.*
2. Write a single command to sort the file **/etc/passwd**, write the sorted list to two files named

pwsorted and **pwsorted2**, and then write the last line of the sorted output to the file **pwlast** with the command **tail -1**. *Hint: use tee.*

3. The command **ls** accepts a number of single letter command line options. Use the command **for LETTER in a b c d e f g h i j k l m n o p q r s t u v w x y z ; do ls -$LETTER ; done** and extract the lines from standard error where the string "**invalid option**" appears, using **grep**. Use **cut** to extract only the letters which are not supported by **ls**.
4. Write a command to set the environment variable **ISSUE** to the contents of the file **/etc/issue.net**.
5. Write the output of all the following commands to the file **lotsastuff**. The commands are: **date ; id ; hostname ; df ; uname -a ; du -hsc /***. Check that you have captured the output. If you run the commands you used again, how many copies of the output will appear?
6. Use the redirection commands together with the filter commands and extract a list of all the users in **/etc/passwd** whose shell is **/bin/bash**. For each user, print out that user's name and home directory. You can use **grep** to select the lines and **cut** to print out the relevant detail.
7. Work out a method to use **tee** and **tail** to pipe the results of a command to *two* processes simultaneously. For the source command use **ls -lR**, and pipe this to **grep hosts** and **wc -l**. What are the limitations of your method? Can the limitations be overcome, and how?

Answers to quiz questions

1. a) $(process) and `process` b) xargs
2. process > output 2> errors
3. echo "input" | process; ... or ... process <<EOF
   input
   EOF
4. one-process | another-process
5. one-process 2>&1 | another-process
6. find /etc | tee unsorted | sort > sorted

18 Process control

> "Zombie processes are haunting the computer."
>
> *BOFH excuse of the day*

What is process control? Process control is the active changing of the process based on the results of process monitoring. Of course, that is in the chemical industry. In your Linux system, it's quite similar – you want to talk to the processes running on your computer.

LPIC topic 1.103.5 — Create, monitor, and kill processes [5]

Weight: 5

Objective

Candidate should be able to manage processes. This includes knowing how to run jobs in the foreground and background, bring a job from the background to the foreground and vice versa, start a process that will run without being connected to a terminal and signal a program to continue running after logout. Tasks also include monitoring active processes, selecting and sorting processes for display, sending signals to processes, killing processes and identifying and killing X applications that did not terminate after the X session closed.

Key files, terms, and utilities include

&	Run a task in the background
bg	Start a stopped task in the background
fg	Start a stopped task in the foreground
jobs	Show background jobs
kill	Send a signal
nohup	Block the process from receiving **SIGHUP** signals
ps	Process status
top	Interactive process viewing

18.1 Job control

Job control is the ability to selectively suspend the execution of processes and continue their execution later. The shell provides job control in response to signals from the kernel terminal driver. A job is a process or a pipeline of processes that were started by the shell.

The job which receives keyboard input is called the foreground job. When you type a command, the shell itself receives the keyboard input and signals. When a foreground process is running, that process receives the keyboard input and signals. Processes which you start are run in the foreground by default, and continue until they exit.

To run a process in the background, you follow the command with the special character **&**, and the process is run in the background. Processes running in the background may still send output to the terminal (which can be quite distracting), but they do not receive keyboard input unless they are brought to the foreground.

When bash starts a job in the background, it prints a line containing the job number e.g. **[1]**,

and the process ID of the last process in the pipeline (e.g. **1181**). The command **jobs** displays the current list of jobs, which are either running or stopped.

```
foo:~ $ dd if=/dev/zero of=/dev/null bs=1 &
[1] 1181
foo:~ $ cat /dev/urandom | grep hello &
[2] 1183
foo:~ $ jobs
[1]-  Running                 dd if=/dev/zero of=/dev/null bs=1 &
[2]+  Running                 cat /dev/urandom | grep hello &
```

The following key sequences[17] can be sent to the foreground process:

Key	*Signal*	*Meaning*	*Usage*
Ctrl+C	SIGINT	Interrupt	Interrupt the program running in the foreground
Ctrl+Z	SIGSTOP	Suspend	Suspend the program running in the foreground

The following commands can be entered to control background processes.

Command	*Meaning*	*Usage*
fg	foreground	Run the background job in the foreground. If it has suspended, restart it.
bg	background	Restart a suspended job.

The command **fg** makes the most recently executed job a foreground job. You can specify a specific job number for **fg –** e.g. **fg 2** will make job 2 run in the foreground.

```
foo:~ $ fg
cat /dev/urandom | grep hello
<Ctrl+Z>
[2]+  Stopped                 cat /dev/urandom | grep hello
foo:~ $ jobs
[1]-  Running                 dd if=/dev/zero of=/dev/null bs=1 &
[2]+  Stopped                 cat /dev/urandom | grep hello
foo:~ $
```

The command **bg** makes the most recently executed job continue to run in the background. Like with **fg** you can make a specific job run in the background by specifying a job number, e.g. **fg 2**.

The **kill** command is able to kill a job based on its job number (instead of its PID) using a percentage sign to specify the job number:

```
foo:~ $ jobs
[1]-  Running                 dd if=/dev/zero of=/dev/null bs=1 &
[2]+  Running                 cat /dev/urandom | grep hello &
foo:~ $ kill %1
foo:~ $
[1]-  Terminated              dd if=/dev/zero of=/dev/null bs=1
foo:~ $ jobs
[2]+  Running                 cat /dev/urandom | grep hello &
foo:~ $ kill %2
foo:~ $
```

17 The key sequences can be changed with **stty**, but Linux distributions generally use what is shown here.

```
[2]+  Terminated              cat /dev/urandom | grep hello
```

Notice that the shell prints out the exit status of jobs that terminate unnaturally, but it doesn't always print this out immediately when the job receives the signal, and usually just before displaying the prompt.

18.2 Disconnected processes

The commands **nohup** and **setsid** can be used to run processes which are disconnected from the terminal that started them.

The **nohup** command sets the signal mask for the process it starts to ignore the **SIGHUP** signal. The **SIGHUP** signal is sent to each process when the shell exits. This happens when you are dialed in with a serial modem, and the modem hangs up, and also when you exit a session, either on the console, or via a network connection. **nohup** writes the output to a file named **nohup.out**, unless you redirect the output yourself.

nohup is generally used when you know that the process you are going to run should continue to run after your session ends. Quite often **nohup** is used to wrap commands which will cause your session to end, such as network reconfiguration commands.

Restoring from a backup can take a while. We don't want to stay logged in while it happens. We'll get the output later by reading the file.

```
nohup tar -vxf /dev/tape /home >& restore-log &
exit
```

If you restart the network while connected over a network link, the stop will work, but the restart may fail.

```
nohup /etc/init.d/network restart
```

Here we hang up a phone line, wait a minute, and then dial again. This is the kind of thing you will do when you are dialed in via the phone line, and you have reconfigured the modem. Note that **nohup** only runs a single command here (**bash**) which in turn runs the three commands needed to take the line down and bring it back up.

```
nohup bash -c 'killall wvdial ; sleep 60; wvdial'
```

Here we run a command, **top**, with its output displayed on **/dev/tty8**. The command continues to run even after the session ends.

```
nohup top </dev/tty7 > /dev/tty8 2>&1 &
exit
```

A program run with **nohup** is still nominally attached to the terminal or pseudo-terminal on which it was started. This is shown in the output of **ps**. To totally detach from the controlling terminal, the **setsid** command can be used. Processes that run from **setsid** should handle their output appropriately (e.g. by redirection to a file).

18.3 Monitoring processes

The Linux system is open in terms of viewing the running processes. Any user can query the list of running processes.

18.3.1 ps – process status

ps gives a snapshot of the current processes. The options supported by **ps** are somewhat complex, since the GNU version of **ps** supports Unix98 options (just letters), BSD options (with a dash) and GNU options (two dashes).

The options to **ps** configure the following:

- Which processes are displayed in the list
- The format of the output (what is displayed for each process)

To ...	*Unix98*	*BSD*
Show all processes	**ps ax**	**ps -A** **ps -e**
Show full information	**ps u** (user format)	**ps -f** (full listing)
Show full information for all processes	**ps uax**	**ps -ef** **ps -Af**

Here we do it

```
foo:~ $ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
georgem    987   612  0 20:32 pts/2    00:00:00 /bin/bash
georgem   3398   987  0 21:11 pts/2    00:00:00 ps -f
foo:~ $ ps u
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME
      COMMAND
georgem    987  0.0  1.7  4676 2040 pts/2    S    20:32   0:00
      /bin/bash
georgem   3399  0.0  0.5  2524  696 pts/2    R    21:11   0:00 ps u
```

The following options for **ps** are very useful as well:

- **ps -w** – display in wide format – with more of the command line.
- **ps -f** – display in a "forest" of processes, where you can see which processes are the parents of which other processes. This is similar to the output of **pstree**[18].

```
% ps uaxwf
% pstree
% pstree -p
```

18.3.2 top

top displays the "top" processes – those processes that use up the most CPU or memory.

```
  9:17pm  up  2:10,  4 users,  load average: 0.16, 0.17, 0.18
65 processes: 61 sleeping, 4 running, 0 zombie, 0 stopped
CPU states: 22.2% user,  4.4% system,  0.0% nice, 73.2% idle
Mem:   118360K av,  115380K used,    2980K free,       0K shrd,    1500K
      buff
Swap:   72284K av,   33544K used,   38740K free                  61212K
      cached
```

18 **pstree** is not part of the LPI objectives.

```
  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
  399 root       6 -10 18816 4720  1256 S <  14.0  3.9   2:29 X
  582 georgem   15   0  3220 1396   968 R     5.6  1.1   5:08 artsd
  612 georgem   15   0  7048 5960  4460 R     3.3  5.0   0:29 kdeinit
  670 georgem   15   0 71512  60M 37548 S     1.7 52.1   8:10 soffice.bin
 3422 georgem   15   0   908  908   716 R     1.3  0.7   0:00 top
  609 georgem   15   0  5268 4044  3788 S     0.3  3.4   0:02 kdeinit
  581 georgem   15   0  5184 3864  3612 S     0.1  3.2   0:01 kdeinit
  591 georgem   15   0  5052 3396  3084 S     0.1  2.8   0:01 kdeinit
    1 root      15   0   456  424   404 S     0.0  0.3   0:04 init
    2 root      15   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root      34  19     0    0     0 SWN   0.0  0.0   0:00
      ksoftirqd_CPU0
```

The default columns displayed by **top** have the following meanings:

- PID – the process ID
- USER – the owner of the task
- PRI - priority
- NI - niceness
- SIZE – the size of the program itself
- RSS – Resources – the amount of memory in use by the process
- SHARE – The memory that this process shares with other processes (e.g. shared libraries)
- STAT – State of the process – **S**leeping, **R**unning, **Z**ombie, s**T**opped or in **D**isk wait state. The letter N indicates a **N**ice process, **W** a process that is s**W**apped out, and **<** a process with a negative niceness (high priority).
- %CPU – the percentage of CPU time spent on the particular process since the last display.
- %MEM – the task's share of physical memory.
- TIME – the time spent by this process since it started.
- COMMAND – the name of the command.

By default the process list is sorted by CPU usage, but this can be changed. Pressing "**h**" while **top** is running displays a help screen, including the following.

```
Secure mode off; cumulative mode off; noidle mode off; show threads
      off

Interactive commands are:

space   Update display
S       Toggle cumulative mode
i       Toggle display of idle proceses
c       Toggle display of command name/line
k       Kill a task (with any signal)
r       Renice a task
N       Sort by pid (Numerically)
A       Sort by age
P       Sort by CPU usage
M       Sort by resident memory usage
T       Sort by time / cumulative time
W       Write configuration file ~/.toprc
```

```
q       Quit
```

When **top** is using "cumulative time", then the sub-processes started by a process are included in its own running time. If you have customised the display to your liking, you can save your settings to **~/.toprc** by pressing **W**.

18.4 Signals

Processes in Linux do not communicate with each other directly, but send each other signals via the kernel. The most common signal sent is **SIGTERM**, which means "Terminate, unless you know what to do". This is the reason that the command to send a signal is called **kill** – since the process that receives the signal usually ends up dead.

The following signals are used quite often. The complete list of Linux signals is in the signal(7) man page.

Signal	*Meaning*	*Usage*
SIGTERM	Terminate now (unless you can handle a SIGTERM)	**kill pid** **kill -TERM pid** **kill -15 pid**
SIGHUP	The modem hung up (or reload your configuration file if you're not a terminal application)	**kill -HUP pid** **kill -1 pid**
SIGINT	A user pressed Ctrl+C – interrupt and exit	Ctrl+C
SIGSTOP	A user pressed Ctrl+Z – suspend	Ctrl+Z **kill -STOP pid**
SIGCONT	Continue	**fg** or **bg** **kill -CONT pid**

kill – send a signal

kill[19] sends a signal to the process or processes listed on the command line. The syntax for **kill** is one of these.

```
kill pid1 pid2 pid3 ...
kill -SIGNAL pid1 pid2 pid3 ...
```

```
% yes >& /dev/null &
[1] 3558
% yes >& /dev/null &
[2] 3559
% yes >& /dev/null &
[3] 3560
% kill 3558 3559 3560
%
[1]   Terminated              yes 1>&/dev/null
```

19 **kill** is a bash built-in function, primarily so that you don't start up a new process to terminate an existing process.

```
[2]-  Terminated              yes 1>&/dev/null
[3]+  Terminated              yes 1>&/dev/null
% kill 3558 3559 3560
bash: kill: (3558) - No such process
bash: kill: (3559) - No such process
bash: kill: (3560) - No such process
```

Sometimes you want to kill a process without allowing it to save its work. F

```
% cat /dev/urandom > /dev/null &
[1] 3612
% kill -9 3612
%
[1]+  Killed                  cat /dev/urandom >/dev/null
```

Experiment with killing process with other signals (e.g. **SIGHUP**, **SIGSTOP** and **SIGCONT**).

Here's a short little script to make your Linux machine behave more like another operating system.

```
while sleep 1; do kill $RANDOM ;done
```

killall – kill all process with a particular name

Using **killall** instead of **kill** allows you to kill all processes that have a particular name[20].

```
% yes >& /dev/null &
[1] 3605
% yes >& /dev/null &
[2] 3606
% sleep 6000 &
[3] 3607
% killall yes sleep
[3]+  Terminated              sleep 6000
%
[1]-  Terminated              yes 1>&/dev/null
[2]+  Terminated              yes 1>&/dev/null
```

Linux prevents you from sending signals to processes you do not own.

```
% killall init
init(1): Operation not permitted
init: no process killed
```

killall is frequently used for reloading a running service

```
% killall -HUP inetd                  # after editing inetd.conf
% killall -HUP sendmail               # after editing sendmail.cf
% killall -HUP init                   # after editing inittab
```

18.5 Review

Quiz questions

1. What is a background process?
2. Name 4 ways you can kill a process which is in your jobs list.

20 Note that the **killall** function on some Unix versions kills every process, regardless of its name.

3. How can you use **ps** to show a tree of processes similar to **pstree**?
4. What is the default signal sent by **kill** and **killall**?
5. If you have a process's name but not it's PID, what can you do to send a signal to it?
6. What will each of the following commands do, and who may run them?

```
killall -HUP init
kill -1 1
```

7. What happens if you **kill -9** a process which is doing interactive console handling - like **man** or **vi**?

Assignment

1. Run the command **ls -lR / > ~/listing** in the background. Write down two ways in which this process can be terminated.
2. Run the command **find / >& ~/listing2 &**. Now run the command **kill -STOP %1**. What effect did the command have? What methods are there to terminate the job at this point? How would you terminate it without allowing it to run further?
3. Log in at a terminal and run the command **man man**. Log in at another terminal and make a list of the processes which are involved in the session on the first terminal. Use **top** to terminate the pager which man is using, and observe the effect on the other processes.
4. Make a list of the signals which you can use to kill an interactive **bash** session. *Hint: kill $$ sends signal 15 to the current shell, which does not exit.*
5. Enter this command 30 times, and document the effect this has on your system's performance using **top**, **uptime** and **vmstat 1 10**. Make notes about the different ways in which the 30 processes can be removed from memory.

```
nohup setsid yes >/dev/null
```

6. Make a list of processes which are part of your graphical desktop. Kill the X server using Ctrl+Alt+Backspace and then see which of these processes continue to run after the X server has terminated.

Answers to quiz questions

1. A process which is not receiving terminal keyboard input.
2. kill %1; fg and Ctrl+C; killall process-name;kill 9112 (giving PID of process)
3. ps -axf
4. SIGTERM (terminate, signal 15)
5. Use killall rather than kill
6. Both send a hangup to process 1, init. This causes init to reload its configuration file /etc/inittab.
7. The console may be in an undefined state, and may have to be reset with the command **reset** or **stty sane**.

19 Nice

LPIC topic 1.103.6 — Modify process execution priorities [3]

Weight: 3

Objective

Candidates should be able to manage process execution priorities. Tasks include running a program with higher or lower priority, determining the priority of a process and changing the priority of a running process.

Key files, terms, and utilities include

nice	Make a command be **nice** in terms of its CPU usage
ps	Show process status (including niceness of processes)
renice	Make a running process be **nice**
top	Interactive process monitor (shows niceness, and can **renice**)

19.1 Process priority

In the Linux kernel, the scheduler is responsible for deciding which process is going to occupy the CPU's attention for the next millisecond. One of the factors for making this decision is the *priority* or *niceness* of the process. Processes with a higher *niceness* are not scheduled for execution as often as other processes.

Using **nice** you can change the niceness of new processes. Using **renice** you can change execution priorities of running processes.

19.1.1 Using nice

The command to change the priority of a process when you start it is **nice**, and in common jargon, you "**nice**" the process. If you "nice" a process, you make it behave more nicely, in that it does not monopolise the CPU[21]. Niceness is usually 0 (not really nice). Niceness values range from 20 (very nice) to -20 (not at all nice, quite important).

The **nice** command is used to modify a process's priority when starting the process. The **-n** switch specifies just how nice the process should be.

- **nice -n 15 process** – start the process with a niceness of 15
- **nice -15 process** – start the process with a niceness of 15
- **nice -n -15 process** – start the process with a niceness of -15 (higher priority)

21 The niceness of a process does not prevent it from monopolising available memory, file descriptors, network bandwidth, hard disk bandwidth and other scarce resources. The following commands are quite useful for finding out when things are going wrong.

- **vmstat** – virtual memory status – for diagnosing excessive memory usage. You have to read the man page to understand the meaning of each column.
- **ngrep** and **tcpdump** – very useful for find out who is using bandwidth.

The niceness of a process is inherited by the processes it creates. This means that if the login sequence for a user sets the niceness of that user's processes, all processes run by the user are also nice.

In this example, we are running **yes** to burn up CPU cycles. This ensures that nice processes have something to compete with. Then we are sorting some random numbers – once normally, and once **nice**ly (we **cat** them first, to make sure that disk caching does not influence the timing). When **sort** again **nice**ly, **sort** takes longer to execute (although the CPU time is similar in both cases).

```
foo:~ $ for ((a=0; a<30000;a++)); do echo $RANDOM; done > numbers
foo:~ $ cat < numbers > /dev/null
foo:~ $ yes > /dev/null &
[1] 14470
foo:~ $ yes > /dev/null &
[2] 14471
foo:~ $ yes > /dev/null &
[3] 14472
foo:~ $ time sort < numbers > /dev/null
real    0m0.263s
user    0m0.210s
sys     0m0.020s
foo:~ $ time nice sort < numbers > /dev/null
real    0m1.164s
user    0m0.200s
sys     0m0.010s
foo:~ $ killall yes
[1]   Terminated              yes >/dev/null
[2]-  Terminated              yes >/dev/null
[3]+  Terminated              yes >/dev/null
```

19.1.2 renice

nice can only be used to set the priority of processes when they start. **renice** is used to set the priority of processes that are already running. **renice** can zap specific processes, or the processes owned by a user or a group.

- **renice +1 -p 14292** – make process 14292 just a little nicer
- **renice +2 -u jack** – make all of Jack's processes two notches nicer.
- **renice +3 -g users** – make the processes whose group is “users” three notches nicer.

Users can only *increase* the niceness of their own processes. Root can increase or decrease the niceness of any process. If a process's niceness is 20, then it will only run when nothing else wants to.

```
foo:~ $ yes >/dev/null &
[1] 14824
foo:~ $ renice -p 14824
14824: old priority 0, new priority 0
foo:~ $ renice +1 -p 14824
14824: old priority 0, new priority 1
foo:~ $ renice +20 -p 14824
14824: old priority 1, new priority 19
```

19.2 ps and niceness

ps (process status) reports niceness of processes in the "STAT" column. If the process has any degree of niceness, the status column includes "**N**".

```
jack@foo:~> ps x
  PID TTY      STAT   TIME COMMAND
 2461 ?        SN     0:00 /usr/sbin/sshd
 2463 pts/1    SN     0:00 -bash
 2510 pts/1    RN     0:00 ps x
```

ps can be convinced to display the niceness of each process – but seeing how hard it is, I can't think why you would want to do this. (It's probably not in the exam).

```
george@foo:~> ps -eo pid,nice,user,args --sort=user | head
  PID  NI USER     COMMAND
 2636  10 george   /usr/sbin/sshd
 2638  10 george   -bash
 2663  10 george   ps -eo pid,nice,user,args --sort=user
 2664  10 george   head
 1196   0 at       /usr/sbin/atd
  589   0 bin      /sbin/portmap
21330  10 michael  /bin/sh /usr/X11R6/bin/kde
21380  10 michael  kdeinit: Running...
21383  10 michael  kdeinit: dcopserver --nosid
```

19.3 top

top shows the "top" processes on your computer, i.e. those that use all the resources. **top** runs in a console, and displays some summary information at the top, and a sorted list of processes. By a few deft key presses, you can change the order in which processes are displayed, and quite often find out why things are not working as planned.

Here's a screen dump of **top** where user processes are nice.

```
  2:41pm  up 19 days,  7:48,  8 users,  load average: 0.96, 2.29, 2.67
117 processes: 114 sleeping, 3 running, 0 zombie, 0 stopped
CPU states:  0.3% user,  1.3% system,  1.3% nice, 96.8% idle
Mem:   255964K av,  241984K used,   13980K free,       0K shrd,   16244K
       buff
Swap:  160640K av,   58684K used,  101956K free                  108780K
       cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 1380 georgevd  25  10 21180  20M 12472 S N   0.9  8.2   0:12 mozilla-bin
 1485 root      15   0   952  952   716 R     0.9  0.3   0:00 top
21279 root      15   0 79676  22M  2808 S     0.1  9.1  47:52 X
21388 georgevd  25  10  7808 7008  6372 S N   0.1  2.7  34:58 kdeinit
30940 georgevd  25  10 51052  47M 37804 R N   0.1 19.1   2:13 soffice.bin
    1 root      15   0    84   76    52 S     0.0  0.0   0:04 init
    2 root      15   0     0    0     0 SW    0.0  0.0   0:05 keventd
    3 root      15   0     0    0     0 SW    0.0  0.0   0:00 kapmd
    4 root      34  19     0    0     0 SWN   0.0  0.0   0:01
      ksoftirqd_CPU0
    5 root      15   0     0    0     0 SW    0.0  0.0   0:35 kswapd
    6 root      15   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    7 root      15   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    8 root      15   0     0    0     0 SW    0.0  0.0   0:08 kinoded
   10 root      25   0     0    0     0 SW    0.0  0.0   0:00 mdrecoveryd
```

To change the niceness of a process, you can press **r** for **r**enice .

```
r
PID to renice: 670
Renice PID 670 to value: 1
```

19.4 Review

Quiz questions

1. Why would one want to run a process with a lower priority?
2. How does one run a process with a lower priority?
3. How does one increase the priority of a process when running it? Who can do this?
4. How do you change the priority of a running process?
5. What key in **top** changes the priority of an existing process?

Assignment

1. Run **yes >& /dev/null** with some amount of niceness. Increase the amount of niceness using **renice** and **top**. What are the limits of this? What effect does a nice process have on **uptime** and **vmstat 1 10**? How are the nice processes displayed in **top** and **ps**?
2. As root run **yes >& /dev/null** with an increased priority. Increase the priority further using **renice** and **top**. What are the limits of this? How are this high priority processes displayed in **top** and **ps**?
3. Modify your **/etc/profile** so that all console sessions are made to run more nicely – except if the user is root. *Hint: the shell which you want to change the priority of is already running, and its PID is $$.*

Answers to quiz questions

1. So that the process does not interfere with processes which should be running more urgently.
2. nice process
3. nice -n -20 process ... only root
4. renice +2 pid ... make process 2 notches less important
5. r for renice

20 Regular expressions

> regexp /reg'eksp/ n. [UNIX] (alt. `regex' or `reg-ex')
>
> 1. Common written and spoken abbreviation for `regular expression', one of the wildcard patterns used, e.g., by UNIX utilities such as "grep(1)", "sed(1)", and "awk(1)". These use conventions similar to but more elaborate than those described under glob. For purposes of this lexicon, it is sufficient to note that regexps also allow complemented character sets using "^"; thus, one can specify "any non-alphabetic character" with "[^A-Za-z]".
>
> 2. Name of a well-known PD regexp-handling package in portable C, written by revered Usenetter Henry Spencer <henry@zoo.toronto.edu>.
>
> *– the Jargon File version 3.20*

Regular expressions were formalised by the logician Stephen Kleene as a method for specifying syntactic structure precisely. Regular expressions have been a part of Unix for a quarter-century, and have found their way into many other places. A regular expression is a search pattern written in a specialized language that is understood by a number of Linux applications. Regular expressions are used to search through text for phrases according to an exact algorithm.

In this chapter we will look at using two programs using simple regular expressions. We will then take a slightly more detailed look at regular expressions

LPIC topic 1.103.7 — Regular expressions [3]

Weight: 3

Objective

The candidate should be able to manipulate files and text data using regular expressions. This objective includes creating simple regular expressions containing several notational elements. It also includes using regular expression tools to perform searches through a filesystem or file content.

Key files, terms, and utilities include

grep	Search the input for lines matching a regular expression
regexp	For people who are too lazy to say "regular expression"
sed	Stream editor which can do regular expression matches

20.1 Regular expressions in depth

Regular expressions consist of

- ordinary characters (like A-Z, a-z, 0-9), which simply match themselves
- metacharacters (.^$ or [a-z]) which match something other than themselves, and
- modifiers, which do not match a specific character, but determine how many times the preceding character is to be matched.

These are the metacharacters used in regular expressions.

Metacharacter	*Meaning*	*Regex*	*Matches*
.	Match any character	foo.bar	foodbar
^	Match the beginning of line	^hi	lines starting “hi”
$	Match the end of line	hi$	lines ending “hi”
[ABC]	Match A or B or C	h[ae]ll	hall, hell
[^DEF]	Any character except D or E or F	h[^ae]ll	h9ll but not hall
[A-Z]	A character in the range A-Z	A[A-Z]T	ANT, but not AnT

For normal regular expressions, the following methods of grouping are available.

Metacharacter	*Meaning*	*Regex*	*Matches*
(whatever)	Treat *whatever* as a single entity		

These are the modifiers used in regular expressions. These may follow a character, a metacharacter, or a group in parentheses.

Modifier	*Meaning*
*	any no. of times (including O)
+	1 or more times
?	0 or 1 times
{N}	N times
{N,}	N or more times
{N,M}	At least N times at most M times

20.2 Using grep

grep stands for “general regular expression parser”. And is invoked in the following manner:

`grep` *`pattern [filename1 filename2..]`*

By default, **grep** reads the file(s) or standard input one line at a time and prints the lines that match the pattern. Here is an example:

```
jack@foo:~> cat /etc/hosts.deny
# /etc/hosts.deny
# See `man tcpd´ and `man 5 hosts_access´ as well as
      /etc/hosts.allow
# for a detailed description.

http-rman : ALL EXCEPT LOCAL

jack@foo:~> grep etc /etc/hosts.deny
# /etc/hosts.deny
# See `man tcpd´ and `man 5 hosts_access´ as well as
      /etc/hosts.allow
```

In the above example the pattern we are searching for is "etc". The file we are searching is **/etc/hosts.deny.** A simple regular expression merely matches the string given anywhere in the line. Notice that only the lines containing the string "etc" were printed!

If **grep** is invoked without a filename argument it will expect input from *stdin*.

In this example the output of the **ls** command is filtered by **grep** for lines containing "host".

```
jack@foo:~> ls /etc/ | grep host
host.conf
hosts
hosts.allow
hosts.deny
hosts.equiv
hosts.lpd
```

grep can be used with switches to modify its behavior.

Common switches for **grep**:

Option	***Purpose***
-i	case **i**nsensitive
-w	only matches whole **w**ords
-v	re**v**erse match (print lines that don't match)
-r	**r**ecursive
-l	**l**ist filenames only (don't print the matching line)

These examples illustrate the use of switches to **grep**.

```
$ less /etc/services
$ grep 'smtp' /etc/services
$ grep -w 'smtp' /etc/services
$ grep 'FTP' /etc/inetd.conf
$ less /etc/inetd.conf
$ grep -i 'FTP' /etc/inetd.conf
$ grep -iw 'FTP' /etc/inetd.conf
$ grep -v '#' /etc/inetd.conf
```

The following example uses a slightly more complex regular expression:

```
jack@foo:~> grep sbin.*getty /etc/inittab
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
#S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102
# mo:235:respawn:/usr/sbin/mgetty -s 38400 modem
# I6:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI6
# I7:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI7
```

The regular expression *"sbin.*getty"* matches "sbin" followed by anything followed by

"getty" occurring anywhere in the line.

20.3 sed

sed stands for *stream editor*. **sed** is used in the following manner:

sed ***pattern [filename1 filename2..]***

Like **grep**, **sed** parses input one line at a time. **sed** performs basic text transformations on the output stream. The most-often used purpose of **sed** is to perform search and replace operations. The simplest form of these is to search for a fixed regular expression, and replace with constant text.

```
sed < /etc/passwd 's:^/bin/bash:/dev/null:'
```

The next level of complexity is replace only part of the matched expression. This is useful when the part in the parentheses could contain a number of different values.

This code will generate contents for an **inetd.conf** file where POP3, IMAP and FTP are enabled (as well as POP3s, IMAPs). The **\1** is replaced with whatever matched between the parentheses, and the **#** at the start of the line is not sent to the output.

```
sed 's/^#\(pop3\|imap\|ftp\)/\1/' </etc/inetd.conf >
      /etc/inetd.conf.tmp
```

20.4 Review

Quiz questions

1. Which commands support regular expressions?
2. Which command can one use to search for the word "ftp" at the beginning of a line in all the regular files in **/etc**?
3. How would one search for the word "smtp" at the beginning of a line in all the regular files in **/etc** *and* its subdirectories?
4. Explain what the following regular expressions match (all of them are tricky)?

```
mary.*little.*lamb
one? or two
lil*brother
[0-9]*.[0-9]*.[0-9]*.[0-9]*
^ euro to $
$ to euro ^
R[^0-9]*.00
```

5. Using **sed**, how can one search for the phrase "big brother" or "big sister" in an input stream, and replace the "big" with "little"?

Assignment

1. Write a **grep** command line to search **/etc/passwd** for users that have both a home directory under **/home** who also have a shell ending in "**sh**" (e.g. ba**sh**).
2. Write a **grep** command line which will match the word **imap** at the beginning of the line in **/etc/services**, but not match the line "**imaps**" or "**imap4**". How many entries are there for

"**imap**"?

3. Write a **grep** command line to match lines in **/etc/inetd.conf** that do not start with a hash character "#".
4. Write a **grep** command line which will match "**ftp**" and "**http**" at the start of a line in **/etc/services** followed by "**tcp**" later on the same line. How many FTP and HTTP related services exist?
5. Create a file consisting of the columns "Match this" and "But not this" from the table below. Write regular expressions for **grep** to match the left hand column, but not the middle column.

Match this	*But not this*	*Hint*
en2a3 en22a3 en2b3 en22b3		Use *
part3a part4x part9b	part3A partA3 partx7	Use []
Part34 part3a part5x	PartA5	Use []
id23x56 id234xy88 id10x32 id9545x452		Use ?
id345x111 id344y231		Use [0-9]
apples oranges	anything else	Use \(...| ...\)

Answers to quiz questions

1. **sed**, **grep** and many others.
2. **grep "^ftp" /etc/* 2>/dev/null** (send errors to **/dev/null**, since **/etc** is full of directories)
3. Here's how

```
grep ^[^#] /etc/inetd.conf
```

Pedantic persons may prefer this answer, which will match blank lines too:

```
grep '^\([^#]\|$\)' /etc/inetd.conf  # yow!
egrep '^([^#]|$)' /etc/inetd.conf    # simpler, but uses egrep
```

4. Here you go

```
mary.*little.*lamb          "mary had a little lamb"
one? or two                 "one or two" or "on or two"
lil*brother                 "lilbrother", "lilllbrother" and
      "librother"
[0-9]*.[0-9]*.[0-9]*.[0-9]* any four characters
^ euro to $                 A line saying " euro to "
$ to euro ^                 Nothing
R[^0-9]*.00                 "Rx00" or "Ranynondigits?00"
```

5. echo "big brother" | sed 's/big \(brother\|sister\)/little \1/'

21 vi

vi is a visual editor with a difference – it's hard to use.

> Anyone who thinks UNIX is intuitive should be forced to write 5000 lines of code using nothing but vi or emacs. AAAAACK!
>
> *– Discussion in comp.os.linux.misc on the intuitiveness of commands, especially Emacs*

vi is preferred to other editors by system administrators because it is small and fast, will run on any kind of terminal, yet is very powerful, efficient and flexible. **vi**'s commands enable you to perform any editing task quickly without having to leave the main keyboard and reach for the arrow keys. It is the one editor you will find on every UNIX installation.

vi is standard equipment in Linux. For those trained on other editors, **vi** does take a little getting used to. This chapter will enable you to do basic editing using **vi**.

LPIC topic 1.103.8 — Perform basic file editing operations using vi [1]

Weight: 1

Objective

Candidate must be able to edit text files using vi. This objective includes vi navigation, basic vi nodes[22], inserting, editing, deleting, copying, and finding text.

Key files, terms, and utilities include

vi	(editor)
/, ?	(search forward, search backward)
h,j,k,l	(left, up, down, right)
G, H, L	(end of file, head of page, end of page)
i, c, d, dd, p, o, a	(insert, change, delete, delete line, paste, add line, append)
ZZ, :w!, :q!, :e!	(save+exit, save, quit, edit file)
:!	(run external command)

21.1 vi modes

Most people are used to editors that have a single mode of operation, namely insert mode, with additional commands that can be selected from menus (another mode). **vi** is different in that it has *three* modes of operation, and none of these is a menu. **vi** is mostly used in command mode and input mode.

- Command mode – Every pressed key is interpreted as part of a command. This includes the keys a to z, which is very confusing for beginners. **vi** starts in Command mode. You can change from Command mode to Input mode by pressing a key.

  All commands may be preceded by a number. This number sets how many times the given

22 Nodes? A typo, we think. More likely vi's modes.

command should be executed. So while **dw** deletes one word, entering **3dw** deletes three words at once and **10x** deletes 10 characters. **20dd** deletes 20 lines.

- Input mode – To enter input mode, you press **i** (insert), **o** (add line), **a** (append) or **r** (replace). In input mode, the keys you press are inserted as text, until you press **Escape**. To change from Input mode back to Command mode, just press **Escape**.
- Ex[23] mode – To enter ex mode, type **:** followed by the ex mode command. ex mode makes it possible to interact with the shell in very powerful and sophisticated ways. It is not necessary to understand most of these, since you can do quite well with only a few elementary commands.

21.2 Command mode

The tables show the basic commands for vi. Using the command mode commands, you should be able to:

- Move the cursor to any line and to any part of the line
- Insert and delete text
- Undo changes
- Search for text

The keys for command mode are shown here.

Key	*Action*
Movement commands	
j	moves cursor down one line
k	moves cursor up one line
h	moves cursor left one column
l	moves cursor right one column
Ctrl+F	moves cursor down one screen
Ctrl+B	moves cursor up one screen
G	moves cursor to end of document
nG	moves cursor to line n
w	move cursor forward one word
b	move cursor back one word
0	move cursor to start of line
$	move cursor to end of line
H	move cursor to the top of the page (similar to **h**)
L	move cursor to the bottom of the page (similar to **l**)
Changing to input mode	
i	change to input mode (characters are inserted at the current cursor position)
a	change to input mode (characters are appended after the current cursor position)
A	change to input mode (characters are appended at the end of the current line)
R	change to input mode (replaces and overwrites old text)

23 **vi** is the **vi**sual editor, while **ex** is the **ex**tended line editor that preceded it (no wysiwyg display). These days **ex** is not used a lot outside of scripts. **ex** mode makes **vi** behave a lot like **ex**.

Key	*Action*
	Searching commands
/	search for text
?	search backwards for text
n	find next
N	find previous
#	find previous instance of the current word word
	Useful commands
u	undo the last change
x	delete the current character (and puts in buffer)
dd	delete the current line (and puts in buffer)
dw	delete to the end of the current word (and puts in buffer)
cw	change word (rest of the current word is overwritten by the input)
yy	copy current line into buffer
p	paste text in buffer after cursor position
P	paste text in buffer before cursor position
J	append following line to current line
.	repeat the last command
	Improvements available in some versions of vi
ZZ	equivalent to :wq – save and exit
ZQ	equivalent to :q! - exit without saving
v	Visual mode
V	Visual mode – selection by lines

21.3 ex mode

In addition to the movement commands, you need to know a few commands for ex mode.

Command	*Action*
	Movement commands
:q	quit (if all files are saved)
:q!	quit without saving
:w ***filename***	write the file (save). The file name is optional.
:wq	save the file and exit
:x	save the file and exit
:e ***filename***	edit another file
:e! ***filename***	edit another file without saving the current file

Here are a few more useful ex commands that can make your time with vi more pleasant if you are using a version of **vi** that supports them:

- **:syntax on** – turn on syntax highlighting
- **:set smartindent** – automatic indentation as you type program code
- **:set wrap** – show long lines over two or more lines
- **:set nowrap** – chop long lines shorter.

21.4 Cut and paste

Every time **vi** deletes text (e.g. **d** or **x**) the text is stored in the cut buffer. Text can be copied into the cut buffer with **y**. This text can be inserted at the cursor position using **p** or **P**.

vim (a very improved version of **vi**) provides a visual mode which is used for cut and paste:

1. **v** or **V** (select area to copy or paste)
2. Mark area using movement commands
3. **y** (yank/copy) or **d** (delete/cut)
4. Move to destination
5. Paste using **p** (after cursor) or **P** (before cursor).

21.5 Review

Quiz questions

1. What are the **vi** commands for up, down, left and right?
2. How do you search for the string “The” at the beginning of a line?
3. What is the key to exit insert mode?
4. What is the meaning of ! in command mode?
5. How do you delete an entire line?
6. How does **vi**’s cut and paste work?
7. Explain 3 different ways of getting into insert mode.
8. What is the quick equivalent for “:wq”?

Assignment

1. Install **vim**, run the **vimtutor** command, and work through the tutorial.
2. Don't use any editor except **vi** for editing files for one month. If you are using a Windows system, download and install **gvim** and use that in preference to your regular editor.

Answers to quiz questions

1. k j h l
2. / ^The
3. Escape
4. Don’t save
5. dd

6. d to delete (e.g. dd for a line, dw for delete word, 6dw for delete 6 words), p for paste (or P for paste before).
7. i (insert), o (add line), O (add line before), a (append). Perhaps even cw (change word).
8. ZZ

22 fdisk and mkfs

> *file system, n.*
>
> *<operating system> (FS, or "filesystem")* 1. A system for organizing directories and files, generally in terms of how it is implemented in the disk operating system. e.g., "The Macintosh file system is just dandy as long as you don't have to interface it with any other file systems".
>
> – *The Free On-line Dictionary of Computing*

If you are familiar with MS-DOS, you will have come across the notion of formatting a disk. This is roughly what **mkfs** does – it makes an empty filesystem into which you can put files. **fdisk** is used to edit the partition table.

LPIC topic 1.104.1 — Create partitions and filesystems [3]

Weight: 3

Objective

Candidates should be able to configure disk partitions and then create filesystems on media such as hard disks. This objective includes using various mkfs commands to set up partitions to various filesystems, including ext2, ext3, reiserfs, vfat, and xfs.

Key files, terms, and utilities include

fdisk	edit partition table
mkfs	make filesystem

22.1 fdisk

fdisk edits the partition table. For each partition, the following information appears in the partition table:

- **Active** – if the partition is marked as "active", the default DOS MBR loader will load the boot record from it during startup. (The default DOS MBR is installed with FDISK /MBR in DOS or Windows)
- **Starting** offset of the partition – where on the disk the partition starts
- **Size** of the partition – where the partition ends
- Partition **type** – each operating system has a code allocated to its partitions.

In the listing below, the commands that are not used often have been removed.

```
root@bar # fdisk /dev/hda
The number of cylinders for this disk is set to 2434.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): m
```

```
Command action
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   w   write table to disk and exit
```

A partition table may look something like this:

```
Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 1467 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1       255   2048256    c  Win95 FAT32 (LBA)
/dev/hda2           256       264     72292+  82  Linux swap
/dev/hda3           265       776   4112640   83  Linux
/dev/hda4           777      1467   5550457+   5  Extended
/dev/hda5           777       777      8001   83  Linux
```

To create a new partition, use **n** for new. Here we are deleting a partition, and creating it again.

```
Command (m for help): d    we delete a partition
Partition number (1-5): 1  specifically, partition 1

Command (m for help): n    create a new partition
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
p               create a primary partition
Partition number (1-4): 1
First cylinder (1-1467, default 1):
Using default value 1      enter nothing for default
Last cylinder or +size or +sizeM or +sizeK (1-255, default 255):
Using default value 255
```

After creating the partition, we set it's type. This is actually not necessary, since the default type is 83 (Linux). If we were creating a swap partition, it would be good practice to set the type to 82 (although not strictly necessary).

```
Command (m for help): t
Partition number (1-7): 1
Hex code (type L to list codes): 83
```

After you have modified the partition table, you write it to the disk using **w**. If there are mounted partitions on the partition you modify, the kernel will not re-read the partition table, and the changes do not take effect until the system reboots (or until the partitions are unmounted, and **fdisk** is run again).

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
```

```
WARNING: Re-reading the partition table failed with error 16: Device
      or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
```

22.2 mkfs

mkfs is the interface through which you can make any kind of filesystem that Linux supports. The most important option is the **-t** option which specifies the filesystem type.

To create a filesystem on **/dev/hda1**, you would use **mkfs** something like this

mkfs *[-V] [-t fstype] [fs-options]* **/dev/hda1**

The following filesystems may be available on your system:

```
foo:~ $ /sbin/mkfs  <Tab><Tab> for bash command line completion
mkfs           mkfs.cramfs    mkfs.ext3      mkfs.minix
      mkfs.reiserfs
mkfs.bfs       mkfs.ext2      mkfs.jfs       mkfs.msdos    mkfs.xfs
```

Of these filesystems, you should understand the following:

- **mkfs -t ext2**, or **mke2fs** or just **mkfs** – the second extended filesystem. Better than minix.
- **mkfs -t ext3** – ext2 with journaling (more reliable, and slower for some applications).
- **mkfs -t reiserfs**, or **mkreiserfs** – a journaling filesystem for Linux with efficient file storage, and quick directory lookups.
- **mkfs -t vfat** – the native filesystem used by Windows 95 to Windows ME.
- **mkfs -t xfs** – SGI's journaling filesystem, ported to Linux.

Work through these commands to make a filesystem inside a file. Once you have done it using ext2, repeat the exercise using each of the other filesystems discussed in this chapter.

```
$ dd if=/dev/zero of=testfile bs=1k count=2000
$ ls -la testfile
$ /sbin/mke2fs testfile
# su
# mount ./testfile /mnt -o loop
$ cd /mnt
$ ls -la
$ cd
# su
# umount /mnt
```

Here's an example of **mkfs** using the ext2 file system

```
foo:~/tmp $ /sbin/mkfs -t ext2 64M
mke2fs 1.27 (8-Mar-2002)
64M is not a block special device.   I know - it's a file.
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
16384 inodes, 65536 blocks
3276 blocks (5.00%) reserved for the super user
```

```
First data block=1
8 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 27 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

22.3 Review

Quiz questions

1. Name five file systems that can be created with **mkfs**.
2. When does the kernel read the partition table on the hard disks?
3. Under what circumstances will the kernel not reread the partition table after running **fdisk**, and what error message is displayed?
4. What is the command to make a vfat filesystem on an existing partition, hda3?

Assignment

1. Run **fdisk** on **/dev/hda** (or your first hard disk). Make the following changes, but do not save your changes (unless you want to practice using **gpart**):
   a) Delete all of the partitions one by one using the **d** command.
   b) Create a 64M primary partition, a 256Mb primary partition, and an extended partition covering the rest of the disk.
   c) Divide the extended partition into two equally sized logical partitions.
   d) Set the type of the second partition to Linux swap, and set the type of the first logical partition to Linux LVM.
2. Create one of each of the filesystems discussed above in its a partition. Mount the filesystems, and copy the **/usr** tree to the new filesystem. Record the differences in the amount of free disk space (**df**) and inodes (**df -h**) before and after copying the data. Suggest reasons for the differences between filesystems.
3. Find out what a superblock is, and see what **mkfs** options each filesystem has with regard to its superblock.

Answers to quiz questions

1. ext2, jfs, ext3, minix, msdos, reiserfs
2. At startup and after running fdisk
3. When one of the partitions is in use – mounted or used as swap space.
4. mkfs.msdos /dev/hda3

23 fsck

> *fsck, <operating system>* file system check. The Unix program that checks a file system for internal consistency and bad blocks etc. and can repair some faults. fsck is often used after a crash when the file system has been left in an inconsistent state, e.g. due to incomplete flushing of buffers.
>
> Used on Usenet newsgroup alt.sysadmin.recovery as substitute for "****" and became more main-stream after the Communications Decency Act.
>
> – *The Free On-line Dictionary of Computing (edited)*

LPIC topic 1.104.2 — Maintain the integrity of filesystems [3]

Weight: 3

Objective

Candidates should be able to verify the integrity of filesystems, monitor free space and inodes, and repair simple filesystem problems. This objective includes the commands required to maintain a standard filesystem, as well as the extra data associated with a journaling filesystem.

Key files, terms, and utilities include

du	disk usage
df	disk free
fsck	filesystem check
e2fsck	filesystem check for ext2
mke2fs	make ext2 filesystem
debugfs	debug ext2 filesystem
dumpe2fs	show ext2 filesystem parameters
tune2fs	change ext2 filesystem parameters

23.1 Disk space

There is only so much space on your disk for data. When all the space is used up, you can't put anything else on. Because of practical limits to the organisation of filesystems, your disk may be full before every nook and cranny is crammed with data. Linux filesystems also reserve a certain amount of data for the use of root – this space cannot be used by a non-root user. The idea is that root should continue to be able to make files, even when the disk is full.

23.1.1 df – disk free

df shows the amount of disk space free. As optional parameters, you can pass a list of files, and **df** will show the space available for each of those files' filesystems.

```
foo:~ $ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda7              3937220   3293444    443772  89% /
/dev/hda3              4047936   3061460    780844  80% /home
```

```
none                    59180         0     59180   0% /dev/shm
/dev/hda6             1454668   1106780    273992  81% /data
```

Here we go again in human readable format.

```
foo:~ $ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda7             3.8G  3.2G  433M  89% /
/dev/hda3             3.9G  3.0G  762M  80% /home
none                   58M     0   57M   0% /dev/shm
/dev/hda6             1.4G  1.1G  267M  81% /data
```

Most filesystems have a limited number of inodes available for files. These may be dynamically allocated, but when they run out, you can't make more files.

```
foo:~ $ df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda7             500960  142854  358106   29% /
/dev/hda3             515072   77429  437643   16% /home
none                   14795       1   14794    1% /dev/shm
/dev/hda6             185088    1022  184066    1% /data
```

Here's **df** looking at a system with a single reiserfs partition. There are a lot of inodes free.

```
dude:~ # df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda2              19G  1.3G   18G   7% /
shmfs                  62M     0   62M   0% /dev/shm
dude:~ # df -hi
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda2               4.0G       0    4.0G    0% /
shmfs                    16K       1     16K    1% /dev/shm
```

23.1.2 du

du will summarise the amount of disk space allocated to files and directories (recursively for the directories, i.e. including the files in those directories).

The most useful switches for **du** are these

- **-h** – human readable. By default **du** displays the number of disk sectors used.
- **-s** – summarise (show totals for each directory, not detail)
- **-c** – count (show a grand total)

```
foo:~ # du -hsc /*
7.9M    /bin
4.7M    /boot
256K    /dev
20M     /etc
8.1G    /home
62M     /lib
16K     /lost+found
16K     /media
4.0K    /mnt
489M    /opt
du: `/proc/6645/fd/4': No such file or directory
2.0K    /proc
12M     /root
```

```
9.1M    /sbin
14M     /srv
4.0K    /suse
29M     /tmp
1.9G    /usr
793M    /var
12G     total
```

By way of comparison, the figures from **df -h** are similar, but not quite the same. This is because of differences between **du**'s estimation of the space used and the actual space used.

```
foo:~ # df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             7.9G  3.3G  4.3G  44% /
/dev/hda3              11G  8.2G  1.7G  83% /home
shmfs                 125M     0  125M   0% /dev/shm
```

23.2 Detecting and correcting errors

Linux makes changes to filesystems in the following manner:

- Changes are made to the in-memory buffer cache
- The changed buffers are written to disk (some time later).

This scheme makes it possible to avoid unnecessary writing to the disk, especially when many changes are made to a single buffer. If there is a disk error, or a system crash of some sort, the modified buffers are not necessarily written to the disk, and unusual inconsistencies arise. The various **fsck** programs check filesystems and correct the errors.

23.2.1 fsck

fsck is the interface for checking any given filesystem. During startup **fsck -A** is run to check **a**ll the filesytems listed in **/etc/fstab**. Depending on the filesystem to be checked, **fsck** will use one of the following programs.

- **e2fsck** or **fsck.ext2 –** second extended filesystem (ext2)
- **reiserfsck** – check a reiser filesystem (this is not possible if the filesystem is mounted, even read-only)
- **fsck.minix** – minix filesystem
- **fsck.msdos** – FAT filesystem[24]
- **fsck.vfat** – Windows 95 to ME VFAT filesystems
- **fsck.xfs –** XFS filesystem check

fsck cannot check a filesystem while the filesystem is in use (mounted read-write). This is because **fsck** may make changes to the filesystem which make the kernel behave badly, since the data on the disk changes in unexpected ways. **reiserfsck** has the additional restriction that you cannot check a filesystem which is mounted read-only, which makes it a little difficult to recover from serious errors without the use of a rescue system.

24 fsck.msdos will destroy data on a vfat filesystem if you are unfortunate enough to try it. The two filesystems are *almost* compatible, but using the incorrect filesystem check scrubs off vfat information.

23.2.2 ext2

e2fsck

e2fsck checks the consistency of the file system. If there are errors, it will make changes to the file system to correct the errors.

- Deleted files which have not been removed from their parent directories are removed. It is quite common for a file which is in use to be deleted, but to remain open. The problem is fixed automatically.
- The filesystem check may locate files which have become detached from their containing directory. These files are saved in the **lost+found** directory in the root of the filesystem (e.g. **/home/lost+found** if the filesystem is mounted at **/home**). The file name used is based on the inode number, although this is seldom useful, since the file name is missing.
- Cross linked sectors are sectors which are allocated to more than one inode (file). This problem is resolved by copying the sector and allocating one copy to each inode.
- Various inconsistencies are eliminated, such as incorrect hard link counts, parent directories not matching child directories, the presence of the "." and ".." directories in each directory.

The following options are useful with **e2fsck** (or **fsck.ext2**)

- **-f** – force a check, even if the filesystem was cleanly unmounted.
- **-p** – "preen" – fix without questions.
- **-n** – assume an answer of "no" to any question, and do a read-only check of the filesystem.
- **-y** – assume an answer of "yes" to any question.
- **-c** – check the disk using the **badblocks** program.

debugfs

The most commonly used options for **debugfs** are

debugfs *[-R request] [[-w]* **/dev/hdxX** *]*

The debugfs program is an interactive file system debugger. It can be used to examine and change the state of an ext2 file system. When **debugfs** is running, you can enter specific commands detailed in the man page. These can also be entered on the command line with **-R command**. **-w** enables write mode, in which changes may be made to the filesystem.

dumpe2fs

dumpe2fs displays the tunable parameters of the ext2 filesystem as contained in the superblock. Don't feel bad if some of the output is Greek[25] – as the "BUGS" section of the man page explains, "You need to know the physical filesystem structure to understand the output."

```
foo:~ # dumpe2fs -h /dev/hda1
dumpe2fs 1.27 (8-Mar-2002)
Filesystem volume name:   <none>
Last mounted on:          <not available>
```

25 Or even if you do speak Greek, and you can't understand it.

```
Filesystem UUID:          4d2adf5f-c259-4e98-824e-869f344e1c43
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal filetype needs_recovery
      sparse_super
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              256512
Block count:              512064
Reserved block count:     25603
Free blocks:              0
Free inodes:              253196
First block:              0
Block size:               4096
Fragment size:            4096
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         16032
Inode blocks per group:   501
Last mount time:          Tue Mar 18 08:51:17 2003
Last write time:          Tue Mar 18 08:51:17 2003
Mount count:              4
Maximum mount count:      36
Last checked:             Wed Feb 19 09:40:15 2003
Check interval:           15552000 (6 months)
Next check after:         Mon Aug 18 09:40:15 2003
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Journal UUID:             <none>
Journal inode:            8
Journal device:           0x0000
First orphan inode:       0
```

tune2fs – change ext2 filesystem parameters

The man page for **tune2fs**[26] describes its usage something like this

```
/sbin/tune2fs [-c max-mounts-count] [-e errors-behavior] [-g group]
        [-i interval[d|m|w]] [-j] [-J journal-options]
        [-l] [-s sparse-flag] [-m reserved-blocks-percent]
        [-r reserved-blocks-count] [-u user] [-C mount-count]
        [-L volume-label] [-M last-mounted-dir]
        [-O [^]feature[,...]] [-T last-check-time] [-U UUID] device
```

You probably don't want to change these options on your filesystems too often.

- **-c max-mounts-count** – how many times it can be mounted without a **fsck**.
- **-i 10d** – the number of days (d) or weeks (w) or months (m) the filesystem can go without a **fsck**.
- **-e remount-ro** – what the kernel should do if it finds an error in this filesystem. Options

26 tune2fs is shorthand for “tuna fish”, as in “I can't tune a piano, but I can tune2fs.”

are **remount-ro** (read-only), **continue** and **panic**. The default is to continue.

- **-g group -u user -r reserved-blocks-count** – which group and user can use the reserved blocks on the disk. The standard options reserve a few blocks for the user root and the group root so that the system still functions when the filesystem is full.
- **-U UUID** – set the universally unique identifier for the filesystem. This can be used in **/etc/fstab** and for locating the journal of a filesystem.

Here is how you add a journal to an existing filesystem using **tune2fs**. This converts it from ext2 to ext3.

```
[jack@foo jack]$ dd if=/dev/zero of=filesystem bs=1M count=4
4+0 records in
4+0 records out
[jack@foo jack]$ /sbin/mke2fs filesystem
mke2fs 1.27 (8-Mar-2002)
filesystem is not a block special device.
Proceed anyway? (y,n) y
# mke2fs says lotsa stuff ...
[jack@foo jack]$ /sbin/tune2fs -j filesystem
tune2fs 1.27 (8-Mar-2002)
Creating journal inode: done
This filesystem will be automatically checked every 30 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

23.2.3 reiserfs

If you ever need to run **reiserfsck** manually, you may have to use the command **reiserfsck --rebuild-tree**.

```
[root@foo jack]# dd if=/dev/zero of=filesystem bs=1M count=60
60+0 records in
60+0 records out
[root@foo jack]# /sbin/mkreiserfs -f filesystem

<-------------mkreiserfs, 2002------------->
reiserfsprogs 3.6.2

filesystem is not a block special deviceContinue (y/n):y
# ... snip ...
UUID: ef129659-35e8-451b-a1a2-61e6ae1a452c
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..ok
# ... snip ...
[root@foo jack]# mount -o loop filesystem /mnt/
[root@foo jack]# cp -a /usr/ /mnt/ 2>/dev/null
[root@foo jack]$ /sbin/reiserfsck --rebuild-tree filesystem

<-------------reiserfsck, 2002------------->
reiserfsprogs 3.6.2
# ... snip ...
Do you want to run this program?[N/Yes] (note need to type Yes):Yes
Replaying journal..
No transactions found
############
```

```
reiserfsck --rebuild-tree started at Wed Apr 23 11:26:34 2003
############

Pass 0:
####### Pass 0 #######
Loading on-disk bitmap .. ok, 14861 blocks marked used
Skipping 8211 blocks (super block, journal, bitmaps) 6650 blocks
      will be read
0%....20%....40%....60%....80%....100%                        left
      0, 3325 /sec
"r5" got 931 hits
        "r5" hash is selected
Flushing..done
        Read blocks (but not data blocks) 6650
                Leaves among those 151
                Objectids found 916

Pass 1 (will try to insert 151 leaves):
####### Pass 1 #######
Looking for allocable blocks .. ok
0%....20%....40%....60%....80%....100%
      left 0, 0 /sec
Flushing..done
        151 leaves read
                151 inserted
####### Pass 2 #######
Flushing..done
Pass 3 (semantic):
####### Pass 3 #########
Flushing..done
        Files found: 786
        Directories found: 4
        Symlinks found: 125
Pass 3a (looking for lost dir/files):
####### Pass 3a (lost+found pass) #########
Looking for lost directories:
Flushing..done done 1, 1 /sec
Pass 4 - done           done 0, 0 /sec
Flushing..done
Syncing..done
############
reiserfsck finished at Wed Apr 23 11:26:38 2003
############
```

23.3 Review

Quiz questions

1. In which two ways can a Linux file system become full?
2. Where are lost files stored when they are recovered during a filesystem check?
3. How does **du** differ from **df**?
4. What is **debugfs**, and for which filesystems is it available?

5. Why are deleted files often found during a filesystem check?

Assignment

1. Set up an ext2 filesystem in a partition. Set the label of this partition to CRASHME and mount the filesystem at **/mnt/crashme**. How large is the partition? How many inodes are available? How much space is available for files?
2. Copy a large directory tree (e.g. **/usr**) to your filesystem (**cp -a /usr /mnt/crashme**). While this copy process is in process, reboot the computer forcibly by pressing the reset button. Use the diagnostic commands to examine the effect of doing this.

   Now set up a journal for your e2fs partition, and repeat the operation above. Make notes on what you observe.
3. Repeat the previous assignment with the other filesystems you have available. Do any of these survive the treatment?
4. Create an **ext3** filesystem, and copy some files to it. Use **tune2fs** to remove the journal from the ext3 partition and convert it into a ext2 partition.

Answers to quiz questions

1. Run out of space, and run out of available inodes.
2. lost+found in the root directory of the filesystem.
3. du reads the disk like ls, while df display summary information only.
4. debugfs shows filesystem information. It's available for ext2/3. Tools do exist for other filesystems, but do not have the name.
5. Linux allows open files to be deleted. The files stay on the disk, but their directory entries are removed.

24 Mounting

> Recursive traversal of loopback mount points.
>
> *– BOFH excuse of the day*

Linux does not have A: for its floppies, C: for its hard disk, D: for another hard disk and H: to Z: for network connections. Instead all files are arranged in a big tree, the file hierarchy, rooted at /. The files can be spread out over several devices. The **mount** command attaches the filesystem found on a device to the file hierarchy. The **umount** command detaches filesystems from the file hierarchy.

This chapter is primarily concerned with hard disk partitions and removable media, and omits mention of the various network filesystems that Linux supports.

LPIC topic 1.104.3 — Control mounting and unmounting filesystems [3]

Weight: 3

Objective

Candidates should be able to configure the mounting of a filesystem. This objective includes the ability to manually mount and unmount filesystems, configure filesystem mounting on bootup, and configure user mountable removeable filesystems such as tape drives, floppies, and CDs.

Key files, terms, and utilities include

/etc/fstab	file system table
mount	mount a filesystem
umount	unmount a filesystem

24.1 mount

The **mount** command without any options shows the filesystems which are mounted as part of the filesystem.

On this system, partitions of the first IDE hard disk **/dev/hda** are mounted at **/** and **/home**. The additional entries are kernel virtual filesystems (**/proc**, **/dev/pts**, and **/dev/shm**).

```
[jack@tonto jack]$ mount
/dev/hda7 on / type ext3 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/hda3 on /home type ext3 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
```

mount is used to add a filesystem to *the* filesystem.

```
mount [ -t type ] [ -o options ] /dev/what /where
```

The **mount** commands which correspond to the mounts above are these

```
mount -t ext3 -o rw /dev/hda7 /
```

```
mount -t ext3 -o rw /dev/hda3 /home type ext3
```

24.2 fstab

During the boot sequence, the command **mount -a** is run. **mount -a** reads **/etc/fstab** and, for each line, mounts the appropriate filesystem at the designated mount point.

```
# what          mount-point  filesystem   options       dump pass
LABEL=/         /            ext3         defaults         1 1
/dev/hda3       /home        ext3         defaults         1 2
none            /dev/pts     devpts       gid=5,mode=620   0 0
none            /proc        proc         defaults         0 0
none            /dev/shm     tmpfs        defaults         0 0
/dev/hda2       swap         swap         defaults         0 0
/dev/cdrom      /mnt/cdrom   iso9660      user,noauto,owner,kudzu,ro
      0 0
/dev/fd0        /mnt/floppy  auto         noauto,owner,kudzu 0 0
```

The columns in **/etc/fstab** are

- What to mount – usually a partition
- Mount point – where to mount it. "swap" is a special mount point for swap files and swap partitions (virtual memory).
- Filesystem – the **-t** type of the filesystem must be given. If the filesystem is "auto", mount will try all the filesystems supported by the kernel (listed in **/proc/filesystems**).
- Options – the **-o** options required for mounting the partition are given. If no special options are required, the word "defaults" is used. "noauto" tells **mount -a** that the filesystem should not be mounted when the system starts up. "ro" is read-only, and "rw" is read-write.
- Dump – if you use **dump** for backups of your filesystems, you will set this value accordingly.
- Pass – at which point must this filesystem be checked by **fsck** during boot-up. "0" means no check, "1" means first, and is used for the root filesystem, and higher values are used for the other filesystems.

24.3 Options for mount

The following switches for mount are available regardless of the type of filesystem being mounted.

- **mount -t filesystem** – mount as if the device contains a specific filesystem, e.g. iso9660, reiserfs, ext2, ext3, etc.
- **mount -V** – show the version of mount (doesn't actually mount anything).
- **mount -a** – mount all filesystems
- **mount -o option1,option2,option3** – this sets the options for mounting.
- **mount -n** – mount without changing **/etc/mtab**. You need to do this when **/etc** is on a read-only filesystem, i.e. **mount /dev/hda2 / -n -o remount,rw**

Each filesystem has a number of **-o** options which are supported by it.

The following **-o** options usually appear in **/etc/fstab** to control the mounting behaviour of the

filesystem.

- **auto** – this option is used in **/etc/fstab** to specify that the system must be mounted automatically at boot-up. The opposite is **noauto**.
- **user** – this option specifies that a regular user may mount and unmount the filesystem. The default is **nouser**. The **users** option specifies that *any* user can mount and unmount a filesystem (not only the user that mounted it).

The following **-o** restrict the way in which a mounted filesystem can be used. Often the **nosuid**, **noexec** and **nodev** restrictions are used for removable media for security reasons.

- **ro** – read-only (no changes are written to the filesystem) (opposite is **rw** for read-write).
- **suid** – set-uid bits on file permissions are honoured (opposite is **nosuid**).
- **exec** – executable files are allowed on the filesystem (opposite is **noexec**).
- **dev** – character and block devices are allowed on the filesystem (opposite is **nodev**).

These **-o** options don't fit into the categories above.

- **async** – changes are written to the filesystem asynchronously, and not as they occur (opposite is **sync**, where any changes are written to the filesystem before processing continues).
- **remount –** When the filesysem you mount is already mounted, to change the mount options, you use **-o remount**.
- **defaults** – equivalent to **rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, and **async**.

24.4 Removable media

Floppy disks, CD-ROMs and other removable media need to be mounted before they are used, unless you use some tool such as the **mtools** package (for floppy disks), **cdparanoia** or **dd** to read the device directly.

Usually, a line like this appears in **/etc/fstab** (sometimes specifying **/floppy** or **/mnt/floppy** instead of **/media/floppy**).

```
/dev/fd0     /media/floppy        auto       noauto,user,sync      0
      0
```

This means that any user can access the floppy disk with any of these commands. The user cannot specify any options to override the options listed in **/etc/fstab**.

```
$ mount /dev/fd0 /media/floppy
$ mount /dev/fd0
$ mount /media/floppy
```

The lines for a CD-ROM are typically something like this (from various machines). All of the **fstab** lines specify **user** (so that a regular user can access the media) and **noauto** (so that the media is not mounted by root during startup).

```
/dev/cdrom       /mnt/cdrom       iso9660 user,noauto,owner,ro  0 0
/dev/cdrecorder  /mnt/cdrecorder  auto    ro,noauto,user,exec   0 0
/dev/scd0        /mnt/cdrom1      auto    noauto,user           0 0
```

24.5 Review

Quiz questions

1. Write the full command to mount the first partition of the third scsi disk on the **/mnt** directory
2. How would you allow a non root user to mount a cdrom on **/mnt/cdrom**?
3. How does one prevent a filesystem from being mounted at bootup?
4. Who can use the **umount** command?

Assignment

1. Create an ext2 filesystem on **/dev/fd0** using **mke2fs**. Mount the floppy disk as read-write and copy **/etc/passwd** onto the floppy. Remove the floppy disk and set the read-only tag. Configure your system to mount the floppy disk read only when you boot up. Does it work? If not, make it work. What can a floppy disk like this be used for? What happens if the floppy disk is not in the drive when the system is booted?
2. Configure your system so that a regular user can mount and unmount the floppy disk, but ensure that no executable files on the floppy disk can be run. Test whether your changes work. (You will need to undo the effects of the previous assignment.)
3. Create a loopback filesystem, and mount it on **/mnt/loop**. To create the filesystem you can use the following script or any other method.

```
dd if=/dev/zero of=bigfile bs=1M count=20
mke2fs bigfile
```

   Which option is required to **mount** in order to use the filesystem you have created?

Answers to quiz questions

1. mount /dev/sdc1 /mnt
2. Create an entry in **/etc/fstab** with “user” and “noauto” in the options column.
3. Add “noauto” in the options column in **/etc/fstab**.
4. Only root, unless the filesystem was mounted by the user and the options column in **/etc/fstab** includes “user”.

25 Quotas

"I hate quotas."

– Ralph Waldo Emerson, when his system administrator said he couldn't have another 50 Megs

Quotas are the nemesis of disk hungry users and programs. Quotas enable the administrator to restrict the amount of disk space available to a given user or a given group. Quotas are often used on web servers to restrict the size of a user's web directory. Quotas on mail servers prevent a single user's mail from filling up the disk. Because they are effective, quotas cause administration problems.

LPIC topic 1.104.4 — Managing disk quota [3]

Weight: 3

Objective

Candidates should be able to manage disk quotas for users. This objective includes setting up a disk quota for a filesystem, editing, checking, and generating user quota reports.

Key files, terms, and utilities include

quota	Show disk usage and limits
edquota	Modify quotas for a user or group
repquota	Report on quotas
quotaon	Turn quotas on for a filesystem

25.1 Overview

In order to use quotas, the kernel must be compiled with quota support. When quotas are enabled, the kernel notes each operation that causes a change in the amount of disk space allocated to a user or to a group.

The quota commands are used in the following sequence.

1. Add the **usrquota** or **grpquota** option to **/etc/fstab** for the filesystems that require quota support.
2. Run **quotacheck -avug** to determine the amount of disk space allocated to each user and group, and create the quota files **quota.user** and **quota.group** in the root directory of the filesystems.
3. Run **quotaon** to enable quotas.
4. Run **edquota** to set quotas for users
5. Run **repquota** to view the usage of quotas.

The quota commands discussed in this section have the following switches in common:

- -v – verbose
- -a – all file systems with quotas as listed in /etc/fstab
- -u – user quotas (the default)
- -g – group quotas

25.2 Enabling Quotas

When a filesystem is mounted with the **usrquota** or the **grpquota** option, the kernel applies quota control on that filesystem. The quota system must be initialised before the system is mounted.

If you set up quotas on an existing system, you might do it something like this. The first step is to modify **fstab**.

```
foo:~ # cat /etc/fstab
LABEL=/         /                ext3    defaults           1 1
/dev/hda3       /home            ext3    defaults           1 2
none            /dev/pts         devpts  gid=5,mode=620     0 0
none            /proc            proc    defaults           0 0
none            /dev/shm         tmpfs   defaults           0 0
/dev/hda2       swap             swap    defaults           0 0
foo:~ # vi /etc/fstab
foo:~ # cat /etc/fstab
LABEL=/         /                ext3    usrquota,grpquota  1 1
/dev/hda3       /home            ext3    usrquota,grpquota  1 2
none            /dev/pts         devpts  gid=5,mode=620     0 0
none            /proc            proc    defaults           0 0
none            /dev/shm         tmpfs   defaults           0 0
/dev/hda2       swap             swap    defaults           0 0
```

Then we update the quota files.

```
foo:~ # quotacheck -avug
quotacheck: Can't find filesystem to check or filesystem not mounted
      with quota option.
```

Bother! It has to be mounted with the **usrquota** and/or **grpquota** options.

```
foo:~ # mount / -o remount
foo:~ # mount /home -o remount
foo:~ # quotacheck -avug
quotacheck: Cannot get quotafile name for /dev/hda7
```

Now it can't find the file. So let's give it the file. For version 1 quotas, it would be **quota.user** and **qouta.group**. We'll use version 2, **aquota.user** and **aquota.group**.

```
foo:~ # touch /aquota.user /aquota.group /home/aquota.user
      /home/aquota.group
foo:~ # quotacheck -avug
quotacheck: WARNING - Quotafile /home/aquota.user was probably
      truncated.  Can't save quota settings...
quotacheck: Cannot remount filesystem mounted on /home read-only so
      counted values might not be right. Please stop all programs
      writing to filesystem or use -m flag to force checking.
```

Okay, so lets use the **-m** flag, since there aren't any programs writing to the filesystem. If you want accurate quotas (usually) you would shut down to single user mode with **init single** before running **quotacheck**, without **-m**.

```
foo:~ # quotacheck -avugm
quotacheck: Scanning /dev/hda7 [/] done
quotacheck: Checked 12071 directories and 133110 files
quotacheck: Scanning /dev/hda3 [/home] done
quotacheck: Checked 7347 directories and 70493 files
```

```
foo:~ # ls -l /aq* /home/aq*
-rw-r--r--    1 root     root     9216 Apr 24 13:51 /aquota.group
-rw-r--r--    1 root     root     9216 Apr 24 13:51 /aquota.user
-rw-r--r--    1 root     root    24576 Apr 24 14:53
      /home/aquota.group
-rw-r--r--    1 root     root    34816 Apr 24 14:15
      /home/aquota.user
```

To make these quotas come into effect, we run **quotaon -avug**. This requires quota support in the kernel, which may be compiled as a module.

```
foo:~ # modprobe quota_v2
foo:~ # quotaon -avug
```

25.3 Setting quotas

The **edquota** command allows the administrator to edit the quota for a user or a group. **edquota**'s default mode of operation is to edit the quotas for a user

edquota allows you to edit a temporary file that sets the quota for the user. The blocks parameter indicates the number of blocks in use. The actual size allocated depends on the size of the blocks on the filesystem. The inodes limit allows the administrator to limit the number of files the user may create.

```
# edquota joe
Disk quotas for user joe (uid 514):
  Filesystem   blocks      soft      hard    inodes     soft
      hard
  /dev/hda7         0         0         0         1        0
      0
  /dev/hda3        40         0         0        10        0
      0
~
"/tmp//EdP.azZAqUH"
```

The “blocks” and “inodes” columns specify the amount of blocks and inodes actually in use, according to the kernel quota system.

Disk quotas are a really good reason not to get on the wrong side of the system administrator.

```
Disk quotas for user joe (uid 514):
  Filesystem   blocks      soft      hard    inodes     soft
      hard
  /dev/hda7         0        30        40         1        0
      0
  /dev/hda3        40        50        60        10       20
      40
~
"/tmp//EdP.aAvNEZ7"
```

There's not much one can do with 60 blocks – it turns out to be 60kbytes on this system.

```
[joe@foo joe]$ dd if=/dev/zero of=$RANDOM bs=1k
ide0(3,3): warning, user block quota exceeded.
ide0(3,3): write failed, user block limit reached.
dd: writing `20086': Disk quota exceeded
```

```
17+0 records in
16+0 records out
[joe@foo joe]$ du -hsc .
60K     .
60K     total
```

Let's create a stack of files (using **touch**). We only get so many. So the quota does seem to be working.

```
[joe@foo joe]$ while touch $RANDOM; do echo -ne '' ; done
ide0(3,3): warning, user file quota exceeded.
ide0(3,3): write failed, user file limit reached.
touch: creating `8786': Disk quota exceeded
[joe@foo joe]$ ls          # we're missing 8786 ...
1239   13313  1595   18389  20855  24789  29816  5927  7061  8841
12942  13325  17158  18863  21371  28547  31968  6370  7327
13021  14995  18339  20086  23493  29722  4481   6897  7717
[joe@foo joe]$ find | wc   # how many inodes are in use?
     40      40     391
```

Here is the quota system's view of the affair.

```
[joe@tonto joe]$ quota
Disk quotas for user joe (uid 514):
    Filesystem  blocks  quota  limit  grace  files  quota  limit
      grace
     /dev/hda7      0     30     40            1      0      0
     /dev/hda3     60*    50     60           40*    20     40
[joe@tonto joe]$ quota -q
Disk quotas for user joe (uid 514):
        File limit reached on /dev/hda3
        Block limit reached on /dev/hda3
```

quota -q can be included in **/etc/profile** so that the user is notified about problems with her disk space usage.

If we remove a file, we should have some inodes spare, and some space too.

```
[joe@foo joe]$ rm .bashrc
[joe@foo joe]$ quota -q
Disk quotas for user joe (uid 514):
        In file grace period on /dev/hda3
        In block grace period on /dev/hda3
```

Quotas version 2 allows you to set the grace time during which the soft limit can be exceeded.

```
% edquota  -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem             Block grace period     Inode grace period
  /dev/hda7                     7days                  7days
  /dev/hda3                     7days                  7days
```

25.4 Reporting with repquota

The **repquota** command produces a report for user (default or **-u**) or group (**-g**) quotas:

```
foo:~ # repquota /home
*** Report for user quotas on device /dev/hda3
Block grace time: 7days; Inode grace time: 7days
```

```
                     Block limits                File limits
User            used    soft    hard  grace    used  soft  hard
      grace
-----------------------------------------------------------------------
      --
root      --   33464       0       0           7174     0     0
lp        --      20       0       0              5     0     0
news      --       4       0       0              1     0     0
uucp      --       4       0       0              1     0     0
bob       --     592       0       0             56     0     0
demo      --    4656       0       0            634     0     0
jack      +-   31384      20      40  6days    1034     0     0
joe       --      40      50      60             10    20    40
```

25.5 Review

Quiz questions

1. Name 2 Linux filesystems that support quotas, and name one filesystem that does not.
2. What happens when the soft limit is reached?
3. What happens when the hard limit is reached?
4. Who can adjust filesystem quotas?
5. If changes are made to the filesystem while quotas are not enabled, which command must be used for quotas to work correctly?

Assignment

1. Set up quotas on a filesystem as explained in the notes above. Edit quotas for a user and allow the user 50Mb more space and 50 more files than the user currently has available. Run the command **dd if=/dev/zero of=dir/$RANDOM.junk bs=1k count=1000** until the quota is reached, and write down an explanation of the behaviour you see.
2. Set the grace period for users exceeding the soft limit to one minute, and explain the behaviour of commands when the quota is exceeded.

Answers to quiz questions

1. reiserfs does not support quotas in some kernel versions. Ext2, ext3 and others do support quotas. Read-only file systems do not support quotas.
2. Nothing happens. **quota -q** begins to display a warning message.
3. The filesystem behaves as if it is full.
4. Only root.
5. **quotacheck -avug** (or specify the specific filesystem)

26 Permissions

"No."

– Mom

Linux restricts access to the filesystem based on one of these ...

- The owner of the files
- The permissions of the files
- The permissions and ownership of the parent directories.

This chapter discusses the permissions of filesystem objects.

LPIC topic 1.104.5 — Use file permissions to control access to files [5]

Weight: 5

Objective

Candidates should be able to control file access through permissions. This objective includes access permissions on regular and special files as well as directories. Also included are access modes such as suid, sgid, and the sticky bit, the use of the group field to grant file access to workgroups, the immutable flag, and the default file creation mode.

Key files, terms, and utilities include

chmod	Change file mode
umask	Change user mask
chattr	Change file attributes (ext2 only)

26.1 Ownership and permissions

The items that affect the permissions of a file are displayed when using the **ls -l** command:

- The file type (e.g. a regular file, a directory, a symbolic link, etc). This is the first character on each line in the output of **ls**.
- The permissions of the file (e.g. **rwxr-xr-x**).
- The user and group of the file.

```
[jack@foo jack]$ ls -ld /usr/bin/pas* /etc/ho* /bin/ba*
     /usr/bin/wal* ~
-rwxr-xr-x    1 root     root      10680 Aug 29  2002 /bin/basename
-rwxr-xr-x    1 root     root     626188 Aug 23  2002 /bin/bash
lrwxrwxrwx    1 root     root          4 Oct 14  2002 /bin/bash2 ->
     bash
-rw-r--r--    1 root     root         17 Jul 23  2000 /etc/host.conf
-rw-rw-r--    2 root     root        291 Mar 31 15:03 /etc/hosts
-rw-r--r--    1 root     root        201 Feb 13 21:05
     /etc/hosts.allow
-rw-rw-r--    1 root     root        196 Dec  9 15:06 /etc/hosts.bak
-rw-r--r--    1 root     root        357 Jan 23 19:39
     /etc/hosts.deny
drwxr-xr-x    4 root     root       4096 Oct 14  2002 /etc/hotplug
drwx------   21 jack     jack       4096 Apr 30 09:15 /home/jack
```

```
-rwxr-xr-x    1 root     root       4467 Aug 20  2002
      /usr/bin/passmass
-r-s--x--x    1 root     root      15368 May 28  2002
      /usr/bin/passwd
-rwxr-xr-x    1 root     root      18774 Jul  1  2002 /usr/bin/paste
-r-xr-sr-x    1 root     tty       10224 Jul 19  2002 /usr/bin/wall
```

Every file has one owner and one group[27]. Only the owner or root can change a file's permissions using **chmod**. The owner can change the group of the file to any group of which he is a member (using **chgrp**).

File permissions are represented with the letters r (read), w (write) and x (execute).

User	*Group*	*Others*
rwx	rwx	rwx

The interpretation of the permissions are as follows.

- **r** – read permission (for a directory, that means you can view the file listing)
- **w** – write permission
- **x** – execute permission (for a directory, that means you can use the files inside the directory)
- **-** – no permission (if there is a dash in the place of the letters r, w or x, then that permission is absent).
- **s** – sticky bit (this can appear in the place of **x** for the user or group).
- **t** – sticky bit (this can appear in the place of **x** for "others").

The sticky bits are explained in detail for each file type.

26.2 chmod

chmod changes file access permissions.

Symbolic chmod

The symbolic syntax for **chmod** allows you to change the permissions of a file for the category user, group or others.

```
chmod [options] [ugoa][+-=][rwxXstugo] FILE
```

You can specify multiple changes by separating them by comas.

Symbol	*Usage*
ugoa	**u**ser, **g**roup, **o**thers[28] or **a**ll. Default is all, but limited by **umask** value.
+-=	add, subtract or set permissions

27 Kernel patches that implement access control lists for certain filesystems are available. These are not standard equipment on a Linux installation. Having only one owner and one group makes Linux a simpler and more robust system.

28 It is rather easy to think you are changing the owner's permissions with "**o**", which actually stands for "other".

Symbol	*Usage*
rwuxXstu g o	**r –** read permission **w** – **w**rite permission **x** – e**x**ecute permission **X** – e**x**ecute permission provided the file has execute permissions already **s** – **s**et user id or set group id **t** – s**t**icky **u** – same as user's permissions **g** – same as user's permissions **o** – same as other people's permissions

And some practice ...

```
$ cp /dev/null file     # create a file with nothing in it
$ ls -la file
$ chmod +x file         # set the execute permission for the file
$ ls -la file
$ chmod go-rw file      # remove read and write permission for
      everyone but the owner
$ ls -la file
$ chmod ugo+rwx file    # give everyone read, write, and execute
      permission.
$ ls -la file
```

Octal chmod

The octal mode of **chmod** sets the file permissions to exactly the value specified as an octal number[29].

```
chmod [options] OCTAL-MODE FILE
```

Every letter in "**rwx**" corresponds to a bit: "**---**" is 0, "**--x**" is 1, "**-w-**" is 2, "**r--**" is 4. The idea is to add the digits together as well, so "**-wx**" is 3 and "**r-x**" is 5.

Sticky			*User*			*Group*			*Others*		
s	**s**	**t**	**r**	**w**	**x**	**r**	**w**	**x**	**r**	**w**	**x**
4	2	1	4	2	1	4	2	1	4	2	1

Some examples of symbolic and octal permissions.

Symbolic	*Octal*	*Example*
rwxr-x---	0750	**/home/user**
rwxr-x--x	0751	**/etc/ppp/peers**
rw-r--r--	0644	**/etc/bashrc**

29 Octal is a base-8 number system, using the digits 0 to 7. Decimal is base 10 (0 to 9), binary base 2 (0 and 1) and hexadecimal base 16 (0 to 9 and A to F). The reason octal is used is that each permission is represented as a bit which is either set or unset, and octal conveniently allows you to group three bits into one digit.

Symbolic	*Octal*	*Example*
rw-r-----	0640	**/etc/aliases.db**
rwsr-xr-x	4755	**/bin/ping**
rwxr-sr-x	2755	**/usr/bin/write**
rwxrwxrwx	1777	**/tmp**

Recursive chmod

chmod -R changes the permissions of a directory and its files and subdirectories.

Here we remove permissions from all other users for jack's home directory, and make sure he has write and execute permissions on his own files. We discover that we can't change the mode of the immutable file (see the section on ext2fs attributes).

```
foo:~ # chmod -R go-wrx,u+rX ~jack
chmod: changing permissions of `/home/jack/attr/dfjqw': Operation
      not permitted
foo:~ # su - jack
[jack@foo jack]$  ls -l | head
total 96
-rw-------    1 jack     jack          956 Apr 10 10:59 ANOTHERCOPY
-rw-------    1 jack     jack         2010 Apr  8 11:46 COPY-OF-
      PASSWD
drwx------    3 jack     jack         4096 Apr 22 15:57 Desktop
-rw-------    1 jack     jack          570 Apr 10 11:04 GREPRESULTS
drwx------    2 jack     jack         4096 Apr  8 11:57 JOE
drwx------    2 jack     jack         4096 Apr 22 11:51 Junk
drwx------    7 jack     jack         4096 Apr 22 13:47 Mail
-rw-------    1 jack     jack           76 Apr 10 11:14 TELNET-LOG
```

26.3 File types

File types on Linux are identified by the left hand character in the output of **ls**. The following possibilities exist:

- regular files
- **d** – directory
- **l** – symbolic link
- **b** – block device
- **c** – character device
- **p** – named pipe (made with mkfifo)
- **s** – socket (unix domain socket)

Each type of file has slightly different behaviour based on its permissions.

Regular files

Regular files are identified by "**-**" before the permissions.

```
[jack@foo jack]$ ls -l /
-rw-r--r--    1 root     root         9216 Apr 29 14:58 aquota.group
```

The following permissions are quite common for a file.

- **rw-r--r-- –** 0644 **–** only the file owner may modify the file
- **rw-rw-r--** – 0664 – the user and the group may read and write the file
- **rwxr-xr-x** – 0755 **–** This file is executable, and can be modified by the owner.
- **rwx------** – 0700 **–** The file is executable, but only for the owner.

For a executable files, the set-uid bits have the following meanings:

- Set user-id bit: the program may choose to set its real user id to the effective user id of the file. This allows a program to do privileged operations as root, or as some other user.
- Set group-id bit: the program may choose to set its real group id to the effective group id of the file. The file can then do actions requiring the permissions of the specific group.

Under Linux, effects of settings the other sticky bits are largely undefined. On non-Linux systems, setting these bits requests in-memory caching of the file, and mandatory locking (except for NFS).

Directories

Directories are identified by a "d" in the directory listing. The interpretation of permissions for directories is as follows.

- **r** – Read permission means a user can observe the list of files in the directory.
- **w** – Write permission means a user can create files in the directory.
- **x** – Execute permission means a user can open files inside the directory (provided the user has permissions to read the actual file).

The following permissions are quite common for a directory:

- **drwxr-xr-x** – 0755 – The owner can make files in the directory, and others can read the files that are there.
- **drwxr-x---** – 0750 – The group can read from the directory, but not modify it, and all others are barred.
- **drwx------** – 0700 – A directory for the exclusive use of the owner.
- **drwx--x--x** – 0711 – All users, except the owner, can open files in this directory, if they happen to know the file names. This permission is suitable for a web server.

For directories files, the set-uid and sticky bits have the following meanings:

- Set group-id bit – New files created in the directory are given the group id of the directory, rather than the group id of the process that created them.
- Sticky bit – Files in the directory can only be deleted by their owner. This permission is used for the **/tmp** directory, to prevent denial of service and symbolic link race attacks between users.

Symbolic links

Symbolic links are created with the **ln -s** command. The owner of the symbolic link is the user that created the link, but the permissions for a link always look like this.

```
[jack@foo jack]$ ln -s /etc/passwd
[jack@foo jack]$ ls -la passwd
```

```
lrwxrwxrwx    1 jack     jack    11 Apr 30 13:22 passwd ->
      /etc/passwd
```

When **chmod** is used on a symbolic link, it acts on the file that the link points to.

```
[jack@foo jack]$ touch file
[jack@foo jack]$ ln -s file link
[jack@foo jack]$ chmod 600 link
[jack@foo jack]$ ls -la file link
-rw-------    1 513      jack       0 Apr 30 13:23 file
lrwxrwxrwx    1 513      jack       4 Apr 30 13:23 link -> file
```

The permissions for a symbolic link are therefore largely irrelevant, since they can not be changed. Having a link pointing to a file does not provide a user any additional permissions to the file.

```
[jack@foo jack]$ ln -s /etc/shadow mylink
[jack@foo jack]$ ls -l mylink
lrwxrwxrwx    1 jack     jack      11 Apr 30 13:27 mylink ->
      /etc/shadow
[jack@foo jack]$ cat mylink
cat: mylink: Permission denied
```

Pipes, sockets and devices

Named pipes made with **mkfifo** ("**p**") allow programs to stream information to each other. Both programs require permissions on the FIFO for this to work, and the sticky bits have no particular significance.

UNIX domain sockets (file type "**s**") allow programs on a single machine to communicate using sockets. On Linux UNIX domain sockets respect the permissions of the directory they appear in. The set-id and sticky bits have no particular significance.

Character devices (file type "**c**") and block devices (file type "**b**") behave like regular files as far as their permissions are concerned.

26.4 umask

The **umask** setting of a process determines the permissions with which a file is created by default. During login an initial value is set – usually in **/etc/profile**.

If the **umask** is **000**, then files are created with permissions of **0666** (**rw-rw-rw-**) and directories with permissions of **0777** (**rwxrwxrwx**). The **umask** value is subtracted from the permissions that the file would be created with. Usually **umask** digits are either "0", "2" or "7".

- 0 – read, write and execute allowed.
- 2 – no read permissions for files or directories.
- 7 – no permissions at all.

The more common values for **umask** are shown in the table.

Umask	*Directories*	*Files*
002	rwxrwxr-x	rw-rw-r--

Umask	*Directories*	*Files*
022	rwxr-xr-x	rw-r--r--
027	rwxr-x---	rw-r-----
077	rwx------	rw-------

Of course, the **umask** values can be set to any value from 0 to 7.

```
[jack@foo umask]$ for UMASK in 0 1 2 3 4 5 6 7; do umask $UMASK ;
     touch file.$UMASK; done
[jack@foo umask]$ ls -l file.*
-rw-rw-rw-    1 jack     jack            0 Apr 30 11:52 file.0
-rw-rw-rw-    1 jack     jack            0 Apr 30 11:52 file.1
-rw-rw-r--    1 jack     jack            0 Apr 30 11:52 file.2
-rw-rw-r--    1 jack     jack            0 Apr 30 11:52 file.3
-rw-rw--w-    1 jack     jack            0 Apr 30 11:52 file.4
-rw-rw--w-    1 jack     jack            0 Apr 30 11:52 file.5
-rw-rw----    1 jack     jack            0 Apr 30 11:52 file.6
-rw-rw----    1 jack     jack            0 Apr 30 11:52 file.7
```

And here we do it for directories.

```
[jack@foo umask]$ for UMASK in 0 1 2 3 4 5 6 7; do umask $UMASK ;
     mkdir dir.$UMASK; done
[jack@foo umask]$ ls -lad dir.*
drwxrwxrwx    2 jack     jack         4096 Apr 30 11:51 dir.0
drwxrwxrw-    2 jack     jack         4096 Apr 30 11:51 dir.1
drwxrwxr-x    2 jack     jack         4096 Apr 30 11:51 dir.2
drwxrwxr--    2 jack     jack         4096 Apr 30 11:51 dir.3
drwxrwx-wx    2 jack     jack         4096 Apr 30 11:51 dir.4
drwxrwx-w-    2 jack     jack         4096 Apr 30 11:51 dir.5
drwxrwx--x    2 jack     jack         4096 Apr 30 11:51 dir.6
drwxrwx---    2 jack     jack         4096 Apr 30 11:51 dir.7
```

26.5 Ext2 attributes

The **chattr** command manipulates ext2 filesystem attributes. **lsattr** displays these attributes. The following attributes can be set on the ext2 filesystem (and not on other filesystems):

- **i** – immutable – the file cannot be changed, unless the immutable bit is removed first
- **a** – append only – this file can only be appended to, not rewritten.
- **s** – secure deletion – after the file is deleted, the blocks that stored its data are blanked out.

Here's an example of setting the immutable flag on a file.

```
[jack@foo attr]$ ls -l
-rw-rw-r--    1 jack     jack            0 Apr 30 14:25 dfjqw
-rw-rw-r--    1 jack     jack            0 Apr 30 14:25 qwoo1
-rw-rw-r--    1 jack     jack            0 Apr 30 14:25 woow
[jack@foo attr]$ lsattr
-------------- ./dfjqw
-------------- ./qwoo1
-------------- ./woow
[jack@foo attr]$ chattr +i dfjqw
chattr: Operation not permitted while setting flags on dfjqw
```

```
[jack@foo attr]$ su
Password:
[root@foo attr]# chattr +i dfjqw
[root@foo attr]# rm dfjqw
rm: remove write-protected regular empty file `dfjqw'? y
rm: cannot remove `dfjqw': Operation not permitted
[root@foo attr]# lsattr
---i---------- ./dfjqw
-------------- ./qwoo1
-------------- ./woow
[root@tonto attr]# ls >> dfjqw
bash: dfjqw: Permission denied
```

26.6 Review

Quiz questions

1. What is the effect of setting the sticky bit for a file?
2. What is the meaning of **chmod 640 filename**?
3. Which users may change the permissions on a given file?
4. What are the default permissions for a file and for a directory when the **umask** is 0027?
5. What is the effect of the set group-id bit for a directory?
6. Which filesystems support **chattr** and **lsattr**?

Assignment

1. Write down the commands do do the following. Create files using **touch** and set the following permissions:

   file1 **-rw-rw-rw-**; file2 **-rwxrwxrwx**; file3 **-rwsr-xr-x**; file4 **-rwx------**; file5 **---------x**; file6 **-r-xr-Sr-x**; file7 **-rws------** (that's an entirely useless set-uid bit that is).

   Create directories using **mkdir** and set the following permissions:

   dir1 **drwxrwxrwx**; dir2 **drwx--x--x**; dir3 **drwxrwsr-x**; dir4 **drwxrwxrwt**; dir5 **----------**; dir6 **drwxr-xr-x**; file7 **drwxr-x---**
2. Run the command **chmod -R 000 ~**. What happens, and why? How can you fix it? *Hint: find -type d | xargs chmod something {} \; ... repeat ...*
3. Use **find** to create a list of the files on your disk which are world writable.
4. Sort this list into different types of files. *Hint: use ls -ld to show the file type ...*

Answers to quiz questions

1. Nothing happens, except that the sticky bit is set.
2. Set permissions to rw-r----- for the file.
3. Root and the file owner. Not group members.
4. For a file 0666 - 027 = 0640 (rw-r-----), for a directory 0777 - 027 = 0750 (rwxr-x---).
5. Files and directories in the directory receive the group of the directory.

6. ext2 and ext3

27 File ownership

I really hate this damned machine
I wish that they would sell it.
It never does quite what I want
But only what I tell it.

(All the good quotes were taken)

In Linux, you get your own files. Nobody else can fiddle them, unless you allow them to. This chapter is about setting the ownership of files.

LPIC topic 1.104.6 — Manage file ownership [1]

Weight: 1

Objective

Candidates should be able to control user and group ownership of files. This objective includes the ability to change the user and group owner of a file as well as the default group owner for new files.

Key files, terms, and utilities include

chmod	Change file mode (permissions)
chown	Change file owner (and maybe group too)
chgrp	Change file group

27.1 File ownership

Every file on the Linux filesystem is assigned a single owner and a single group – yes, only one group. Linux allows you to assign permissions for the owner (user), group, and other users, namely permission to read, write and execute (or search, for directories).

The user that creates a file is the owner of the file, and the assigned group is the primary group of the user (by default). If the permissions on the file permit reading, writing or executing by the group, any member of the group can exercise those permissions.

The owner of a file can change its group to any other group of which he is a member. The command for this function is **chgrp**. Root can change the group of any file to any value.

The user root is able to change ownership of files using **chown**.

One day jack deletes his **.bashrc**. Oops.

```
[jack@foo jack]$ rm .bashrc
```

So root made another one for him (because jack didn't know how to do it himself, I suspect).

```
[jack@foo jack]$ su
Password:
[root@tonto jack]# cp /etc/skel/.bashrc .
[root@tonto jack]# ls -l .bashrc
-rw-r--r--    1 root     root          124 May  1 14:03 .bashrc
```

Unfortunately, the ownership was incorrect – but that was fixed with **chown**. In fact, **chown** can set the group as well.

```
[root@tonto jack]# chown jack.users .bashrc
[root@tonto jack]# exit
```

If jack doesn't like the file's group being “users”, he can change it.

```
[jack@foo jack]# ls -la .bash*
-rw-------    1 jack     jack        10655 Apr 30 15:38
      .bash_history
-rw-------    1 jack     jack           24 Apr  8 11:45 .bash_logout
-rw-------    1 jack     jack          213 Apr 29 14:58
      .bash_profile
-rw-r--r--    1 jack     users         124 May  1 14:03 .bashrc
[jack@foo jack]$ chgrp jack .bashrc
```

A user can only change the group to another group of which he is a member. Here jack discovers he is not a user.

```
[jack@foo jack]$ chgrp users .bashrc
chgrp: changing group of `.bashrc': Operation not permitted
[jack@foo jack]$ id
uid=513(jack) gid=514(jack) groups=514(jack),101(boneheads)
[jack@foo jack]$ chgrp boneheads .bashrc
[jack@foo jack]$ ls -la .bashrc
-rw-r--r--    1 jack     boneheads      124 May  1 14:03 .bashrc
```

27.2 Default group

There are three distinct ways of setting the group of a new file:

- Set the set-gid bit on the directory
- Use **newgrp** before running the command.
- Change the group after the fact with **chgrp**.

To set the set-gid bit on a directory, use **chmod** after setting the group with **chgrp**.

```
foo:~/tmp $ id
uid=500(jack) gid=500(jack) groups=500(jack),101(boneheads)
foo:~/tmp/test $ mkdir -p tmp/test; cd tmp/test
foo:~/tmp/test $ chgrp boneheads .
foo:~/tmp/test $ ls -la
total 8
drwxrwxr-x    2 jack  boneheads 4096 Apr 30 16:45 .
drwxr-xr-x   12 jack  jack      4096 Apr 30 16:45 ..
foo:~/tmp/test $ touch newfile
foo:~/tmp/test $ ls -la
total 8
drwxrwxr-x    2 jack  boneheads 4096 Apr 30 16:45 .
drwxr-xr-x   12 jack  jack      4096 Apr 30 16:45 ..
-rw-rw-r--    1 jack  jack         0 Apr 30 16:45 newfile
foo:~/tmp/test $ chmod g+s .
foo:~/tmp/test $ touch newfile2
foo:~/tmp/test $ ls -la
total 8
drwxrwsr-x    2 jack  boneheads 4096 Apr 30 16:46 .
drwxr-xr-x   12 jack  jack      4096 Apr 30 16:45 ..
```

```
-rw-rw-r--    1 jack  jack         0 Apr 30 16:45 newfile
-rw-rw-r--    1 jack  boneheads    0 Apr 30 16:46 newfile2
```

Here we do it with **newgrp**. **newgrp** creates a session with a different primary group, which is why we have to "exit".

```
foo:~ $ mkdir -p tmp/newgrp
foo:~ $ cd tmp/newgrp
foo:~/tmp/newgrp $ id
uid=500(jack) gid=500(jack) groups=500(jack),101(boneheads)
foo:~/tmp/newgrp $ touch file
foo:~/tmp/newgrp $ ls -l
total 0
-rw-rw-r--    1 jack  jack         0 Apr 30 16:50 file
foo:~/tmp/newgrp $ newgrp boneheads
foo:~/tmp/newgrp $ id
uid=500(jack) gid=101(boneheads) groups=500(jack),101(boneheads)
foo:~/tmp/newgrp $ touch file2
foo:~/tmp/newgrp $ ls -l
total 0
-rw-rw-r--    1 jack  jack         0 Apr 30 16:50 file
-rw-r--r--    1 jack  boneheads    0 Apr 30 16:50 file2
foo:~/tmp/newgrp $ exit
foo:~/tmp/newgrp $ id
uid=500(jack) gid=500(jack) groups=500(jack),101(boneheads)
```

And of course **chgrp** after the fact would work as well.

27.3 Review

Quiz questions

6. Under what circumstances would one use the command **chmod 2775 directoryname**?
7. Which users may change the group of a file?
8. Which users may change the ownership of a file?

Assignment

1. Create a new user using **useradd username**, but do not create a home directory for the user. Manually copy **/etc/skel** to the user's home directory using **cp -r /etc/skel ~username/.**. Now fix the ownership of the files you copied so that they belong to the user.
2. As a regular user, use the **id** command to see which supplementary groups you are a member of. Create files named after each of the groups you are a member of, with the group set appropriately. Set the permissions on these files to **-rw-rw-----**. Log in as a different member of the group, and see whether you can do the following:
   a) Delete the file.
   b) Rename the file
   c) Edit the file using **vi**
   d) Change the permissions of the file
   e) Change the owner of the file

f) Change the group of the file

3. Change the permissions and ownership of your **/bin/su** so that only members of the group **trusted** can use **su** to become root (you will have to create the group). Check that **su** still works for members of the group and for root. Check that users who are not members of the group **trusted** cannot use **su**.

Answers to quiz questions

1. To set the directory to give its group away to all the files created in it.
2. The owner and root.
3. Only root.

28 Links

> "And we can always supply them with a program that makes identical files into links to a single file."
>
> — *Larry Wall in <199709292012.NAA09616@wall.org>*

LPIC topic 1.104.7 — Create and change hard and symbolic links [1]

Weight: 1

Objective

Candidates should be able to create and manage hard and symbolic links to a file. This objective includes the ability to create and identify links, copy files through links, and use linked files to support system administration tasks.

Key files, terms, and utilities include

ln	Link command for hard and soft (**-s**) links

28.1 Hard links

The data for a file in Linux is located by its *inode number*. If two files have the same inode number, then they refer to the same data[30]. Modifying one file will cause both files to change. This is a hard link. Linux keeps a link count for each file in the filesystem so that it only deletes the data when all the links have been removed.

To create a hard link using the **ln** command the syntax is

```
ln original duplicate
```

or, more simply, to link to an existing file in another location,

```
ln /some/dir/original
```

There are a few provisos when making hard links –

- The permissions and ownership of the new link is the same as the permissions and ownership of the original file. If a user creates a link to a file he does not own, he will retain his "own" copy of that file if the original is deleted.
- You can't make hard links to directories (usually – but you probably shouldn't want to either.)

Hard links can be identified in the output of **ls** by the link count, which is the second column (just after the permissions). For regular files the link count is 1. For files with hard links, the link count is 2 or more. (The link count for directories is based on the number of files in the directory).

```
[jack@foo link]$ echo Hello world > hello
[jack@foo link]$ ls -l
-rw-rw-r--    1 jack     jack           12 May  2 09:07 hello
[jack@foo link]$ ln hello greeting
```

30 Hard links are like cross-linked files in older filesystems – with the difference being that the operating system understands they are supposed to be there.

```
[jack@foo link]$ ls -l
-rw-rw-r--    2 jack     jack           12 May  2 09:07 greeting
-rw-rw-r--    2 jack     jack           12 May  2 09:07 hello
[jack@foo link]$ cat greeting
Hello world
[jack@foo link]$ echo Farewell cruel world > greeting
[jack@foo link]$ cat hello
Farewell cruel world
```

When one of the files pointing to the data is removed, the data remains accessible via the other file.

```
[jack@foo link]$ rm hello
[jack@foo link]$ ls -l
-rw-rw-r--    1 jack     jack           12 May  2 09:07 greeting
[jack@foo link]$ cat greeting
Farewell cruel world
```

When using **ln** to create a link to a file in another directory, specifying the file name is sufficient.

```
[jack@foo link]$ ln ../.bashrc
[jack@foo link]$ ls -la .bashrc
-rw-r--r--    2 jack     boneheads      124 May  1 14:03 .bashrc
[jack@foo link]$ ls -la .bashrc ../.bashrc
-rw-r--r--    2 jack     boneheads      124 May  1 14:03 .bashrc
-rw-r--r--    2 jack     boneheads      124 May  1 14:03 ../.bashrc
```

You cannot create hard links between filesystems.

```
[jack@foo link]$ ln /etc/passwd
ln: creating hard link `./passwd' to `/etc/passwd': Invalid cross-
      device link
```

This example shows how permissions are shared between files with hard links. (Files cannot be removed from **/tmp** because of the sticky bit on the directory permissions).

```
[jack@foo tmp]$ ln /etc/passwd
[jack@foo tmp]$ ln /etc/shadow
[jack@foo tmp]$ ls -ld . passwd shadow
drwxrwxrwt  203 root     root         8192 May  2 09:11 .
-rw-r--r--    2 root     root         2045 Apr 25 10:05 passwd
-r--------    2 root     root         1558 Apr 25 10:05 shadow
[jack@foo tmp]$ cat shadow
cat: shadow: Permission denied
[jack@foo tmp]$ rm -f passwd shadow
rm: cannot remove `passwd': Operation not permitted
rm: cannot remove `shadow': Operation not permitted
```

Using **ls -i** (show inode numbers) you can see that the link and the "original" file share the same inode number, which makes them equivalent.

```
[jack@foo tmp]$ ls -li /tmp/passwd /etc/passwd /tmp/shadow
      /etc/shadow
 214686 -rw-r--r--  2 root   root         2045 Apr 25 10:05
      /etc/passwd
 214684 -r--------  2 root   root         1558 Apr 25 10:05
      /etc/shadow
 214686 -rw-r--r--  2 root   root         2045 Apr 25 10:05
      /tmp/passwd
```

```
 214684 -r--------  2 root   root         1558 Apr 25 10:05
      /tmp/shadow
```

28.2 Symbolic links

To create a soft link using the **ln** command the syntax is

```
ln -s original duplicate
```

or, more simply, to create a symbolic link to an existing file in another location, where the symbolic link has the same name as the original file,

```
ln -s /some/dir/original
```

Symbolic links much like hard links, with the following differences

- Symbolic links can cross filesystems
- Symbolic links can point to directories
- The link is owned by the user that created it
- The destination does not have to exist. A symbolic link becomes useless if the original file is moved, removed or renamed.

Symbolic links are identified in the output of **ls** by a "**l**" in the first column, just before the permissions.

```
[jack@foo link]$ echo Hello world > hello
[jack@foo link]$ ls -l
-rw-rw-r--    1 jack   jack           12 May  2 09:07 hello
[jack@foo link]$ ln -s hello greeting
[jack@foo link]$ ls -l
lrwxrwxrwx    1 jack   jack            5 May  2 09:17 greeting ->
      hello
-rw-rw-r--    1 jack   jack           12 May  2 09:17 hello
[jack@foo link]$ cat greeting
Hello world
[jack@foo link]$ echo Farewell cruel world > greeting
[jack@foo link]$ cat hello
Farewell cruel world
[jack@foo link]$ rm hello
[jack@foo link]$ ls -l
lrwxrwxrwx    1 jack   jack            5 May  2 09:17 greeting ->
      hello
[jack@foo link]$ cat greeting
cat: greeting: No such file or directory
```

When using **ln** to link to a file in another directory, specifying the file name is sufficient.

```
[jack@foo link]$ ln -s ../.bashrc
[jack@foo link]$ ls -l .bashrc ../.bashrc
lrwxrwxrwx    1 jack   jack        10 May  2 09:24 .bashrc ->
      ../.bashrc
-rw-r--r--    1 jack   boneheads  124 May  1 14:03 ../.bashrc
```

You cannot create hard links between filesystems.

```
[jack@foo link]$ ln -s /etc/passwd
[jack@foo link]$ ls -l passwd
lrwxrwxrwx    1 jack   jack        11 May  2 09:25 passwd ->
      /etc/passwd
```

There is a limit to the number of symbolic links that Linux will follow before arriving at the file.

```
[jack@foo link]$ touch zero
[jack@foo link]$ ln -s zero one
[jack@foo link]$ ln -s one two
[jack@foo link]$ ln -s two three
[jack@foo link]$ ln -s three four
[jack@foo link]$ ln -s four five
[jack@foo link]$ ln -s five six
[jack@foo link]$ ls -la
drwxrwxr-x    2 jack     jack         4096 May  2 09:05 .
drwx------   26 jack     jack         4096 May  2 09:04 ..
lrwxrwxrwx    1 jack     jack            4 May  2 09:05 five -> four
lrwxrwxrwx    1 jack     jack            5 May  2 09:05 four ->
      three
lrwxrwxrwx    1 jack     jack            4 May  2 09:05 one -> zero
lrwxrwxrwx    1 jack     jack            4 May  2 09:05 six -> five
lrwxrwxrwx    1 jack     jack            3 May  2 09:05 three -> two
lrwxrwxrwx    1 jack     jack            3 May  2 09:05 two -> one
-rw-rw-r--    1 jack     jack            0 May  2 09:05 zero
[jack@foo link]$ cat *
cat: six: Too many levels of symbolic links
```

28.3 Review

Quiz questions

9. How does the output of **ls** identify a hard link?
10. How can you use **ls** to identify which two files are hard links to each other?
11. Why does an empty directory show contain 2 links?
12. Which flag for **ln** will request it to overwrite an existing file or link?
13. What happens when you delete a file to which a symbolic link points?
14. What happens when you delete a file to which a hard link points?

Assignment

1. Create a symbolic link to **/etc/profile**. Copy the link using **cp**. Do you get a regular file or a symbolic link? What option to **cp** changes this?
2. Create symbolic links in your **/etc/skel** directory pointing to **/usr/share/doc** and **/etc**. What effect does this have when you add a new user using **useradd** or **adduser**? Does the user get a symbolic link or a regular file?
3. Run the commands below, and find the reason for the output:

```
[jack@foo link]$ touch 0
[jack@foo link]$ for ((a=1;a<10;a++)) ; do ln -s $((a-1)) $a; done
[jack@foo link]$ ls -l
```

Hint: check which of the files you can read...

4. Copy **/etc/hosts** to your home directory. Create a hard link to this file named **link1**. Now

create a hard link to **link1** named **link2**. Change the permissions of **link1** to 0600. What happens to the other files? What happens if you change the group of the link **link2**? What happens to the original file and to **link2** if you delete **link1**? Explain why the behaviour is as you observed.

5. Create a regular file named **symlink** using **ls > symlink**. Now create a symbolic to the current directory named **symlink** using a single command. What happens to the previous contents of the file **symlink**?

Answers to quiz questions

1. The second column contains a number other than 1.
2. ls -i shows inode numbers. If the inode numbers are the same, the files are the same.
3. The link count for the directory is the number of inodes in it – one for "." and one for "..".
4. -f
5. The symbolic link is unchanged, but points to nowhere.
6. The other part of the hard link remains the same.

29 Finding files

> Making files is easy under the UNIX operating system. Therefore, users tend to create numerous files using large amounts of file space. It has been said that the only standard thing about all UNIX systems is the message-of-the-day telling users to clean up their files.
>
> *– System V.2 administrator's guide*

With hard disk sizes getting awfully big, it is wonderful to have an orderly arrangement of where-data-goes. The basic separations are between program code and data, between static data and dynamic data, and between shareable files and non-shareable files.

LPIC topic 1.104.8 — Find system files and place files in the correct location [5]

Weight: 5

Objective

Candidates should be thoroughly familiar with the Filesystem Hierarchy Standard, including typical file locations and directory classifications. This objective includes the ability to find files and commands on a Linux system.

Key files, terms, and utilities include

find	find by searching the filesystem
locate	find by querying the locate database
slocate	secure version of locate (enforce user permissions)
updatedb	update locate's database
whereis	look in common locations for a file
which	find the path name of a command
/etc/updatedb.conf	configuration file for slocate

29.1 Filesystem hierarchy standard

The Filesystem hierarchy standard (FHS) is a collaborative document that defines the names and locations of many files and directories in Linux systems[31]. The FHS guidelines are intended to support interoperability of applications, system administration tools, development tools, and scripts as well as greater uniformity of documentation for these systems.

If you examine the root filesystem of your computer, you will find the following directories, most of which are specified by the FHS:

```
jack@foo:~ > ls /
bin  boot  dev  etc  home  mnt  opt  proc  root  sbin  srv  tmp  usr
     var
```

The root hierarchy

- **/** — The root directory. Everything is “in” this directory.

31 Other UNIX systems do not enjoy the same level of standardisation as Linux does.

- **/etc** — This directory contains configuration files for the system. Many programs have their own subdirectory of **/etc** for their configuration files, such as **/etc/X11**.
- **/home**, **/root** — Each user has a directory here, where they can store their files. The super-user root has a separate directory outside the **/home** directory to maintain the separation between users and administrators.
- **/bin**, **/sbin**, **/lib** — These are the directories containing the essential programs for booting the system. Program files (binaries) are stored in directories named **bin**. The main **bin** directory contains files which are required by scripts (and also the script interpreter **/bin/sh**). The **sbin** directory contains supervisor binaries, such as network configuration and testing programs. The essential library files are stored in **/lib**.
- **/usr** — This is the secondary hierarchy of the filesystem. Just about everything ends up in **/usr** – everything users need while operating the system. In the **/usr** directory, there are sub-directories for binaries (**/usr/bin**), supervisor binaries (**/usr/sbin**), and library files (**/usr/lib**). For example, **/usr/bin** contains programs, similar to **/bin**, but not vital to the system's basic operation.
- **/var** — Variable files that are modified by programs are stored here including, log files and files in transit.
- **/opt** — Complete add-on applications like netscape and kde tend to end up here. Each program is stored neatly in its own sub-directory, not overtly interfering with the rest of the system.
- **/mnt**, **/cdrom**, **/floppy** — These directories are used for file systems that are mounted temporarily. To access a CD-ROM, you run mount /cdrom and its files appear at /cdrom.
- **/dev** — The “files” in **/dev** represent devices which may be attached to the system, such as hard disks, etc. The files are generally character devices and block devices.
- **/boot** — Files used by the boot loader (e.g. LILO or GRUB).
- **/proc** — Originally for providing an interface to **proc**ess information, **/proc** now provides a general interface to the kernel. The **/proc** interface is used to set networking and virtual memory parameters, for example.

In addition to specifying directories, the filesystem hierarchy standard also specifies the presence of specific files in specific directories. Here are a few ...

- **/bin** – cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, false, hostname, kill, ln, login, ls, mkdir, mknod, more, mount, mv, ps, pwd, rm, rmdir, sed, sh, stty, su, sync, true, umount, uname.
- **/etc** - csh.login, exports, fstab, ftpusers, gateways, gettydefs, group, host.conf, hosts, hosts.allow, hosts.deny, hosts.equiv, hosts.lpd, inetd.conf, inittab, issue, ld.so.conf, motd, mtab, mtools.conf, networks, passwd, printcap, profile, protocols, resolv.conf, rpc, securetty, services, shells, syslog.conf (if the associated programs are installed).
- **/sbin** – fastboot, fasthalt, fdisk, fsck, fsck.*, getty, halt, ifconfig, init, mkfs, mkfs.*, mkswap, reboot, route, swapon, swapoff, update (all of these are optional, but usually do appear).

The /usr hierarchy

If a system is FHS compliant, it is possible to mount **/usr** as a read-only filesystem. This means it is possible to share a single **/usr** filesystem using NFS (since there is no conflict between systems wanting to make modifications). Apart from essential files for booting up, the bulk of the files making up a standard installation appear in **/usr**.

- **/usr/bin**, **/usr/sbin**, **/usr/lib** – Binaries that are not essential to the system starting up appear under the **/usr** tree. Again, **bin** is for everyone, **sbin** for the supervisor, and **lib** is for the libraries that these programs require.
- **/usr/local** – this is the third hierarchy, for files which are used only by the local site – e.g. programs compiled from source code, local scripts and such like. **/usr/local** is often used for installing plug-in applications like **/opt** is. In most distributions, **/usr/local** is untouched by the standard packaging tools.
- **/usr/X11R6** – XFree86 is installed here, with its own subdirectories **/usr/X11R6/bin** and **/usr/X11R6/lib**. XFree86 is a large program which *could* be installed under **/opt**, but it never is.
- **/usr/games** – various games make their own files under this directory, for high scores, etc.
- **/usr/include** – Include files for compiling from source code are stored in this directory. If you install the development packages for various libraries (e.g. glibc-devel) this directory will be populated with files.
- **/usr/src** – This is where program source code is stored. The kernel source code is often found in **/usr/src/linux**. If source code is to be modified, this directory is often implemented as a separate and writeable partition (since **/usr** should be read-only).
- **/usr/share** – This contains shared data files, such as documentation which are not specific to the architecture of the installation. As the name implies, this means that the data can be *shared* between different Linux platforms.

The /var hierarchy

The **/var** hierarchy is for things that change during the running of the system. This includes mail and printer spool directories, administrative and logging data, and temporary files.

- **/var/lib** – state information for programs. Programs such as logrotate, NFS client utilities and the PCMCIA utilities store their state here.
- **/var/lock** – Lock files. Lock files are a way for programs accessing a single resource to make sure that they don't fight with each other.
- **/var/spool** – Various message queues go here, such as printer queues,
- **/var/log** – Log files (what has been happening). The most important log files are **/var/log/messages** and **/var/log/mail** or **/var/log/maillog**.
- **/var/run** – Data used by running processes. Mostly this contains files with the current process ID (PID) of processes (such as **/var/run/crond.pid**).
- **/var/tmp** – This is very similar to **/tmp**, but files might not be deleted quite as often.

29.2 find

The **find** command searches the specific directories for files that match your criteria.

```
find [dir1 dir2 ...] [criteria]
```

To find a file on Linux, you often know the name of the file, and something about its location. To use this information with **find** you would use the **-name** criterion, or **-iname** (case insensitive).

```
find / -name hosts.deny
find /usr/share/ -iname "*mail*"
```

The **find** command is discussed in more detail in the chapter on "File Management".

29.3 locate

Although **locate** is not installed by default on many distributions, it is a useful tool. When **locate** is installed and set up, you can use it something like this:

```
jack@foo:~> locate motd
/etc/motd
/lib/security/pam_motd.so
/usr/share/man/man5/motd.5.gz
```

You may notice that **locate** runs significantly faster than **find**. The reason for this is that **locate** does not search the entire filesystem, but searches a database which was created by **updatedb**. This works out significantly faster. The one drawback is that the database is not always up to date. If you try to find a file that was created recently, it will not be in the database.

```
locate COPYING
locate COPYING | grep findutils
locate printcap
locate locate
locate /users
```

The command that updates the **locate** database is **updatedb**.

```
touch averynewfile
su
updatedb
exit
locate averynewfile
```

29.4 slocate

slocate is just like **locate**, except that it records the permissions and ownership of each file so that users cannot see files they do not have access to. (**locate** achieves the same result by running **updatedb** as the user nobody when run from the cron.daily script.)

In order to have permissions to read the database, which contains privileged information, **slocate** is a set-gid application. If your system has **slocate** installed, you can still use the **locate** command, which is simply a symbolic link to **slocate**. **updatedb** is also a link to **slocate**, but it has the good sense to update the database when you run it in this way.

```
[jack@foo jack]$ ls -l /usr/bin/{slocate,locate,updatedb}
```

```
lrwxrwxrwx 1 root  slocate     7 Oct 14 2002 /usr/bin/locate ->
      slocate
-rwxr-sr-x 1 root  slocate 31661 Jun 24 2002 /usr/bin/slocate
lrwxrwxrwx 1 root  slocate     7 Oct 14 2002 /usr/bin/updatedb ->
      slocate
```

When **slocate** updates its database, it uses the file **/etc/updatedb.conf**. This file explains which directories and filesystem types should be excluded from the **slocate** database[32].

```
[jack@foo jack]$ cat /etc/updatedb.conf
PRUNEFS="devpts NFS nfs afs proc smbfs autofs auto iso9660"
PRUNEPATHS="/tmp /usr/tmp /var/tmp /afs /net"
export PRUNEFS
export PRUNEPATHS
```

The **updatedb** command has to be run by root. It is run periodically (usually daily by **cron**), and it takes quite a while.

```
[jack@foo jack]$ updatedb
fatal error: updatedb: You are not authorized to create a default
      slocate database!
jack@foo jack]$ su
Password:
[root@foo jack]# updatedb
```

29.5 Finding files with whereis

The **whereis** command searches a list of commonly used locations for the file name specified. It tends to find man pages, configuration files and similar lost objects.

```
[jack@foo jack]$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
[jack@foo jack]$ whereis whereis
whereis: /usr/bin/whereis /usr/share/man/man1/whereis.1.gz
[jack@foo jack]$ whereis hosts
hosts: /etc/hosts.allow /etc/hosts.deny /etc/hosts /etc/hosts.bak
      /usr/share/man/man5/hosts.5.gz
```

Here are some **whereis** commands to work through.

```
whereis bash
whereis passwd
whereis apache
whereis httpd
whereis netscape
whereis inetd.conf
whereis password
whereis wally
```

29.6 Finding programs with which

When you enter a command, the shell searches the directories listed in the **$PATH** environment variable for the program to run. The command **which** searches for, and prints out the location of the command which is executed.

32 This same file is used by the original GNU **locate**.

```
[jack@foo jack]$ echo $PATH
/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/home/jack/
      bin
```

Here's a few examples of using **which** for satisfying idle curiosity.

```
[jack@foo jack]$ /usr/bin/which ls
/bin/ls
[jack@foo jack]$ /usr/bin/which pwd
/bin/pwd
[jack@foo jack]$ /usr/bin/which echo
/bin/echo
[jack@foo jack]$ /usr/bin/which cd
/usr/bin/which: no cd in
      (/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/home
      /jack/bin)
```

Many distributions define useful or unusual aliases for **which**. The "redhat" example below corresponds with the recommended usage from the **which** man page.

```
jack@suse:~> alias which
alias which='type -p'
jack@redhat:~> alias which='alias | /usr/bin/which --tty-only --
      read-alias --show-dot -show-tilde'
```

One of the most useful applications for which is together with **rpm**, to find the package which provides a specific command.

```
[jack@foo jack]$ rpm -qf $( which ppmtolj )
netpbm-10.5-66
```

29.7 Review

Quiz questions

15. What is the distinction between **/usr/bin** and **/bin**?
16. What determines whether an executable is in **/usr/bin** or **/usr/sbin**?
17. When should files appear in **/usr/local**?
18. Why is **/lib** separate from **/usr/lib**?
19. What is the relationship between the **which** command and the **PATH** environment variable?

Assignment

1. Write down where would you expect to find the following files, and them find them using the tools listed in this chapter.
   a) **cat** (show file contents)
   b) **ls** (show directory listing)
   c) **traceroute** (find route to given network)
   d) **ifconfig** (set up network interface configuration)
   e) **syslog.conf** (configuration file for **syslogd**)

f) **cat.1.gz** (man page for the **cat** command)

g) **messages** (growing file of various system events)

2. Write a command to find all the files in **/etc** which are smaller than 12kb. Write a command to list all files in **/etc** which are larger than 12kb. Do the two lists comprise all the files in the **/etc** directory?
3. Download the filesystem hierarchy standard document from the internet or find it on your distribution media. Read it and make notes.

Answers to quiz questions

1. **/bin** contains files which are essential for booting when **/usr** may not be available.
2. The contents of **/bin** is standardised, but anything can appear in **/usr/bin**.
3. When the files have been prepared for local conditions, e.g. customised programs, etc.
4. boot time vs run time when **/usr** is available.
5. which searches the directories in the **PATH** variable.

30 XFree86

> If the designers of X-window built cars, there would be no fewer than five steering wheels hidden about the cockpit, none of which followed the same principles – but you'd be able to shift gears with your car stereo. Useful feature, that.
>
> *– From the programming notebooks of a heretic, 1990.*

LPIC topic 1.110.1— Install & Configure XFree86 [5]

Weight: 5

Objective

Candidate should be able to configure and install X and an X font server. This objective includes verifying that the video card and monitor are supported by an X server, as well as customizing and tuning X for the video card and monitor. It also includes installing an X font server, installing fonts, and configuring X to use the font server (may require a manual edit of /etc/X11/XF86Config in the "Files" section).

Key files, terms, and utilities include

XF86Setup	ncurses based X configuration tool
xf86config	command line X configuration tool
xvidtune	video mode tuning (for modeline)
/etc/X11/XF86Config	default X server configuration file
.Xresources	

30.1 X11 architecture

The X Window System[33] is the display system used by Linux and other Unix systems and hardware. The X11 architecture consists of a number of elements:

- X servers, which have a display (screen), input devices (keyboard and pointer devices) and other resources. On Linux, the most commonly used program which runs as an X server is called XFree86[34]. There are now two flavours of this – the original XFree86 project, and the xorg.org server. Commercial X servers are also available, but are seldom used.
- X clients, which are programs capable of displaying their output on a X server's display. It is possible to run programs on a number of machines distributed over the network, with their output displayed on one X server. X clients include xterm, xeyes, OpenOffice, Gnome and KDE.
- The X11 protocol, by which X clients talk to an X server. The X11 protocol works over networks as well as over the local connection (TCP/IP and UNIX sockets are supported). Because the X Window System uses a well defined protocol, X applications are

33 And don't you forget it. It's the X Window System. It's not X Window**s**. Some overeducated psychopathical purist will have your head for calling it anything "Windows". Be careful out there.

34 There is an intentional pun here – it's **X** for the 80**386**. Actually, XFree86 is used on more than just 80386 based machines these days.

independent of the graphics hardware on which they rely.

While X client applications generally require no special configuration to work with an X server, installing an X server can be quite difficult. The problematic aspects are usually:

- Display adapter configuration (choosing software to match your hardware).
- Configuration for monitor (setting up the software to match the monitor, usually without the monitor's specifications).
- Font installation

30.2 X server

The task of configuring an X server means setting up the following:

- Software installation (XFree86, version 3.x or 4.x)
- Graphics card
- Monitor
- Keyboard
- Mouse
- Other input devices (joystick, graphics tablet, etc).

Configuration programs

The following programs are capable of configuring XFree86:

- **xf86config** – a text based program which is part of the XFree86 distribution. **xf86config** asks a number of questions, and then creates a configuration file in accordance with the supplied answers.
- **XF86Setup** – this is a graphical program which is distributed with XFree86 version 3.x. It runs in graphical mode (16 colours, 640x480), and emits a configuration file based on your choices. After creating the configuration file, the X server is started using your choices, allowing you to test whether it works.
- **Xconfigurator** – this is the configuration program for Red Hat Linux. It is an improved version of **xf86config** which scans the PCI bus to see which graphics card is installed.
- **dexconf** – the Debian distribution uses this configuration program. It is a menu based program which emits a configuration file according to the chosen settings – similar to **xf86config**. To run this program once Debian is installed, the command is **dpkg-reconfigure xserver-xfree86**.
- **sax** and **sax2** – the SuSE X configuration programs, based on XF86Setup with some additional PCI bus scanning. **sax** is used for configuring version 3.x of the X server, and **sax2** for version 4.x of the X server.
- **X -configure** – XFree86 version 4.x will generate a configuration file according to what it autodetects in your hardware.

Things to configure

- Keyboard – the relevant information is the number of keys (pc101, etc), language (US,

UK), options

```
Please select one of the following keyboard types that is the better
description of your keyboard. If nothing really matches,
choose 1 (Generic 101-key PC)

  1  Generic 101-key PC
  2  Generic 102-key (Intl) PC
  3  Generic 104-key PC
```

- Mouse – where the mouse is connected (serial, PS/2 port, USB), type of mouse and the protocol used (Microsoft, PS/2, imps/2 wheel mouse), number of buttons, 3-button emulation. If you set up a wheel mouse, the wheel corresponds to the "z-axis mapping".

  Choosing the mouse protocol

```
First specify a mouse protocol type. Choose one from the following
      list:

 1.  Microsoft compatible (2-button protocol)
 2.  Mouse Systems (3-button protocol)
 3.  Bus Mouse
 4.  PS/2 Mouse
```

  The country or language specification refers to the keyboard language, rather than the language you actually speak.

```
  1  U.S. English
  2  U.S. English w/ ISO9995-3
  3  U.S. English w/ deadkeys

Enter a number to choose the country.
Press enter for the next page
```

  Emulate3Buttons is an excellent idea if you have a two-button mouse. ChordMiddle is seldom useful, unless you have a mouse that requires this because of its design.

```
You have selected a two-button mouse protocol. It is recommended
      that you
enable Emulate3Buttons.

Please answer the following question with either 'y' or 'n'.
Do you want to enable Emulate3Buttons?
```

  The mouse is usually connected to one of **/dev/ttyS0**, **/dev/psaux**. Whatever the correct setting is **/dev/mouse** is generally set up to be a symbolic link to the actual device.

```
Now give the full device name that the mouse is connected to, for
      example
/dev/tty00. Just pressing enter will use the default, /dev/mouse.
```

- Graphics card – For version 3.x it is necessary to specify the X server to use (although XF86SVGA supports most of the supported graphics cards). Newer hardware is a great deal easier to configure and you will not have to enter ClockChip values, which are nearly impossible to obtain.

  The XFree86 server supports an enormous range of graphics cards.

```
  0  2 the Max MAXColor S3 Trio64V+                    S3 Trio64V+
  1  2-the-Max MAXColor 6000                           ET6000
```

```
  2  3DLabs Oxygen GMX                                   PERMEDIA 2
  3  3DVision-i740 AGP                                   Intel 740
  4  3Dlabs Permedia2 (generic)                          PERMEDIA 2
Enter a number to choose the corresponding card definition.
Press enter for the next page, q to continue configuration.
```

By default, XFree86 will detect the amount of video memory you have on your card. Just occasionally, it can't and you have to tell it manually.

```
How much video memory do you have on your video card:
 1  256K
 2  512K
 3  1024K
 4  2048K
 5  4096K
 6  Other
```

- Monitor – older monitors *can* be destroyed by incorrect settings for horizontal and vertical sync range. These settings are used as protection against the X server asking the video card to drive the monitor too hard. Unfortunately, you have to set your own levels of protection when you configure the system.

  The horizontal sync specification is the rate at which the monitor moves from one pixel to the next.

```
    hsync in kHz; monitor type with characteristic modes
 1  31.5; Standard VGA, 640x480 @ 60 Hz
 2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
 3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no
      800x600)
 4  31.5, 35.15, 35.5; SVGA, 1024x768@ 87 Hz interlaced, 800x600 @
      56 Hz
 5  31.5 - 37.9; Extended SVGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
 6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72
      Hz
 7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
```

  The vertical sync range is the rate at which your monitor is capable of drawing the screen.

```
 1  50-70
 2  50-90
 3  50-100
 4  40-150
 5  Enter your own vertical sync range
```

Symbolic link /usr/X11R6/bin/X

The command **X** is used to start the X server[35], and is what ends up running at some stage after running **startx**. For version 4.x, each video driver is supplied as an independent module, which is loaded when X is run. For version 3.x, a large number of video drivers were provided in a single precompiled binary – the most widely used being XF86SVGA. Because of this, it was necessary to set up **X** as a link to the correct binary.

For version 4.x, **X** is a link to XFree86 binary. For version 3.x, **X** must be set to point to the X

35 For XFree86 version 3.x, the command **X** can only be run by root. Other users must run **Xwrapper**.

server with compiled-in support for your particular hardware.

```
gabriel:~ $ which X
/usr/X11R6/bin/X
gabriel:~ $ ls -o /usr/X11R6/bin/X
lrwxrwxrwx 1 root 16May  7  2002 /usr/X11R6/bin/X ->
      /var/X11R6/bin/X
gabriel:~ $ ls -o /var/X11R6/bin/X
lrwxrwxrwx 1 root 22May 7 2002 /var/X11R6/bin/X ->
      /usr/X11R6/bin/XFree86
```

30.3 Configuration file

The configuration file for XFree86 is **/etc/X11/XF86Config**. A number of other configuration file locations are also possible, although these are used only in special circumstances. The complete list is give in the XF86Config(5x) man page. If both version 3.x and version 4.x of XFree86 are installed, the file **/etc/X11/XF86Config-4** is used by version 4.x, while version 3.x uses **/etc/XF86Config**.

The XF86Config configuration file consists of the following sections:

30.3.1 Version 3.x

For version 3.x, the following sections may appear:

- Files (File pathnames) - location of fonts and font servers

```
Section "Files"
   FontPath        "unix/:7100"         # local font server
   # if the local font server has problems, we can fall back on
      these
...snip...
   FontPath        "/usr/lib/X11/fonts/75dpi"
EndSection
```

- Module (Dynamic module loading) – in version 3.x modules are only used for certain input devices.
- ServerFlags (Server flags) – this section sets X server options, some of which are helpful in debugging (e.g. with AllowMouseOpenFail set, the server will start if the mouse is misconfigured).
- Keyboard (Keyboard configuration) – Specifies the keyboard input device, parameters and mapping options.

```
Section "Keyboard"
        Protocol        "Standard"
        XkbRules        "xfree86"
        XkbModel        "pc104"
        XkbLayout       "us"
EndSection
```

- Pointer (Pointer configuration) - Mouse or other pointing device configuration. The two most important parameters in this section are the **Protocol** parameter and the **Device** parameter (e.g. **/dev/mouse**).

```
Section "Pointer"
```

```
        Device          "/dev/gpmdata"
        Protocol        "Microsoft"
        Emulate3Buttons
        ZAxisMapping    4 5
EndSection
```

- Monitor (Monitor description) – one or more “Monitor” sections provide the specifications of a monitor and list the video modes which can be used. Important parameters in this section are ...
  HorizSync 30-57 – the horizontal frequency (kHz) range supported by the monitor
  VertRefresh 43-72 – the vertical refresh rate (Hz) range supported by the monitor
  ModeLine “800x600” ... – the exact timing for the mode.

```
Section "Monitor"
        Identifier      "Generic Monitor"
        VendorName      "Generic"
        ModelName       "Monitor"
        HorizSync       30-57
        VertRefresh     43-72
# 640x480 @ 60Hz (Industry standard) hsync: 31.5kHz
ModeLine "640x480"  25.2 640 656 752 800    480 490 492 525 -hsync
      -vsync
# 800x600 @ 56Hz (VESA) hsync: 35.2kHz
ModeLine "800x600"  36.0 800 824 896 1024   600 601 603 625 +hsync
      +vsync
...snip...
EndSection
```

- Device (Graphics device description) – this defines the video adapter (e.g. VGA display adapter). In the example below, the graphics card is automatically detected, because support for it is compiled into the X server.

```
Section "Device"
        Identifier      "Generic Video Card"
        VendorName      "Generic"
        BoardName       "Video Card"
EndSection
```

- Screen (Screen configuration)

```
Section "Screen"
        Driver                  "SVGA"
        Device                  "Generic Video Card"
        Monitor                 "Generic Monitor"
        DefaultColorDepth       15
        SubSection "Display"
                Depth           15
                Modes           "1024x768" "800x600" "640x480"
        EndSubSection
EndSection
```

- XInput (Extended Input devices configuration) – this is optional, and used for strange input devices.

30.3.2 Version 4.x

In the configuration file for XFree86 version 4.x there is a new top level section, the

"ServerLayout" section. The "Keyboard" and "Pointer" sections have been replaced by "InputDevice" sections. The "Device" section has been supplemented by module-specific sections, "DRI" and "VideoAdapter".

```
Section "Files"          File pathnames
Section "ServerFlags"    Server flags
Section "Module"         Dynamic module loading
Section "InputDevice"    Input device description
Section "Device"         Graphics device description
Section "VideoAdaptor"   Xv video adaptor description
Section "Monitor"        Monitor description
Section "Modes"          Video modes descriptions
Section "Screen"         Screen configuration
Section "ServerLayout"   Overall layout
Section "DRI"            DRI-specific configuration
Section "Vendor"         Vendor-specific configuration
```

- ServerLayout section – what to include when starting the X server. This section lists the input and output devices to be used. Each input device is described in its own "InputDevice" section. The components of output devices (e.g. graphics board with a "Device" section, and a monitor with a "Monitor" section) are bound together in a "Screen" section which is listed in the "ServerLayout" section.

```
Section "ServerLayout"
        Identifier      "Default Layout"
        Screen          "Default Screen"
        InputDevice     "Generic Keyboard"
        InputDevice     "Configured Mouse"
        InputDevice     "Generic Mouse"
EndSection
```

- InputDevice section – for a mouse or a keyboard. Here's the "InputDevice" section for a Keyboard.

```
Section "InputDevice"
        Identifier      "Generic Keyboard"
        Driver          "keyboard"
        Option          "CoreKeyboard"
        Option          "XkbRules"      "xfree86"
        Option          "XkbModel"      "pc104"
        Option          "XkbLayout"     "us"
EndSection
```

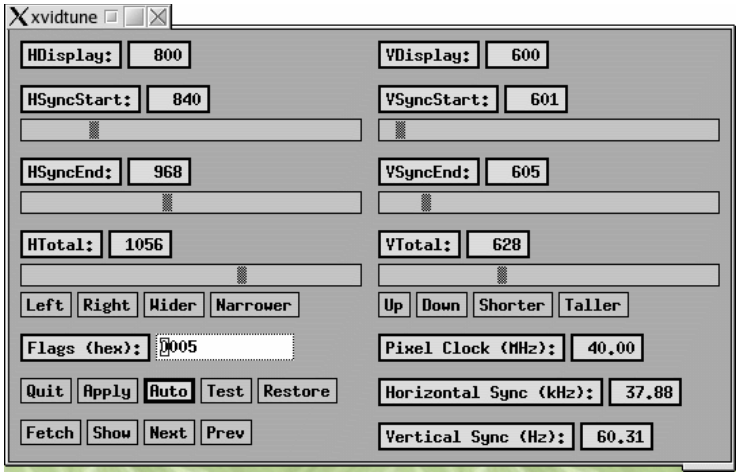
  Here's the "InputDevice" section for a mouse.

```
Section "InputDevice"
        Identifier      "Configured Mouse"
        Driver          "mouse"
        Option          "CorePointer"
        Option          "Device"                "/dev/ttyS0"
        Option          "Protocol"              "Microsoft"
        Option          "Emulate3Buttons"       "true"
        Option          "ZAxisMapping"          "4 5"
EndSection
```

The "ModeLine" configuration lines usually are not necessary for configuring XFree86 version 4.x since the standard VESA modes are built into the server. As a result, the configuration file looks a great deal simpler, and the "ModeLine" parameters are omitted.

30.4 Video card and monitor tuning

The **xvidtune** utility produces tuned mode lines for your monitor. It looks something like this,



Controls:

- Left, Right, Wider, Narrower – adjust the horizontal position and size of the display
- Up, Down, Shorter, Taller – adjust the vertical position and size of the display.
- Test – try the settings and see what the effect is.
- Auto – try the settings as you make changes. This usually leads to an unusable display.
- Show - When you click "Show" **xvidtune** prints output which is appropriate for the **Modeline** parameter in the XF86Config file.

```
Vendor: Monitor Vendor, Model: Monitor Model
Num hsync: 1, Num vsync: 1
hsync range 0:  31.50 -  48.50
vsync range 0:  50.00 -  70.00
"800x600"   40.00  800  840  968 1056    600  601  605  628 +hsync
     +vsync
```

The corresponding entry in **XF86Config** has the keyword "ModeLine" added to the front.

```
ModeLine "800x600" 40.00  800 840 968 1056  600 601 605 628 +hsync
     +vsync
```

It is not always possible to tune your display using **xvidtune**, the usual reason being that the Horizontal Sync and Vertical Sync ranges are incorrect for your monitor.

30.5 Installing fonts

Fonts in X have long and unwieldy names specifying the exact details of the font, separated by "-".

```
-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1
```

Font names can contain the "*" wildcard, allowing the above to be abbreviated to the equally obscure version below.

```
-*-courier-medium-r-normal--*-100-*-*-*-*-iso8859-1
```

xlsfonts lists fonts that meet a specific font specification.

```
% xlsfonts -fn -*-courier-medium-r-normal--*-100-*-*-*-*-iso8859-1
```

```
-adobe-courier-medium-r-normal--10-100-75-75-m-0-iso8859-1
-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1
-adobe-courier-medium-r-normal--14-100-100-100-m-90-iso8859-1
-bitstream-courier-medium-r-normal--10-100-75-75-m-0-iso8859-1
-bitstream-courier-medium-r-normal--10-100-75-75-m-0-iso8859-1
-urw-courier-medium-r-normal--10-100-75-75-p-0-iso8859-1
```

To install a new font in X:

1. Make a new directory for the fonts, either under **/usr/X11R6/lib/X11**, or under **/usr/local/fonts**.
2. Copy the font files into this directory.
3. Run **mkfontdir** or similar in this directory.
4. Include this directory in the "FontPath" element in the Files section of the XF86Config file, or the "catalogue" value in **/etc/X11/fs/config**, or use **xset +fp pathname**.

In the font directory, the following files may exist (and are made by mkfontdir):

- **fonts.dir** – bitmap fonts directory, made by **mkfontdir** or **ttmkfdir > fonts.scale**
- **fonts.scale** – scalable fonts directory, made by **mkfontscale**
- **fonts.alias** – font alias names (you can make this directly)

To temporarily add a directory to the running X font server font path using **xset**, the usage is

```
xset +fp directory
xset fp+directory
```

30.6 X font server

The font server does the hard work of translating a font from the many available formats to a bitmap which can be displayed on the screen. Here's the configuration file **/etc/X11/fs/config**.

```
# Turn off TCP port listening (Unix domain connections are
# still permitted)
no-listen = tcp
# paths to search for fonts
catalogue =
      /usr/lib/X11/fonts/misc/,/usr/lib/X11/fonts/cyrillic/,/usr/lib/
      X11/fonts/100dpi/:unscaled,/usr/lib/X11/fonts/75dpi/:unscaled,/
      usr/lib/X11/fonts/Type1/,/usr/lib/X11/fonts/CID,/usr/lib/X11/fo
      nts/Speedo/,/usr/lib/X11/fonts/100dpi/,/usr/lib/X11/fonts/75dpi
      /
# For fonts that don't specify a resolution, use something default.
default-point-size = 120
# x1,y1,x2,y2,...
default-resolutions = 100,100,75,75
```

Many X servers may share a single font server. In **/etc/X11/XF86Config**, the location of fonts is given in the **"Files"** section:

```
Section "Files"
  FontPath     "/usr/X11R6/lib/X11/fonts/misc:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/local"
  FontPath     "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
```

The use of a font server is configured by specifying "tcp" and a port number:

```
FontPath "tcp/xserver:7100"
FontPath "tcp/xserver:7101"
```

It is quite common to use 127.0.0.1 as the font server, or to use the “unix” socket,

```
FontPath "unix/:7100"
```

30.7 Review

Quiz questions

The answers to these questions appear in this chapter.

1. Is **xterm** an X client application, or an X server?
2. What are the physical meanings of the **HorizSync** and **VertRefresh** parameters in the **XF86Config** file?
3. What is the purpose of an X font server?
4. What must you do to install a new font for your X server?

Assignment

1. Set up an X font server, and configure your X server to use it.
2. Find a font on the internet (search for “truetype font download”) and install it into your system’s X font server. Can you use the font you installed?
3. Configure your X server from scratch using **xf86config**. Set your graphics card to run your monitor at the highest resolution and refresh rate possible. Increase the number of available colours for your X server, until you have the most colours possible for your video card.
4. Install and configure the version of XFree86 that you do not currently have installed – i.e. switch from XFree86 version 3.x to 4.x or visa versa.

Answers to quiz questions

1. Client
2. Limits of pixel to pixel clock and screen redraw frequency.
3. To translate vector fonts into bitmaps (pictures) for use by an X server.
4. Set the font path to include the directory, and create the appropriate directory file.

31 X display manager

I have never seen anything fill up a vacuum so fast and still suck.

– Rob Pike, on X.

Steve Jobs said two years ago that X is brain-damaged and it will be gone in two years. He was half right.

– Dennis Ritchie

Dennis Ritchie is twice as bright as Steve Jobs, and only half wrong.

– Jim Gettys

(/usr/share/games/fortune/computers)

LPIC topic 1.110.2 — Setup a display manager [3]

Weight: 3

Objective

Candidate should be able to setup and customize a Display manager. This objective includes turning the display manager on or off and changing the display manager greeting. This objective also includes changing default bitplanes for the display manager. It also includes configuring display managers for use by X-stations. This objective covers the display managers XDM (X Display Manger), GDM (Gnome Display Manager) and KDM (KDE Display Manager).

Key files, terms, and utilities include:

/etc/inittab	Configuration file for init which may start xdm
/etc/X11/xdm/*	XDM configuration
/etc/X11/kdm/*	KDM configuration
/etc/X11/gdm/*	GDM configuration

31.1 What is a display manager

If a system has X installed, it can generally be started by a regular user logged into the console with the command **startx**. **startx** starts an X server on the local machine, and then runs **~/.xinitrc** (or **/var/X11R6/lib/xinit/xinitrc** if the user does not have a **.xinitrc** file).

The display manager makes it possible to rearrange things slightly.

Console login and startx	*Display manager login*
1. **startx** starts X server	1. **xdm**, **kdm** or **gdm** started by boot scripts
2. **startx** runs **~/.xinitrc** for the user	2. X server started by display manager
3. Run the window manager	3. Set up session (change ownership of console devices, etc)
	4. Start selected window manager (**kdm** and **gdm**) or run **~/.xsession** for user

After the X server is started, the display manager is run by root, and presents a login window to users.

For remote displays, **xdm** and other display managers do the following:

- Wait for a XDMCP query from a X server requesting a session on its display. This will only happen if the display manager is listening for TCP connections (the initial connection is actually UDP).
- Display a menu of available machines to log in to (the chooser). This happens when the remote machine uses **X -indirect xdmhost2**
- Send a XDMCP query to the selected machine to display a login screen on the X server (if the selected machine is the machine running xdm, then it displays the login screen itself). The query that is sent is the same as sent by **X -query xdmhost2**.
- A login window is displayed by the display manager on the selected host.
- Start a session for the user by running the appropriate window manager.

31.2 Runlevels and display managers

Although **xdm** and other display managers are regular programs which happen to start a graphical login, they are treated specially in the runlevel configuration. For other applications it is sufficient to make sure there is an appropriate link in **/etc/init.d/rc*.d**. For the display manager, the convention is to change the default runlevel to one that includes the display manager.

The parameter in **/etc/inittab** that sets the default runlevel is "initdefault". The system here is running a display manager.

```
foo@bar:~> grep :initdefault /etc/inittab
id:5:initdefault:
```

Linux distributions use different standard values for the runlevels. Debian is the one distribution that does use special runlevels for graphical login – if you install the display manager it will be started during boot-up.

Distribution	*Console login runlevel*	*Display manager runlevel*
RedHat and Mandrake	3	5
Caldera	3	5
Slackware	3	4
SuSE Linux version 7.2 and above	3	5
SuSE Linux up to version 7.1	2	3
Debian	2	2

31.3 Configuring XDM

The configuration directory for **xdm** is **/etc/X11/xdm**, and the main configuration file is **/etc/X11/xdm/xdm-config**, which references the other configuration files.

Listening to remote requests

In order to serve remote XDMCP requests, the following line must be adjusted in **xdm-config**.

The default configuration is that **xdm** does not request a listening TCP port. **kdm** shares this configuration, and is adjusted in the same way.

```
! 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
```

xdm Xresources

When **xdm** starts up, it loads **/etc/X11/xdm/XResources** as a X resources database (using **xrdb**). This defines the greeting that the system displays, and other interface settings which can be modified such as the border width.

```
xlogin*greeting: Welcome to CLIENTHOST
xlogin*namePrompt: \040\040\040\040\040\040\040Login:
xlogin*greetFont: -adobe-helvetica-bold-o-normal--24-240-75-75-p-
      138-*
xlogin*font: -adobe-helvetica-medium-r-normal--18-180-75-75-p-103-*
xlogin*promptFont: -adobe-helvetica-bold-r-normal--18-180-75-75-p-
      103-*
xlogin*failFont: -adobe-helvetica-bold-r-normal--18-180-75-75-p-103-
      *
xlogin*borderWidth: 1
xlogin*frameWidth: 5
xlogin*innerFramesWidth: 2
xlogin*shdColor: grey30
xlogin*hiColor: grey90
xlogin*background: grey
xlogin*greetColor: Blue3
xlogin*logoFileName: /etc/X11/xdm/pixmaps/XFree86.xpm
xlogin*logoPadding: 10
```

One non-obvious thing in the above is that the “logoFileName” parameter must refer to a XPM file (X pixmap).

XSetup

In **/etc/X11/xdm/xdm-config**, a line will appear specifying the script to run to the display for the login window to appear on.

```
DisplayManager.*.setup:          /etc/X11/xdm/Xsetup
```

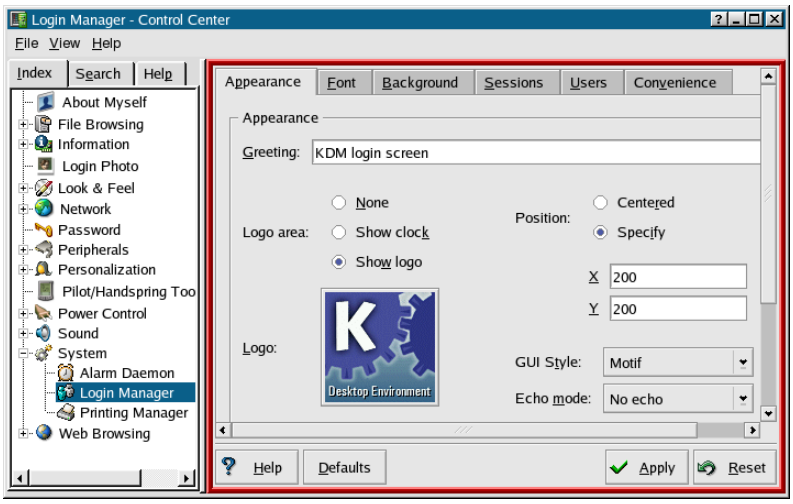
The script **/etc/X11/xdm/Xsetup** script will generally set the image on the root window using **xsetroot** or **xsri** and open up **xconsole** to show console messages. It could be abbreviated something like this.

```
#! /bin/bash
/usr/X11R6/bin/xpmroot /etc/X11/xdm/BackGround.xpm
setsid /usr/X11R6/bin/xconsole -notify -nostdin -verbose -exitOnFail
      &
```

31.4 Configuring KDM

Configuring **kdm** is a great deal easier than configuring **xdm** – primarily because the KDE project has written a graphical client to modify the configuration file **/etc/X11/xdm/kdmrc**.

Of course, your distribution may hide **kdmrc** somewhere else, such as **/etc/kde/kdm** or **/opt/kde3/share/config/kdm**.
Because **kdm** shares its code base with **xdm**, you will find the same files as used by **xdm** along with the **kdm** configuration file – Xresources, Xsession and friends.



31.5 Configuring GDM

The Gnome display manager's configuration is stored in **/etc/X11/gdm/gdm.conf**. **gdm** is derived from **xdm**, but many of the functions have been rewritten. The program **gdmconfig** edits this file, so generally you don't have to do it directly.

31.6 Connecting to a remote display manager

In order for an X terminal to initiate a login session from a remote display manager, the display manager must support the XDMCP protocol – usually via a TCP connection.

To connect to a X display manager via XDMCP, there are three different queries that the X server can choose to send:

- **X -query somehost** – request a login session on a particular machine "somehost".
- **X -indirect somehost** – request a chooser screen from a machine named "somehost", which will in turn send the request for the login window to the host you choose.
- **X -broadcast** – send a broadcast request for a chooser screen. (You can get the same behaviour using **X -indirect 255.255.255.255**). A host that feels capable of popping up a chooser window on the terminal will do so.

Xaccess

xdm and **kdm** use the **Xaccess** file to determine whether a XDMCP request is accepted or refused. For **gdm** there is a XDMCP configuration tab in the configuration tool. **Xaccess** also determines the fate of indirect requests, which may be diverted to other hosts.

The **Xaccess** file assumes that you are running a number of application servers, and a number of X terminals. The key elements in the file are:

- hosts – machines which run a display manager
- terminals – machines which run X servers (which want to connect to display managers)

There are two types of request which may be received from a terminal. These are handled differently in the **Xaccess** file.

- A direct query (**-query**). For this, the hosts's name must appear on a line in **Xaccess**, or the line * must appear. In the example the terminals **terry**, **fred** and **joe** can access the display

manager. Usually, a * appears to indicate that any terminal may open up a login window.

```
terry
fred
joe
```

- An indirect query (**-indirect** or **-broadcast**). In the following example, **terry**, **fred**, **pam** or **joe**, on connecting to the display manager, will be offered connections to the three application servers **fasthost**, **slowhost** and **officehost**.

```
terry      CHOOSER fasthost slowhost officehost
fred       CHOOSER fasthost slowhost officehost
pam        CHOOSER fasthost slowhost officehost
%hostlist  fasthost slowhost officehost
joe        CHOOSER %hostlist
```

The most common patterns in the **Xaccess** files allow access for any host to open a login session, or get a window. The special name **BROADCAST** below means that the list of hosts is obtained by broadcasting on the network for other display managers.

```
*                                # any host can get a login window
*          CHOOSER BROADCAST     # any indirect host can get a
      chooser
```

As the comments in the file suggest, **Xaccess** can allow access for only specific hosts. The file is (at its simplest) a list of machines that can connect to the display manager. Additional lines can be given to allow access to the host chooser.

```
host-a                             # host-a is allowed
host-b                             # host-b is allowed
host-c                             # host-c is allowed
```

Here we allow access to the chooser.

```
%hostlist      host-a host-b
*              CHOOSER %hostlist       #
```

This particular configuration in the above will allow only the machines **host-a** and **host-b** to connect to the chooser.

31.7 Review

Quiz questions

1. Why do remote terminals use a display manager, and not just run applications with **export DISPLAY=myname:0**?
2. What are the configuration files for the display managers discussed?
3. How does **/etc/inittab** get used for starting a display manager?
4. How do you change the display manager greeting for **xdm**?
5. What are bitplanes, and how do they relate to the display manager?

Assignment

1. Change your display manager to **xdm**, and set up a custom greeting and a custom background colour.

2. Set up **xdm** to serve direct and indirect remote sessions. Test whether this works. Once you have done this, set up the display manager to serve direct and indirect remote sessions for only one host. Test whether it works, and test that it only works for the one host.
3. Repeat the previous assignment using either **kdm** or **gdm**.
4. If you have a second machine available, set it up as a display manager and get the chooser on both display managers to show the other. Log on to one of the display managers, and then connect to the other for a login session.

Answers to quiz questions

1. Changing the value of the **DISPLAY** environment variable assumes that you can log in to the server that will be running the application. If you can't then you need the display manager to send the application to you.
2. **/etc/X11/xdm/***, **/opt/kde3/share/config/kdm** (or similar) and **/etc/X11/gdm/gdm.conf**
3. In some distributions, the runlevels either include or do not include **xdm**. In some distributions xdm is started from **/etc/inittab** for a particular runlevel.
4. Modify the resources **/etc/X11/xdm/Xresources**
5. Bitplanes are the number of bits per pixel, and you will want to have configured your X server with the correct values for your display card by the time you run **xdm**.

32 GUI environment

"Hallo. I em ze Viper. I huf kome to vipe your vindows."

"The Viper is Coming!" (Unpublished work)

LPIC topic 1.110.4 — Install & Customize a Window Manager Environment [5]

Weight: 5

Objective

Candidate should be able to customize a system-wide desktop environment and/or window manager, to demonstrate an understanding of customization procedures for window manager menus and/or desktop panel menus. This objective includes selecting and configuring the desired x-terminal (xterm, rxvt, aterm etc.), verifying and resolving library dependency issues for X applications, and exporting X-display to a client workstation.

Key files, terms, and utilities include

.xinitrc	User's configuration file for **startx**
.Xdefaults	X resources, for configuring X applications like **xterm**
xhost	Host based access control for the X server
DISPLAY environment variable	Determines which X server receives output

32.1 Window managers

An X window manager is responsible for the following tasks:

- Title bars and borders around windows, and the functions available on the title bar and border (move, resize, minimise, close)
- Controlling the focus of keyboard and mouse input
- Icon management (associating icons with windows)
- Determining the initial position and size of windows

Many window managers implement additional functions:

- Desktop menus which can be activated by mouse clicks
- Global keyboard and mouse shortcuts
- Virtual desktops
- Desktop decoration
- Desktop configuration programs
- Sound support (pops and whistles for desktop events)

Here are a few window managers which you may encounter under Linux (and there are more).

- twm – the simple window manager that shipped with the X project
- kwm – KDE's window manager
- enlightenment – Gnome's default display manager
- metacity – A window manager with good Gnome integration.
- sawfish – An extensible window manager that uses a Lisp-based scripting language. Sawfish is configured by Lisp code in a personal .sawfishrc file, or using a GTK+ interface.

- WindowMaker – A window manager which emulates the look and feel of the NeXTSTEP (TM) graphical user interface.
- blackbox – A very small and fast window manager.
- hackedbox – The bastard son of blackbox, just in case blackbox wasn't small enough for you.

There is no universal configuration interface for configuring all Linux window managers, but most window managers have their own dot configuration file or directory in the user's home directory.

32.2 .xinitrc and the system-wide window manager

When the X server is started with **startx**, the following files determine which window manager to load. (Actually, other applications can be loaded, in addition to the window manager.)

1. **~/.xinitrc** – if this file exists, it is run after the X server starts up. When it exits, the session is over.
2. **/etc/X11/xinit/xinitrc** – if no **~/.xinitrc** exists, this file is run. It determines which window manager will run.

The usual change which is made to these files is to change the window manager which is run when **startx** is selected. Many Linux distributions use a display manger such as **xdm** to control login. In this case, the contents of **.xinitrc** are largely irrelevant. Changing **/etc/X11/xinit/xinitrc** will change the default behaviour for users that have no **~/.xinitrc**. Creating **/etc/skel/.xinitrc** will change the behaviour for new users.

32.3 X applications

In the original design of the X window System, it was intended that X applications should be customised by editing X resources[36]. X resources are simply configuration settings that are loaded into the X server, usually at startup (using **xrdb**, but you generally don't worry about that).

In ancient times[37], when **twm**, **mwm** and **olwm** were the window managers of choice, window managers were customised by editing the X resource files. In modern times, editing X resource files is still the way to customise a number of useful applications using X, including **xterm**. X resources make it possible to change these values on the fly as well, without editing the files.

When X starts, the X resources are loaded from the following files into the X server by **xinitrc**:

1. **/usr/X11R6/lib/X11/Xresources** – system-wide X resources
2. **~/.Xdefaults** – default resource settings users don't usually edit (only in some distributions)
3. **~/.Xresources** – resource settings that users do edit

36 The **xrdb** man page says "Most X clients use the RESOURCE_MANAGER and SCREEN_RESOURCES properties to get user preferences about color, fonts, and so on for applications."

37 Ancient times – i.e. before last year

The format of X resource files is as follows:

- The name of the resource, followed by a colon, followed by the value.
- The name of the resource can be abbreviated with a *, in which case any number of characters can match (actually, only a complete section name can match).
- The name of each resource is in the form application.item, e.g. xterm*font.

You should have different values to these if you run the commands on your server.

```
foo:~ $ xrdb
... lotsa output ...
foo:~ $ xrdb -query | grep -i xterm
XTerm*Scrollbar*borderWidth:    3
XTerm*Scrollbar*height: 16
XTerm*Scrollbar*shadowWidth:    2
XTerm*Scrollbar*width:  16
xterm*SaveLines:        2000
xterm*ScrollBar:        off
xterm*background:       White
xterm*font:     fixed
xterm*jumpScroll:       on
xterm*multiScroll:      on
xterm.eightBitInput:    true
xterm.eightBitOutput:   true
foo:~ $ xrdb -query | grep '^\*' | head
*Command*highlightThickness:    2
*Command.background:    grey77
*Form.background:       grey67
*Label*borderWidth:     2
*Label*shadowWidth:     2
*MenuButton*highlightThickness: 2
*MenuButton.background: grey77
*MenuButton.foreground: black
*Panner*shadowThickness:        2
*PushThumb:     False
```

32.4 X terminal emulators

A terminal emulator lets you run console based programs on a virtual console.

- You can run a couple of terminal emulators at the same time.
- You can set the font and the console size.
- Some terminal emulators let you run multiple sessions in a single window.
- An X terminal emulator usually has a large scroll-back buffer – more than the Linux console.

Here are some of the terminal emulators available for X on Linux.

- **xterm** – the original X terminal. Other terminal emulators are based on this one. Some copy its code.
- **rxvt** – a lightweight clone of xterm (based on **xvt**)
- **aterm** – aterm is a pseudo transparency terminal. It is supposed to be resource friendly and has many features like fading (darkening/lightening of colors when **aterm** is losing focus) and tint color (pseudo transparency background color).

- **konsole** – KDE's terminal
- **gterm** – Gnome's terminal

Each session gets a different virtual terminal. As a curiosity, the **tty** command can be used to establish which particular console is active.

```
foo:~ $ tty
/dev/pts/2
```

If you run that same command after pressing **Ctrl+Alt+F1**, you will be on a different (pseudo)terminal.

```
foo:~ $ tty
/dev/tty2
```

32.5 X application library dependencies

Every X application communicates with an X server using the API provided by the **X11** library. This library must be present for the application to work.

```
foo:~ $ ldd /usr/X11R6/bin/xterm
        libXft.so.1 => /usr/X11R6/lib/libXft.so.1 (0x40024000)
        libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x40032000)
        libXrender.so.1 => /usr/X11R6/lib/libXrender.so.1
      (0x4007b000)
        libXaw.so.7 => /usr/X11R6/lib/libXaw.so.7 (0x40080000)
        libXmu.so.6 => /usr/X11R6/lib/libXmu.so.6 (0x400da000)
        libXt.so.6 => /usr/X11R6/lib/libXt.so.6 (0x400f0000)
        libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0x40143000)
        libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0x4014c000)
        libXpm.so.4 => /usr/X11R6/lib/libXpm.so.4 (0x40163000)
        libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x40172000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x40180000)
        libncurses.so.5 => /usr/lib/libncurses.so.5 (0x4025e000)
        libutempter.so.0 => /usr/lib/libutempter.so.0 (0x4029e000)
        libc.so.6 => /lib/libc.so.6 (0x402a0000)
        libfontconfig.so.1 => /usr/lib/libfontconfig.so.1
      (0x403be000)
        libdl.so.2 => /lib/libdl.so.2 (0x403e3000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
        libexpat.so.0 => /usr/lib/libexpat.so.0 (0x403e6000)
```

It is quite common for X applications to interface to the X11 system via a number of additional libraries. So, for example, KDE applications require the **qt** libraries, which insulate the application from the intricacies of talking to X directly.

The following libraries are quite popular for X applications:

- **qt** – this is the underlying display library used by KDE applications. Some applications are built directly on **qt**.
- **gtk** – the **gimp** toolkit. This is the library originally written for the graphics program **gimp**, but now used by a number of other applications.
- **gnome** – the gnome user interface has its own set of libraries.
- **kde** – the K Desktop Environment also comes with a set of libraries supplying functions to K applications.

Here are more than just a few examples. You will notice that KDE and gnome are both heavy interfaces consisting of many interrelated components, while GTK is a light interface.

```
foo:~ $ ldd /usr/bin/gimp
        libgtk-1.2.so.0 => /usr/lib/libgtk-1.2.so.0 (0x40024000)
        libgdk-1.2.so.0 => /usr/lib/libgdk-1.2.so.0 (0x4017e000)
        libgmodule-1.2.so.0 => /usr/lib/libgmodule-1.2.so.0
      (0x401b7000)
        libglib-1.2.so.0 => /usr/lib/libglib-1.2.so.0 (0x401ba000)
        libdl.so.2 => /lib/libdl.so.2 (0x401e0000)
        libXi.so.6 => /usr/X11R6/lib/libXi.so.6 (0x401e3000)
        libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x401ec000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x401fa000)
        libm.so.6 => /lib/libm.so.6 (0x402d8000)
        libc.so.6 => /lib/libc.so.6 (0x402f9000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
foo:~ $ ldd /usr/bin/kalarm | head
        libkcal.so.2 => /usr/lib/libkcal.so.2 (0x40013000)
        libkio.so.4 => /usr/lib/libkio.so.4 (0x4009e000)
        libkdeui.so.4 => /usr/lib/libkdeui.so.4 (0x4030b000)
        libkdefx.so.4 => /usr/lib/libkdefx.so.4 (0x40523000)
        libXrender.so.1 => /usr/X11R6/lib/libXrender.so.1
      (0x4054a000)
        libkdesu.so.4 => /usr/lib/libkdesu.so.4 (0x4054f000)
        libutil.so.1 => /lib/libutil.so.1 (0x4057b000)
        libkalarmd.so.0 => /usr/lib/libkalarmd.so.0 (0x4057f000)
        libkdecore.so.4 => /usr/lib/libkdecore.so.4 (0x40585000)
        libDCOP.so.4 => /usr/lib/libDCOP.so.4 (0x406d9000)
foo:~ $ ldd /usr/bin/gnome-text-editor  | head
        libgnomeprintui-2.so.0 => /usr/lib/libgnomeprintui-2.so.0
      (0x40024000)
        libgnomeprint-2.so.0 => /usr/lib/libgnomeprint-2.so.0
      (0x40047000)
        libeel-2.so.2 => /usr/lib/libeel-2.so.2 (0x40494000)
        libgnomeui-2.so.0 => /usr/lib/libgnomeui-2.so.0 (0x40529000)
        libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0x405ae000)
        libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0x405b7000)
        libgailutil.so.17 => /usr/lib/libgailutil.so.17 (0x405cf000)
        libglade-2.0.so.0 => /usr/lib/libglade-2.0.so.0 (0x405d6000)
        libbonoboui-2.so.0 => /usr/lib/libbonoboui-2.so.0
      (0x405eb000)
        libgnome-2.so.0 => /usr/lib/libgnome-2.so.0 (0x4064d000)
```

32.6 Remote applications

When an X application start up, the X11 library reads the command line and examines environment variables to decide what should be done with the output of the application. The X11 library does not particularly care whether the display which is used is on the computer running the application or not. All it needs to know is where that display is.

From the server side, the X server needs to trust the client application to connect[38]. There are

38 A security note – if a malicious application connects to your X display, it can do strange things like monitor all keystrokes, and take snapshots of the screen. Apart from this, one X application generally cannot subvert another.

two methods of access control that are commonly used:

- Host based access control, controlled by the **xhost** application.
- MIT cookie access control, controlled by the **xauth** application.

32.6.1 DISPLAY environment variable

The **DISPLAY** environment variable sets the display which is to be used for X11 applications. The format of this variable is

```
DISPLAY=host:display.screen
```

If the host is **localhost**, it can be omitted. The screen part is used for X servers which have multiple monitors attached. If the screen is "**.0**", that part can be omitted as well. The most common value for the **DISPLAY** environment variable is to point to the first local display:

```
foo:~ $ echo $DISPLAY
:0
foo:~ $ export DISPLAY=bar:0.0 ; xterm
foo:~ $ export DISPLAY=bar:0 ; xterm
foo:~ $ DISPLAY=bar:0 xterm
```

Instead of setting the DISPLAY environment variable, you can use the **-display** switch on the command line.

```
foo:~ $ xterm -display bar:0
```

32.6.2 Host based access control

Host based access control is the simplest form of access control available for X. The X server maintains a list of hosts which may connect to it. This list is manipulated with **xhost**.

```
% xhost +hostname          # Add the host to the list
% xhost -hostname          # Add the host to the list
% xhost +localhost         # All local users can use the server
% xhost -                  # Allow only hosts on the list
% xhost +                  # Any host can connect to the server
      (eek!)
```

Using host based access control is recommended in the following circumstances:

- You trust all users on the machine you add to the list (or there is only one user).
- You need to get something done, which requires X.
- You computer is not connected to a public network, and trust all computers on the local network (**xhost +**).

Only the controlling machine can use **xhost**. The controlling machine is usually the machine running the X server, or in the case of an X terminal, the login host that supplied the login window.

32.6.3 MIT cookie access control

The file **~/.Xauthority** contains a list of authentication secrets which are used for various displays. The **xauth** application manipulates this database, and allows you to do the following things:

- Generate an X authentication secret (a cookie).
- List X authentication secrets (a jar of cookies)
- Export the X authentication secret (a cookie)
- Import X authentication secrets

Taken together, these functions allow you to provide authentication credentials to an application on a remote host.

```
foo:~ $ xauth list
foo:0  MIT-MAGIC-COOKIE-1  1a5b56785802584e6740787c066c7549
foo.acme.com/unix:0  MIT-MAGIC-COOKIE-1
      1a5b56785802584e6740787c066c7549
```

One can log into another server, and then add these credentials there.

```
foo:~ $ xauth extract - $DISPLAY | ssh otherhost xauth merge -
foo:~ $ ssh otherhost
Last login: Mon May 26 13:50:26 2002 from foo
otherhost:~ $ xterm -display foo:0             # pop up on other
      display
```

Similarly, as root, you can obtain access to another user's display (this is the default behaviour on some distributions).

```
foo:~ $ export DISPLAY=:0
foo:~ $ su username -c "xauth extract - $DISPLAY" | xauth merge -
foo:~ $ xterm
```

X terminals automatically set up MIT-MAGIC-COOKIE authentication.

32.7 Review

Quiz questions

1. When is **~/.xinitrc** used?
2. How do you change the window manager for users that log on using **xdm**?
3. Which library do all X applications use?
4. What program is used to set up MIT cookie based access control?
5. What must one do to change the display used by an X application.
6. How does one modify X resources?
7. How does one allow a specific host permission to open windows on an X display?
8. How does one allow two users to use the same X display?

Assignment

1. Change the system-wide window manager from KDE to Gnome or from Gnome to KDE.
2. Install icewm, FVWM2 or BlackBox and make this the system-wide window manager.
3. Log into a remote machine using **telnet** or **ssh** (without the **-X** option) and make an X application such as **xterm** run on your local display. Do this using both the **-display** option and the **DISPLAY** environment variable. You can use host based authentication for this exercise.

4. Set up MIT magic cookie access control and test it by repeating the previous assignment.
5. Change the foreground and background colours for **xterm** by editing your X resources file. You can import changes to your X resources file with **xrdb** and then test whether they were successful.
6. Modify your **.xinitrc** to run **xlock** before starting the window manager. (This can be quite useful for users which are automatically logged in by **kdm** or **gdm**).

Answers to quiz questions

1. When X is started with **startx**.
2. create **~/.xsession** for the user to start the window manager.
3. the X11 client library
4. **xauth**
5. modify the **DISPLAY** environment variable, or pass -display on the command line.
6. edit the resource file, and then load into the X server with **xrdb**
7. **xhost +hostname** (or use MIT cookies)
8. provide both users with MIT cookies

33 Glossary

0x00c0ffee Hexadecimal numbers are frequently written with the prefix "0x", to indicate the base of 16. This is the notation used by the C programming language. The digits 0 to 9 are used, and "a" to "f" represent digits for 10 to 15. Hexadecimal notation is convenient for computer applications, because a hexadecimal digit corresponds neatly to 4 bits.

1024 1024, or 2^{10} is the number of numbers you can represent with 10 binary digits, and the number of bytes in a kilobyte (kb). A Megabyte (Mb) is similarly 1024 kbytes, or 1048576 bytes. Hard disk sizes are sometimes quoted in millions of bytes, rather than in true megabytes to achieve the marketing edge that 4.9% can give.

about The first word in many glossaries.

ASCII The American standard code for information interchange, a standardised set of meanings for the bytes 32 to 127. Included in this are the letters A to Z in upper and lower case, the digits, and most common punctuation symbols. Text files in Linux are usually encoded with ASCII.

argument Arguments appear after a command on the command line, and are sometimes called "parameters". If the command is "**ls -l -a /etc** then the command is **ls** and the command line arguments are "**-l**", "**-a**" and "**/etc/**".

bash An improved version of the Bourne Shell (sh), known as the Bourne Again Shell.

BIOS Basic Input Output System. This was originally intended to be some kind of hardware independent layer providing kernel-like facilities. For most systems the BIOS is usually used for loading the operating system which provides these facilities.

bleeding edge If the leading edge software you choose doesn't work quite right, it might still need a bit of debugging.

boot When you turn on your computer, it "boots up". The origin of this is that the computer pulls itself up by its own bootstraps and gets the operating system running.

buffer A place for storing data in transit. This is a technical term used by program which occasionally escapes into regular usage. Linux write filesystem changes directly to the disk, but writes them to buffers which are later written to the disk.

bus A set of parallel electrical conductors used in a PC for communication between devices, such as between the CPU and memory and peripherals.

cat A Linux command that shows the contents of a file. This is the only Linux command in the glossary, and it's only here because I like cats. All the other commands are listed in the index.

character A letter in the alphabet, a digit, a punctuation symbol – anything that can be represented by one or two bytes (in this sense, characters are rather one-dimensional). The most commonly used representation of characters is ASCII.

UTF8 uses more than one byte to represent a single character.

client In networking, the side that initiates the connection is the client. The client talks to a server, which (hopefully) serves it.

CLUE A command line user environment, as opposed to a GUI, a graphical user interface.

CMOS Complementary Metal-Oxide Semiconductor – usually refers to something managed by the BIOS. (CMOS is not that stuff you get at the bottom of the ocean).

comment Comments are written by people in configuration files to explain what the meaning of the rest of the file is. Configuration file comments usually start with a hash character (#).

```
# /etc/hosts.deny
# See `man tcpd´ and `man 5 hosts_access´ as well as /etc/hosts.allow
# for a detailed description.
sshd: ALL
```

Reading configuration file comments is often more instructive than reading other documentation such as man pages.

console A console is a terminal which is directly connected to a server.

CPU Central Processing Unit. Linux originally ran on the Intel 80386 CPU.

database A database is an ordered arrangement of computer data. The password file **/etc/passwd** is correctly called a database (not all databases are relational databases using SQL for queries.)

default What it is if you don't change it. The default behaviour of **cp** is not to be verbose, but telling **cp -v** overrides the default and makes **cp** print information about the files it copies.

display A display is strictly just a monitor for viewing computer output. In X windows a display includes the facilities provided by an X windows server – monitor, keyboard and mouse. X Windows applications allow you to specify a display on the command line or in the environment.

```
export DISPLAY=remoteserver:0.0 ; xterm
xterm –display remoteserver:0.0
```

distribution Because the vast majority of software for Linux is free, it is possible for anyone with the technical capability to distribute it. A number of distributions exist as a result, varying primarily in the method of distribution (e.g. CDROM, download), the method of software packaging (e.g. RPM, deb, tar.gz, cvs, other). Some distributions are commercial (e.g. RedHat, Mandrake, SuSE), and others are deliberately non-commercial (e.g. Debian).

DMA Direct Memory Access – where your peripherals talk directly to the memory, rather than via the CPU.

DNS Names on the Internet such as www.disney.com are turned into network addresses using the DNS system. From the client point of view, you simply need to know what DNS server to ask for questions of DNS names. On Linux, the DNS configuration is mostly in **/etc/resolv.conf**.

DOS Microsoft MS-DOS is an cut-down operating system for the Intel 8088 family of processors. Microsoft Windows 3.1, 95 and 98 contain designs influenced

by MS-DOS. DOS is also an acronym for Denial of Service (deliberately making a service fail).

environment The environment a program is provided with is a list of variables in the form **NAME=VALUE**. When the shell starts a program it passes all the "exported" environment variables in the environment. The program may choose to react to the environment variables it receives.

execute To start or run a program. Programs start by being executed, and they end when they are killed.

FAQ FAQs or Frequently Asked Questions are lists of questions that are frequently asked, and sometimes answered. If you have a question about a particular product, and it has an associated FAQ, you may find an answer in the FAQ.

field A field is an area of space, or in a database, a record in which the same type of information is regularly recorded. In text files, fields are separated by delimiters such as colons, tabs or spaces. In the **/etc/passwd** file the fields are separated by colons (the fields are user, password, user id, group id, name, home directory and shell).

```
root:x:0:0:root:/root:/bin/bash
```

filesystem A system for arranging files, either in a fixed space, like a partition or a CD ROM, or referring to the entire Linux filesystem, consisting of a number of mounted filesystems.

font A font specifies the shape of each symbol in a script. Vector fonts describe symbols in terms of the outline. Bitmap fonts specify the exact map of bits for each letter. Bitmap fonts look rather blocky if they are enlarged.

foobar A foobar is a widget that does something. Sometimes written as just foo or just bar. When you can't think of a name for something, foobar is just fine.

fortune A Chinese fortune cookie is a biscuit containing a text, presumably related to your future. The **fortune** program prints out a message which is almost as random as what you may find in a cookie. Some administrators configure their system to print a fortune cookie when you log in.

geometry For computer disks, geometry refers to the coordinate system of finding your data, the coordinates being the cylinder, sector and head to use. The geometry of a disk is the number of cylinders, sectors and heads (disk read and write heads). LBA addressing is more common. This geometry is more like the Cartesian plane than Euclidean geometry.

GPL The GNU public licence – a licence document which allows you to redistribute computer software that is copyrighted – with the proviso that you must distribute the source code as well.

group A group is defined by the contents of **/etc/group**. Every file on the filesystem and every process has a group assigned. A user may belong to a number of groups (the primary group, and a number of secondary groups).

home directory Each user has a directory which is allocated to them for their personal files. For a user named **joe**, this will usually be **/home/joe**, but it is possible to change this. The user's home directory is specified in **/etc/passwd**. In the shell, the abbreviation for your own home directory is "**~**".

host A machine connected to a network. Similarly, the term hostname refers to the name of a host (of a machine).

HOWTO A HOWTO is a long-ish document explaining how to do something. This something is often a relatively complex task, such as setting up a firewall, or configuring modem communications. Mini-HOWTO's are usually only a few pages long, and not as extensive as a full HOWTO.

I/O Input and output – what the CPU does when it talks to devices. Devices are generally identified by their I/O address, a number to which only the device responds.

IDE Integrated Drive Electronics – IDE usually means a hard disk or a CD ROM drive, which unlike archaic disks, has the electronics integrated into the drive unit.

interrupt When a device wants to communicate some data to the CPU, it triggers an interrupt. The CPU interrupts what it is doing, and makes sure that it gets the data from the device.

journal For regular filesystems, changes are written immediately to the disk. In a journalling filesystem, some or all changes are written to an intermediate buffer called the journal, and then written to the disk. Using a journal makes it possible to recover from errors if the process is interrupted, since the journal contains information about what was happening.

Kernel The lowest level of the operating system that does all the hard work.

LBA Logical block addressing – using a single number to represent a location on a disk, rather than physical parameters.

library A library is a repository for computer program code, particular code which is used by a number of applications. Libraries are used for inclusion into code when it is compiled. Linux supports dynamic loading of shared libraries, such as the GNU C library /lib/libc.so.6. This library is loaded into memory once, and shared between all programs that use it.

Linux The kernel of the Linux operating system, by Linus Torvalds and others.

LPI Linux Professionals Institute, see www.lpi.org

LPIC LPI Certification, awarded by the LPI on passing the certification examinations. Various levels of certification exist.

LUN Logical Unit Number – a SCSI host may provide a number of logical units, such as a number of hard disks.

make **make** is a program which automatically determines which part of a large program need to be recompiled. It is configured by a **Makefile**.

man pages Manual pages, stored on-line and accessed with the **man** command. To access the **crontab** man page in section 5 of the manual, the command is **man 5 crontab**. The commonly used abbreviation for this is **crontab(5)**.

media Something physical on which data is stored, e.g. magnetic media

metacharacter A character which takes on a special purpose. Regular expressions are composed of regular characters and metacharacters such as “*.[]^$()\”.

modem A device that converts digital signals to analogue signals, usually for transmission over a telephone link.

mount To permanently attach, for example to attach a filesystem on a device to the Linux filesystem. You can also mount a shelf on a wall, a camera on a car.

network address Each machine on a network is uniquely identified by a network address. Since most networks run IP (Internet protocol), this is usually an IP address. Data directed to this network address arrive at the machine using it.

NIC Network interface card (e.g. an Ethernet card such as a 3COM 3c509 adapter)

ownership Every file and every process in a Linux system is owned by a user. For files and directories, the user can set the group of the files, and configure the permissions that are granted.

package All the components of a program in an uninstalled form. Often **.rpm**, **.deb** or **tar.gz**.

parallel Parallel communication involves sending a number of bits simultaneously, i.e. in parallel. This is the method used by the parallel port for communication with printers.

parameter A parameter is anything that can be changed or chosen by a user. Examples include command line parameters, file system format parameters, kernel tuning parameters and configuration parameters. Often parameters have default values, and these values sometimes work.

PCI Peripheral Component interconnect – a BUS architecture that is a little bit newer than ISA.

peripheral A device on the periphery of a computer, such as a keyboard, printer, mouse or display.

permission In general, the kernel does not allow users are to interact with processes and files that they do not own. File permissions can be set for the owner, a group and for everyone else. Permission to read a file means that the contents can be viewed. Permission to write a file means that is contents can be altered. If execute permissions are set, the kernel will load the file into memory and use it as a program.

pipe Programs can pass information using pipes. One side writes to the one side of the pipe, while the other reads from the other. For the program, this is exactly the same as using a file.

PnP Plug and play – the notion that no configuration is required to use a device. This has been implemented with various degrees of success for the ISA bus, the PCI bus and the USB bus.

ppp Point to point protocol – this is what is used when you dial from one modem to another, such as connecting to an ISP.

proc The **proc** filesystem is mounted at **/proc**, and provides information generated by the kernel about processes, and various other interesting things.

process A program running on your computer. Linux, as a multi-tasking operating system, runs many processes simultaneously.

protocol An agreed method of communication between two parties. A computer protocol may describe the method that one system sends mail to another, or the method that a device notifies another device about an event.

rc A run control file (“rc file”) configures a program. Configuration files in **/etc**

are sometimes called rc files. The directory **rc2.d** contains **rc scripts** used to configure the initialisation sequence of the system (**rc2.d** specifically controls runlevel 2). The **.d** extension indicates a directory.

recursive In Linux, recursive refers to acting on a directory and the files and directories it contains. To understand recursion, you must first understand recursion.

resources Anything that is limited, such as memory, I/O addresses, DMA channels, file descriptors, locked file space.

root The administrator on a Linux system is named "root". The base directory on the file system is also called the root directory. The user "root" has a home directory off the filesystem root named "/root".

rwx Read, write and execute permissions. This is often seen in the output of **ls**.

script A script is a list of commands that the computer should perform. A shell script is run by the shell interpreter (/bin/sh)

SCSI Small Computer System Interface

serial In serial communication, data is passed one piece at a time, and in order, i.e. serially. PC's have a serial communications port which can send and receive data in a serial fashion, using the RS232 protocol. Typically a RS232 serial port is used for communication to a mouse or modem. USB, the Universal Serial Bus, uses serial communication too.

server In networking, the side that accepts an incoming the connection is the server. The client talks to a server, which (hopefully) serves it.

setuid A program which has permissions to change its user ID. This is indicated by a "s" in the place of the "x" permission.

shell A shell is a program which allows you to run commands by typing in the command's name. The Bourne shell will allow you to configure the input and output for commands you enter, and run string multiple processes together by joining their input and output.

source code Source code is human readable instructions for what your computer should do with its time. This is compiled into useful executable form by a compiler. The term "open source" refers to the freedom of this source code.

ssh Secure shell – this is a network protocol which encrypts what you type in a shell and the responses. There are two parts – ssh, the client and sshd the server.

standard input, outut and error By default, a program's standard input and output is connected to the terminal. When the program chooses to write or read something, it will go through the terminal function calls. The shell can redirect standard input and output to files and processes, so that commands operate on file contents rather than typed contents. Programs that want to print error messages print them to standard error which is usually the terminal. The shell allows you to redirect the errors as well.

swapfile Disk space which is used as memory space. As more memory is required for applications, idle memory may be swapped out (exchange with disk space). When the memory on the disk is required again, it is swapped in. Swap is American for swop.

switch A command line option of a single character. In the command **ls -la** the **-l** switch and the **-a** switch are selected.

syntax Syntax is the way in which language is structured. Computer languages are rather pedantic about their grammar. You need to use the expression in pretty much the way that it is expected by the computer. When syntax is explained in written text, optional items are written in [square brackets], e.g. ls [options], meaning that the options are optional; and "..." indicates that you can add more items, e.g. ls [file] ... means list the file, and you can specify more files.

tab space A special character (ASCII code 09) which is meant to work something like the tab key on a typewriter – to fill up with space up to the next tab stop. Tabs are used in some files to separated between fields.

terminal A terminal consists of a keyboard and a screen. In the original days of Unix, many consoles were connected to a single Unix server via serial cables. Linux provides a number of terminals that can be used simultaneously, accessible by pressing Alt+F1 up to Alt+F7 (usually). Since these terminals only exist in the mind of the user and the operating system they are called virtual terminals.

text Data that a human has a reasonable chance of being able to read, as opposed to something exclusively for machine consumption.

timeout A timeout occurs when a program stops waiting for something to happen, simply because it has waited too long.

tree The wiser computer science people seem to think that a tree, like the type that grows in soil, is a useful analogy for explaining the hierarchical organisation of filesystems (and DNS, and some database structures). Like a filesystem a tree has a root, and where trees have branches with leaves and more branches, filesystems have directories with files and more directories. Unfortunately the analogy breaks down too easily, for example if a bird flies in through your serial port and perches on your filesystem tree.

variable Something that varies. In programming, variables have names, and the value of the variable is substituted where the name is used. In the shell, environment variables are set using VARIABLE="something". After this statement, the shell substitutes "something" in the place of $VARIABLE.

verbose A program which is being verbose will print out extra information. For example, **cp file1 file2** will quietly copy the files, while **cp -v file1 file2** will print out the names of the files as they are copied.

wildcard A special character which matches any character. For file names, the wildcards "*" and "?" match anything or a single character. For regular expressions, the only wildcard is "." (matching any character), while ".*" matches any number of characters. In card games, the joker card is a wild card.

zzz After reading an entire glossary, you will notice a tendency towards sleepiness. The other option for this last entry was zebra, but that's not only an animal, but an implementation of BGP and RIP.

34 Index

An index is an alphabetical list of words and the places they occur in the text. This is what you can peruse in this section. The list at the front of the notes was a table of contents.

V

W

X

Z

`

/

$