Basic GNU/Linux

PID_00148393

Universitat Oberta de Catalunya
www.uoc.edu

Index

1. Introduction

1.1. What is GNU?

In order to understand the whole free software movement, we should go back to the late sixties, early seventies. At that time, large computer firms did not give software the same value it has today. They were mostly computer manufacturers whose main revenue was generated by selling their large machines, which came with some type of operating system and applications. Universities had permission to take and study the operating system's source code for academic purposes. Users themselves could request the source code from drivers and programs, for adaptation to their needs. Software was not considered to have an intrinsic value and simply came with the hardware that supported it. In this context, Bell laboratories (AT&T) designed an operating system called UNIX, characterised by its efficient management of the system's resources, its stability and compatibility with the hardware made by different manufacturers (to homogenise all their systems). This last factor was very important (before then, every manufacturer had their own operating systems that were incompatible with the rest), since it made UNIX extremely popular.

Stallman, without drivers

Stallman himself recalls how annoyed he was when he discovered that a company that had supplied a new printer for the laboratory where he was working did not want to give him the driver source code. All he wanted was to modify it so that the printer would automatically alert him when the paper was jammed. The company refused to provide him with the source code.

Gradually, the big firms started realising the value of software: the first to do so was IBM, which in 1965 stopped offering its operating system's source code, while at the end of the seventies Digital Research started selling its own. This made all companies become aware of the fact that software could be very profitable and provide enormous benefits. Then most companies started being reluctant to give out the source code for their programs and operating systems and started selling their programs as an added value to their hardware. In this increasingly closed environment, Richard Stallman (who was working at MIT, the Massachusetts Institute of Technology) was indignant to find that it was becoming increasingly harder to obtain the programs' source code, which he used to adapt the programs to his needs, as he had done up until then.

At that moment, Stallman decided to stand up for his ideals and to launch a grand project to try to reopen programs' source code. Aware of the fact that he would not convince companies to give in on this point, he proposed to create his own operating system and applications, initiating a project called GNU.

Stallman's first manifesto is particularly interesting for understanding the reasons why he created GNU; in this document he explains to the entire community what the new project consisted of, how it would be oriented, and why he had to do it. In the manifesto he started to describe the concept of free software and why he believed programmers and developers from all around the world should contribute with him. Although on many occasions the concept of free software is confused with the concept of free-of-charge or unpaid-for software (in English, *free* can mean both), in later documents it has been made very clear that there is no reason why free software should be free of charge. We need to understand free software to mean programs whose source code we can obtain to study, modify and redistribute, without being obliged to pay for it. What we do need to understand is that we can ask for the money we want for the programs and their source code, the support we may offer users, the books we sell or the material that we provide, just like the many companies that distribute GNU/Linux do. However, at no time, can we oblige the users not to distribute the software that we have sold them. This must be able to be distributed freely. It is a different way of understanding software to what we are used to. Many of the FSF (Free Software Foundation) texts deal more with ethics than with engineering. We need to understand the entire movement as a way of thinking or doing things rather than as just another software company.

The FSF's principles with regards to software are defined according to the following four freedoms:

- Freedom 0 refers to the freedom to be able to use the program for any purpose.

- Freedom 1 allows us to study how the program works and adapt it to our own needs. Access to the source code is a pre-requisite for guaranteeing this freedom.

- The second freedom allows us to freely distribute copies of the software, helping our neighbours.

- The final freedom allows us to improve the program and to make our own improvements public, for the benefit of the entire community. Access to the source code is also a pre-requisite for guaranteeing this freedom.

The GPL (General Public License) was drafted to guarantee these freedoms for the software developed by the project and for the end users of the software, and has since been used to protect all of this type of programs. This licence puts the above-mentioned ideas down on paper.

The project started to produce software in 1984, by developing all the tools required to implement a complete operating system. Although realising a project of these characteristics is a long and complex process, from the out-set many software programmers and developers were captivated by Stallman's idea and started collaborating with him for free. The community did not stop growing, and gradually the necessary tools appeared (editors, compilers etc.) to implement the operating system's kernel, for which the tools under development were required. From the start the idea was to build an operating system similar to UNIX following the POSIX norms (Portable Operating System Interface). Although UNIX also had its own problems and faults, it was, and continues to be, good enough to adapt to most requirements. The task of designing and writing the operating system's kernel was left until last. It is still pending definitive completion and the GNU kernel, called Hurd, remains in the development phase.

Kernel

As its name indicates, the operating system's kernel is the core that makes it function. The software kernel is what manages the computer's resources: it communicates with installed devices and applications, administrates the memory correctly, distributes processing time for all programs, communicates with storage devices for saving files etc.

Activity

1.1 Read the first message written by Stallman in 1983 announcing his project: http://www.fsf.org/gnu/initial-announcement.es.html.

Activity

1.2 Read The GNU manifestoStallman's original.

Activity

1.3 Read General Public License.

1.2. What is GNU/Linux?

Against this backdrop, while the FSF still did not have a stable kernel for its operating system, a professor of the University of Holland, **Andrew Tanenbaum**, decided to write an operating system so that his students could study it. Like Stallman, up until then he had been able to use the source code of AT&T's UNIX to help teach his pupils to design operating systems. His idea was to write an operating system that could be studied and modified by anyone who wished to do so. In 1987, he got down to work and called his project mini UNIX, resulting in **MINIX**. Because he did not use a single line of AT&T's Unix code, there is no restriction to take the code and to use it and modify it freely.

Micro-kernel technology

Micro-kernel technology is based on splitting the different functionalities of an operating system's kernel into totally separate inter-communicating programs. This makes it very modular, which makes testing, detecting and correcting bugs, maintenance etc., a lot easier. Currently, some operating systems such as Amoeba, Chorus, Mach or Windows-NTTM have incorporated this type of technology.

Tanenbaum wanted to create a system for academic purposes, so he designed it using a micro-kernel architecture, ideal for ease of comprehension, using innovative technology for the period, which provided versatility, was multiplatform etc. This has been one of MINIX's strengths and weaknesses at the same time; although the system is a little jewel for study and design purposes, it is unlikely to be used in real environments. It was made easy to understand, modular and good for teaching purposes, but not fast. In any case, that was not Tanenbaum's intention; over the years MINIX has evolved and nowadays it continues to be studied by many pupils in universities worldwide.

This is where **Linux** comes into play. While the FSF continued with its grand project providing tools for the construction of an operating system, Tanenbaum was designing MINIX for academic purposes and many companies continued to evolve their own versions of UNIX. **Linus Torvalds**, a student at the University of Helsinki, decided to create his own kernel for a new operating system, Linux, in August 1991. His idea was to create a UNIX for PC so that anyone who wished to could use it on their computer. His first appearance on the scene was in a debate about MINIX and operating systems, where he put forward the following ideas:

```
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Date: 25 Aug. 91 20:57:08 GMT
Organization: University of Helsinki
Hello everybody out there using minix.
I'm doing a (free) operating system (just a hobby,
won't be big and professional like gnu) for 386(486)
AT clones. This has been brewing since april, and
is starting to get ready. I'd like any feedback on
things people like/dislike in minix, as my OS res-
embles it somewhat (same physical layout of the
file-system (due to practical reasons) among other
things).
I've currently ported bash(1.08) and gcc(1.40), and
things seem to work.
This implies that I'll get something practical
within a few months, and I'd like to know what fea-
tures most people would want. Any suggestions are
welcome, but I won't promise I'll implement them :-)
```

If we enter the forum where this first message appeared, we will see how quickly people from all over the world became interested in this new system, which also had the characteristics of free software because it used the GNU compiler and commands interpreter. Although in Torvalds's own words, if he had known the amount of effort involved for his idea to work, he would never have done it: the efforts of many IT experts all around the world made this project possible.

Linux's monolithic kernel

Linux, the kernel of GNU/Linux, is of the monolithic type. This tells us that its different functionalities are not split into separate modules, but rather that it all forms part of the same program. The main inconvenience with this type of design is that locating bugs and maintenance is very costly. On the other hand, its performance is much better than with other types of design.

In fact, during the first few years of its existence, GNU/Linux was identified as the operating system used by hackers. The fact that it was difficult to install, handle and its lack of drivers made it a tool suitable only for people who were very knowledgeable about operating systems. It was these first users who designed the drivers for the disks, printers, cards etc., and who started generating awareness of this system around the world. Bit by bit, the number of users started to grow and now there are many companies and groups of users who create their own GNU/Linux distributions.

1.3. Distributions

At present, there are many different distributions based on GNU/Linux. They exist for all types of computers and electronic devices: laptop or desktop computers, pocket PCs or PDAs, wireless access network points etc. The nature of free software allows this: anyone can take the code developed up until now and adapt it to their own requirements. It is a fact that an increasing number of companies and users choose systems based on GNU/Linux due to its good performance and the amount of available software.

Linux and the GNU project

Although many GNU/Linux distributions are called just Linux, it is important to point out that Linux is just the kernel of the operating system and that the GNU project is what has really contributed much of the structure for it to function.

At all events, although there are dozens of distributions, some are more popular and have become widely used. The free software philosophy has led many companies, creators of their own GNU/Linux distributions, not to restrict access to their code. Even so, the support that they offer and the material that they sell provide profits, allowing them to subsist. At the same time, we should bear in mind that many of these distributions include proprietary software that some users prefer, although in many cases there may be similar programs with a Free Software license.

Next we will briefly describe a few GNU/Linux distributions:

- Slackware: one of the first distributions to appear. It was created by Patrick Volkerding and was enormously successful during its first years of existence.

- Debian GNU/Linux: one of the first distributions of GNU/Linux to appear and which continues to exist and evolve. The packages system allows us to clearly distinguish between free and non-free software, helping us to set up the system with exclusively Free Software licence programs. Developed by a group of collaborators distributed all over the world without the backing of any company. Although it is one of the most stable and secure in existence, its installation and configuration system requires previous knowledge.

- RedHat Linux: together with SuSE, it is one of the most popular distributions. It is created by a company in the US and provides great quality software. It has a very intuitive environment that makes it very easy to install and configure.

- SuSE Linux: although it is a fairly recently created distribution, it has found wide acceptance. It is developed by a German company and contributes a lot of quality proprietary software. It is very complete and easy to install and maintain, although in some aspects it does not follow some of the community's standards.

- Knoppix: distribution on a live-CD based on Debian. It automatically detects all sorts of hardware and provides the latest KDE desktop and the office applications suite OpenOffice.org. Very useful for demonstrations and new system users.

- Ubuntu: it is a Linux distribution that offers an operating system primarily aimed at desktop computers although it also provides support for servers. Based on Debian GNU/Linux, Ubuntu's main focus is ease of use, freedom in the restriction of use, regular releases (every 6 months) and ease of installation. It exists in both live-CD format and installable format. It automatically detects all sorts of hardware (even the latest).

We should not forget that there are other operating systems compatible with UNIX and with currently observed standards. Many of the concepts and tools that we will encounter throughout this course will also serve for these other ones. In particular, we would mention GNU/Hurd (kernel developed by the GNU project) and FreeBSD.

Activity

1.4 Read the description of some of the current distributions based on GNU/Linux: http://www.linuxhq.com/dist.html.

1.4. Programs and documentation

Internet has always been the main means of communication between free software developers and users. For this reason, since the beginning of GNU/Linux's enormous expansion it has been possible to find a lot of information about the operating system on the Internet. We can download most programs from the Internet, packaged using any of the usual systems or directly from source code so that we can compile it on our own system. Also, most distributions can be downloaded from the Internet without having to purchase any special package from specialised magazines or from the companies that produce them. It is also true that if we wish to have the support offered by some of the distributions, it is better to buy all the material that can be provided (CD, manuals etc.) and to register.

As we become familiar with free software and GNU/Linux, we will realise the importance of knowing how to find the documentation we want. When we face a problem, before starting to go round in circles wondering how to solve it, we should think that someone else has probably already come across something similar. Searching for and finding the documentation that best responds to the problems we encounter from time to time will save us a lot of time and effort. The free software community generates hundreds of documents that we can download freely from the Internet, in addition to discussion forums, news pages and gossip pages etc.

Some of the most popular references that can best help us are:

- Documentation
  - The Linux Documentation Project. We can find most existing manuals, HOWTOS, FAQS etc. on this site, which additionally is in several languages. http://www.tdlp.org
  - LinUx in Spanish . Large documentation project for GNU/Linux HOWTOS, guides etc. in Spanish. http://lucas.linux.org.mx
  - The HOWTO of HOWTOS. http://lucas.linux.org.mx
  - Linux.com. Page with different sections of news, documentation etc. http://www.linux.com
  - Documentation for Debian GNU/Linux. http://www.debian.org/doc/
- News
  - Slashdot. News and gossip from GNU/Linux community. In English. http://slashdot.org
  - Barrapunto. The Slashdot equivalent in Spanish. http://barrapunto.com
  - Puntbarra. The Slashdot equivalent in Catalan. http://puntbarra.com

  - Bulmalug. New Linux users from Mallorca and thereabouts. News and sections dedicated to specific topics. http://bulmalug.net

  - GNU news in Spanish. http://www.es.gnu.org

  - Linuxtoday. Another very practical news page to be up to date with the latest news. http://www.linuxtoday.com

  - Libertonia. News page. Its News Sources section is particularly interesting with numerous links to other similar style pages. http://libertonia.escomposlinux.org

- Forums
  - Foroslinux.org. Several GNU/Linux forums about all sorts of topics.

  - Linux Security forums. Forums focused on security issues and similar.

- Search
  - Linux on Google. The biggest search engine in the world also for GNU/Linux. http://www.google.com/linux

  - Buscadoc. Search engine for IT documents in Spanish.

- Distributions
  - Official web page of the Free Software Foundation. http://www.fsf.org

  - Official web page of Debian GNU/Linux. http://www.debian.org

  - Official web page of RedHat Linux. http://www.redhat.com

  - Official web page of SuSE. http://www.novell.com/es-es/linux/

  - Official web page of Slackware Linux. http://www.slackware.com

  - Official web page of Knoppix. http://www.knoppix.com

  - Official web page of Ubuntu. http://ubuntu.org

- Downloads
  - Sourceforge. The biggest page with free software projects. http://sourceforge.net

  - Linux on Softonic. GNU/Linux download section from one of the many downloads pages. http://www.softonic.com

  - Download. Downloads page. http://www.download.com

- Others
  - Linux Security. A page with pertinent information on all types of security issues with GNU/Linux. http://www.linuxsecurity.com
  - LinuxHQ: General information on GNU/Linux distributions, security etc. http://www.linuxhq.com
  - Linux Journal. Page of news and articles regarding GNU/Linux. http://www.linuxjournal.com
  - Linux Gazette. GNU/Linux magazine. http://www.linuxgazette.com
  - Linux-mag. GNU/Linux magazine. http://www.linux-mag.com
  - Official web page of the XFree86 project. http://www.xfree86.org

2. Basic concepts and commands

2.1. Introduction

In this section we will learn the basic ideas and instructions needed for moving around the system. If we are not used to using the command line to deal with the operating system, at first it may seem a bit complicated, but as we use it we will see that command lines are very useful and that they allow us to perform any task we need. Also, knowing how to use the command line correctly will be very useful when we need to connect remotely to a machine where we will, similarly, be able to design small programs (shell scripts) in order to automate the most common administration tasks.

Command

A command is a program that performs a specific action related to the operating system.

Most commands that we will find in this section form part of the standard (IEEE POSIX norms) and are shared by all GNU/Linux and UNIX systems. Although each distribution has its own administration and management applications, many of the actions they perform can also be carried out with the commands that we will see. Using them, we will be able to manipulate almost all of the system's features and to move through the system efficiently. By learning how to use these commands correctly, we will also learn to navigate through any system based on GNU/Linux, regardless of what distribution we use.

Each of the system's commands tends to have a large amount of different parameters. By passing parameters, we can use the same command to carry out many different actions, even if they are all of the same style. In this document we will not specify the different parameters of each command that we look at, since the text would become too long; neither does it make sense to know exactly all of the possible parameters for each one. They all have an extensive manual, specifying every option, so that any time we need to perform a specific action we can turn to it. In the workshops spread throughout this course, we will see some of these options, although it is important to know that with the manual we can always discover many others that can help us to carry out everything we need.

Command parameter

A parameter is a particular option of a command, which we add following the command, preceded by a space, and often by a hyphen. For example, if the command is `list`, we could add a parameter such as `list all`.

2.2. Users and groups

Currently, most operating systems in use are multi-user and multi-task. This means that more than one user can work on the system at the same time, carrying out one or more tasks simultaneously. For this reason, it is very important for the operating system to have mechanisms that allow us to control and

Names policy

A frequently used names policy is to put as a login the first initial of the user's name followed by the user's surname.

manage users correctly: the access and identification system (login), the programs that each user can run, security mechanisms to protect the equipment's hardware, protection for user files etc.

Operating systems based on UNIX organise all this information into users and groups. To enter the system, we need to identify ourselves using a login and a password. The login tends to be a name that identifies the **user** unequivocally. For systems with more than just a few users, it is important to have a good names policy so that all users can be clearly identified. The password must consist of a combination of letters, numbers, and special characters. It should not consist of any dictionary word or similar because this would constitute a serious security risk. The passwords system is of the unidirectional type. This means that our password is not stored as text, instead it is ciphered and saved as it is. When we enter the system and write our password, it is ciphered and compared to the one stored. If they match, identification is positive, if they don't, there is no identification. What is important about this system is that through the ciphering we cannot obtain the original password in any way. All that programs designed to break user passwords can do is to cipher words based on dictionaries (using automated systems to derive them and search for variants) and test whether they coincide with the ciphering of any user password. Therefore, we need to choose our passwords carefully; otherwise we could compromise the security of the whole system.

NIS

NIS are a series of applications that allow us to manage all users of the same network in a centralised manner on a single server.

Currently, on GNU/Linux systems we can choose two possible types of ciphering for user passwords. The one used since the beginnings of UNIX is 3DES. The only inconvenience with this type of ciphering is that it only allows us to have 8-letter passwords (if we write more, they are ignored), unlike the other type of ciphering, called MD5, which allows us to use passwords of any length we like (in fact, MD5 is a hashing system, but can also be used to cipher passwords unidirectionally). The longer the password the safer it is, so we recommend using the second type of ciphering. In any case, we should remember that if we need to use a special program to manage users, such as NIS, it may not be compatible with MD5.

Although a user is an individual that can access the system, a group is a set of users that share the same characteristics, meaning that it is useful to group them together so that we can give them a series of special permissions on the system. A user must belong to at least one group, but can belong to more than one. The system also uses this entire mechanism of users and groups to manage installed applications servers and other mechanisms. For this reason, in addition to real users, a system will have many other users associated to tasks that need to be carried out on the operating system. Generally, this type of user will not be able to enter the system (using a normal login).

Server

A server is a program responsible for providing some type of service (such as to serve web pages, allowing users to connect remotely etc.), generally linked to the Internet.

Any operating system must have a super-user (root). The root user has maximum privileges to carry out any operation on the operating system. It needs to exist, since the root will be responsible for administrating and managing servers, groups etc. This account should not be used for working normally on the system. We should only enter as root when it is really necessary, and will use other accounts for normal user work. This will prevent us damaging the system through erroneous operations or in testing malicious programs etc.

All user and group information is stored in the following files:

- `/etc/passwd`: information (name, directory, home etc.) of the user.
- `/etc/group`: information about user groups.
- `/etc/shadow`: users' ciphered passwords and configuration for validity, changing them etc.

Using the `shadow` file is optional. Originally, ciphered user passwords were saved in the same `passwd` file, but for security reasons (a lot of mechanisms need to be able to read this file, meaning that it was very easy to get hold of it and try to crack the passwords) it was decided to change this mechanism so that the `shadow` file was only accessible for a few users with special privileges on the system. This option can be configured during the system's installation and it is normally advisable to use it. All of these files are organised in lines, where each one identifies a user or a group (depending on the file). On each line there are several fields separated by the character ":". For administration tasks, it is important to know what these fields are, which is why we are going to look at them in a bit more detail:

Password configuration

It is also possible to configure the system to use a `shadow` file for groups (in the event that they need to be given a password). This file would be named `/etc/gshadow`. Generally, the password configuration is given when the system is installed, although it can all be changed and adapted to our taste, using the Pluggable Authentication Modules for Linux (PAM), which are the programs responsible for the entire user authentication system.

- `passwd`

  1) Login: the user name. There cannot be any two identical names, although there can be a name that coincides with a system group.

  2) Ciphered password: if we do not use the `shadow` file, ciphered passwords will be stored in this field. If we use the `shadow` file, all existing users in this file need to also be in the `shadow` file and we need to type an "x" in this field.

  3) User ID: user identification number. This is the number that the system uses to identify the user. 0 is the only one that is reserved for the root.

  4) Group ID: the number of the group to which the user belongs. Since a user can belong to more than one group, this group is called primary.

Cracking a password

Cracking a password means obtaining the password through the use of specially designed programs. These programs are also used by system administrators to discover which users are using passwords that are too easy to discover (good passwords cannot be cracked in any way without using large supercomputers).

5) Comments: field reserved for entering the comments we wish about the user. It is normally used to enter the full name or some type of personal identification.

6) Home directory: the user's home directory is where the user can save all his files. They are normally all saved into a system folder (generally `/home/`) and organised by groups.

7) Commands interpreter: a commands interpreter (shell) is a program that reads everything we write using the keyboard and runs the programs or commands that we specify. There are dozens of them, although the one most commonly used is undoubtedly, `bash` (GNU Bourne-Again SHell). If in this field we write `/bin/false` we will not allow the user to execute any command on the system, even if the user is registered by the system.

- `group`

  1) Group name.

  2) Ciphered password: a group's password is used to allow the users of a specific group to change to another group or to execute some programs with the permissions of another group (on condition of having the password).

  3) Group ID: group identification number. This is the number with which the system identifies groups internally. 0 is the only one reserved for the root group (administrators).

  4) User list: the names of the users belonging to the group, separated by commas. Although all users should belong to a specific group (indicated in the fourth field of the `passwd` file), this field can be used for users of other groups to have the same permissions as the group in question.

- `shadow`

  1) Login: it should be the same name as used in the `passwd file`.

  2) Ciphered password.

  3) Days, since 1st January 1970, until the password was last changed.

  4) Days to pass before the password can be changed.

  5) Days to pass before the password should be changed.

6) Days before the password's expiry when the user will be notified to change it.

7) Days that can pass following the password's expiry, before the user's account is blocked (unless the password is changed).

8) Days, since 1st January 1970, since the account was blocked.

9) Reserved field.

When a user enters the system, he is placed in his home directory and the configured shell is executed. The user can now start working. Only the system's root (or the root group's users) have permission to handle user and group information, to register and deregister them etc. There are lots of commands for dealing with this. Additionally, each one has several different parameters for managing all the fields we have seen before. Next, we are going to take a look at some of these commands:

Dates on UNIX systems

On UNIX systems, dates are normally shown according to the number of seconds elapsed since 1st January 1970.

Disk quota

For systems with hundreds of users it is common to implement some sort of mechanism to restrict the disk space that each one can use. On GNU/Linux systems, this system is called a quota.

- `adduser`: this allows us to add a new user to the system. The way of adding it (if we do not specify anything else) can be configured in the file `/etc/adduser.conf`. A number of different options can be specified referring to the home directory, the shell to be used etc.

- `useradd`: creates a new user or changes users' default configuration. This command and the preceding one can be used to achieve the same tasks.

- `usermod`: with this command we can modify most of the fields in the `passwd` file and `shadow`, such as the home directory, the shell, password expiry etc.

- `chfn`: changes the user's personal information in the comments field of the file `passwd`.

- `chsh`: changes the user's shell.

- `deluser`: eliminates a user from the system, deleting or saving all the users' files depending on the parameters we set, with a backup copy of them or not etc. The configuration used by default with this command is specified in the file `/etc/deluser.conf`.

- `userdel`: command with the same possibilities as the preceding one.

- `passwd`: we can use it to change a user's password, information on password expiry or to block or unblock a particular account.

- `addgroup`: allows us to add a group to the system.

- `groupadd`: the same as the previous command, but with different parameters.
- `groupmod`: allows us to modify the information (name and GID) of a particular group.
- `delgroup`: eliminates a particular group. If a user still has it as primary, it cannot be eliminated.
- `groupdel`: as in the previous case.
- `gpasswd`: we can use it to change the group's password.

To know what user we are we can use the `whoami` command, which will show us our login. `groups` allows us to know which groups we belong to and `id` will show us user and groups. It is also interesting to be able to become another user without having to exit the session (`login` or `su` command) or to change groups with the `newgrp` command. We should only use this command when we do not belong to the group in question and know the password (which ought to be activated in the `group file`). If all we need are the permissions of the group in question to run a particular command, we can also use `sg`.

GNU/Linux versatility

As we can see, with GNU/Linux we have more than one way of carrying out certain tasks. This is the system's overall approach: we can edit the files directly and modify them ourselves, use some of the existing commands, create them ourselves etc. Of course, we can also choose what we like best.

As we mentioned previously, GNU/Linux is a multi-user operating system, meaning that at any given time there can be several users connected to the system simultaneously. To know what users there are at any specific moment, we can use the `who` command, which will show us the list of users on the system. `w`, it also shows us what they are doing. We can communicate with them using the `write` command, with which the message that we have written on the specified user screen or `wall` will appear, writing the content of the file that we have specified to all users on the system. In order to activate or deactivate the receive messages option we have the `mesg` command. We can also have a personal chat with a user by using the `talk` command.

2.3. The file system

2.3.1. The file system hierarchy

Any operating system needs to save numerous files: from system configuration files, to log files or user files etc. In general, each operating system uses its own file system characterising it in many aspects, such as performance, security, reliability etc. GNU/Linux is capable of reading and writing files with any of the file systems currently in existence, although for its own root and main directories it requires a file system that allows certain operations. Generally, we use the type ext2, ext3 or ReiserFS. The ext2 is the most common and widely used. Its performance is fairly good, it incorporates all sorts of mechanisms, security and tuning and is very reliable. ext3 is its evolution and incorporates a technology known as journaling. One of the main advantages of this technology is that if the power supply is cut off and the computer switches off without the correct shut down, the file recovery systems are more effective. ReiserFS is a new type of system incorporating new design technologies to make it faster. During the installation process the operating system will ask us which of these three we wish to use. Normally, we will use ext2 or ext3 because they have been tested more than ReiserFS.

Filesystem

The file system is the program (or modules of the operating system's kernel) responsible for all operations related to file handling and storage. These are the functions that deal with the computer's physical storage devices, such as the hard disk.

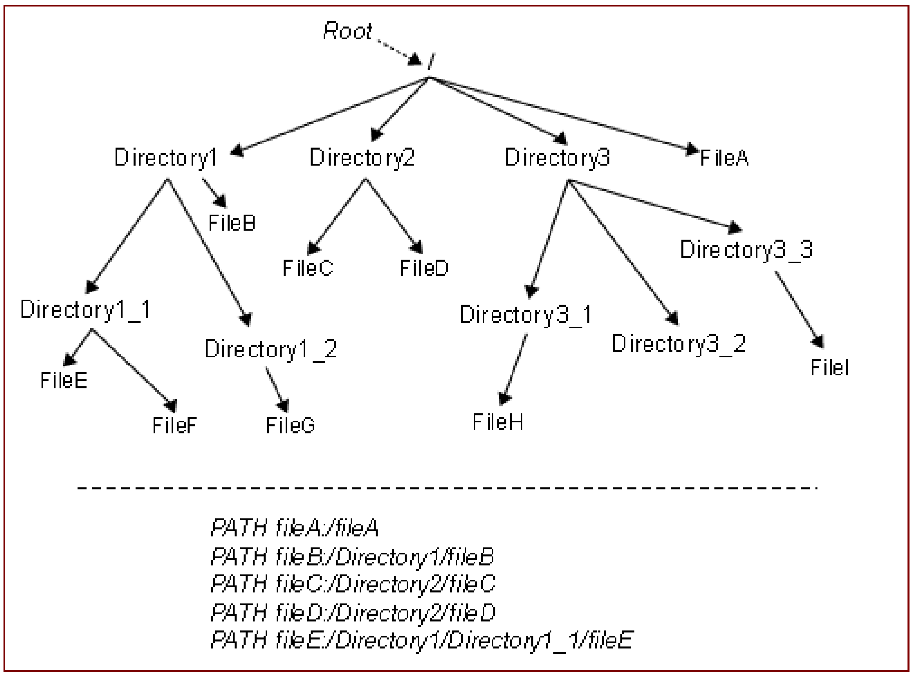
A very important characteristic of all operating systems based on UNIX is that all the system's devices can be handled as if they were files. Likewise, when we wish to access the content of a CD, diskette or any other storage device, we will have to mount it on an already existing system directory and will navigate through it as if it were merely one more file (use of the different units A:, B:, C:, D: etc. is a scheme that only exists in Windows^{TM} type operating systems.

ext2 filesystem

The ext2 file system has been designed to handle small files, which is what an operating system tends to have more of, quickly. It is not so good at handling and manipulating large multimedia files, although we can always do a bit of tuning to adapt it to our needs.

The first thing we need to understand clearly is that any file system starts from the same root, which we refer to with the character "/". This is the origin of the entire file system and there is only one. To correctly organise the files, the system provides what we call directories (or folders), where we can place files and more directories. This allows us to obtain a hierarchical organisation as shown in the following figure:

Figure 2.1



2.3.2. System directories

Most operating systems on the market follow the FHS standard, which specifies the main characteristics that any operating system should have. These include how we should distribute our files into directories in order to organise them correctly and to be able to find them quickly and easily. Most distributions based on GNU/Linux follow these recommendations and we can find the following main directories:

- `/bin/`: basic commands for all system users.
- `/boot/`: static files required for booting the system.
- `/dev/`: system devices.
- `/etc/`: system configuration files and the applications installed on the system.
- `/home/`: directory for users' home files.
- `/lib/`: essential libraries for the system's kernel and its modules.
- `/mnt/`: temporary mounting point for devices.
- `/proc/`: system kernel processes and variables..
- `/root/`: home directory for the system's root.

- `/sbin/`: essential commands for the system's root.
- `/tmp/`: temporary files. Depending on the distribution (or configuration) that we use they are deleted when the system is booted or every certain period of time.
- `/usr/`: second hierarchical structure, used for storing all of the software installed on the system.
- `/var/`: directory for the printing spoolers, log files etc.

It is highly recommended to save and not eliminate any of these directories (or those that the distribution we are using creates by default), since they are fundamental for the correct functioning of the system. Generally, the new application installation processes require this organisation and many program configuration files need to be in specific directories. What we can do without any type of restriction is to create new directories on the system's root or in any other folder.

2.3.3. Moving about

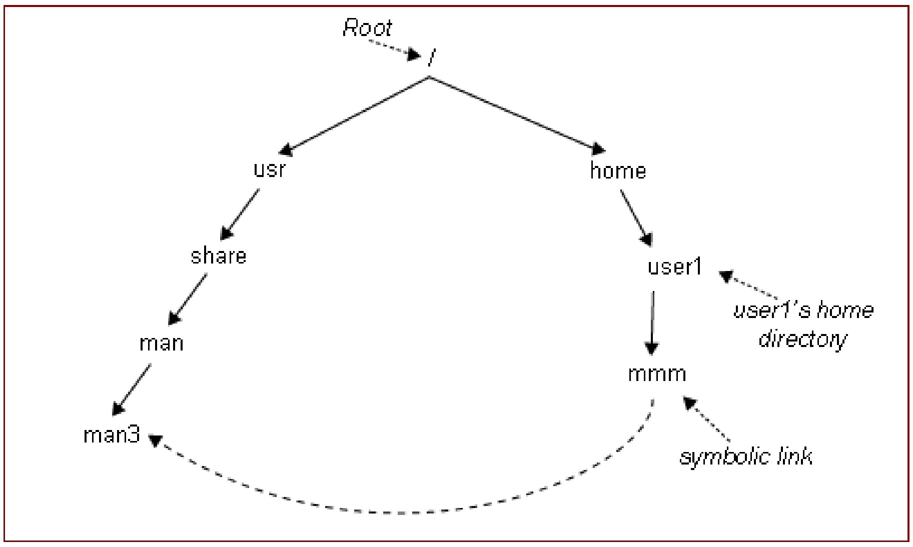
In order to move about the directories structure we need to use the commands for listing contents and changing folders. When we enter the system, the login will normally place us in our home directory, which is generally referred to with the character "~". If we want to see what there is in the directory we are in, we can list the contents using the `ls` command. We should remember that by default the command will not show us files that start with a dot. By using the `-a` parameter all files will be listed. In all directories there is a "." entry and another ".." entry. The dot refers to the current directory, whereas the two dots refer to the directory immediately above the current one (in the hierarchy tree). Naturally, when we are in the file system root, the entry ".." will not exist because we are at the top level.

To change directory we can use the `cd` command. If we do not add any parameter we will be taken to our home directory by default. Generally, we tend to indicate where we want to go, in absolute or relative terms. In relative terms, it means that we will start from the directory that we are in at the moment of executing the command. For example, if we are in the `/usr/bin/` directory and we want to go to the `/root/`, we should enter the following command: `cd ../../root` (the first two dots indicate `/usr/` and the next ones, the system's `/` root, from where we can access `/root/`). In absolute terms, we always start from the root, meaning that the command we would use in the previous example would be: `cd /root`. To know what directory we are in we can use the `pwd` command.

2.3.4. Links

Other mechanisms that most file systems provide are known as links. A link is a bridge to a file or directory belonging to the system; a reference that we can place anywhere we wish and that acts as a direct access to any other. This mechanism allows us to access files or folders more quickly and conveniently, without having to move through the directories hierarchy. Let's look at it with an example: let's imagine that we are a user (user1) that frequently needs to access directory `/usr/share/man/man3/`. Instead of writing the long command that takes us to the directory in question every time we need to go there, we can create a link in our own directory that immediately redirects us to it. The `ln -s /usr/share/man/man3 mmm` command would create this bridge for us, called `mmm`. The user will just have to write (from the home directory) `cd mmm` and the system will automatically redirect him or her to `/usr/share/man/man3/`. It is important to bear in mind that when executing a `cd ..` command to go to the directory above, we would return to the home directory and not to `usr/share/man/`, since we accessed it from our link. We can visualise this scheme graphically as follows:

Figure 2.2



When we created the link in the previous example we passed the `-s` parameter to the command. This indicates that we wish to create a symbolic link. Symbolic links mean that we are just creating a pointer or bridge towards the file or directory, so that if we deleted the destination file, the link would point nowhere. If we do not add the `-s` parameter we would create what is known as a **hard link** which, unlike the preceding one, makes a duplicate of the file. In fact, internally it is not exactly a duplicate; it is like two entries that point to the same data. This means that if we modify one or the other, the two are left the same. The advantage of this type of link is that if we delete either of the two copies of the file, the other one will still be kept. This type of link is not commonly used because it complicates file management and handling (it is always better to have a single copy of the files). Furthermore, if we make a

Creating hard links

A hard link can only be created between files or directories on the same unit due to the internal mechanism used to manage them.

hard link to a directory, all files and subdirectories within it would also have to be referenced. For this reason, only the system's root can create hard links to directories. Another difference is that with a symbolic link we can see what file we are pointing to, but with a hard link we cannot (due to the mechanism used for them internally).

2.3.5. Permissions

In any multi-user operating system we need the files saved on our disk to have a series of features that allow us to see them, modify them or execute them for the users that we define. Although there are several alternatives for achieving this, GNU/Linux uses the classical users and groups system, allowing us any possible configuration. What interests us is to define, for every file or directory, what user and group it belongs to and what permissions it has for each one of them, as well as for the rest of the system's users By executing `ls -l` we will see how in each file of the directory that we are in, a line similar to the following appears:

```
-rwxr-xr-x 1 user1 group1 128931 Feb 19 2000 gpl.txt
```

The first ten characters (starting from the left) tell us the file's permissions as follows:

- Character 1: this entry tells us whether it is a file or a directory. If it is a file, the "-" character will be displayed, whereas for directories a "d" will appear.

- Characters 2, 3, 4: tell us, respectively, read, write, and execution permissions for the file owner. In the event of not having the corresponding permission enabled, we will find the character "-" otherwise "r", "w" or "x", depending on whether we can read, write or execute. For the third character, we can also find an "s", which tells us whether the file is of a SetUserId type, which means that when we execute it, it will obtain the file owner's permissions. If the permission is only "x", when the program is executed it will do so with the permissions of whoever has launched it.

- Characters 5, 6, 7: these characters have exactly the same meaning as the preceding ones, but refer to the permissions granted to the users of the group to which the file belongs.

- Characters 8, 9, 10: as in the preceding case, but for other system users.

Following these 10 characters we have a figure that tells us the number of hard links that the file has. For directories, this number shows how many folders there are in it as well as the number of hard links it has (when there are none, the number is 2, for reasons of the operating system's internal management). Next we see the file's owner and group, followed by the size it occupies (in bytes) and the date it was last modified. For all files the date of creation, latest access and modification is saved, which we can manipulate using the `touch` command. At the end we have the file name, which is case sensitive and we can have any type of characters without any problem.

The SetUserId mechanism

The SetUserId mechanism is very useful when a program requires the owner's permissions in order to access certain files or to carry out some kind of operation on the system. In any case, we need to be very careful with this type of file because it can cause system security failures if used inappropriately.

In order to change the permissions of a particular file we can use the `chmod` command. We need to remember that only the file owner (or root) can change these permissions, since otherwise, the mechanism would make no sense. We can use this command in many different ways, but the two most frequent ones are as follows:

- The first way of using it is in the form of `chmod` *`XXX filename.`* The *X*s have to be three numbers between 0 and 7. The first number indicates the permissions we wish to establish for the user, the second one, for the group, and the third one, for all others. To interpret the permissions that we will apply using the numbers 0 to 7 correctly, we need to make use of the binary representation of the number in question, meaning that the first digit will indicate the permission to write, the second one to read and the third one to execute. In each case, 0 means that the permission in question is not given and 1 that it is. We can see this relationship in the following table:

Table 2.1

Decimal representation	Binary representation	Meaning
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

- The other way of using the command is to indicate explicitly the permission we wish to apply or eliminate from the file. The way of doing so is by indicating first, whether we refer to the permissions of the user, or the group, or all others, with the letters "u", "g" or "o" respectively. Next, we will add a plus sign (+) or a minus sign (-) depending on whether we wish to add or eliminate the attribute, which we will indicate with "r", "w", "x" or "s" (the latter in the case of SetUserId). Also, we can carry out all possible combinations, referring to more than one permission and/or user. For example, `chmod go+r gpl.txt` would give a read permission to the group and to other users for the file `gpl.txt`.

In order to change the file owner we have the `chown` command, which can only be used by the root for security reasons. In order to change the group of a particular file, we can use the `chgrp` command. As we can imagine, when a user creates a new file, the system will put the user who creates it as the owner and assign it as belonging to the user's primary group. The permissions established by default when a new file is created can be configured through the `umask` command, which we will have to give the same notation of three decimal numbers between 0 and 7 that we looked at above but with accessories. For example, if we want our files to be initialised with the permissions `rwr--r--`, we should write `umask 133`.

File security policy

If users are allowed to change file owners, the system's security could be compromised, since malicious actions could be carried out and then the file owner changed in order to blame another user.

2.3.6. Handling, patterns and searches

Now that we know how to move properly through the directories hierarchy, we also need to know how to copy, eliminate and handle other file aspects in the right way. The `rm` command is the one we use to eliminate the files we specify. In order to eliminate a directory, we can use the `rmdir` command, although it will only delete it once the directory is empty (if we wished to fully delete a directory and all of its content, we could use `rm r`). In order to copy files from one place to another, we have the `cp` command, which needs to be told which is the file or directory of origin and the place or name of the destination, even if it is within the current directory. Therefore, if we wish to copy the file `/home/user1/gpl.txt` into the current directory (and with the same name) we should type `cp /home/user1/gpl.txt`, i.e. using the dot (.). If instead of copying the files we wish to move them, we can use the `mv` command. A very useful mechanism provided by the system is patterns. Up until now we have looked at how to carry out certain operations on a particular file. When we are handling a system, we will often be interested in applying one of the operations that we have looked at, but over a large group of files. Patterns will allow us to apply the operations that we wish specifying with just one instruction several files that fulfil a series of specific characteristics. We should look at them as names templates, with the character "*" meaning any chain of possible characters with the "?" serving as a joker for any character. Thus, if we wish to list all files beginning with "s", followed then by any other character, then followed by an "a", and then any other chain, we could use `ls s?a*`. Between "[ ]" we can include other characters, indicating that the pattern is successful if we find one of them in the name. For example, if we wish to refer to all files beginning with "a" or "b" and that continue with any other chain, we could write the pattern `[ab]*`. If after the "[" we type the character "!" (`[!ab]*`) we would be indicating the pattern coinciding with any

Pattern syntax

Pattern syntax can be very complex, allowing us to refer to any set of files that we wish.

file not starting with "a" or "b". Finally, to help with certain searches, within the "[ ]" we can specify classes of characters as follows: `[:class:]`, where the `class` can be any of the ones shown in the following table:

Table 2.2

class	meaning	class	meaning
alnum	[A-Za-z0-9]	alpha	[A-Za-z]
blank	[\]	cntrl	control cars
digit	[0-9A-Fa-f]	graph	printable cars (without spaces)
lower	[a-z]	print	printable cars (with spaces)
punct	[.,¡!¿?:;] ...	space	[]
upper	[A-Z]	xdigit	[0-9A-Fa-f]

A-Z indicates characters from A to Z, \t is the tabulator and \n is a line feed.

Of course, we can use the patterns with any of the commands that we have seen and most of the ones that we will look at next. Also, most of the commands to list, eliminate, copy etc. files can also be passed recurrently. Therefore, entry and execution of the corresponding instruction will be made on all files and directories from where we are until we reach the last level in the hierarchy.

Another type of very useful operation is the file search. We have several commands to allow us to make different types of searches over all the system's files.

Updatedb order

If we wish to update the internal database that uses the `locate` command, we can use the `updatedb order`.

Table 2.3

`find`	This is the most versatile command for performing this operation. It allows us to filter files to find ones with a particular name, modified or created on a particular date, having certain permissions etc. Its only disadvantage is that it does not use any type of mechanism to accelerate the search, meaning that it can take quite a long time.
`locate`	This is another command, which unlike the preceding one, uses an internal database that is periodically updated allowing us to make quicker searches. We need to remember that the results will not always be updated, however, in addition to the fact that we cannot make searches as versatile as with `find`.
`whereis`	Finally, `whereis` is oriented at searching for binary files (executables), rather than source code files for a particular program.

2.3.7. File type and content

We can have many different types of files on our system: executables, text files, data files etc. Unlike other systems, which use the file extension in order to determine the type of file, GNU/Linux uses a system called magic numbers,

File extension

Using the extension to determine what type of file we have is not very efficient, since anyone can change it generating confusion and system errors.

which through a magic number determines the type of file according to its data (a series of tests are run to determine what type of file it is). The `file` command shows this.

If we need to see a file's content, one of the basic commands is `cat`. By passing the name or names of the files that we want to see, it will display them on screen. We should try not to show executable or data files on screen, since the dumping of non-printable characters can leave the console full of incomprehensible characters (we can always reinitialise by keying in `reset` or `tset`). For very large files, it is better to use the `less` or `more` commands, that allow us to move progressively through the file. If the file is of the binary type and we wish to see what it contains, we can use the `hexdump` or `od` commands to see the content in hexadecimal format or other representations. `strings` will search for chains of characters within a binary file for us and display them on screen.

Another type of commands that are useful are those that look for a certain pattern in the file content. With the `grep` command we can add the file name as the second parameter and as the first the pattern that we wish to look for (using the syntax that we looked at previously, extended to other options). Also, the command allows us to perform many other tasks, such as counting the number of lines on which the pattern appears (`-c` parameter) etc. With `cut` we can separate the content of each file line into fields specifying the separating character, which is very useful for automating system administration tasks. We can also take a particular number of lines from the beginning or end of a file using the `head` and `tail` commands, respectively. With `wc` we can count the number of lines or words, the maximum length of a file line etc. Finally, to conclude this subsection on file handling, we will look at how to compare different files. As with all other operations, we have several commands for doing this. `diff`, `cmp` and `comm` make different types of comparisons using different methods over the files that we specify. `sdiff`, also allows them to be merged, as we choose.

2.4. The processes

The fact that the operating system is multi-task implies that we can launch more than one program at once. A process is no more than a program or application that is loaded in the memory and being executed. Although our computer may have just one CPU, the operating system takes charge of sharing its processing time so that several processes can carry out their operations, giving the impression that they are all being executed at the same time.

Manage processes

Process management is a vital aspect of the operating system, since it determines our applications' response times, efficient memory and CPU use etc.

In order to identify each process unequivocally, the system's kernel assigns them a PID number (Process IDentification). Although we could think of referring to them by name only, this number is essential because this way we can execute the same program as many times as we like, and also run different instances of it. To know what processes are being executed, we can use the `ps` command. To explore this processes mechanism in more depth, we will discuss some of the parameters that we can apply to this command in more detail:

- `T`: this option comes by default and tells us that we will be shown only the processes being executed on the terminal that we are on or from which the processes have been launched.

- `-a`: shows us the processes of all system terminals.

- `-`: shows us all the system processes. If we execute the command, we will see that in addition to the programs being run by the users, there are others. Many of these execute the functions required for the operating system to function correctly, others are configured application servers etc.

- `-l`: it displays extensive information for each process, such as the CPU time used, the terminal on which it is executed etc. In the second column we can also see the process status. Although the system may have many processes being executed at the same time, this does not mean that they all require constant CPU time. For example, when a web pages server has no requests, it does not need to perform any operation at all. Although it is in the memory ready to execute the moment it receives a request, it is better for it not to pass through the CPU at any time, since the CPU can then be used for other processes that do need it. Internally, the operating system has a series of very efficient mechanisms implemented for managing all these types of operations. Thus, a process can have any of the following statuses (displayed with the corresponding character):
  – `D`: uninterruptible process. This type of process generally tends to belong to the input/output of some type of device that could be damaged if left unattended.

  – `R`: process that when the command is executed is also being executed, in other words, all those in the execution queue. The process execution queue is where we place all the processes that share the CPU time.

  – `S`: process dormant or pending some type of event for the system to wake it and place it in the execution queue.

  – `T`: process stopped by the user or the system.

  – `Z`: zombie process. This status indicates that the process has had some type of failure and does not function correctly. Usually, it is better to eliminate this type of process.

- Another very useful command is `top`, which tells us interactively about the system processes, CPU use, used and free memory, RAM used by each process etc. This program is very appropriate when the system is not responding adequately or when we notice a strange fault, since it helps us to find quickly the process that is negatively affecting the system's performance.

As we have seen, the system tells us about all possible aspects of the system's processes. In addition, we can send certain signals to the processes to inform them of some event, to take them out of the execution queue, eliminate them, give them higher priority etc. Knowing how to handle all these aspects correctly is also very important, since it will allow us to use our computer more efficiently. For example, if we are the administrators of a calculation centre, where most of the executed applications require a lot of CPU time, we could configure the system so that the most urgent ones are executed with higher priority than others and finish first. The `kill` command allows us to send signals to the processes in which we are interested. In general, all programs are designed to receive this type of signal. Thus, depending on the type of signal received they know what type of operation to perform. There are many different types of signals that we can find in the `kill` manual, although the most frequently used ones are those to oblige a process to end or to pause its execution. With the signal `TERM` (`kill -15 PID`), we are telling the process that we want it to end, so that when it receives the signal it will save what is necessary and finish executing. If there is some type of problem or the program is not prepared to receive this type of signal, we can use `kill` (`kill -9 PID`), which will automatically eject it from the execution queue. `killall` is used to refer to the name of several processes at the same time instead of referring to them by their PID and, thus, we can send a signal to all of them at the same time. With the `skill` command we can also send signals to the processes, but using a different syntax. For example, if we wish to stop all executions of a particular user, we could use `skill -STOP -u` *`LoginName`*, which would stop all processes of the user in question. To reinitiate them again, we could use the `CONT signal`. When we are executing a program on a console and we wish to send the `TERM` signal to it, we can use the Ctrl+C key combination. With Ctrl+Z we can pause a program and revive it with `fg`.

Process handling

With the process handling commands we can carry out any action that interests us: from pausing a specific user's processes, to eliminating those we are not interested in or making some occupy more CPU time to run quicker.

trap command

To handle signals in a shell script (see later how to program them), we can use the `trap` command.

Another way of looking at processes is from their hierarchy. As in the case of the file system, the processes follow a certain hierarchy from parent to child. Any process must be launched from another one, whether the commands interpreter itself, the graphic environment etc., creating a parent-child relationship. With the `pstree` command we can see this hierarchy in graphic terms. If we execute it, we will see how the parent of all processes is one called `init`. All others start from it, and in turn can have more children. This hierarchical structure is very useful, since, for example, by killing a parent process that has many children, we will also kill all its children. We can also use it to identify where certain processes start from etc. If we do not pass any parameter onto the command, by default it will compact all processes with the same name so as not to show too large a structure, although this can also be configurable on the basis of its parameters.

All system processes have a certain priority. As we said before, this priority indicates the CPU time the process will be allowed. The higher the priority of the process, the more execution time it will have in relation to the rest. The range of priorities goes from 20 to 19, from higher to lower. To launch a process with a particular priority, we can use the `nice` command. If we want to give a different priority to a process that is already being executed, we can use `renice`. Only the root can use the negative priority range; this makes sure that the root always has the possibility of executing processes faster than users. By default, the priority with which programs are executed is 0. An aspect to be considered is that with this priorities mechanism we cannot measure a process's real execution time because the CPU is shared by all the processes in the execution queue. In calculation centres, which charge according to the time the machines are used, it is very important to be able to measure this aspect accurately. Therefore, the system offers us the `time` command, which when we pass the program to be measured to it, returns the real CPU time that the program has used.

2.5. Other useful commands

2.5.1. System help

As we have mentioned throughout this document, all commands have many different options and parameters so that we can handle them as we wish. From the beginning it was carefully considered that good documentation would be needed for all of them. Likewise, all this information is needed for the system configuration files, the new applications that we use etc. Therefore, the system itself incorporates a manuals mechanism that we can consult on almost any aspect of the programs, utilities, commands and existing configurations. The most used command is `man`, which shows us the manual for the program that we specify as a parameter. By default, this documentation is shown through the `less` program, which we can use to move forwards and backwards using the PgUp and PgDn keys, to search for a word using the character "/" followed by the word ("n" to search for next occurrences and "N" for the previous ones), "q" to quit etc. The system manuals are divided into different sections:

1) Executable programs (applications, commands etc.).

2) Calls to the system provided by the shell.

3) Calls to system libraries.

4) Special files (generally for devices).

5) Format of the configuration files.

6) Games.

7) Macro packages.

8) System administration commands (generally those that only the root can use)

9) Kernel routines.

If there is more than one manual available for the same word, we can be more specific by passing the number corresponding to the section we want in front of the word, for example `man 3 printf`. As with all other commands, `man` has numerous different options documented in its own manual (`man man`), which we can use for automatic searches, creating a manual file in printable format etc. One of these options that can come in very handy when we know exactly what program we are looking for is `-k` (the `apropos` command does almost exactly the same). With `man -k` followed by a word that refers to the action we wish to carry out, a search will be made through all the system's manuals and all the manuals whose description or name includes the specified word will be shown. Thus, we can find anything we want without having to resort to a book or reference outside the system.

`mandb` command

For fast searches, the `man` application uses an internal database that will search through the files that contain the manuals and index them appropriately. If we wish to update this manual (although normally the system itself already does this automatically), we can use the `mandb` command.

If the manual does not provide us with all the information that we need, we can use the `info` command, which is the same as manual but even broader. If all we want is a brief reference to what a particular program, library etc., does, we can use the `whatis` command.

2.5.2. Packaging and compression

Compressing a file, grouping several files into a single one or seeing what a compressed file contains are tasks that we will carry out frequently in order to make backup copies, move files from one place to another etc. Although there are many different programs that allow us to carry out this type of operations, generally in all GNU/Linux systems we will find the `tar tool.`. This program will allow us to handle one or several files in any way in order to compress them, group them etc. Although the options are interminable and extremely flexible, here we will explain just a few of the more basic ones so that we can have an idea of what we can do with this program. The syntax it uses is as follows: `tar` *`options DestinationFile OriginFile`*, where the destination file will be the new file that we wish to create and the origin files will be those for grouping or compression. It is important to bear in mind that if we wish to group an entire folder, by default the process is recursive, meaning that when it is packaged it will cover all its levels and group everything it contains. To create a new file, we need to pass the `c` parameter, and if we want to save it in a file, we need to pass the `f` parameter. Thus, `tar cf final.tar o*` will package all the files in the current directory that start with an "o". If we also want to compress them, we could use `czf` which would use the `gzip` program after packaging them. To depackage a particular file, the parameter we need is `x`, so we should type `tar xf` indicating the packaged file. If it is compressed, we will have to pass `xzf`.

Although we can use `tar` to compress files, the application is not designed primarily for compression. As we have mentioned, for compression it uses external programs like `gzip`. The `gzip` program uses its own compression format different from the popular `zip`, which we can also use by installing the corresponding application. Another fairly common compression application which gives very good results is `bzip2`. In the following table, we can see the extension that is normally used to identify what format a compressed or packaged file uses:

Table 2.4

Extension	Format
.tar	tar
.gz	gzip
.tgz	tar + gzip
.bz2	bzip2
.zip	zip
.z	compress

2.5.3. Disk operations

Managing and handling the computer's hard disks is another fundamental aspect of system administration tasks. Although later we are going to take a look at how to properly configure the disks that we have installed on the computer, in this sub-section we are going to discuss the commands required to obtain information about them. All hard disks are divided into **partitions**, which we can access as if we were dealing with an independent device, which we will refer to as a **unit**. This is very useful because it allows us to separate information we have on the system appropriately, so that we can have more than one operating system installed on the same disk, for instance. The `df` command shows us, for each unit mounted on the system, the space used and space available. Let's interpret the following `df output`:

Partition configuration

When we format a hard disk partition (with the ext2 or ext3 system), we can configure the size of the block and many other parameters. These parameters can be adjusted to make the system adapt better to our needs and to achieve greater efficiency.

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda1	7787712	421288	6970828	6%	/
/dev/hdb1	19541504	5742384	13799120	29%	/info
/dev/hdc	664432	664432	0	100%	/CD-ROM

As we can see, for each partition or device mounted on the system, the command tells us the number of blocks available and used. The disk block is a unit that is used internally on the storage devices to make handling them more effective. By default, this command shows us the information in 1k blocks, although by passing the `-h` parameter (human readable) we can see it more easily. The first line always shows us the root of the file system and then the other devices. Observe how it also shows us the anchoring point (in the last column), which is the folder we should go to in order to view its content.

Another very useful command is `du`, which shows us what a file really occupies on the disk. For a clearer understanding of what we mean by this, we need to look in a bit more detail at the internal organisation of the disks and how the operating system handles them. As mentioned previously, for efficiency reasons the operating system divides the disk space into smaller parts known as blocks. The size of the block can be configured and generally depends on the size of the disk, although we can also configure it to adapt better to our requirements. Every time we want to add a new file, the operating system will assign it to a block. This way, when reading or acting on it, the operating system can read an entire block directly (of the configured size) in just one step. When the file occupies more than one block, it is assigned more, trying to make sure that they are as close together as possible, so that they can be read consecutively thus increasing the read speed. The only inconvenience with this type of system is that the blocks are wasted when the files are very small, since if a particular file does not occupy an entire block, the remaining space cannot be used for any other. In any case, this type of organisation is the one used by all existing file systems, since it is the most viable for making use of the hard disk. The `du` command, therefore, shows us the number of blocks that a particular file really uses on the disk.

To know what parameters we have configured on our disk units formatted with the ext2 or ext3 extensions, we can use the `dumpe2fs` command, adding the specific partition. We will see how there are many different options that allow us to adjust carefully its behaviour (we can find what each option means in the manual). In any case, once we have formatted a partition, we will not be able to modify hardly any of these options. If we wish to change them, we will have to copy all of the partition's information, reformat and copy the original files again.

Defragging a disk

Defragging (defragmenting) a disk is no more than reorganising the blocks of files to place them in consecutive places to make access quicker. In the file systems that we use with GNU/Linux it is not necessary to defrag the disks (although there are normally programs for this purpose) because the system automatically takes care of organising them properly.

The kernel functions that take care of managing files use a series of methods to speed up the read and write processes. One of them is to use a disk cache, so that we do not constantly have to be reading and writing on the physical device, which is a slow and costly process. All that the cache mechanism does is to keep a copy of the file on which we are working in the RAM (much faster), so that the process is transparent for the user (the copy to disk is done following a policy implemented in the kernel). The only problem with this handling is that if there is a cut in the power supply and we have not shut the system down properly, some files may not have been saved on the physical disk and we may have an **inconsistency** in the file system. The `fsck` program will check and fix a file system that has been left in this state. Although we can execute it whenever we wish, generally the operating system itself will run it when the boot process detects that the system did not shut down properly (before switching off the computer, we must run the `shutdown` command, which takes care of launching all the processes needed for the programs to end, to unmount the file system etc.). In this regard, the ext3 file system is more efficient than its predecessor because the **journaling** allows it to recover more information on lost files and faster.

Naturally, if the files we are dealing with on our system are highly critical and we cannot, in any circumstances, allow ourselves to lose them, we can also configure the operating system not to use the disk cache system. At all events, it is highly advisable to use this mechanism because it significantly improves the system's performance. If at any point we are interested in synchronising the disk cache information with the physical disk, we can use the `sync` command. Finally, we can also check a partition's physical integrity using the `badblocks` command, which carries out a check on the indicated device to make sure that it has no damaged zones.

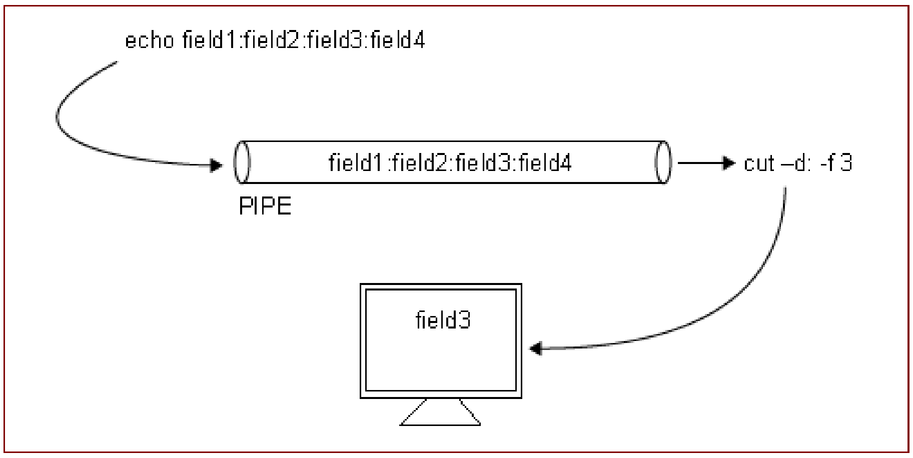
Most of the commands we have described in this subsection require special permissions to be executed, meaning that only the root will be able to use them.

2.6. Operations with commands

2.6.1. Reroutings

Once we have learned to use some of the system's commands, it is likely that we will want to use them simultaneously in some cases so as to speed up the actions we wish to carry out. A very interesting operation consists of being able to take a command's output to serve as another's input and process it appropriately. The operating system uses a **pipes** mechanism, which allows us to reroute the outputs of any command or program to wherever we want. It works very simply: all we have to do is place the "|" character between the commands, so that the first one's output serves as the input for the second. Let's look at it with an example: when we write the `echo field1:field2:field3:field4` command, all we would achieve would be a screen displaying "field1:field2:field3:field4". If from this output we only wanted to take "field3", we could reroute it with a pipe towards the `cut` command, for it to select just the field we are interested in as follows: `echo field1: field 2: field 3: field 4 | cut -d: -f 3`. In the following diagram, we can visualise this example graphically:

Figure 2.3



Of course, we can connect as many pipes as we need in order to carry out more practical tasks than the one we have just seen. Another type of very practical rerouting is related to files. This type of rerouting allows us to take all the output of a command or program and to save it in a file using the character ">", as we did with the "|". For example, if we wish to save in a new file everything we write until pressing Ctrl+C, we could use the following: `cat > test.txt`. With ">>" we can do exactly the same, but instead of always creating the new file, if the file already exists, it would add information to the end of it. With "<" the rerouting is made in the opposite direction, directing the content of the file that we specify towards the indicated command or program.

A very interesting aspect that we ought to know about UNIX type systems is that they separate the normal output of a program from the errors output. Although by default both outputs are directed at the console where the program was executed, we can manipulate them to direct them wherever we want. To see this in practical terms, we will try to delete a file that doesn't exist using the following instruction: `rm file > results`. Although we are rerouting the command's output to the results file, the screen will display an error message telling us that the file has not been found. This is because by default, reroutings only accept the program's standard output and not the error output, which by default is also displayed on screen. In order to reroute the error output, we should indicate, in front of the character ">", the number "2", which is the error output ("1" is the normal output). Thus, by executing `rm file > results` we would manage to reroute the output to the results file. We can also save the normal output and error output in two different files: `rm file 1> results 2> errors`. If on the other hand we wanted all outputs to be directed towards the same file, we could use >&. Furthermore, with the ampersand character (&) we could direct outputs of one type towards other types; for example, if we wish to direct the errors output to the normal output, we could specify this as follows: `2>&1`.

It is important to remember that the order of reroutings is important: they are always executed from left to right.

2.6.2. Specific bash commands

Although some of the commands that we have looked at are specific to bash, this commands interpreter has other commands that can help us carry out many more interesting operations. A very useful mechanism is to execute the processes in what is known as **background** mode. This mode simply tells us that the process is being executed, but that the shell is returning the command line so that we can continue to execute other programs. To tell the bash program this, we need to type the character "&" after the command or program that we are about to execute. Once the process has been launched in background mode, a line is displayed telling us the job number and PID of the launched process.

With the `jobs` command we can see what processes are launched in background mode (by passing the `-l` parameter we can also see its PID). If we wish to pass one of these processes into the foreground mode (as if we had launched it from the command line without the "&" character), we can use the `fg` command indicating the process PID. We also have `bg`, which sends a particular process into background mode for us. This is useful, for example, when we are running a program in foreground mode and we pause it by pressing Ctrl+Z. If then we execute `bg` giving the PID, the process will continue to run in background mode. As we have seen in preceding subsections, processes also have a parent child hierarchy. When we run a program in background mode we are not interfering with this hierarchy, meaning that if we sign out of the session, all of these processes will finish because the parent (the commands interpreter from which we have launched them) will no longer be running. If we wish to unlink a process from its parent, we can use `disown`.

Another useful bash mechanism is the commands history. When we use the system we usually have to repeat many instructions that we have written already. Using the Up and Down keys we can see all the commands we have been using and we can repeat one by pressing "Return". We can also use `history`, which will display all executed commands on screen, numbered in order of appearance. By writing `!NUM` the one corresponding to that history will be executed. We can also write `!` followed by the initial letters of a previously executed program and the program will search for the most recent one in order to execute it.

Bash

The bash command interpreter provides us with endless tools for regulating any aspect of the command interpreter. In its extensive manual we can find the documentation required to learn how to manipulate them correctly.

The bash command interpreter also has fast access keys or shortcuts that allow us to execute certain actions without even having to write them. Some of the most common ones are:

- Tab: we don't need to write the name of a file, directory or command in full. If we write the first characters and then press the tab key, it will write the rest for us. If there is more than one coincidence it will display the different possibilities.
- Ctrl+L: clears the screen (like the `clear` command).
- Shift+PgUp: shows half the previous screen.
- Shift+PgDn: shows half the next screen.
- Ctrl+W: eliminates the last written word.
- Ctrl+T: interchanges the order of the last characters.
- Ctrl+U: deletes all characters in front of the cursor.
- Ctrl+D: exits the command interpreter (equivalent to executing a `logout` command).
- `ulimit` this is a command that allows us to configure some of the internal aspects related to the bash. For example, it allows us to specify the amount of memory that the command interpreter can use, the maximum number of files that can be opened etc. This command can help us to somewhat restrict the actions that our system's users can carry out (in the case of administrating servers with lots of users).

2.6.3. Shell scripts with bash

The shell scripts are files where we write a series of commands (any of the ones we have seen in this section) to be executed. Although their syntax may become quite complex and we should go into programming aspects in order to understand it clearly, in this subsection we summarise some of their essential characteristics so that we can understand them and use them at least (if we want to go into greater depth we can go to the bash manual). The first line of the shell script should specify the command interpreter that is used:

`fc` command

The `fc` command allows us, like the shell scripts, to write a series of commands to be executed but without having to save the file.

```
#!/bin/bash
```

After this line we can already start writing the commands we want to execute, one on each line. As in any programming language, we can use variables, conditional structures and loops. In order to declare a variable we will use the following syntax:

```
variableName=content
```

If the content is a chain of characters, we will need to put it between quotation marks, if it is a number, we do not have to put anything and if we want to save a command output in the variable, we will need to put it between characters. To refer to the variable's content in other instructions, we must always place the "$" character in front of the name. For conditional instructions we can use the following structures:

```
if condition; then instructions else instructions fi
```

where*`condition`* can refer to a file, perform an arithmetical comparison operation (between "(( ))" characters) etc. The `test` command is especially useful in that it allows us to compare files, directories etc., and returns a Boolean. Thus, for example, if we wanted to carry out one action or another depending on the existence of a particular file, we could use the following structure:

```
if test -f /etc/inittab; then echo "The inittab file exists." else echo "The inittab file
does NOT exist." fi
```

Another conditional structure is the selection one:

```
case word in case1) instructions;; case2) instructions;; *) instructions esac
```

In this structure we compare `word` with `case1`, `case2` etc., until finding the one that matches in which case the corresponding actions will be executed. If none is found, it would pass onto section `*)`, which is optional. This structure can come in very handy, for example, when we want a particular script to carry out one action or another according to the parameter that we establish. We can refer to the parameters using `$1` for the first, `$2` for the second and so on consecutively. For loops we can use any of the following structures:

```
#LOOP TYPE FOR for i in list; do instructions where #LOOP TYPE WHILE while condition;
do instructions done
```

Of course, before we can execute a shell script we should give the execution permission to the corresponding file (`chmod 750 fileName` command).

Comments in shell scripts

To write comments in the shell scripts we can use the "#" character followed by the comment we wish. This will be valid until the end of the line.

3. Knoppix workshop

3.1. Introduction

This workshop aims to be your first experience with a UNIX environment. For this reason, it is carried out step by step, but of course leaving the door open for the curious among you to investigate of your own accord.

The main objective of this workshop is to become familiar with the GNU/Linux operating system and to see that everything that we are used to doing with other operating systems, can be done exactly the same with the GNU/Linux operating system. In this workshop, we are also going to start working with the command line, becoming familiar with it and losing our fear of anything that is not a graphic environment. So, let's go ahead, it's time to put into practice everything that has been explained theoretically up until now.

This workshop can be carried out on any computer, since the risk of damaging the information we could have is minimal. We have chosen this distribution because it does not require prior knowledge of the operating system to boot it, and because, once we stop the system, it leaves no trace.

KNOPPIX by default mounts all the hard disk partitions on the system but with read only permissions; therefore, we can neither write nor execute anything, unless we force it to by changing the permissions. Obviously, if we have a different GNU/Linux operating system, we can use it to follow the workshop.

As this is a bootable distribution on CD-ROM or DVD-ROM, we can work without leaving any trace on the computer where it has been executed when the stop process has finished; it means that even though it is based on Debian, the file system does not comply with the Debian Policy in this regard. Nonetheless, these differences will not affect the workshop, and everything we learn will be valid for later ones. Furthermore, its good to become used to working with different distributions from the beginning and to learn to distinguish between what is common to all UNIX-based systems and what is particular to each distribution.

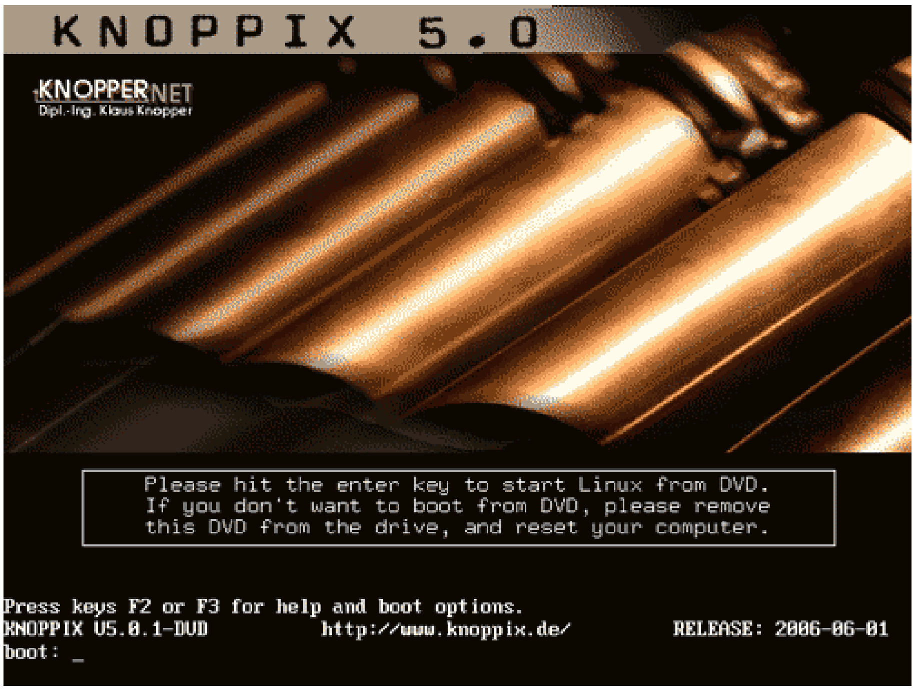
Before we start, just one piece of advice: let's go ahead with our own initiatives, try to answer our questions ourselves, consult the help provided by the man command, carry out tests, if they fail, analyse why, try again, over and over until we achieve the desired results; this is how to learn UNIX, fearlessly, learning from our mistakes.

3.2. System boot

First of all we need to make sure that our computer will boot from the CD-ROM or DVD-ROM reader. To do this we will enter the BIOS (Basic Input Output System), usually by pressing the Del key during the RAM memory check process, and check that the CD-ROM or DVD-ROM is configured as the first boot device; if it is, we can already exit the BIOS without having to save anything and put the KNOPPIX CD-ROM or DVD-ROM into the reader. If it is not, we will make the relevant changes and save them before exiting the BIOS.

After restarting the computer, following several seconds the KNOPPIX boot screen will appear with the following lines at the bottom:

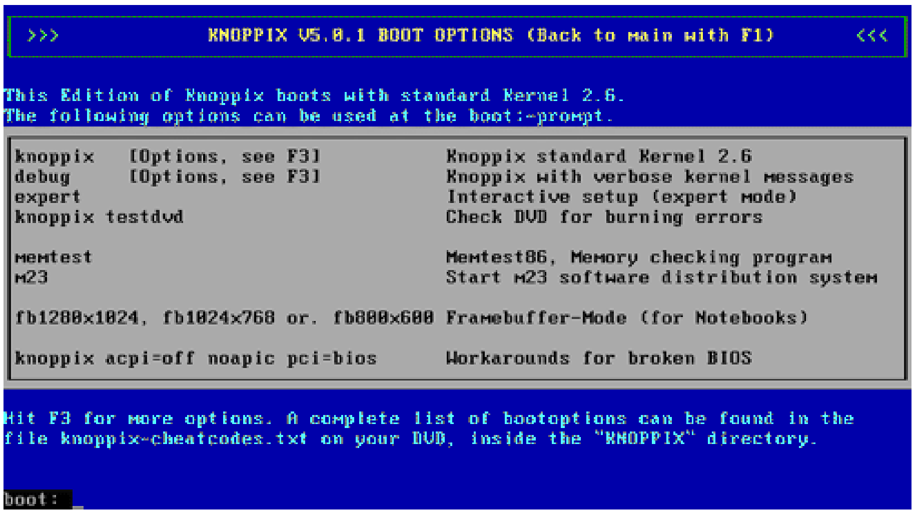
Figure 3.1



```
Please hit the enter key to start Linux from DVD.
If you don't want to boot from DVD, please remove
this DVD from the drive, and reset your computer.

Press keys F2 or F3 for help and boot options.
KNOPPIX V5.0.1-DVD          http://www.knoppix.de/          RELEASE: 2006-06-01
boot: _
```

We can press the F2 or F3 key to enter the screen showing the options accepted by KNOPPIX for booting, next we can see the information presented by pressing the F2 key:

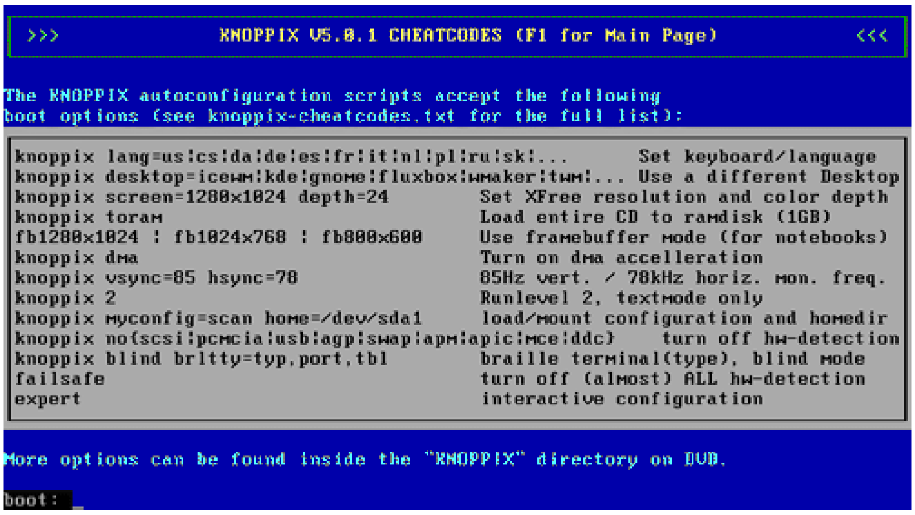
Figure 3.2



```
 >>>          KNOPPIX V5.0.1 BOOT OPTIONS (Back to main with F1)         <<<

This Edition of Knoppix boots with standard Kernel 2.6.
The following options can be used at the boot:-prompt.

knoppix   [Options, see F3]             Knoppix standard Kernel 2.6
debug     [Options, see F3]             Knoppix with verbose kernel messages
expert                                  Interactive setup (expert mode)
knoppix testdvd                         Check DVD for burning errors

memtest                                 Memtest86, Memory checking program
m23                                     Start m23 software distribution system

fb1280x1024, fb1024x768 or. fb800x600 Framebuffer-Mode (for Notebooks)

knoppix acpi=off noapic pci=bios        Workarounds for broken BIOS

Hit F3 for more options. A complete list of bootoptions can be found in the
file knoppix-cheatcodes.txt on your DVD, inside the "KNOPPIX" directory.


boot: _
```

and this is the information that we are shown when we press the F3 key:

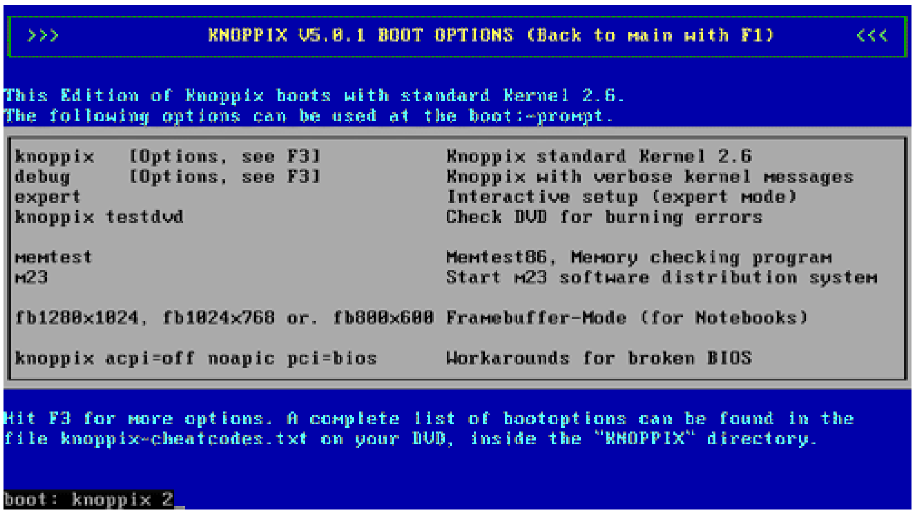
Figure 3.3



```
 >>>            KNOPPIX V5.0.1 CHEATCODES (F1 for Main Page)             <<<

The KNOPPIX autoconfiguration scripts accept the following
boot options (see knoppix-cheatcodes.txt for the full list):

knoppix lang=us|cs|da|de|es|fr|it|nl|pl|ru|sk|...       Set keyboard/language
knoppix desktop=icewm|kde|gnome|fluxbox|wmaker|twm|... Use a different Desktop
knoppix screen=1280x1024 depth=24          Set XFree resolution and color depth
knoppix toram                              Load entire CD to ramdisk (1GB)
fb1280x1024 | fb1024x768 | fb800x600       Use framebuffer mode (for notebooks)
knoppix dma                                Turn on dma accelleration
knoppix vsync=85 hsync=70                  85Hz vert. / 70kHz horiz. mon. freq.
knoppix 2                                  Runlevel 2, textmode only
knoppix myconfig=scan home=/dev/sda1       load/mount configuration and homedir
knoppix no{scsi|pcmcia|usb|agp|swap|apm|apic|mce|ddc}     turn off hw-detection
knoppix blind brltty=typ,port,tbl          braille terminal(type), blind mode
failsafe                                   turn off (almost) ALL hw-detection
expert                                     interactive configuration

More options can be found inside the "KNOPPIX" directory on DVD.

boot: _
```

For example, we could boot with the Spanish keyboard in the Spanish language for interaction with GNOME as windows manager and activate the scroll wheel of the mouse; to do so, we could just type on the command line (`boot:`) as follows `knoppix lang=es gnome wheelmouse`. But we won't; the previous example was just to show the capabilities of KNOPPIX. After inserting a new formatted disk in the disk drive on which we will take our first steps in GNU/Linux, thus safeguarding all the information we may have on our hard disks, we will simply write `knoppix 2` and press Enter to boot the system in text mode.

Figure 3.4



```
>>>            KNOPPIX V5.0.1 BOOT OPTIONS (Back to main with F1)          <<<

This Edition of Knoppix boots with standard Kernel 2.6.
The following options can be used at the boot:-prompt.

knoppix   [Options, see F3]           Knoppix standard Kernel 2.6
debug     [Options, see F3]           Knoppix with verbose kernel messages
expert                                Interactive setup (expert mode)
knoppix testdvd                       Check DVD for burning errors

memtest                               Memtest86, Memory checking program
m23                                   Start m23 software distribution system

fb1280x1024, fb1024x768 or. fb800x600 Framebuffer-Mode (for Notebooks)

knoppix acpi=off noapic pci=bios      Workarounds for broken BIOS

Hit F3 for more options. A complete list of bootoptions can be found in the
file knoppix-cheatcodes.txt on your DVD, inside the "KNOPPIX" directory.

boot: knoppix 2_
```

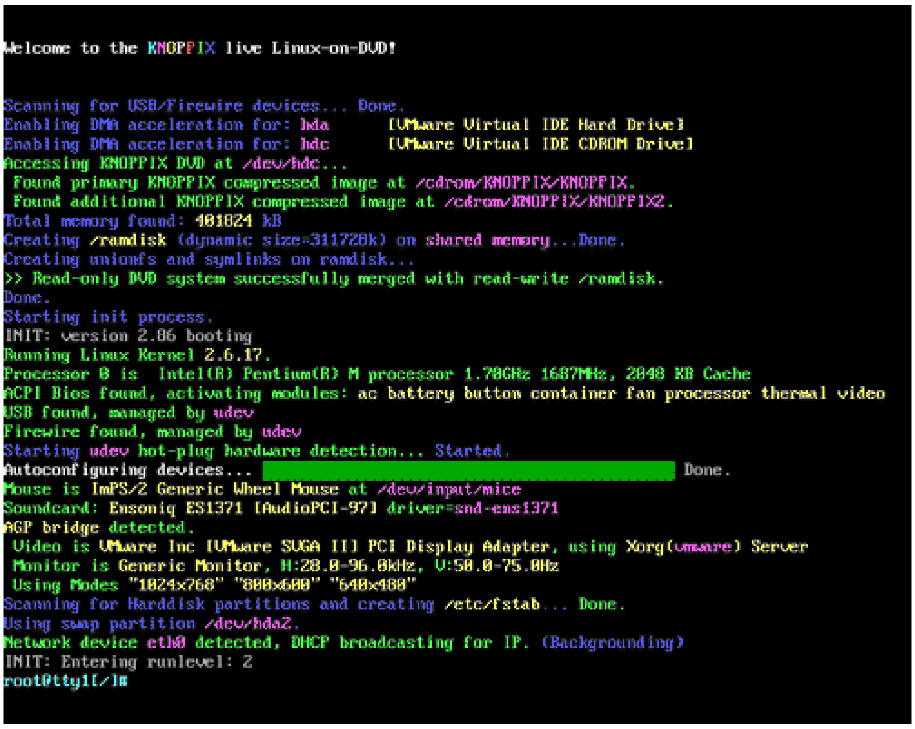
Initially, the keyboard is configured for the United States (us), so some characters differ from the Spanish keyboard; we can fix this immediately, but even so, it is interesting to know where certain US characters are on the Spanish keyboard:

- "=" is on the key ¿.
- "/" on key o on the number pad.
- "-" on key? or on the number pad.

After pressing "Intro", KNOPPIX will start to load the operating system, showing on screen the results of some of the tests that it carries out during the auto-configuration.

Once this process is over, we will have the command line:

Figure 3.5



```
Welcome to the KNOPPIX live Linux-on-DVD!

Scanning for USB/Firewire devices... Done.
Enabling DMA acceleration for: hda       [VMware Virtual IDE Hard Drive]
Enabling DMA acceleration for: hdc       [VMware Virtual IDE CDROM Drive]
Accessing KNOPPIX DVD at /dev/hdc...
 Found primary KNOPPIX compressed image at /cdrom/KNOPPIX/KNOPPIX.
 Found additional KNOPPIX compressed image at /cdrom/KNOPPIX/KNOPPIX2.
Total memory found: 401824 kB
Creating /ramdisk (dynamic size=311720k) on shared memory...Done.
Creating unionfs and symlinks on ramdisk...
>> Read-only DVD system successfully merged with read-write /ramdisk.
Done.
Starting init process.
INIT: version 2.86 booting
Running Linux Kernel 2.6.17.
Processor 0 is  Intel(R) Pentium(R) M processor 1.70GHz 1687MHz, 2048 KB Cache
ACPI Bios found, activating modules: ac battery button container fan processor thermal video
USB found, managed by udev
Firewire found, managed by udev
Starting udev hot-plug hardware detection... Started.
Autoconfiguring devices...                                            Done.
Mouse is ImPS/2 Generic Wheel Mouse at /dev/input/mice
Soundcard: Ensoniq ES1371 [AudioPCI-97] driver=snd-ens1371
AGP bridge detected.
 Video is VMware Inc [VMware SVGA II] PCI Display Adapter, using Xorg(vmware) Server
 Monitor is Generic Monitor, H:28.0-96.0kHz, V:50.0-75.0Hz
 Using Modes "1024x768" "800x600" "640x480"
Scanning for Harddisk partitions and creating /etc/fstab... Done.
Using swap partition /dev/hda2.
Network device eth0 detected, DHCP broadcasting for IP. (Backgrounding)
INIT: Entering runlevel: 2
root@tty1[/]#
```

We are now inside the system. In this case, no user or password has been necessary, we have entered directly as root, and we know this because the prompt ends with the character "#". For any other user than the root, the last character would be "$".

3.3. Stop system

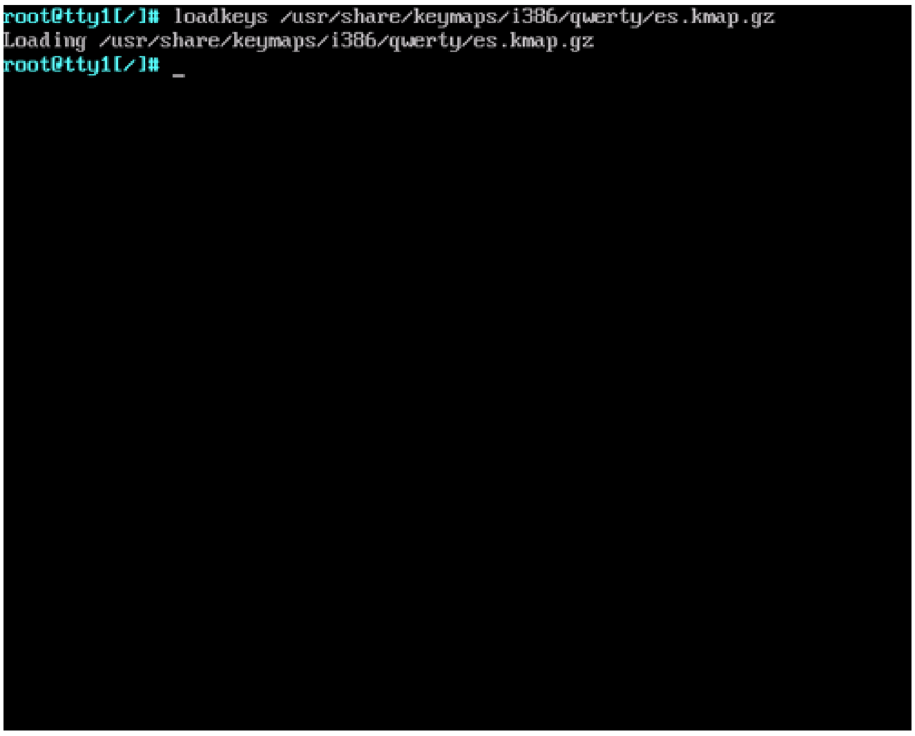
Once inside the system, the first thing that we should know, as we have already mentioned, is how to stop it. Let's remember once more that we cannot stop the computer without previously stopping the operating system (GNU/Linux). There are several ways of doing this. The most common are: pressing the key combination Ctrl+Alt+Del or using the `halt` command (there are many more, through the `reboot` command, changing the runlevel from 0 to 6 etc.). Once we have given the system the order to stop, it will start executing the relevant stop instructions (demounting devices, stopping processes etc.), and at the end of this entire process KNOPPIX will eject the CD-ROM or DVD-ROM and ask us to press Enter in order to stop the computer.

3.4. Keyboard configuration

Once we know how to start and stop the system, we will boot it again following the same steps as before. Once we are inside the system, the first thing we will have to do is to configure the correct keyboard layout. There are several ways of doing this:

- manually and if we boot in console mode (`knoppix 2`):

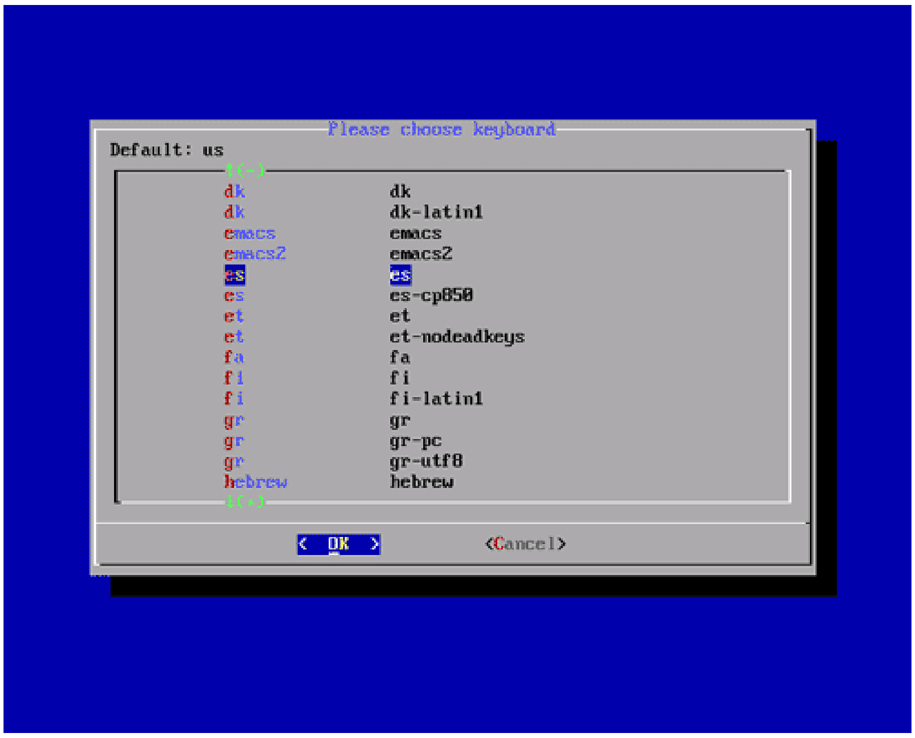
Figure 3.6



```
root@tty1[/]# loadkeys /usr/share/keymaps/i386/qwerty/es.kmap.gz
Loading /usr/share/keymaps/i386/qwerty/es.kmap.gz
root@tty1[/]# _
```

- graphically with the `kbdconfig` command selecting the option `es es`.

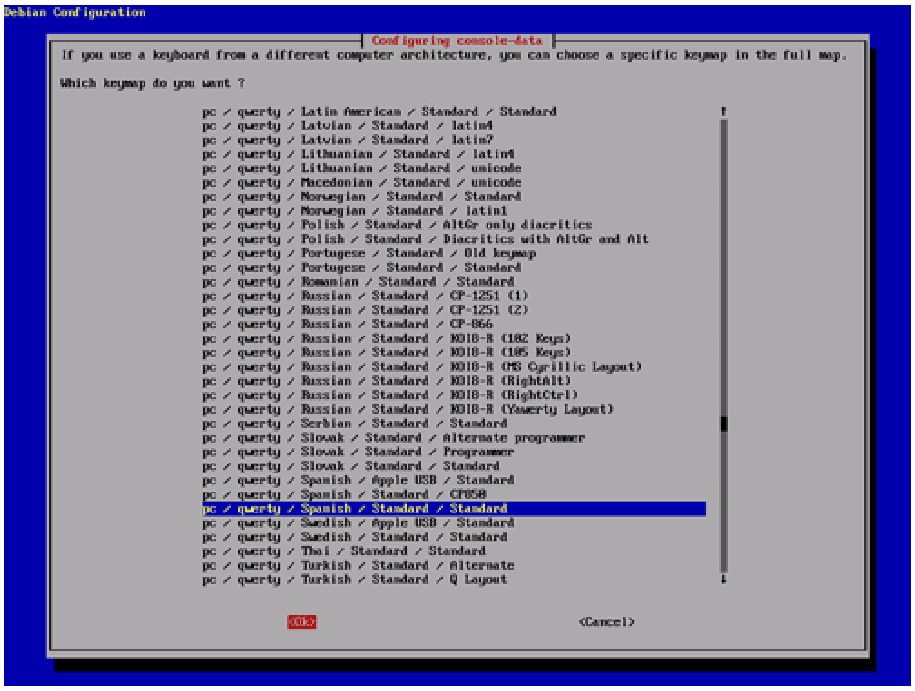
Figure 3.7



```
                    Please choose keyboard
 Default: us
            ↑(-)
            dk               dk
            dk               dk-latin1
            emacs            emacs
            emacs2           emacs2
            es               es
            es               es-cp850
            et               et
            et               et-nodeadkeys
            fa               fa
            fi               fi
            fi               fi-latin1
            gr               gr
            gr               gr-pc
            gr               gr-utf8
            hebrew           hebrew
            ↓(+)

                 <  OK  >          <Cancel>
```

- using the `kbd-config` command and selecting the following options:

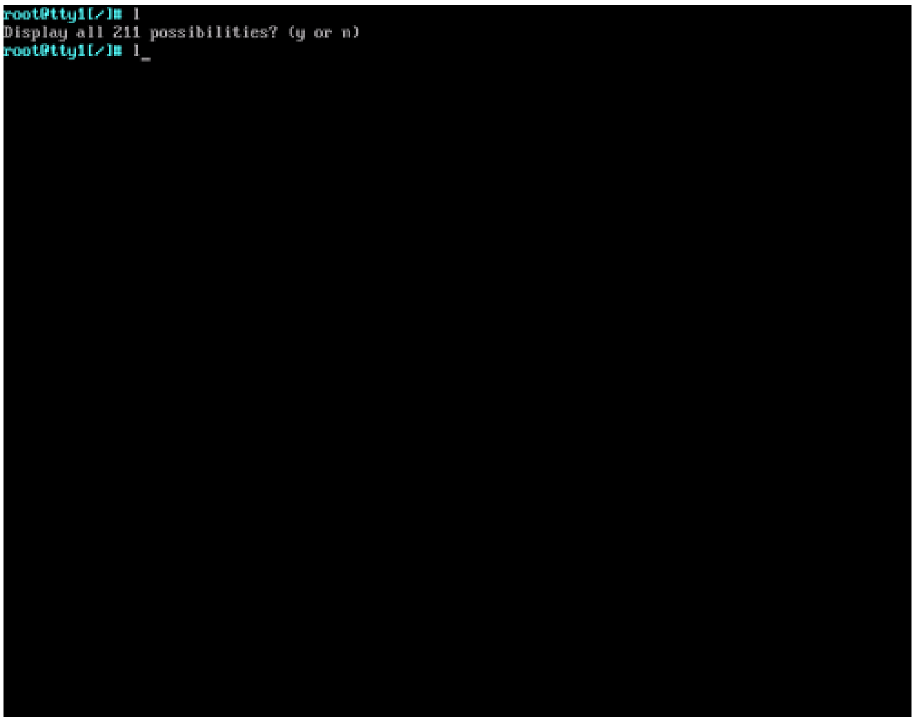
Figure 3.8



```
Debian Configuration

                        ┤ Configuring console-data ├
 If you use a keyboard from a different computer architecture, you can choose a specific keymap in the full map.

 Which keymap do you want ?

                    pc / qwerty / Latin American / Standard / Standard
                    pc / qwerty / Latvian / Standard / latin4
                    pc / qwerty / Latvian / Standard / latin7
                    pc / qwerty / Lithuanian / Standard / latin4
                    pc / qwerty / Lithuanian / Standard / unicode
                    pc / qwerty / Macedonian / Standard / unicode
                    pc / qwerty / Norwegian / Standard / Standard
                    pc / qwerty / Norwegian / Standard / latin1
                    pc / qwerty / Polish / Standard / AltGr only diacritics
                    pc / qwerty / Polish / Standard / Diacritics with AltGr and Alt
                    pc / qwerty / Portugese / Standard / Old keymap
                    pc / qwerty / Portugese / Standard / Standard
                    pc / qwerty / Romanian / Standard / Standard
                    pc / qwerty / Russian / Standard / CP-1251 (1)
                    pc / qwerty / Russian / Standard / CP-1251 (2)
                    pc / qwerty / Russian / Standard / CP-866
                    pc / qwerty / Russian / Standard / KOI8-R (102 Keys)
                    pc / qwerty / Russian / Standard / KOI8-R (105 Keys)
                    pc / qwerty / Russian / Standard / KOI8-R (MS Cyrillic Layout)
                    pc / qwerty / Russian / Standard / KOI8-R (RightAlt)
                    pc / qwerty / Russian / Standard / KOI8-R (RightCtrl)
                    pc / qwerty / Russian / Standard / KOI8-R (Yawerty Layout)
                    pc / qwerty / Serbian / Standard / Standard
                    pc / qwerty / Slovak / Standard / Alternate programmer
                    pc / qwerty / Slovak / Standard / Programmer
                    pc / qwerty / Slovak / Standard / Standard
                    pc / qwerty / Spanish / Apple USB / Standard
                    pc / qwerty / Spanish / Standard / CP850
                    pc / qwerty / Spanish / Standard / Standard
                    pc / qwerty / Swedish / Apple USB / Standard
                    pc / qwerty / Swedish / Standard / Standard
                    pc / qwerty / Thai / Standard / Standard
                    pc / qwerty / Turkish / Standard / Alternate
                    pc / qwerty / Turkish / Standard / Q Layout

                 <Ok>                                      <Cancel>
```

We have an advantage when it comes to writing on the terminals: it is the autocomplete option, we press "Tab" once, and if there is just one possible choice, it autocompletes; or we press it twice in a row, and if there is more than one option we are shown the different choices. We can test it using the `loadkeys` command. We type just an "l" and press the Tab key once, the system makes a beep, and then we press the key a second time to get the following:

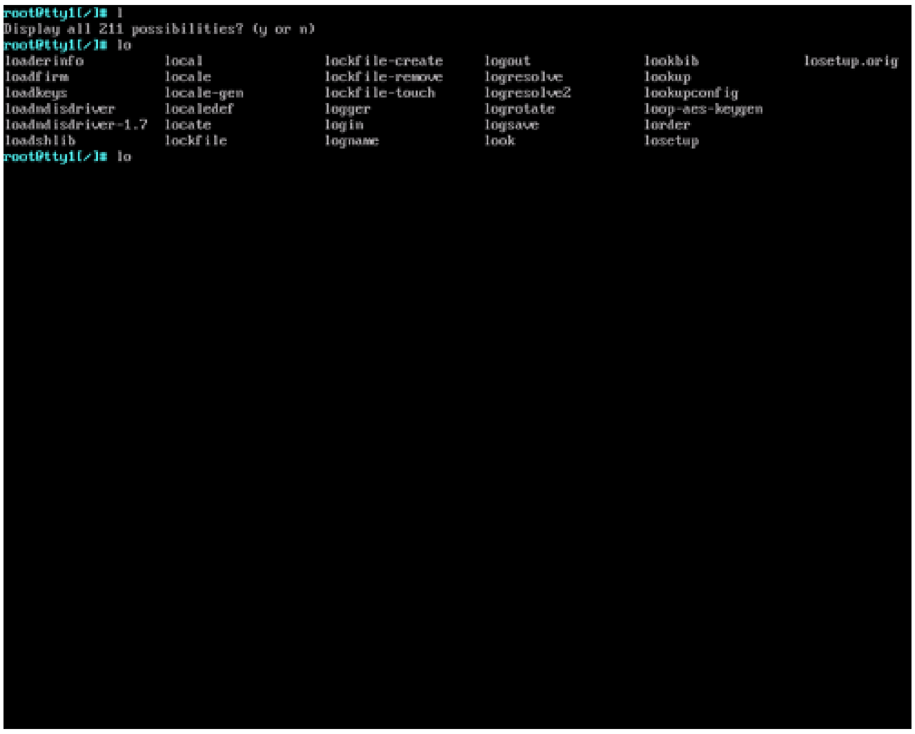
Figure 3.9



```
root@tty1[/]# l
Display all 211 possibilities? (y or n)
root@tty1[/]# l_
```

In this case we do not want to be shown the 211 possibilities. Therefore, we press "n" and add the letter "o", and repeat the previous operation of pressing the Tab key twice. Now we get a different result:

Figure 3.10



```
root@tty1[/]# l
Display all 211 possibilities? (y or n)
root@tty1[/]# lo
loaderinfo           local          lockfile-create   logout        lookbib           losetup.orig
loadfirm             locale         lockfile-remove   logresolve    lookup
loadkeys             locale-gen     lockfile-touch    logresolve2   lookupconfig
loadndisdriver       localedef      logger            logrotate     loop-aes-keygen
loadndisdriver-1.7   locate         login             logsave       lorder
loadshlib            lockfile       logname           look          losetup
root@tty1[/]# lo
```

All we need to do is add "adk" and press the Tab key again for the autocomplete to function giving:

```
root@tty1[/]# loadkeys
```

The autocomplete feature can also be used to refer to files and directories. We just have to type "loadkeys /u" and press Tab to get:

```
root@tty1[/]# loadkeys /usr
```

and press Tab again to get:

```
root@tty1[/]# loadkeys /usr/
```

Autocomplete is a very useful tool not just because it saves typing, but also because it helps to make sure that we have written the commands correctly, as well as directories and files. The lack of this tool is noticeable for those accustomed to using it.

3.5. System inspection

Once we have configured the keyboard, we will proceed to inspect the system a bit. Firstly, where in the directories structure are we at present? We can find out through the pwd command:

```
root@tty1[/]# pwd /
```

We are in the system root, and we know from the command return pwd, but we could also have known by reading the information given by the prompt: root@tty1[/]#. We already learned about the meaning of the "#" character, now let's learn about the rest of the information that it tells us.

"root", in this field the username appears in this case, root, slightly redundant information, since the "#" character is already telling us this. But for other users it is interesting, since all of them end with the "$" symbol, and this way we can know whether it is Isabel, Daniel or another user.

Following the "@" character, which simply serves to separate fields, we have "tty1", this field is telling us what terminal we are on, and this is because, as we have already mentioned, GNU/Linux is multi-user and multi-process; therefore, it is not surprising that we can access the system from different terminals. KNOPPIX in particular offers four terminals by default, which we can access through the following key combinations: Alt+F1 (tty1), Alt+F2 (tty2), Alt+F3 (tty3) and Alt+F4 (tty4); this is extremely useful, since it allows us to have up to four work sessions meaning that for example we can be testing a command on one, have the man for the command in question on another in order to read about its options, while on another we may have an editor open in order to take note of what we are doing (many distributions offer six terminals in text mode and reserve the seventh for the graphics terminal; in KNOPPIX the graphics terminal, if it has been enabled, which is not our case, would be the fifth terminal). So, we try the Alt+F2 combination, observing that "tty2" now appears in the field that we are studying, and execute a command, for example `man man`, to read the man or help of the `man application` . For the moment we will leave it there and return to tty1, checking that everything is as we had left it.

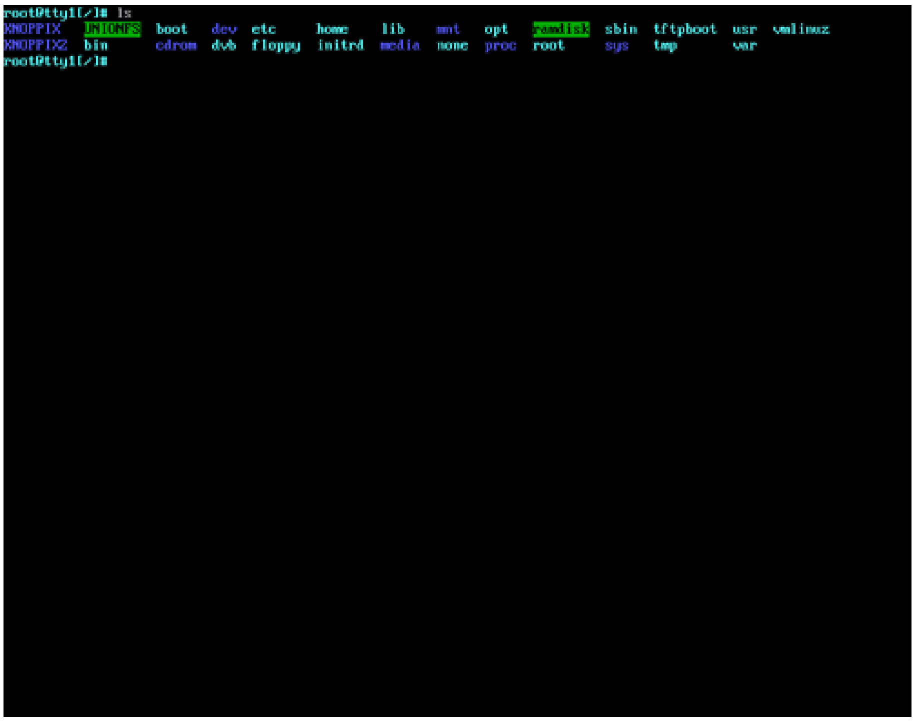
Finally, and continuing with the interpretation of the prompt, in front of the field "#" we already know, we find "[/]": this tells us the current directory, the root in this case, as we were told by the pwd command. The difference stems from the fact that the prompt only gives us the current directory, whereas the pwd command gives us the full path.

For example, if we were in /home/sofia we would have:

```
root@tty1[sofia]# pwd /home/sofia
```

Now we can list the content of the directory that we are in. To do so, we will use the `ls`, first without passing any option onto it:

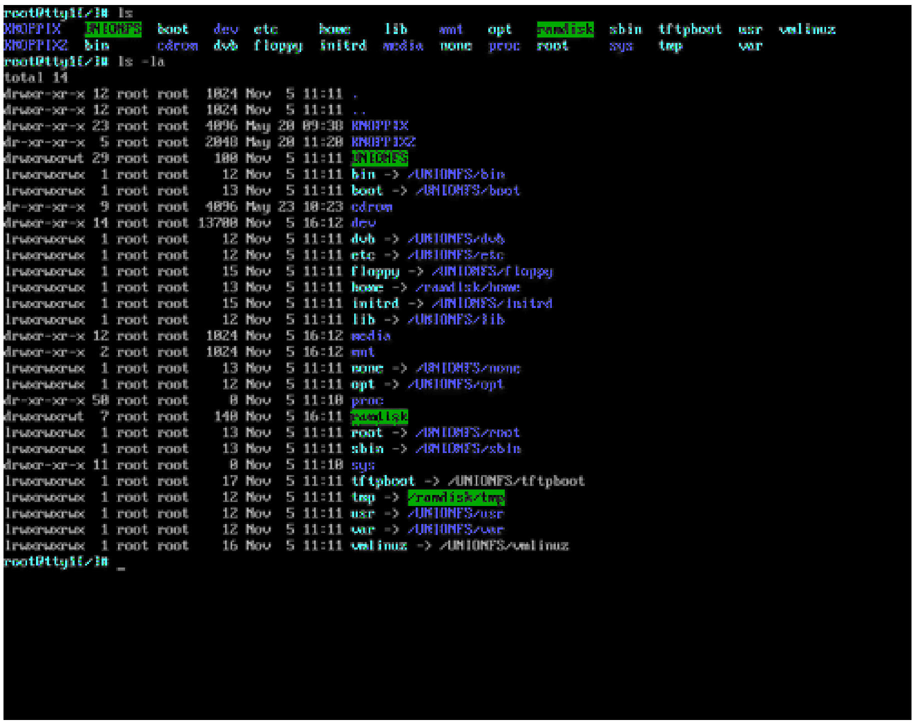
Figure 3.11



```
root@tty1[/]# ls
KNOPPIX   UNIONFS  boot   dev  etc     home    lib    mnt   opt   ramdisk  sbin  tftpboot  usr  vmlinuz
KNOPPIX2  bin      cdrom  dvb  floppy  initrd  media  none  proc  root     sys   tmp       var
root@tty1[/]#
```

and now we will pass two options (normally the options are preceded by a "-", although there are commands that do not require this), to tell it to return more detailed information. So, once we have typed the "-" we will write the options `la` to obtain this information:

Figure 3.12

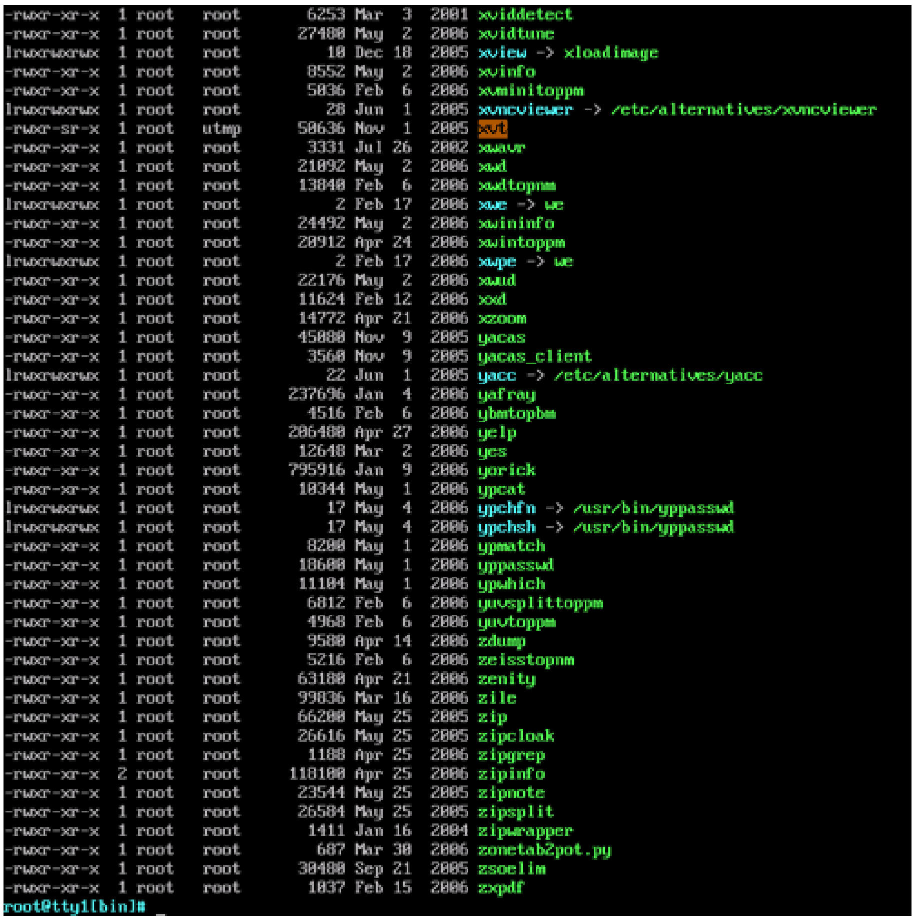


```
root@tty1[/]# ls
KNOPPIX   UNIONFS  boot   dev  etc     home    lib    mnt   opt   ramdisk  sbin  tftpboot  usr  vmlinuz
KNOPPIX2  bin      cdrom  dvb  floppy  initrd  media  none  proc  root     sys   tmp       var
root@tty1[/]# ls -la
total 14
drwxr-xr-x 12 root root  1024 Nov  5 11:11 .
drwxr-xr-x 12 root root  1024 Nov  5 11:11 ..
drwxr-xr-x 23 root root  4096 May 20 09:30 KNOPPIX
dr-xr-xr-x  5 root root  2048 May 20 11:20 KNOPPIX2
drwxrwxrwt 29 root root   100 Nov  5 11:11 UNIONFS
lrwxrwxrwx  1 root root    12 Nov  5 11:11 bin -> /UNIONFS/bin
lrwxrwxrwx  1 root root    13 Nov  5 11:11 boot -> /UNIONFS/boot
dr-xr-xr-x  9 root root  4096 May 23 10:23 cdrom
drwxr-xr-x 14 root root 13700 Nov  5 16:12 dev
lrwxrwxrwx  1 root root    12 Nov  5 11:11 dvb -> /UNIONFS/dvb
lrwxrwxrwx  1 root root    12 Nov  5 11:11 etc -> /UNIONFS/etc
lrwxrwxrwx  1 root root    15 Nov  5 11:11 floppy -> /UNIONFS/floppy
lrwxrwxrwx  1 root root    13 Nov  5 11:11 home -> /ramdisk/home
lrwxrwxrwx  1 root root    15 Nov  5 11:11 initrd -> /UNIONFS/initrd
lrwxrwxrwx  1 root root    12 Nov  5 11:11 lib -> /UNIONFS/lib
drwxr-xr-x 12 root root  1024 Nov  5 16:12 media
drwxr-xr-x  2 root root  1024 Nov  5 16:12 mnt
lrwxrwxrwx  1 root root    13 Nov  5 11:11 none -> /UNIONFS/none
lrwxrwxrwx  1 root root    12 Nov  5 11:11 opt -> /UNIONFS/opt
dr-xr-xr-x 50 root root     0 Nov  5 11:10 proc
drwxrwxrwt  7 root root   140 Nov  5 16:11 ramdisk
lrwxrwxrwx  1 root root    13 Nov  5 11:11 root -> /UNIONFS/root
lrwxrwxrwx  1 root root    13 Nov  5 11:11 sbin -> /UNIONFS/sbin
drwxr-xr-x 11 root root     0 Nov  5 11:10 sys
lrwxrwxrwx  1 root root    17 Nov  5 11:11 tftpboot -> /UNIONFS/tftpboot
lrwxrwxrwx  1 root root    12 Nov  5 11:11 tmp -> /ramdisk/tmp
lrwxrwxrwx  1 root root    12 Nov  5 11:11 usr -> /UNIONFS/usr
lrwxrwxrwx  1 root root    12 Nov  5 11:11 var -> /UNIONFS/var
lrwxrwxrwx  1 root root    16 Nov  5 11:11 vmlinuz -> /UNIONFS/vmlinuz
root@tty1[/]# _
```

Let's look at the different colours in which the results are displayed (we will see these colours on our computer screens as we advance through the Knoppix workshop): navy blue for directories, magenta for symbolic links (whose destination is shown following the combination of characters "->"), green for scripts and executables (although, obviously, we do not expect to find any in the root directory, since as we have seen in UNIX, the order within the filesystem is very rigid) etc.

Let's go into a directory where we are sure to find executables, such as `/usr/bin/` (to do so, let's execute `cd /usr/bin/`). The directory `/usr/` is really a symbolic link over `/KNOPPIX/usr/`; therefore, we could access its content by entering it directly through the `cd` command, or follow the real destination through the same command. However, we will choose the first option since this directory will be a real directory when we are in a UNIX system installed on a hard disk:

Figure 3.13



```
-rwxr-xr-x  1 root   root       6253 Mar  3  2001 xviddetect
-rwxr-xr-x  1 root   root      27400 May  2  2006 xvidtune
lrwxrwxrwx  1 root   root         10 Dec 18  2005 xview -> xloadimage
-rwxr-xr-x  1 root   root       8552 May  2  2006 xvinfo
-rwxr-xr-x  1 root   root       5036 Feb  6  2006 xvminitoppm
lrwxrwxrwx  1 root   root         28 Jun  1  2005 xvncviewer -> /etc/alternatives/xvncviewer
-rwxr-sr-x  1 root   utmp      50636 Nov  1  2005 xvt
-rwxr-xr-x  1 root   root       3331 Jul 26  2002 xwavr
-rwxr-xr-x  1 root   root      21092 May  2  2006 xwd
-rwxr-xr-x  1 root   root      13040 Feb  6  2006 xwdtopnm
lrwxrwxrwx  1 root   root          2 Feb 17  2006 xwe -> we
-rwxr-xr-x  1 root   root      24492 May  2  2006 xwininfo
-rwxr-xr-x  1 root   root      20912 Apr 24  2006 xwintoppm
lrwxrwxrwx  1 root   root          2 Feb 17  2006 xwpe -> we
-rwxr-xr-x  1 root   root      22176 May  2  2006 xwud
-rwxr-xr-x  1 root   root      11624 Feb 12  2006 xxd
-rwxr-xr-x  1 root   root      14772 Apr 21  2006 xzoom
-rwxr-xr-x  1 root   root      45080 Nov  9  2005 yacas
-rwxr-xr-x  1 root   root       3560 Nov  9  2005 yacas_client
lrwxrwxrwx  1 root   root         22 Jun  1  2005 yacc -> /etc/alternatives/yacc
-rwxr-xr-x  1 root   root     237696 Jan  4  2006 yafray
-rwxr-xr-x  1 root   root       4516 Feb  6  2006 ybmtopbm
-rwxr-xr-x  1 root   root     206400 Apr 27  2006 yelp
-rwxr-xr-x  1 root   root      12648 Mar  2  2006 yes
-rwxr-xr-x  1 root   root     795916 Jan  9  2006 yorick
-rwxr-xr-x  1 root   root      10344 May  1  2006 ypcat
lrwxrwxrwx  1 root   root         17 May  4  2006 ypchfn -> /usr/bin/yppasswd
lrwxrwxrwx  1 root   root         17 May  4  2006 ypchsh -> /usr/bin/yppasswd
-rwxr-xr-x  1 root   root       8200 May  1  2006 ypmatch
-rwxr-xr-x  1 root   root      18600 May  1  2006 yppasswd
-rwxr-xr-x  1 root   root      11104 May  1  2006 ypwhich
-rwxr-xr-x  1 root   root       6812 Feb  6  2006 yuvsplittoppm
-rwxr-xr-x  1 root   root       4968 Feb  6  2006 yuvtoppm
-rwxr-xr-x  1 root   root       9580 Apr 14  2006 zdump
-rwxr-xr-x  1 root   root       5216 Feb  6  2006 zeisstopnm
-rwxr-xr-x  1 root   root      63100 Apr 21  2006 zenity
-rwxr-xr-x  1 root   root      99036 Mar 16  2006 zile
-rwxr-xr-x  1 root   root      66200 May 25  2005 zip
-rwxr-xr-x  1 root   root      26616 May 25  2005 zipcloak
-rwxr-xr-x  1 root   root       1188 Apr 25  2006 zipgrep
-rwxr-xr-x  2 root   root     118100 Apr 25  2006 zipinfo
-rwxr-xr-x  1 root   root      23544 May 25  2005 zipnote
-rwxr-xr-x  1 root   root      26504 May 25  2005 zipsplit
-rwxr-xr-x  1 root   root       1411 Jan 16  2004 zipwrapper
-rwxr-xr-x  1 root   root        687 Mar 30  2006 zonetab2pot.py
-rwxr-xr-x  1 root   root      30400 Sep 21  2005 zsoelim
-rwxr-xr-x  1 root   root       1037 Feb 15  2006 zxpdf
root@tty1[bin]# _
```

We have probably not been able to see more than the last few lines of what has been shown on screen. We can visualise a few more results by pressing Shift+PgUp to go back through the list and Shift+PgDn to advance. Even so, the terminal buffer has a limit and using this technique we are unlikely to see all of the information returned by the `ls` command. Therefore, we need to resort to other techniques, rerouting output (in this case on a file) or using pipes ("|") and paginators (`less` in this case, which uses the PgUp and PgDn keys to move the content shown on screen, and the Q key to quit). We will use this last one, since we are not interested at all in saving a directory's content list; we will also take the opportunity to practice another utility that saves us typing: pressing the up and down arrow keys we can move through all the command lines that we have passed to the system; thus, to obtain the line we want, `ls -la | less`, all we will have to do is press the arrow up key once and add `| less`.

`less` is a very powerful paginator that KNOPPIX and most distributions use to show the contents of the man (although, like most Linux services, we can configure it to our own taste). Therefore, we can do a `man less` to find out a bit about this command while getting used to this help's morphology at the same time. Obviously, since we are under less, to quit man all we have to do is to press Q.

Note how for all the directory content lists that we have been doing two rather special directories always appear: `.` and `..`. When we create directories, the system automatically creates these as subdirectories within the created directory, and they contain very special information. The `.` directory, which is known as the current directory, refers to the directory itself (so if we execute a `ls -la .` command, we will get the same information as by executing a `ls -la` command). And the `..` directory, which is known as the parent directory, refers to the parent directory of the directory that we are in (executing a `ls -la ..` command will give us the content list of the parent directory or directory immediately above).

Activity

3.1 Look at the man of the `ls` command and understand the functions of the `-a` and `-l parameters`.

Can we remember where we are in the file system? No? Well the `pwd` command will remind us. Let's go back to the root directory. There are basically two ways of doing this and another especially for this case; in any case, they are all based on using the `cd` command, and differ only in the argument that we pass in each case.

The first way is to rise up repeatedly through the parent directories using `cd ..` (note, the system does not understand `cd..` , because it interprets that we are trying to execute a command called "cd..", which does not exist) until we reach the root directory (we will see it in the prompt).

The second is more efficient, since with just one command line we will achieve our purpose, which is to go to the root directory. To do this, we need to pass the root directory as argument to the `cd` command (`cd /`); this second option is much more powerful and allows us to pass directly to directories belonging to different branches, for example, we could have done `cd /etc/rcS.d/` (the last slash is optional) to go to this directory, from `/usr/bin/`.

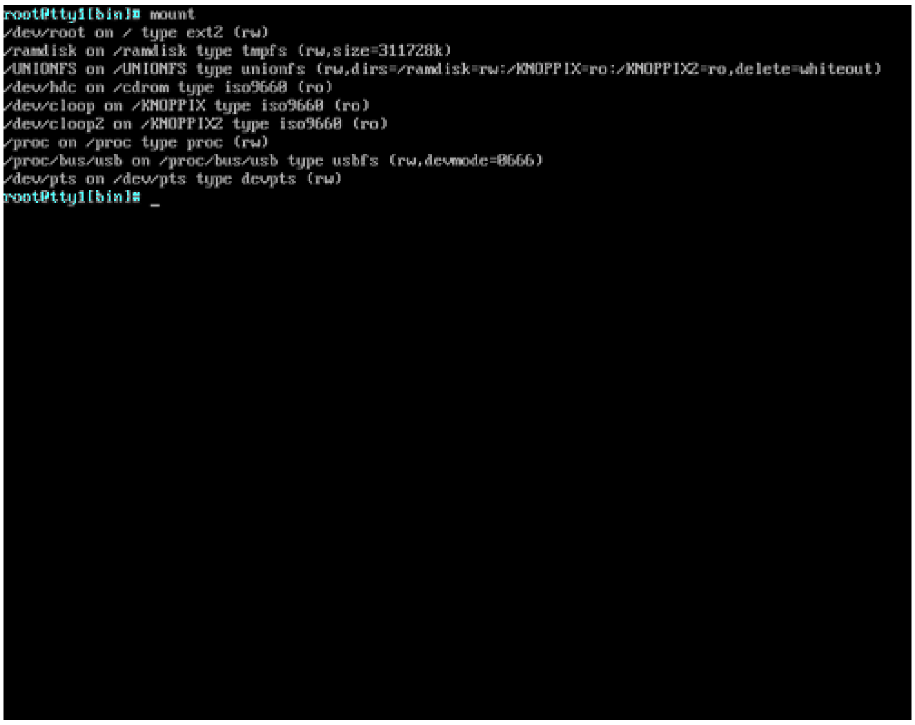
As we have mentioned, for this case (going to the root directory) there is a third option to achieve the same goal with variations: it involves executing the `cd` command without passing any argument, and it would place us directly in the system's root, since in the case of KNOPPIX this is the home directory (an exceptional and somewhat particular case, since in most distributions, and according to the Filesystem Hierarchy Standard, root must have its own home `/root`); a variation of this method is "`cd  `", since " " is the equivalent to the user's home, the root directory in this case. There is yet another option for returning to the root directory, or rather, to return to the directory we've come from, the root, since we had executed `cd /usr/bin`", and it is `cd -`, since in "-" it saves the last directory we were in prior to the current one.

3.6. Handling files and directories

Having examined different options for achieving the same purpose, we are still in the root directory. Now we can learn how to create files and directories, move, copy, delete them etc. We will do this on the disk that we used to boot KNOPPIX, which the system has mounted itself while booting (later we will learn how to mount and unmount devices).

Let's remember that in UNIX before devices can be used they need to be mounted, and that before the support is removed, they need to be unmounted. This last point is extremely important, since otherwise the data's integrity cannot be fully guaranteed. For example, we can attempt to remove the CD-ROM by pressing the eject button before proceeding to work on the disk. Surprise! we can't; we shouldn't think that the device has broken, much less that GNU/Linux has done this. It is simply that the system has mounted this device automatically during the boot, and, therefore, it has become part of the system; it would make no sense to be able to remove the CD-ROM without telling the system beforehand; therefore, the system has taken control of this device and, among other things, has disabled the eject button for the CD-ROM, precisely in order to avoid it being removed without first having been unmounted. This is not the same with the disk drive: for this device the eject button functions mechanically, meaning that the system cannot prevent us from ejecting the disk without telling it first, but this is not advisable, since as we have already said, it could entail the loss of the data that it contains. Therefore, until we learn how to mount and unmount data support devices, the disk must remain in the disk drive from the moment the system boots until it stops, since, along with other operations during the stop process these devices will be unmounted. Let's go to the disk; but, where is it? Where has the system mounted it? To find the answer to these questions we will execute the `mount` command with no arguments or additional options (this is precisely the command used to mount devices, and `umount` to unmount them):

Figure 3.14



```
root@tty1[bin]# mount
/dev/root on / type ext2 (rw)
/ramdisk on /ramdisk type tmpfs (rw,size=311728k)
/UNIONFS on /UNIONFS type unionfs (rw,dirs=/ramdisk=rw:/KNOPPIX=ro:/KNOPPIX2=ro,delete=whiteout)
/dev/hdc on /cdrom type iso9660 (ro)
/dev/cloop on /KNOPPIX type iso9660 (ro)
/dev/cloop2 on /KNOPPIX2 type iso9660 (ro)
/proc on /proc type proc (rw)
/proc/bus/usb on /proc/bus/usb type usbfs (rw,devmode=0666)
/dev/pts on /dev/pts type devpts (rw)
root@tty1[bin]# _
```

This command executed without arguments shows us the devices mounted on the system at the moment of executing it and some additional information regarding them (we can find this same information in the file `/etc/mtab`; consequently, executing `cat /etc/mtab` (to show the content of `mtab`) we would obtain the same results). In the second line returned by the command, we can see that indeed the CD-ROM, `/dev/CDROM`, is mounted on the system and additionally we can see that it is mounted on the directory `/CD-ROM/`. The last returned line answers the questions we made before: the system has mounted the disk, `/dev/fd0/`, on `/mnt/auto/floppy/`. Therefore, we will change to this directory in order to start working on the disk and check that it is effectively empty:

```
root@tty1[/]# cd /mnt/auto/floppy root@tty1[floppy]# ls -la total 8 drwxrwxrwx 3 knoppix
knoppix 7168 Jan 1 1970 . drwxr-xr-x 3 root root 0 Mar 3 19:34 ..
```

We create our first directory, enter it and create a couple of subdirectories:

```
root@tty1[floppy]# mkdir dir00 root@tty1[floppy]# cd dir00/ root@tty1[dir00]# mkdir
subdir00 subdir01 root@tty1[dir00]# ls subdir00 subdir01
```

We enter the first subdirectory and create our first file:

```
root@tty1[dir00]# cd subdir00 root@tty1[subdir00]# touch file00 root@tty1[subdir00]# ls
-la total 1 drwxrwxrwx 2 knoppix knoppix 512 Mar 3 20:21 . drwxrwxrwx 4 knoppix knoppix
```

```
512 Mar 3 20:21 .. -rwxrwxrwx 1 knoppix knoppix 0 Mar 3 20:21 file00
```

In reality `touch` is not for creating empty files although it will if the file does not exist, but rather to change information concerning file times and dates. We can put some content into our first file and check that it has indeed been saved:

```
root@tty1[subdir00]# echo "my fist hello world in Linux" > file00 root@tty1[subdir00]# cat
file00 my first hello world in Linux
```

We add a bit more text:

```
root@tty1[subdir00]# echo "Nice to meet you, we're gonna be good friends" >> file00
root@tty1[subdir00]# cat file00 my first hello world in Linux Nice to meet you, we're
gonna be good friends
```

Let's not forget to use the arrow keys to save typing, or autocomplete. Now we can use a text editor to create our second file. To do this, we will use the vi editor. Let's remember that this editor has two modes: the command mode, that we enter when it boots, and the editing mode. To enter the editing mode we just need to press "i", and to go into command mode we just need to press Esc. If we want to save, we need to go into command mode and write `:w` and `:q` to quit (vi has many more commands, but for the time being these two are enough). It is very interesting to know these basic vi commands, since this editor comes with almost all the basic installation packages of any distribution, and if it fails at any time, it can be useful to modify a file and continue with the installation.

We execute the `vi` command followed by the name that we wish to give our second file, we press I, write what we want, press Esc and `:wq` to save and exit. Using more, another paginator, we can check that everything has worked as expected:

```
root@tty1[subdir00]# vi file01 it seems we're on the right way, mate!!! I agree.
~ ~ ~ :wq
root@tty1[subdir00]# more file01 it seems we're on the right way, mate!!! I agree.
```

USB memory devices. The USB can connect peripherals such as a mouse, keyboard, scanner, digital camera, PDA etc. But we are going to focus on USB memory devices, since these days they are commonly used instead of the older disks. They allow us to store several hundred megabytes of information on a small and easy to carry device: USB memory devices. There are different systems, such as the **flash** system or **memory stick** system.

These systems function perfectly well with the Linux kernel. It is enough to mount them as if they were SCSI disks, normally they will be in `/dev/sda1`, or if there are other SCSI disks, in `/dev/sdb1` or `/dev/sdc1`.

As we have already seen, the `mount` command allows us to mount any file system, and with `umount` we can unmount it. To know how many file systems there are and how they have been mounted, we need to execute the `mount` command without any option. The information is obtained from the `/etc/mtab` file, since this file contains the list of file systems mounted at that time. In fact, it is the `mount` and `umount` programs that keep this list in the `/etc/mtab file.`.

Like the following command:

```
knoppix@0[knoppix]$ mount /dev/root on / type ext2 (rw) /ramdisk on /ramdisk type tmpfs
(rw,size=401312k) /UNIONFS on /UNIONFS type unionfs (rw,dirs=/ramdisk=rw:/KNOPPIX=ro:
/KNOPPIX2=ro,delete=whiteout) /dev/hdc on /cdrom type iso9660 (ro) /dev/cloop on
/KNOPPIX type iso9660 (ro) /dev/cloop2 on /KNOPPIX2 type iso9660 (ro) /UNIONFS/dev/pts
on /UNIONFS/dev/pts type devpts (rw) /proc/bus/usb on /proc/bus/usb type usbfs (rw,devmode=0666)
automount(pid2320) on /mnt/auto type autofs (rw,fd=4,pgrp=2320,minproto=2,maxproto=4)
/UNIONFS/dev/sda1 on /mnt/sda1 type vfat (rw,nosuid,nodev,umask=000,user=knoppix)
```

The file `/etc/fstab` is the one that contains what devices are usually mounted, where and how (what their options are). But when we mount other devices or unmount any device, this is reflected in the file `/etc/mtab`, which is the one that describes the file systems that are mounted at that moment in time.

Activity

3.2 How do we mount the USB stick or memory so that any user can write to it? If we want this type of USB memory to be mounted by default (when the system boots) and to have read and write permissions for all users, what line should we add?

Activity

3.3 Repeat all the actions of activity 3.2 on handling files and directories but instead of using the disk use the USB memory.

Now, from the disk or USB stick let's try to delete our first file. We will do so using the `rm` command:

```
root@tty1[subdir00]# rm file00 rm: remove regular file 'file00'?
```

We certainly did not expect the system to ask us this question. We press Ctrl+C to cancel, n for no (do not delete) and y for yes (delete). Now it is important for us to understand why the system has asked us this question. If we read the man of the `rm` command we will see that it should delete the file without asking us any question (let's remember that we can run this `man` on another tty, and that on tty2 we still have the man open, unless we have quit (pressing q), or switched off the computer since we invoked it, and continued working on the tty that we were on. But let's look at the explanation of option `-i` in this same man. What is happening? It looks as though this option is enabled by default. It is not really so, what happens is that by default the system has established some aliases; to view them all, let's execute the alias command:

```
root@tty1[etc]# alias alias ..='cd ..' alias cp='cp -i' alias l='ls -a --color=auto' alias
la='ls -la --color=auto' alias ll='ls -l --color=auto' alias ls='ls
--color=auto'
alias mv='mv -i' alias rm='rm -i' alias where='type -all' alias which='type -path'
```

Here we will start to understand much more, for example why the return of the `ls` command is in colours, or that to do an `ls -la` we just need to type `la`. And we also understand why when we have executed the previous `rm`, the system has asked us if we really wanted to do this. Through aliases we can establish and modify the default behaviour of commands or even create other new ones:

```
root@tty1[subdir00]# alias hi='echo¨I say hello"' root@tty1[subdir00]# hi I say hello
```

Continuing with the man of the `rm` command, we are about to read about the use of `-f` and `-r options.`. The first is for the command to execute without a prompt, or in other words, disobeying the `-i` options, so that if we did not delete our first file, we can do so now through `rm -f file00`. The second option forces recursivity, in other words, extends the order over potential subdirectories and their content. These options are common to most basic commands for handling files and directories; therefore, we can create a second directory in the root of the disk with all the contents of the first one that we have created through the `cp` command; to do this, we must appeal to recursivity:

```
root@tty1[subdir00]# cp /mnt/auto/floppy/dir00/ /mnt/auto/floppy/dir01 -r
```

Activity

3.4 Repeat this recursive copy of the disk's `/dir01` to the USB stick or memory.

In this case we have used absolute routing (for specifying both the origin and the destination) to carry out the copying operation. In other words, we have specified the full path, starting from the root directory, for both the origin and the destination. Similarly, we could have used relative directioning to specify the origin of the operation, its destination or both. When we use relative directioning, the routing origin is the current position within the file system. As in most cases, the system gives us the possibility of achieving the same results using different methods. The more we know the better and we can choose the most effective one in each particular case. Thus, we would have achieved the same result by using `cp ../../dir00/ ../../dir01 -r`, i.e. telling the cp that the origin is the directory `/dir00/`, which is two branches below our current position, and that the destination is `/dir01/`, which we also want to place two branches below our position. Also, the line `cp -r ../../../dir02/` would have been perfectly valid for the same purpose.

Activity

3.5 Explain the reason for this last assertion.

Now we can go down one subdirectory, placing us in `/mnt/auto/floppy/dir00/` and copy the second file that we have created into the `/subdir00` subdirectory within this same directory:

```
root@tty1[dir00]# cp subdir00/file01 .
```

We need to fully understand the meaning and reasoning behind the previous line. Firstly, we specify the origin of the file that we wish to copy (among many other options, `./subdir00/file00`) would also have been valid) and secondly we must specify the destination, which is the current position, i.e. ".". Through un ls we can check whether we have achieved the desired result. Now we can position ourselves in the parent directory, where it is mounted on the disk, and delete everything that we have created up until now. To do so, we will use the "*" wildcard (we also have the "?" wildcard, which is for just one character, in addition to other methods for specifying ranges of characters, and to refer, in general to more than one file or directory) to save typing:

```
root@tty1[floppy]# rm -rf *
```

Activity

3.6 Read the man of `rm` and understand why `rm -rf *` is so dangerous.

3.7. User administration

We should be alarmed by the fact of having performed all the previous tasks as home, since we already understand clearly that this user should only be used when necessary.

Nevertheless, it is justified since up until now we had not had any direct contact with a UNIX system, and this has allowed us to familiarise ourselves a bit with its basic commands and with its file system. The time has come to start working as we always should as of this moment, i.e. using the home account only when it is strictly necessary, such as in the first step that we will take next, creating a new system user and giving him or her an account, which is an operation that can only be done by home. Therefore, let's go on to create a user.

```
root@tty1[etc]# useradd user00
```

We have created our first user, but the process has been somewhat opaque since the system has not returned any information to us regarding this command that we have just executed. Therefore, we do not know what group or groups this user belongs to, where his home directory is, what default shell has been assigned to him or her etc. We will find a lot of this information in the passwords file (`/etc/passwd`) so let's analyse its content field by field:

```
root@tty1[/]#cd /etc root@tty1[etc]# cat passwd root:x:0:0:root:/home/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh . . .
partimag:x:104:65534::/home/partimag:/bin/false user00:x:1001:100::/home/user00:/bin/bash
```

We can find the information we want from the last line of the file, since our user is the last one to have been added. Let's analyse the content of this line:

- `user00` is the username, which we already know because we have created it ourselves.
- `x` his *password* is in the shadow file, `/etc/shadow`.
- `1001` is his user identification number (UID).
- `100` is the identification number of his group (GID).
- He has no associated comment.
- `/home/user00` this is his home directory, as was to be expected, since all users have their home in `/home/user`.
- `/bin/bash` the shell the user will encounter when logging in is the bash, the same one we have used up until now.

The question we ask immediately upon seeing this information is to do with the user's password. Therefore, we need to consult the `shadow` file, since we are told this by the file `/etc/passwd`. At the same time, we will introduce a new concept, the filter, which in this case materialises in the form of the `grep` (general regular expression processor) command. Filters are extremely potent tools, commonly used in UNIX, and their scope of action extends well beyond what is commonly understood by a filter. In this case, we will pass, through a pipe, the output of `cat` to `grep`, so that it displays on screen the lines that contain the word that we pass as a parameter, `user00`:

```
root@tty1[etc]# cat shadow | grep user00 user00:!:12115:0:99999:7:::
```

As of now we don't need to worry any more: the second field tells us that our user has no password. If the user had, we would not be able to know it, because as we have mentioned, the passwords are saved encrypted and encryption is unidirectional. But this should not concern us either, since, through the `passwd` command, home can change the password of any other user. Therefore, the only thing that we really need to do is to try not to forget the password of home, which does not have a password in this case either, as we are told by `shadow`, although if this were to happen, it can also be resolved. Let's continue investigating the properties of our new user. `/etc/group` will help us to know exactly what groups he belongs to:

```
root@tty1[etc]# cat group | grep user00
root@tty1[etc]# cat group | grep 100 users:x:100:knoppix knoppix:x:1000:
```

With the first command line we ask, with a negative answer, whether `/etc/group` has any explicit reference to *user00* (we could also have done this through the `groups: groups user00` command). With the second we search for the reverse of group 100, the primary group of the *user00* according to `/etc/passwd`, a group called users in this case. Therefore, we can already affirm that our new user belongs to just one group and that this group is user. We can also edit the file `/etc/group` to add users to groups and to create new groups, but we should note that in no circumstances can we put the name of another group in the field of users belonging to a particular group with the intention of adding all the users of the former to the latter. But we do not recommend doing this, since there are specific commands for working with groups (`newgrp`, `addgroup` etc.)

There is just one last issue to be resolved: if we list the contents of `/home/`, we will see that the subdirectory `/user00/` does not exist. Therefore, we need to create it and to give it the properties and relevant attributes manually:

```
root@tty1[etc]#mkdir /home/user00 root@tty1[etc]#chown user00 -R /home/user00
root@tty1[etc]#chgrp users -R /home/user00 root@tty1[etc]# cd /home root@tty1[home]#
ls -la total 0 drwxr-xr-x 5 root root 100 Mar 4 08:12 . drwxrwxrwt 4 root root 80
Mar 4 08:35 .. drwxr-xr-x 2 knoppix knoppix 40 Mar 4 08:35 knoppix drwxr-xr-x 2 root
root 40 Mar 4 08:35 root drwxr-xr-x 2 user00 users 60 Mar 4 08:12 user00
```

New users in Debian

The process of adding new users to the system through Debian is much simpler, as we will see later. But creating one through KNOPPIX has not been in vain since it has helped us to become familiar with the system and to learn new commands.

Activity

3.7 Through `man`, understand the functioning of the `chown` and `chgrp commands`.

The time has come to enter the system as a new user. We will do so through the su command, which opens a child process with the login of the user that we have passed to it. As root we can always use this mechanism to enter as another user without having to know the password, since when we execute su as root the password is not required; also, it so happens that the user that we have created does not have a password, and therefore, it will not be requested. But, if we tried to execute su in any other circumstances we would be asked for the user's password:

```
root@tty1[home]# su user00 su(pam_unix)[4312]: session opened for user user00 by (uid=0)
user00@tty1[home]$
```

First, let's note the changed appearance of the prompt. We have stopped being a home user, to become *user00*, so we have lost the privileges of the root, as shown by the change in the last character of the prompt. Now we can access our home directory, create a file and let's change its attributes so that only *user00* can read, write and execute it. For the time being, the execution permission does not make much sense to us, since we still do not know how to create executable files, but it is good to know in advance for when we need it:

```
user00@tty1[home]$ cd user00@tty1[user00]$ echo "only user00 can read, write and execute
this file." > user00file user00@tty1[user00]$ chmod 700 user00file
```

Activity

3.8 Create a new user following the steps described previously, and check that this new user indeed cannot even read the recently created file or write on it.

3.8. Manage processes

UNIX is characterised by its excellent management of the processes executed over the system. First of all, we should learn to detect what processes are running on the system and their particular features (mode in which they are running, resources consumed, who is executing them etc.). We will execute the ps (process status) command passing several parameters to see how these affect the information returned by the command:

```
root@tty1[/]# ps PID TTY TIME CMD 481 tty1 00:00:00 bash 1559 tty1 00:00:00 ps
```

Without arguments ps tells us what processes are running on the terminal on which it is executed; naturally, the first process will always correspond to the shell.

We could order another process to be executed in the background, `sleep`, for example (a process that simply waits for the number of seconds that we pass to it as a parameter to finish) and observe the `ps return`:

```
root@tty1[/]# sleep 300 [1] 1703 root@tty1[/]# ps PID TTY TIME CMD 481 tty1 00:00:00 bash
1703 tty1 00:00:00 sleep 1705 tty1 00:00:00 ps
```

Now we can ask ourselves about all the processes that are running on the system:

```
root@tty1[/]# ps aux USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND root
1 0.4 0.0 72 72 ? S 15:41 0:07 init [2] root 2 0.0 0.0 0 0 ? SW 15:41 0:00 [keventd] root
3 0.0 0.0 0 0 ? SWN 15:41 0:00 [ksoftirqd_CPU0] root 4 0.0 0.0 0 0 ? SW 15:41 0:00 [kswapd]
root 5 0.0 0.0 0 0 ? SW 15:41 0:00 [bdflush] root 6 0.0 0.0 0 0 ? SW 15:41 0:00 [kupdated]
root 52 0.0 0.0 0 0 ? SW 15:41 0:00 [kapmd] root 59 0.0 0.0 0 0 ? SW 15:41 0:00 [khubd]
root 433 0.0 0.1 1368 616 ? S 15:41 0:00 pump -i eth0 root 475 0.0 0.1 1316 596 ? S 15:41
0:00 /usr/sbin/automount root 481 0.0 0.3 2864 1988 tty1 S 15:41 0:00 /bin/bash -login root
482 0.0 0.3 2864 1988 tty2 S 15:41 0:00 /bin/bash -login root 483 0.0 0.3 2856 1952 tty3 S
15:41 0:00 /bin/bash -login root 484 0.0 0.3 2856 1976 tty4 S 15:41 0:00 /bin/bash -login
root 2086 0.0 0.3 3436 1552 tty1 R 16:06 0:00 ps aux
```

In certain situations it can also be interesting to execute the `top` command, which tells us the activity of the CPU, use of memory etc., interactively.

Now we are going to boot a process, stop it, and order it to be executed in the background. To do this, we will execute a `sleep 20` and press the Ctrl+Z key combination before 20 seconds, to stop it, `jobs` to make sure that it has indeed stopped and obtain its identification number, and `bg` together with its identification number to order it to run in background mode:

```
root@tty1[/]# sleep 20 <Ctrl+Z> [1]+ Stopped sleep 20 \intro root@tty1[/]#jobs [1]+
Stopped sleep 20 root@tty1[/]# bg 1 [1]+ sleep 20 & [1]+ Done sleep 20 root@tty1[/]#
```

Once the 20 seconds have passed, the process has stopped while still in the background, and we have been told this on screen. We can check through `ps` that indeed there is no process running. At this point, we can stop a process and return it to the foreground using `fg`:

```
root@tty1[/]# man ls <Ctrl+z> [1]+ Stopped man ls root@tty1[/]# jobs [1]+ Stopped
man ls root@tty1[/]# fg 1
```

3.9. Enabling and using the mouse

The mouse is an extremely useful tool for working in console mode, since it can be used to select and paste text (this is how we have captured all inputs and outputs to prepare this workshop's text, for example).

Therefore, the first thing that we will do is to configure the mouse, and we will do this using the most commonly used program, the gpm, which already runs automatically in background mode (this is because gpm is a daemon, a term that we will analyse in subsequent sections).

There are many types of mouse devices (mechanical, optical, laser and trackball) and many types of connections (serial port, PS/2 port, USB, laser, wireless: radio frequency, infrared or bluetooth), although currently the ones used most often are the ones that have the USB or PS/2 type connectors (normally they are distributed with the two types of connector). If we have a mouse of the first type, we will execute the following line:

```
#gpm -m /dev/input/mice -t autops2
```

If what we have is a mouse connected to the PS/2 port, to enable it we will use the same procedure as above varying only the device and type of mouse:

```
#gpm -m /dev/psaux -t ps2
```

In principle, after executing one of the two command lines above, as home, by moving the mouse we should see the mouse arrow on all terminals, and by executing:

```
sofia@tty1[/]$ ps aux | grep gpm
```

we should obtain a reply of the type

```
root 3594 0.0 0.1 1632 532 ? Ss 00:28 0:00 gpm -m
/dev/input/mice -t autops2
```

while gpm is running in the background.

If not, it means our mouse is non-standard. In this case we will need to read the man of the gpm carefully and execute `gpm -t help` to try and identify the type of mouse that adapts to the one we have.

Once the mouse has been configured, we can select the part of the text we are interested in, and when we press the middle button we will paste the selected content in the cursor's current position. If we are used to using the mouse for placing the cursor, we could find that the results we obtain are not precisely the ones we wanted, but by practicing a bit, we will soon get used to this way of working.

We can also configure the `gpm` through the `gpmconfig` command or the `dpkg-reconfigure gpm` utility.

Activity

3.9 How would we configure the Touch Pad to emulate a two-button mouse?

We should note that the buffer content of the mouse is kept when we pass from one *tty* to another.

Activity

3.10 Taking advantage of the fact that the mouse buffer content is kept between terminals, open a `vi` session and on another place yourselves in the file system root, list its content in detail and port this data to the text editor. Save the text file's content and using `cat`, check that we have effectively obtained the required results.

3.10. Other operations

To finish this first workshop, we will execute a series of commands to continue with the process of becoming familiar with the system and acquiring resources to solve potential problems in future.

We should find out about the system, the type of machine we are running on, the hardware we have installed etc.

So here you have a few commands, remember that with `man` and the name of the command you have some help to find out what the commands are for and all the options that they can have.

```
knoppix@0[knoppix]$ uname -a Linux Knoppix 2.6.12 #2 SMP Tue Aug 9 23:20:52 CEST 2005 i686
GNU/Linux knoppix@0[knoppix]$ cat /proc/cpuinfo processor : 0 vendor_id : GenuineIntel cpu
family : 15 model : 2 model name : Intel(R) Pentium(R) 4 CPU 2.80GHz stepping : 7 cpu MHz :
2801.835 cache size : 512 KB fdiv_bug : no hlt_bug : no f00f_bug : no coma_bug : no fpu :
yes fpu_exception : yes cpuid level : 2 wp : yes flags : fpu vme de pse tsc msr pae mce cx8
sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe cid bogomips
: 5455.87 knoppix@0[knoppix]$ lspci 0000:00:00.0 Host bridge: Silicon Integrated Systems
[SiS] SiS645DX Host & Memory & AGP Controller (rev 01) 0000:00:01.0 PCI bridge:
Silicon Integrated Systems [SiS] Virtual PCI-to-PCI bridge (AGP) 0000:00:02.0 ISA bridge:
Silicon Integrated Systems [SiS] SiS962 [MuTIOL Media IO] (rev 14) 0000:00:02.1 SMBus:
Silicon Integrated Systems [SiS]: Unknown device 0016 0000:00:02.3 FireWire (IEEE 1394):
Silicon Integrated Systems [SiS] FireWire Controller 0000:00:02.5 IDE interface: Silicon
Integrated Systems [SiS] 5513 [IDE] 0000:00:02.6 Modem: Silicon Integrated Systems [SiS]
AC'97 Modem Controller (rev a0) 0000:00:02.7 Multimedia audio controller: Silicon
Integrated Systems [SiS] Sound Controller (rev a0) 0000:00:03.0 USB Controller: Silicon
Integrated Systems [SiS] USB 1.0 Controller (rev 0f) 0000:00:03.1 USB Controller: Silicon
Integrated Systems [SiS] USB 1.0 Controller (rev 0f) 0000:00:03.2 USB Controller: Silicon
Integrated Systems [SiS] USB 1.0 Controller (rev 0f) 0000:00:03.3 USB Controller: Silicon
Integrated Systems [SiS] USB 2.0 Controller 0000:00:04.0 Ethernet controller: Silicon
Integrated Systems [SiS] SiS900 PCI Fast Ethernet (rev 91) 0000:00:08.0 CardBus bridge: O2
Micro, Inc. OZ6912 Cardbus Controller 0000:01:00.0 VGA compatible controller: ATI Technologies
Inc Radeon R250 Lf [Radeon Mobility 9000 M9] (rev 01) knoppix@0[knoppix]$ df S.files 1K blocks
Used Free Use% Mounted on /dev/root 2471 21 2450 1% / /ramdisk 401312 8388 392924 3% /ramdisk
/UNIONFS 7764900 7371976 392924 95% /UNIONFS /dev/hdc 4067388 4067388 0 100% /cdrom /dev/cloop
4963828 4963828 0 100% /KNOPPIX /dev/cloop2 2399762 2399762 0 100% /KNOPPIX2 /UNIONFS/dev/sda1
255452 14552 240900 6% /mnt/sda1 knoppix@0[sda1]$ free total used free shared buffers cached
Mem: 514720 495232 19488 0 15476 350084 -/+ buffers/cache: 129672 385048 Swap: 497972 0 497972
```

Activity

3.11 In the virtual directory `/proc/` and in its subdirectories we will find a lot of information regarding our system in text files. Place yourselves in this directory, and using `more` or `cat` explore these files' information.

3.11. Conclusion

This has been our first contact with a UNIX environment. It has helped to break the myth that working on this type of operating systems is complicated and difficult. Additionally, it has helped us to start understanding its capabilities and versatility. We have learned how to move through its file system and to carry out operations with the files, to search for information etc. Everything we have learned up until now will be very useful from here on and will serve as a basis for expanding our knowledge.

At the same time, the enormous hardware autodetection and configuration capabilities of KNOPPIX means that it can be very useful to us when it comes to installing Debian in the next workshop, in case hardware problems arise. We can always restart KNOPPIX and study how it has solved this problem (basically by studying the configuration files).

As a last workshop activity, we can reboot KNOPPIX in graphic mode, in other words, without passing parameter 2 when we initiate it (another way of doing this is by passing the `knoppix lang=es wheelmouse` parameters).

4. GNU/Linux installation

4.1. Introduction

In this section we will look at the essential steps followed in most GNU/Linux installation processes. Although each distribution has its own installation environment, all of them share some basic steps. In this section we will look at these steps and explain the concepts we need to carry them out successfully. It is important before we install a new operating system that we are suitably familiar with the main components that we have installed on our computer so that we can configure everything properly, even if the distribution we use includes hardware detection.

It is possible that we may have two totally independent operating systems installed on just one hard disk. Although the process of installing another operating system on the same disk (or on another disk installed on the same computer) should not interfere with the others' partitions, it is advisable to make backup copies of all our documents. Despite the fact that if we follow all the steps we describe below correctly it is practically impossible to lose information, caution is always advisable and so is making backups of important files.

Preparation to install Linux

Before starting with the installation process it is convenient to know the brand and model of the graphic and sound card we have installed; the network card; the brand, type and characteristics of the monitor, as well as any other special hardware that we have. Generally, for the motherboard, the CPU and the RAM, we do not need to know their characteristics. To obtain this information, we can study the manuals given to us when we purchased the computer or, if we have another operating system installed, use it for this purpose (on Windows^{TM} systems we can obtain this data from the control panel).

4.2. Booting

In general, all GNU/Linux distributions provide some type of means for booting the installation process. Currently, it is typical to provide a start-up CD or DVD or if our computer does not have the corresponding reader or if we wish to do a remote installation, a start-up disk is provided.

The first step of the process tends to be selecting the type of installation we wish to carry out. In general, this choice is given in order to identify the type of user installing the system, providing accordingly more or less information and allowing the user to configure the system in more detail or not. If we have sufficient knowledge, it is advisable to do the installation in expert mode (or similar) so that we can adapt the system better to our particular needs. In some distributions, this first step also serves for choosing the version of the kernel that we wish to install or to configure the installation process in graphic mode, text mode etc. It is essential to read carefully all of the information we are provided with in this first phase so that we can choose correctly what best suits the use we wish to give our system.

Next, most distributions will allow us to choose the type of keyboard (or similar configuration) that we wish to use. Although there are many different types, the most common ones are qwerty (the first characters starting from the top left of the keyboard), so we will select `qwerty/ es (Spain)`.

Installation methods

Some GNU/Linux distributions allow us to carry out the installation via any means: CD, DVD, FTP, HTTP, hard disk, NFS etc. There are also installation methods that allow us to boot the process from other operating systems.

4.3. Partitioning the disk

Partitioning the hard disk is one of the most critical parts of the entire process. This step involves dividing the disk into various sections, so that each one is taken as an independent unit. If we already have an operating system installed on our computer, the disk will be partitioned into one or several partitions, but if the disk is new, it generally won't be.

Choice of file system

Although there are ways of installing GNU/Linux using the file system of another operating system, it is not advisable to use this type of installation because the system's performance is significantly reduced.

To install GNU/Linux we need to have at least one partition for it. The fact that when we modify the size of a partition we need to eliminate it and create it again implies losing all the information it contains. For this reason and others, there are programs that allow us to modify the size of partitions without having to eliminate them. `fips` is a program with a GPL license that allows us to resize our partitions formatted with FAT (for Windows^{TM} systems of the non-NT branch) without losing their information. There are also other commercial programs that allow us to carry out this type of operation with any other file system, meaning that if we do not want to lose our system's information, we should use one of them before starting with the entire process.

Disk partitions

Although with one or two partitions we would have enough to be able to install GNU/Linux, it is interesting to divide the disk into more fragments and to place certain directories in different units so that we can manage the resources more efficiently, avoid system crashes due to disk saturation etc. At all events, we will leave these aspects for more advanced courses on administration.

It is advisable for GNU/Linux to use two hard disk partitions. One will serve for saving the system files and the other for swap. Swap is an exchange zone between the computer's RAM and the hard disk. It helps when the operating system's RAM is totally occupied and the programs being executed need more memory. This is when we start using the swap to save zones of the RAM that are not being used, interchanging them so that the applications are not left without available memory. It is also possible to do without swap, but it is not advisable because the system will not be able to manage the resources as efficiently; besides, when we use several applications simultaneously they will be left without enough memory much more easily. Although the size of the swap can be as big as we want, we advise making it the double of the RAM installed on the computer if we have 64 MB or less, and the same if we have more. These calculations are based on system performance tests that show that they reach a point where, if the applications need to use too much swap memory, the performance decreases a lot, causing the system to become almost saturated (to see what amount of RAM and swap memory is being used, the system gives us the `free command`).

There are several applications for fragmenting the disk. One of the first to appear was `fdisk`, although currently there are others such as `cfdisk`, `diskDruid` etc. In some installation processes, we can choose which we wish to use, although they all allow us to do exactly the same thing with changes in the environment's presentation. The way GNU/Linux identifies the disks is `/dev/hd`*`X`* for IDE disks and `/dev/sd`*`X`* for SCSI and Serial ATA, where in both cases the *X* is a letter corresponding to the disk to which we wish to refer as follows:

Types of hard disks

For personal computers there are three types of hard disks: IDE, Serial ATA and SCSI. At present, most motherboards carry Serial ATA controllers. SATA controllers allow up to four or six hard disks. SCSI controllers (Small/smart Computer System Interface) allow up to 8 devices and have higher transfer rates, although they are also much more expensive than IDEs.

Master boot record

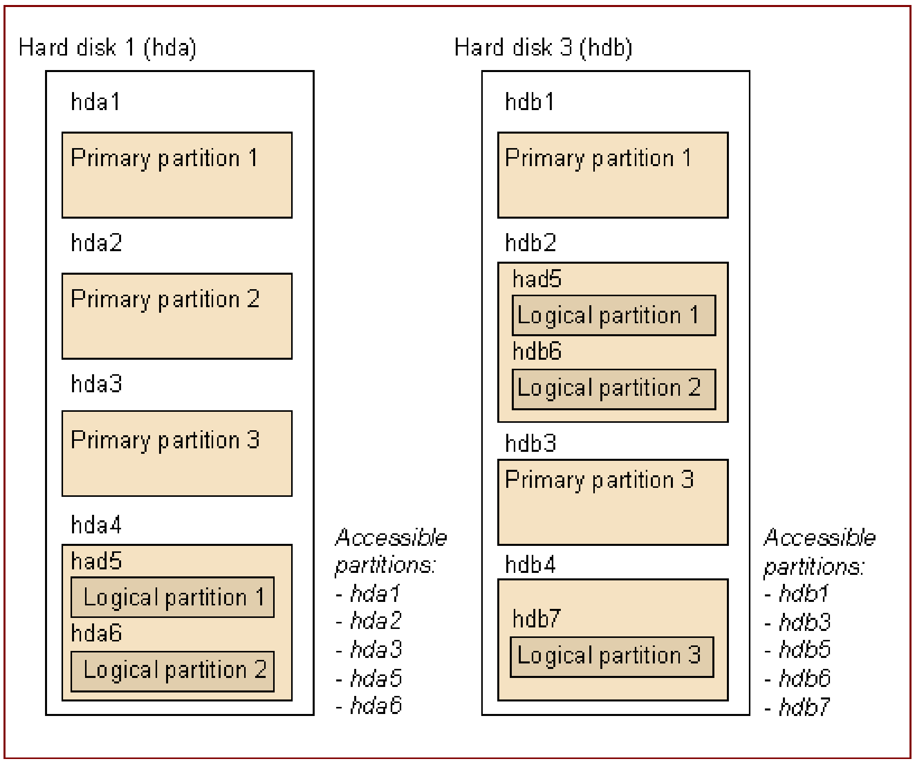
To save the information of partitions and the boot program, the disks have a reserved data zone called MBR (Master Boot Record).

Table 4.1

Device	Meaning
`/dev/had`	Master of the first IDE channel
`/dev/hdb`	Slave of the first IDE channel
`/dev/hdc`	Master of the second IDE channel
`/dev/hdd`	Slave of the second IDE channel
`/dev/sda`	First disk of the SCSI or SATA controller
`/dev/sdb`	Second disk of the SCSI or SATA controller

If we have more than one disk in our computer, before we enter the fragmentation program we can choose which one to work on. When we create a partition we will be asked if it should be primary or logical. On a hard disk we can have up to four primary partitions and up to 64 logical ones. If we do not need more than 4 partitions, we can choose either of the two types. If we need more, we should bear in mind that the logical ones are placed within a primary one (up to a maximum of 16 for each) so we cannot have 4 primary partitions created and then add other logical ones. In this case, we should create 3 primary ones and up to 16 logical ones in the fourth primary partition. The following figure gives us a graphic example:

Figure 4.1



When we create a partition, we need to specify what file system we will use for it (Linux ext2, Linux ext3 or Linux swap). Once we have made the partitions, we will save the configuration and we must tell the installation process where we wish to place the root filesystem and system swap. Once we have carried out these steps, we can continue with the rest of the installation process.

Active partition

Of all the partitions of a hard disk we can select one to be the most active. This flag serves to tell the system's BIOS or EFI which partition should be initiated if it finds no start-up program in the MBR.

4.4. Installation of modules

The kernel's modules are software parts specialised in a particular task. With all manner of different devices and dozens of functions for multiple file systems, online management operations etc., it was decided not to include all of these drivers and functions as standard. In one of the first versions of Linux, when the kernel we had did not include any function, we had to recompile it in full and generate a new one adapted to our needs. With the incorporation of modules this process is no longer necessary and we can directly choose the functions we need.

Generally, the distributions already incorporate hundreds of different modules for the Linux kernel. They tend to be organised into different categories to make finding them easier and normally, with the same name of the module or the brief description that comes with it we can already know what device it is designed for or what function it carries out. This is why we mentioned earlier that it is important to know what hardware our computer has: in this step we can choose in more detail what modules we need. For some of them we may have to pass a special parameter (such as the I/O address, the interruption used by the device etc.), information that we can obtain from the system's BIOS or EFI or through the procedures mentioned before.

It is also true that if the installation process carries some type of program for automatic hardware recognition, the whole process of selecting and loading modules is not necessary because it is already carried out automatically. However, it may not detect correctly a device we have installed, in which case we will need to include the corresponding module manually.

If we forget to include a module (or install some new device) at the time of installing the system, we can always use the following commands: `insmod` (to add a new module), `lsmod` (to list the modules installed), `rmmod` (to eliminate a module) and `modprobe` (to test one, and if it works correctly, include it in the kernel). All of these modules are no more than binary files that we tend to find in the system's `/lib/modules/` directory.

If we have problems discovering what hardware we have installed, a very practical utility that tends to come with the distributions is discover. We can use it to know exactly what components we have and what modules[1] correspond to them.

(1)We can normally find the modules configuration file in `/etc/modules`.

4.5. Basic network configuration

After configuring the modules that will be included in the operating system's kernel, we need to configure the network (if we have the required card). Although in this document we will not go into the details of computer networks, we will provide the necessary ideas to carry out this step without any complications.

The first thing we will be asked for is the name we wish to give the system. This name will serve to refer to it without always having to remember its IP address. Once we have entered the name, we will normally be asked if on our network we use a mechanism called DHCP or BOOTP, which consists of having a special server responsible for automatically assigning IPs to computers that boot. If we use this mechanism, we need to specify that this is the case, and if not, we will be asked for the IP and mask of our computer. If we do not know this data, we must ask our network administrator. Next we will have to enter the IP of our network's gateway. The gateway is a device or a computer that acts as a bridge between our local network and the Internet (if we do not have any device of this type, we can leave this field blank). Next, we should specify the name server or servers that we use. A name server (DNS server) is a machine that gives us the equivalence between a name and an IP address. If we do not know which one we are using we will also have to ask the network administrator.

IP address

An IP address is a computer's identification within a network when we use the TCP/IP protocol, the most commonly used on the Internet.

If we are in a local network, we will need to consult its administrator to provide us with all the necessary information in this regard. If we have another operating system installed on the computer, we can also access its configuration in order to obtain this information. However, we should never invent these values, since the most likely result is that it will not configure properly and we will cause problems on the local network.

4.6. Boot system

Once we have configured all these settings, we will need to install a small program on the hard disk so that during the computer boot process we can choose which of the operating systems we have installed we want to boot. Even if we have only installed GNU/Linux on the system, we should still install this program, since its boot system requires it.

There are various applications for carrying out this process. The most common ones are Lilo (Linux Loader) and Grub (GNU Grand Unified Bootloader). The only thing these applications do is to initiate the process of loading and executing the kernel of the operating system that we specify. In general, all distributions detect whether we have another operating system installed on the hard disks and automatically configure the boot system for us. All we need to bear in mind is that we will have to place this program correctly so that it runs when the computer is started up. Normally, it tends to be placed in the MBR of the master disk of the first IDE channel or SCSI, which is the first place that the computer's BIOS or EFI inspects in search of a program of these characteristics.

4.7. Package selection

Most installation processes include two ways of selecting the programs that we will install on the system: basic or expert. With the basic selection process the available packages will normally be grouped into large sets of programs: administration, software development, office suites, mathematical etc. If we still do not know exactly what programs we are going to use, this type of installation is the most appropriate, since we can choose what type of programs we use in general and without going into detail.

Packages

A package consists of one or several interrelated programs/libraries, grouped to comprise a single block. Most distributions include utilities for handling packages (installation, elimination, configuration etc.). This type of organisation is very useful because when we need a program, utility etc., we can install it all at once.

When we know a bit more about the system and know exactly what we are going to use it for, the expert package selection is better. With this type of selection we can adjust the programs that we need much better, saving disk space and avoiding the system loading more programs than are necessary.

RedHat update

RedHat has also adopted a download or update system in a similar style to Debian's `apt`.

When we install a system for a specific use (http server, cvs etc.) it is advisable to choose only those programs that we will really use, in order to avoid security problems (the fewer doors we leave open, the safer the system).

Some distributions also allow the possibility of obtaining packages from different locations, such as one or several CDs or DVDs, from servers on the Internet etc. Debian GNU/Linux was the first to have a system of this type, called `apt`. This type of management is very useful because it gives us a great deal of flexibility and keeps us informed about the latest program corrections and updates.

4.8. Other aspects

We need to know what to do if we have had any problem with the installation or at some time the boot system has not allowed us to load the operating system. Usually, during the installation process there is a step that asks us if we wish to create a recovery disk. It is always advisable to generate this disk, since it allows us to load the operating system and access the file system in order to fix what is needed.

In some distributions, we can do the same with the start-up CD. In the case of Debian for example, the boot process itself includes a console (we can access it by pressing Ctrl+F2) with the basic commands for us to carry out essential recovery operations. At all events, if we are facing a serious problem that does not allow us to boot the system correctly and we want all the usual tools, we can also boot with a live CD of Knoppix or any other distribution and mount the unit where we have the system installed in order to fix the fault. What is important is to have one of these tools and to have tested it before a serious problem occurs so that we are always prepared.

Most installation processes use an application called bootstrap, which we can also install on the system with the corresponding package. With it we could create our own installation process or see how the system really carries out these installation and configuration tasks. We can reproduce most of them with the `base-config program` or any of the others that it calls itself (you can find further information in the manual), although we can always turn to the operating system's configuration files to do it manually.

5. Debian Etch installation workshop

5.1. Introduction

Whereas the first workshop helped us to take our first steps on a UNIX-type system, and to do so we used a distribution that leaves no trace on our computer, since it booted and was executed from a CD-ROM or DVD-ROM, this one serves to teach us how to install a basic GNU/Linux system on our computer.

The distribution selected for this workshop is Debian GNU/Linux 4.0, known as Etch (the latest one at the time of publishing this work), which was released on 8th April 2007 following 21 months of continuous development. Debian GNU/Linux is a free operating system that supports a total of eleven processor architectures.

In terms of the software, its innovations include the desktop environments GNOME 2.14, KDE 3.5.5a and Xfce 4.4; Office suites OpenOffice.org 2.0.4a, KOffice 1.6, GNUcash 2.0.5, GNUmeric 1.6.3 and Abiword 2.4.6; Coreo Evolution 2.6.3 and instant messaging Gaim 2.0; iceweasel 2.0.0.2 is the Firefox navigator, while icedove 1.5 is the Thunderbird mail client; both programs are versions that the registered trademarks do not use.

It also includes cryptographic programs, it is compatible with FHS version 2.3 and with the programs developed for version 3.1 of the LSB.

Debian GNU/Linux 4.0 introduces a new graphic interface for the installation system which supports spellings that use compound characters and complex languages. The installation system for Debian GNU/Linux is now translated into 58 languages. As of Debian GNU/Linux 4.0, security and the efficiency of the package management system has improved.

Debian GNU/Linux can be executed on computers running from PDAs and pocket systems to supercomputers, via almost any intermediary system. It offers maintenance for a total of eleven architectures: Sun SPARC (sparc), HP Alpha (alpha), Motorola/IBM PowerPC (powerpc), Intel IA-32 (i386) e IA-64 (ia64), HP PA-RISC (hppa), MIPS (mips, mipsel), ARM (arm), IBM S/390 (s390) and (introduced for the first time in Debian GNU/Linux 4.0) AMD64 and Intel EM64T (amd64).

The choice between Debian and RedHat for this workshop and the two remaining ones was not easy, since RedHat offers commercial products that are very well received in the business world, and in general it is considered much simpler to install a Red Hat than a Debian distribution. The fact that we have chosen Debian is due to the following main reasons: firstly, this distribution reflects the philosophy of GNU/Linux more faithfully and everything is achieved thanks to the voluntary work of thousands of people; secondly, it is probably one of the distributions that gives users most freedom when it comes to installing and maintaining it; and as a last fundamental point, for its packages system, which is probably the most consistent one currently available with 18,733 packages. Therefore, Debian has been chosen not because it is considered better than RedHat (there are certainly as many opinions in this regard as there are users of each distribution), but simply because its flexibility makes it suitable for academic purposes. But aware of the influence of RedHat, we have dedicated an appendix to this distribution so that readers can learn about it too and form their own opinions.

Neither should we forget that aside from these two distributions there are many more. But we believe that once you have had an initial contact with any of them, it is each user's job to test different distributions and form an opinion of which best adapts to his or her needs or demands. For this reason, once again we would encourage each individual to experiment and to test and to target efforts in the most appropriate direction. GNU/Linux is not a closed system, on the contrary, GNU/Linux is synonymous with freedom, and for this reason the basic intention of this module is, in general, that once the necessary foundations of knowledge have been established, this freedom can be exercised with no impediments, with the benefit of all its advantages and assuming all of its consequences.

We would note that in Spain, and in the educational context in particular, many autonomous communities have chosen free software. It started in Extremadura, with GnuLinEx, a GNU/Linux distribution based on Debian. And it was followed by Guadalinex in Andalucía, LliureX in the Community of Valencia, Max in the Community of Madrid, Molinux in Castilla-La Mancha etc. Although initially they were all based on Debian, some of them have changed and are now based on Ubuntu, a GNU/Linux distribution that is currently becoming very popular.

So, aware of the growing force of Ubuntu, we have dedicated an appendix to this distribution so that readers can also become familiar with it and form their own opinions.

Activity

5.1 This activity aims for you to visit the web pages indicated above to find out about the actions that are being carried out, in different autonomous communities, to introduce and promote the use of free software in the educational context.

Activity

5.2 Since we are going to install Debian, we advise visiting its web page and becoming familiar with its content. Thus, we propose you visit the web page and its subsections.

Once more we encourage readers to give their curiosity free rein, and on this occasion to pursue the links they find interesting.

5.1.1. Installation systems

The installation system is understood to mean the resources or devices used to install the operating system. Currently, almost all the distributions provide different possibilities for carrying it out, but essentially we can distinguish between two installation systems: CD-ROM/DVD-ROM or Internet. Also, in general, it is possible to combine different systems.

Debian offers various types of installation:

- Using a set of CD-ROMs or DVD-ROMs that use `debian-installer`. This set of CD-ROMs/DVD-ROMs contain all the Debian packages, meaning that we can carry out a full installation without connecting to a network. We can download Debian GNU/Linux using bittorrent (this is the recommended method now), jigdo or HTTP/FTP.

- Using a CD-ROM. There are two different network installation images for CD-ROM ("netinst") that can be used to install Etch with the `debian-installer`. These images are designed so that you can boot from the CD and install the additional packages you wish from the Net, hence the name "netinst". The difference between the two available images is that the full "netinst" image contains the installer and the base system, the "visitor's card" image is smaller and does not contain the base system, meaning it has to download it from the Net. You can obtain a full CD image that does not need a connection to the Net to complete the installation. To do this you will just need to use the first CD of the set of Debian CDs.

- Through the boot disk (`boot.img`), when the disk boots, it will ask you to insert a second disk (`root.img`). If you are going to carry out the network installation, you will need the image `floppy/net-drivers.img`, since it contains additional controllers for many Ethernet network cards, and also includes support for PCMCIA. In the case of carrying out the installation from CD-ROM, and not being able to boot from it, boot first from a disk and use the image `floppy/cd-drivers.img` as controllers disk to complete the installation using the CD-ROM.

- Using the USB memory device. The easiest way of preparing your USB memory device is by downloading `hd-media/boot.img.gz` and using `gunzip` to extract the image. Next, mount the memory device (it will have a FAT file system) and, finally, download an image of the Debian "netinst" CD and copy this file onto the memory device (it should be `.iso`).

- Using the network. The different methods for network booting depend on the architecture and configuration for network booting (for example, by DHCP). The easiest way to configure a network boot is probably using PXE.

- Using the hard disk. It is possible to boot the installer without using removable means, but only if there is an existing hard disk, which could have a different operating system. Download `hd-media/initrd.gz`, `hd-media/vmlinuz`, and a Debian CD image in the highest level directory on the hard disk (the CD image should have a file name ending in `.iso`). Now we just need to boot Linux with `initrd`.

This summary is merely designed to list the different types of installation that Debian offers. But to carry out the installation we advise you to visit its web page, as we have already mentioned, and to carefully read the "HOWTO install".

The degree of interaction that an installation requires will also depend on the system and selected type of installation. As can be expected, each one of these systems has its advantages and its inconveniences: whereas a standard installation allows us to configure step by step, something extremely useful for adapting the system to our needs and possibilities, a totally automatic installation system requires more advanced infrastructure and knowledge, and therefore more of an investment in both time and money, which can only be justified if the number of systems that we are going to mount is very large (for example, we could consider the implementation of this type of installation in a department where computers designed for personal use cohabit with others designed for parallelisation, and where their number often increases).

When it comes to choosing a system and a type of installation, we need to consider several factors, such as: how many installations are we going to carry out?, how many different installations are we going to carry out?, what level of experience do we have? etc. The fact that this is our first installation leads us immediately to the most interactive and frequently used type of installation: the standard interactive installation.

Now, we just need to determine the installation system, and this will depend fundamentally on whether we have an Internet connection and its speed. Obviously, if we do not have an Internet connection or our access speed is very low, we have no choice but to choose the installation based on a set of CD-ROMs or DVD-ROMs; if on the other hand we have an acceptable Internet access speed (such as provided by lines based on ADSL technology) or high access speed (direct Internet connection via a gateway), the best option will be to choose a network installation.

A network installation offers many advantages over one carried out by CD-ROM or DVD-ROM, and especially with Debian, since it will allow us to install the latest available versions of the packages; furthermore, updating all the packages installed on the system is as easy as executing a single instruction. But this fact should not make us abandon an installation using CD-ROM or DVD-ROM if, as we have mentioned, we do not have an Internet connection or the one we have is too slow (we should not underestimate either the packages that these CD-ROMs or DVD-ROMs contain, since Debian is characterised as a distribution that only includes extensively tested packages). And once we have completed the installation we can always change the content of the file `/etc/apt/sources.list` and add the relevant lines to access the Internet also, especially for security updates.

5.1.2. Types of packages

Next we focus on Debian and its packages system. A Debian package can be identified by its `.deb` extension. All packages included in the official Debian distribution are free according to the Debian free software guidelines. This ensures the free use and redistribution of the packages and all of their source code. The Debian distribution distinguishes between four different classes of packages. The distribution's own packages are included in the main class, whereas classes contrib, non-free and non-US are provided by the Debian organisation for the benefit of their users:

- **main.** Packages that comply with the Debian free software guidelines, in other words, free use and redistribution of all the binaries comprising them, in addition to their complete source code, is guaranteed.

- **contrib.** Packages that, while free and therefore like the binaries have their source code available, depend on other packages that are not.

- **non-free.** Packages that although they may be free of charge, have restrictive conditions that in some way limit their use or redistribution.

- **non-US/main.** The packages of this area are free themselves, but cannot be exported from a server in the US.

- **non-US/non-free.** Packages that cannot be exported from the US because they contain ciphered software or software that can affect issues relating to patents.

The official Debian distribution consists of the content in the main section of the Debian file.

5.1.3. Package development status

The package development status will condition the type of distribution that we install. Thus, there are three different types of distribution:

- ***Stable*** **.** this is the most recent official version of the Debian GNU/Linux distribution. It consists of stable well-tested software and changes only to incorporate important security or usability corrections.

- ***Testing*** **.** Distribution that contains the packages expected to become part of the next stable distribution. There are a series of very strict requirements that each package must fulfil before moving from *unstable* to *testing*. *Testing* does not have the security team's updates at the same time as they are released..

- ***Unstable*** . This distribution includes the most recent Debian packages, and consequently, the least tested. Therefore, they can contain serious enough problems to affect the system's stability.

Although all packages have their own dependencies, there is no problem in mixing the packages of different distributions. `apt-pinning` (consult, `man apt-pinning`) makes this task much easier. Even so, in critical systems it is advisable to use only the stable distribution's packages, which are the most reliable.

5.2. Installation of Debian Etch

As commented, Debian presents various types of installation, from DVD-ROM, CD-ROM, USB memories and diskettes, to using a network. The most commonly used types of installation are: CD-ROM or DVD-ROM, minimum CD-ROM or network installation. If we can master the first and the last, the second one is irrelevant. Therefore, we will start by installing our first system using the set of DVD-ROMs and, once finished, we will analyse the differences between this process and the network installation.

For the first time in Debian GNU/Linux, multi-architecture CDs and DVDs are provided that allow multiple architectures to be installed from a single disk.

5.2.1. Debian Etch flavours

Debian GNU/Linux 4.0 or Debian Etch supports twelve main architectures and some variations of each architecture known as flavours (different pre-compiled kernels designed to support different types of hardware).

For example, Debian GNU/Linux 4.0 for Intel x86 architecture comes by default with version 2.6.18 of the kernel. For more information, you can consult the document containing the installation instructions for the Debian GNU/Linux 4.0 system (key name "etch"), for the Intel x86 architecture. Also, it contains links to other sources of information, in addition to information on how to make the most of your new Debian system.

5.2.2. Installing Debian GNU/Linux 4.0 For Intel x86

The basic document that Debian offers us to install it is the *Installing Debian GNU/Linux 4.0 For i386*, included in the DVD1 (`/doc/install/manual/en/index.html`) available in different languages, including Spanish (`/doc/install/manual/es/index.html`). We advise reading this document and having a copy of it at hand in case there is any problem, such as not having a bootable DVD-ROM unit.

5.3. Installation of Debian Etch from DVD-ROM

We will carry out this installation on a standard computer over an ATA hard disk connected as master over the standard primary IDE port. Where controllers and SCSI disks are concerned, these will be detected without any problem during the start-up process, as in the case of Serial-ATA disks. From the Debian web page we can download the three DVDs that belong to this version although for a basic desktop system we will have enough with just the first one.

5.3.1. Before starting the installation

Before starting with the installation itself, we will need to make sure that we have a minimum disk space on our hard disk (we advise, having between at least 2 and 3 GB of free space). If it is new, we can start directly with the installation even if we are also considering installing another operating system on it (all we will have to do is reserve the space for it with the type of partition that it requires).

If we have a space that we had previously reserved, because we already had in mind to install GNU/Linux, or we have a partition of any other operating system where we wish to install it, we can also continue with the installation, in other words boot from the DVD-ROM.

If on the other hand the hard disk is full and has just one partition (something we do not recommend since in general this makes the performance of any operating system decrease considerably, and particularly those with not very consistent file systems), we will need to liberate space so as to install Debian GNU/Linux 4.0. The difficulty of carrying out this operation will depend strictly on what file system the partition contains.

It is likely that the operating system in question belongs to the Microsoft ^{TM} product family; if the file system is of the FAT or FAT32 type (used by MS-DOS^{TM}, Windows95^{TM}and Windows98^{TM}) the problem is relatively simple to overcome, since the distribution itself provides an application (fips20.exe on DVD1 /tools/fips20.zip) which will help us to repartition the hard disk and create space for installing GNU/Linux.

If we already had a GNU/Linux operating system installed, we can use one of the many applications that there are for resizing hard disk partitions and creating new ones. Like partman for example (an original GNU/Linux tool), cfdisk, gparted (hard disk partitioning graphic editor under GNU/Linux and Gnome) or qtparted (hard disk partitioning graphic editor under GNU/ Linux and KDE). These applications manage FAT or FAT32 type file systems as well as NTFS.

If we did not have a GNU/Linux operating system installed we can use the Gparted Live CD, it is a live CD that contains a small and basic GNU/Linux operating system executed directly from the CD-ROM containing the Gparted application.

And as a last option, we can always turn to commercial applications.

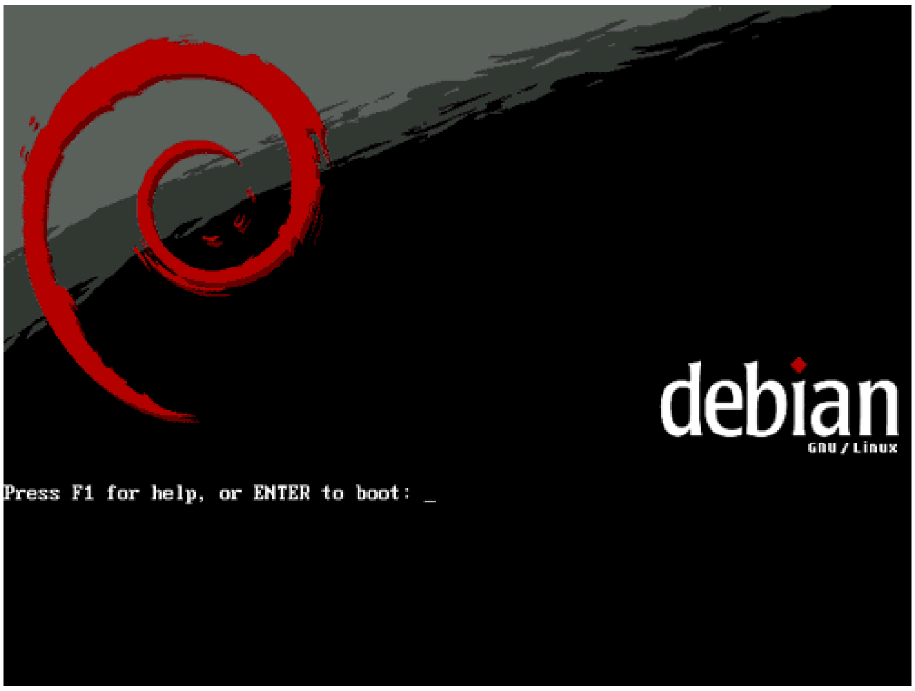
Irrespective of the application that we use, we will always have to **defrag the disk** beforehand. The disk defragger is a utility that analyses local disks, finding and consolidating fragmented files and folders. It can also defrag disks from a command line using the defrag command. This will avoid problems, since we could have fragmented files with part of them at the end of the partition. Therefore, when we resize and take space from this final part, we would delete these files and in the best of cases we would lose the information, but if they are system files, we could render the operating system inoperative.

Therefore, it is very important to defrag the disk before repartitioning, and likewise to make backup copies of all of the disk's information. Better safe than sorry.

5.3.2. Booting the installation system

Having reached this point, we are ready to start with the installation itself. To do this, we will boot the computer making sure that the first device read when it starts-up (boot) is the DVD-ROM unit (by entering the BIOS), and we will place DVD1 in it. In a few moments a welcome screen will appear as follows:

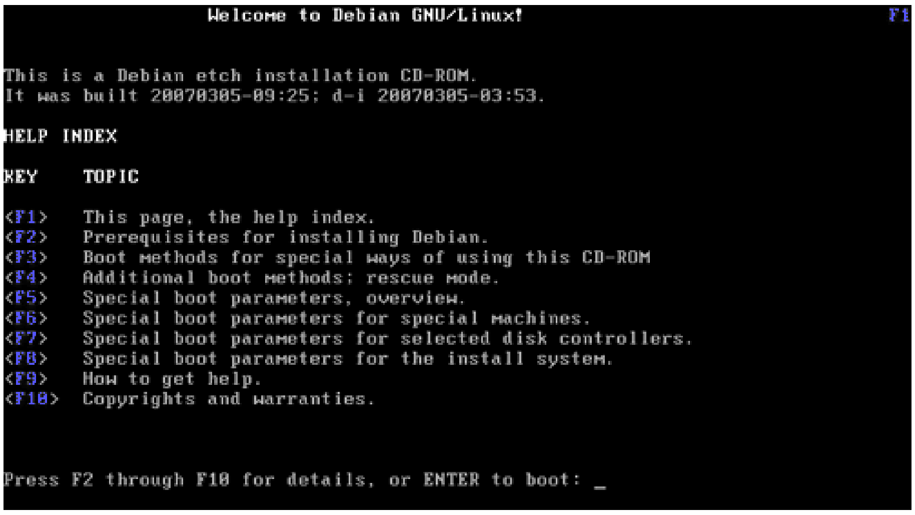
Figure 5.1



```
Press F1 for help, or ENTER to boot: _
```

If we press the F1 key, a screen like the following one will appear:

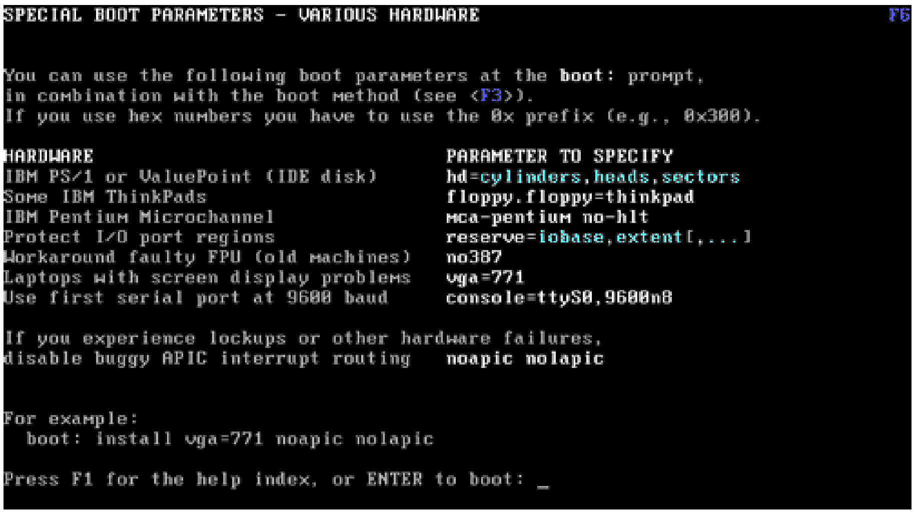
Figure 5.2



```
                    Welcome to Debian GNU/Linux!                        F1

This is a Debian etch installation CD-ROM.
It was built 20070305-09:25; d-i 20070305-03:53.

HELP INDEX

KEY     TOPIC

<F1>    This page, the help index.
<F2>    Prerequisites for installing Debian.
<F3>    Boot methods for special ways of using this CD-ROM
<F4>    Additional boot methods; rescue mode.
<F5>    Special boot parameters, overview.
<F6>    Special boot parameters for special machines.
<F7>    Special boot parameters for selected disk controllers.
<F8>    Special boot parameters for the install system.
<F9>    How to get help.
<F10>   Copyrights and warranties.


Press F2 through F10 for details, or ENTER to boot: _
```

We can press any of the options that the menu provides, if we press F6 for example, the following will appear:

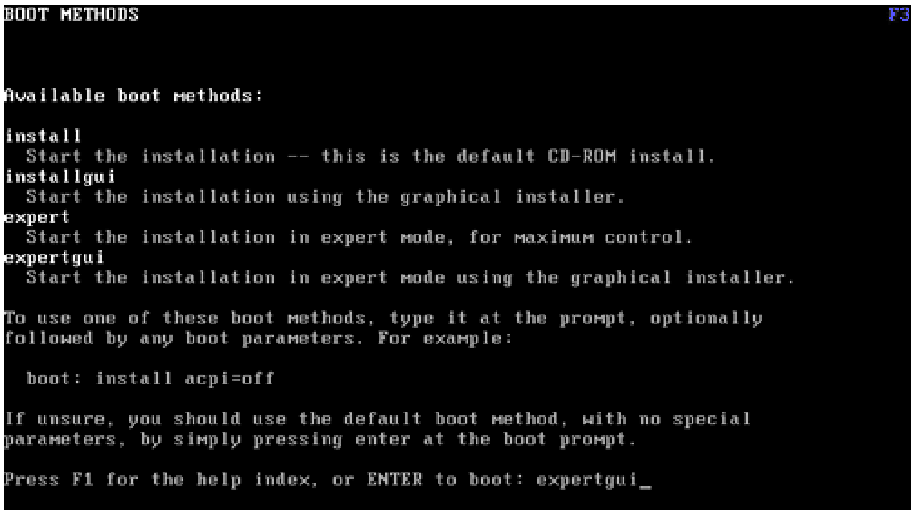
Figure 5.3



```
SPECIAL BOOT PARAMETERS - VARIOUS HARDWARE                                  F6

You can use the following boot parameters at the boot: prompt,
in combination with the boot method (see <F3>).
If you use hex numbers you have to use the 0x prefix (e.g., 0x300).

HARDWARE                                PARAMETER TO SPECIFY
IBM PS/1 or ValuePoint (IDE disk)       hd=cylinders,heads,sectors
Some IBM ThinkPads                      floppy.floppy=thinkpad
IBM Pentium Microchannel                mca-pentium no-hlt
Protect I/O port regions                reserve=iobase,extent[,...]
Workaround faulty FPU (old machines)    no387
Laptops with screen display problems    vga=771
Use first serial port at 9600 baud      console=ttyS0,9600n8

If you experience lockups or other hardware failures,
disable buggy APIC interrupt routing    noapic nolapic


For example:
  boot: install vga=771 noapic nolapic

Press F1 for the help index, or ENTER to boot: _
```

But the most interesting one is F3, which will offer us the different methods permitted for the installation. We can install using the default method, with the graphic installer, in expert mode for greater control or in expert mode with the graphic installer. So, let's key in `expertgui` for an installation in expert mode with the graphic installer:

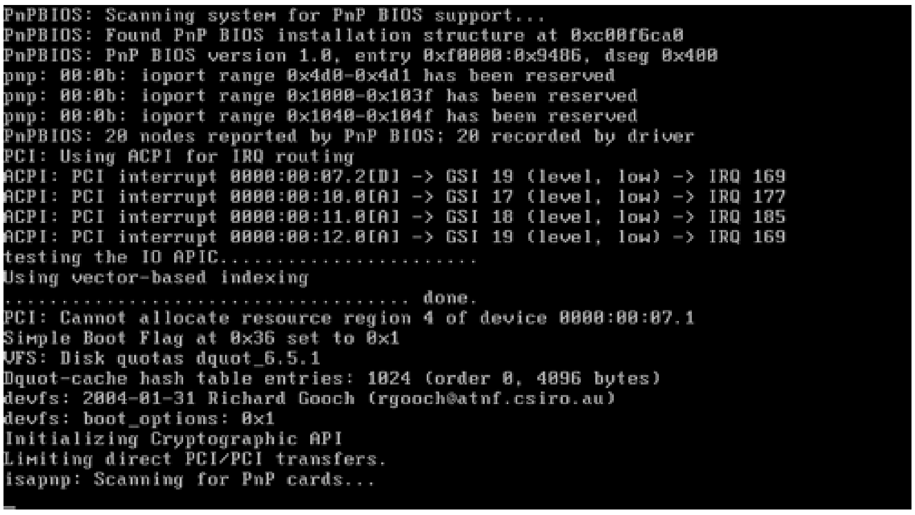
Figure 5.4



```
BOOT METHODS                                                                F3


Available boot methods:

install
  Start the installation -- this is the default CD-ROM install.
installgui
  Start the installation using the graphical installer.
expert
  Start the installation in expert mode, for maximum control.
expertgui
  Start the installation in expert mode using the graphical installer.

To use one of these boot methods, type it at the prompt, optionally
followed by any boot parameters. For example:

  boot: install acpi=off

If unsure, you should use the default boot method, with no special
parameters, by simply pressing enter at the boot prompt.

Press F1 for the help index, or ENTER to boot: expertgui_
```

and let's start with the installation:

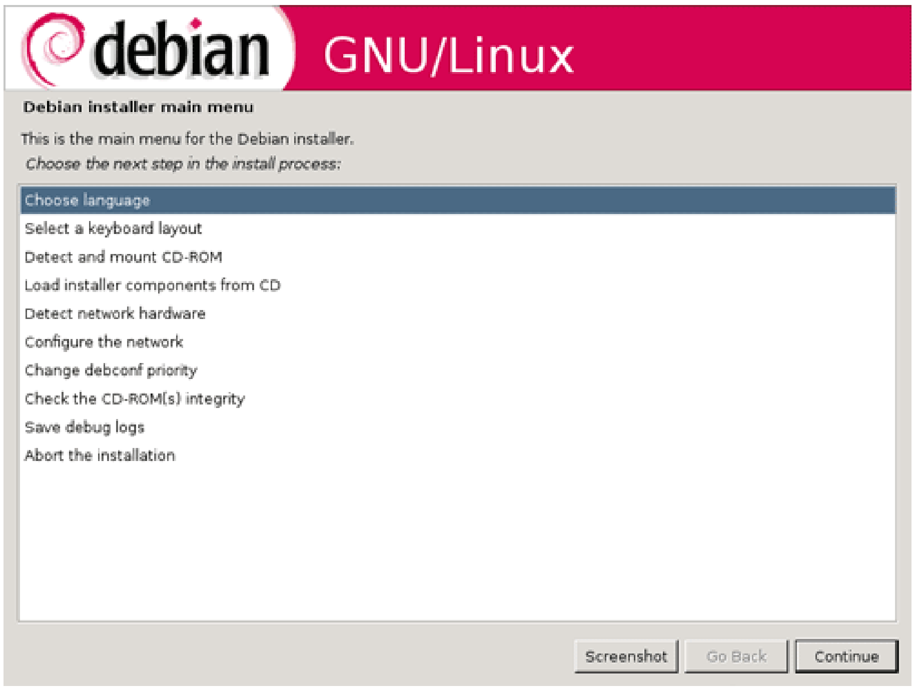
Figure 5.5



```
PnPBIOS: Scanning system for PnP BIOS support...
PnPBIOS: Found PnP BIOS installation structure at 0xc00f6ca0
PnPBIOS: PnP BIOS version 1.0, entry 0xf0000:0x9486, dseg 0x400
pnp: 00:0b: ioport range 0x4d0-0x4d1 has been reserved
pnp: 00:0b: ioport range 0x1000-0x103f has been reserved
pnp: 00:0b: ioport range 0x1040-0x104f has been reserved
PnPBIOS: 20 nodes reported by PnP BIOS; 20 recorded by driver
PCI: Using ACPI for IRQ routing
ACPI: PCI interrupt 0000:00:07.2[D] -> GSI 19 (level, low) -> IRQ 169
ACPI: PCI interrupt 0000:00:10.0[A] -> GSI 17 (level, low) -> IRQ 177
ACPI: PCI interrupt 0000:00:11.0[A] -> GSI 18 (level, low) -> IRQ 185
ACPI: PCI interrupt 0000:00:12.0[A] -> GSI 19 (level, low) -> IRQ 169
testing the IO APIC.......................
Using vector-based indexing
.................................... done.
PCI: Cannot allocate resource region 4 of device 0000:00:07.1
Simple Boot Flag at 0x36 set to 0x1
VFS: Disk quotas dquot_6.5.1
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
devfs: 2004-01-31 Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x1
Initializing Cryptographic API
Limiting direct PCI/PCI transfers.
isapnp: Scanning for PnP cards...
```

Having done this, the kernel will load (for a few instants we will see the outputs of the various tests carried out on screen) and immediately afterwards a Choose Language screen will appear so that we can select the installation language using the cursors.

Figure 5.6



Debian installer main menu

This is the main menu for the Debian installer.

Choose the next step in the install process:

As of now, by pressing the Alt+F2 key combination and Enter, we have a shell that, although very basic, can be used at any time during the installation.

On tty3 (Alt+F3) the kernel leaves its messages (in other words, the content of file `/var/log/messages`).

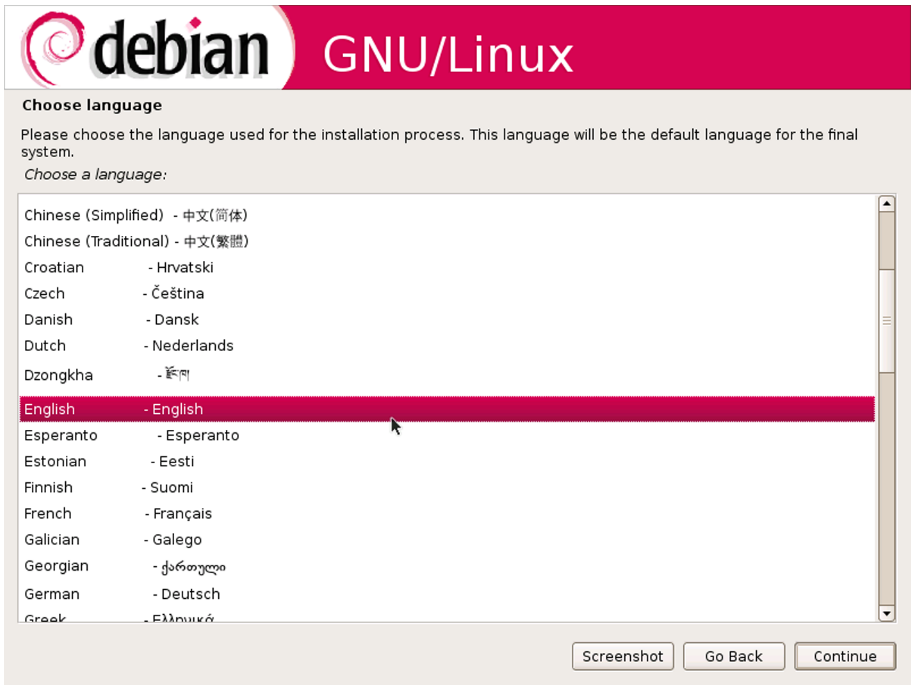
5.3.3. Installation language configuration

It is advisable to select the English option in the Choose language screen in order to avoid potential translation errors. In general, and where possible, it is always preferable to work with the original language; although, once again, readers are free to decide what they consider best; in this case, they can choose the language to be used during the installation.

Because this manual is documented in Spanish, it has followed the same philosophy and all the installation captures have been made in that language.

Now that we are here, we select the language and press Continue:

Figure 5.7



Choose language

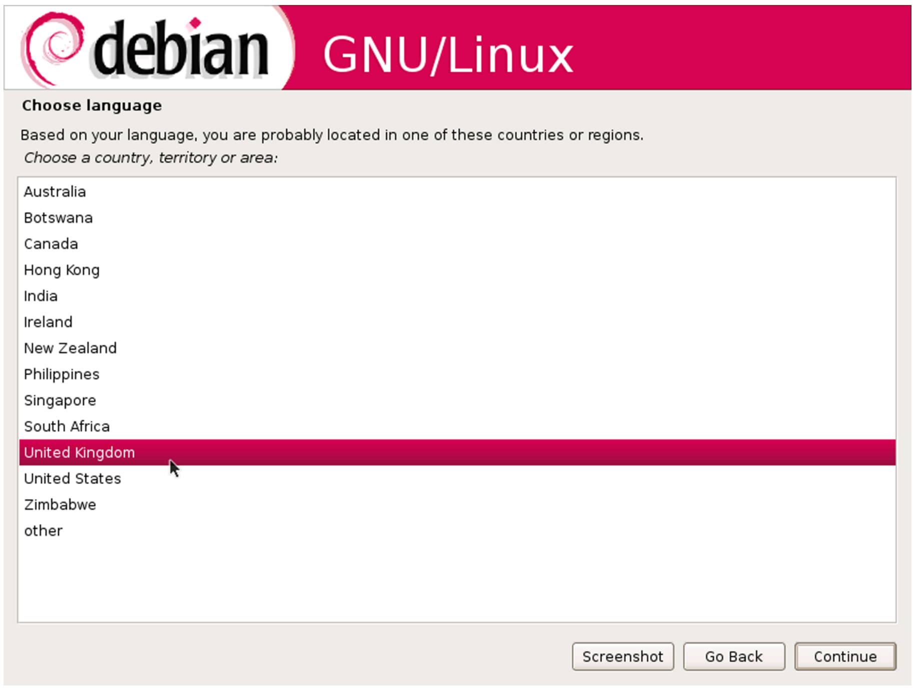
Please choose the language used for the installation process. This language will be the default language for the final system.

Choose a language:

Chinese (Simplified)	- 中文(简体)
Chinese (Traditional)	- 中文(繁體)
Croatian	- Hrvatski
Czech	- Čeština
Danish	- Dansk
Dutch	- Nederlands
Dzongkha	- རྫོང་ཁ
English	- English
Esperanto	- Esperanto
Estonian	- Eesti
Finnish	- Suomi
French	- Français
Galician	- Galego
Georgian	- ქართული
German	- Deutsch
Greek	- Ελληνικά

On the Choose a country, territory or area screen, as with the language, readers are free to decide what to choose. In this case, we have chosen Spain:

Figure 5.8



Choose language

Based on your language, you are probably located in one of these countries or regions.

Choose a country, territory or area:

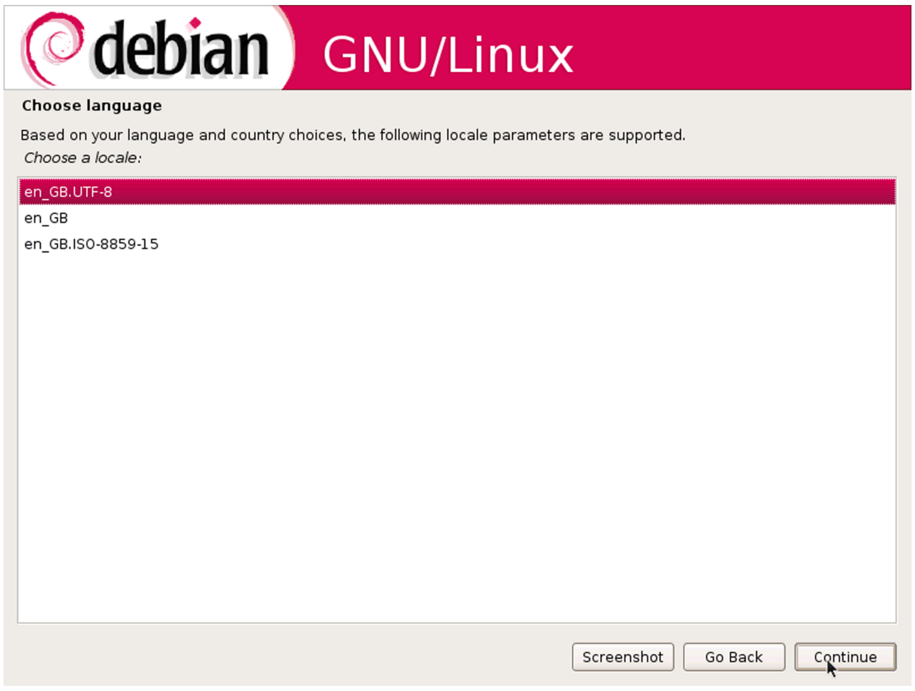
Australia
Botswana
Canada
Hong Kong
India
Ireland
New Zealand
Philippines
Singapore
South Africa
United Kingdom
United States
Zimbabwe
other

Language configuration

The installation language does not condition the keyboard configuration or the language in which we interact with the system or other language configurations. This option is only valid during the installation process.

Next, depending on the language and selected country, we are offered several localisation parameters and we choose the one that supports UTF-8:

Figure 5.9



Choose language

Based on your language and country choices, the following locale parameters are supported.

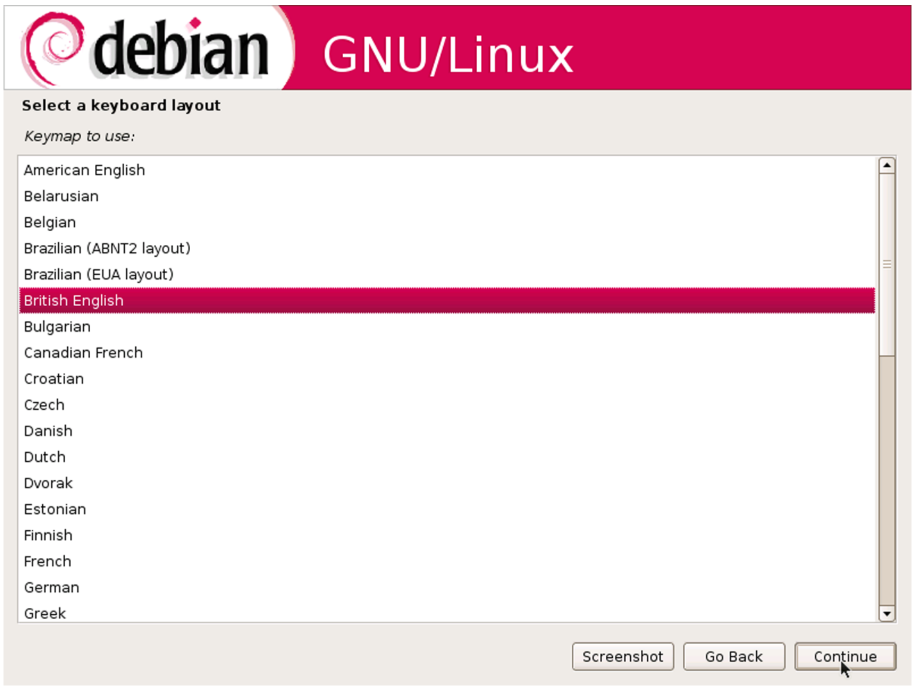
Choose a locale:

en_GB.UTF-8
en_GB
en_GB.ISO-8859-15

5.3.4. Keyboard configuration

As a first suggestion, the installation's interface will prompt us to configure the keyboard on the Choose Keyboard Layout screen and we will select the keyboard layout that we wish to use:

Figure 5.10



Select a keyboard layout

Keymap to use:

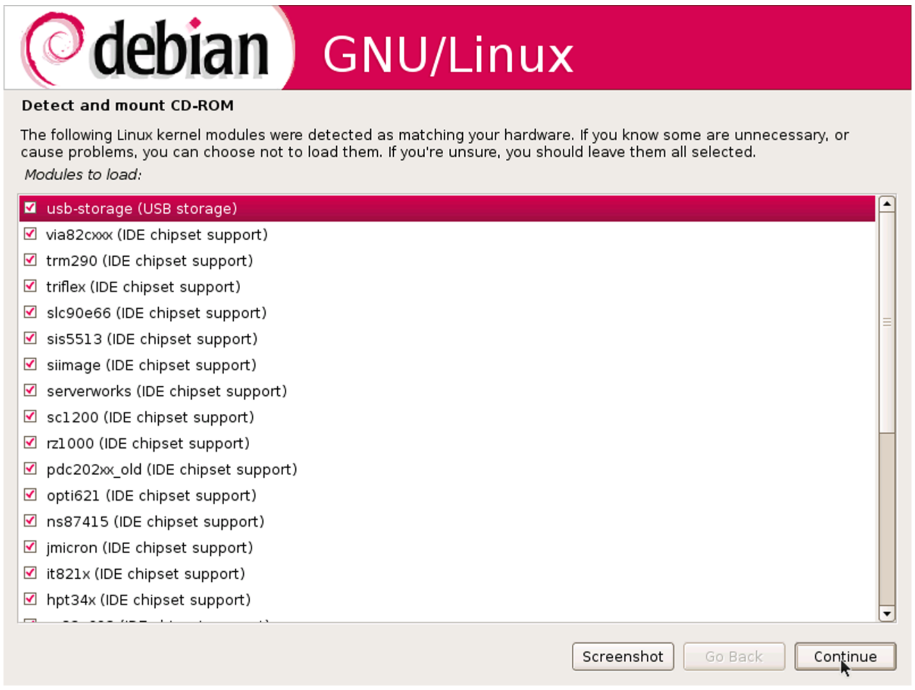
American English
Belarusian
Belgian
Brazilian (ABNT2 layout)
Brazilian (EUA layout)
British English
Bulgarian
Canadian French
Croatian
Czech
Danish
Dutch
Dvorak
Estonian
Finnish
French
German
Greek

We will press Continue so that the input device responds to the Spanish keyboard's mapping of characters. On tty2 we can verify that indeed we have accents, for example.

5.3.5. Detecting and mounting the CD-ROM

The next step is to detect and mount the CD-ROM (DVD-ROM). We are told that it has been detected and are advised to select all the modules that we are going to load:

Figure 5.11



Detect and mount CD-ROM

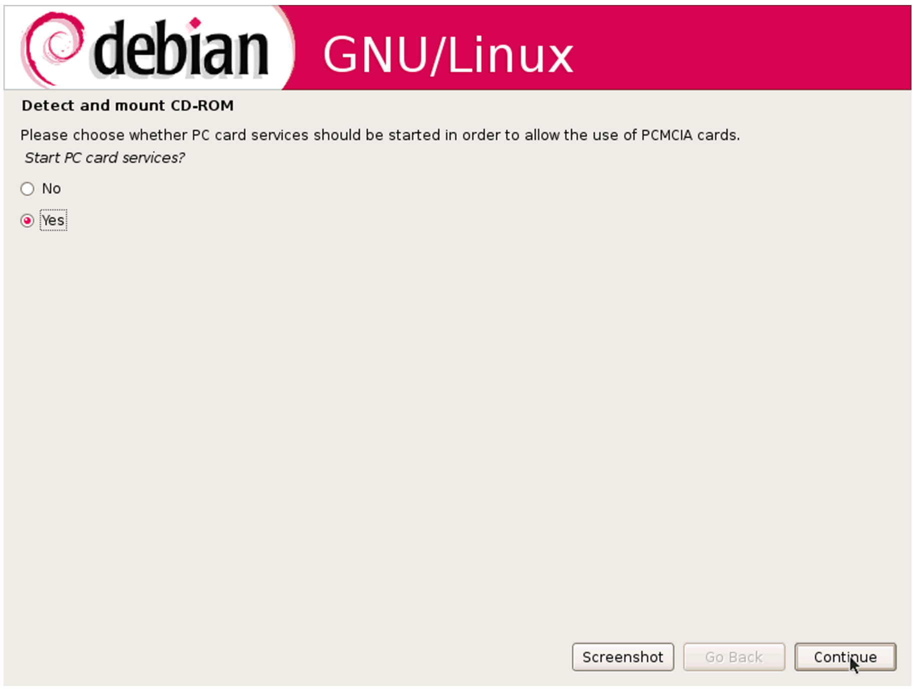
The following Linux kernel modules were detected as matching your hardware. If you know some are unnecessary, or cause problems, you can choose not to load them. If you're unsure, you should leave them all selected.

Modules to load:

- ✓ usb-storage (USB storage)
- ✓ via82cxxx (IDE chipset support)
- ✓ trm290 (IDE chipset support)
- ✓ triflex (IDE chipset support)
- ✓ slc90e66 (IDE chipset support)
- ✓ sis5513 (IDE chipset support)
- ✓ siimage (IDE chipset support)
- ✓ serverworks (IDE chipset support)
- ✓ sc1200 (IDE chipset support)
- ✓ rz1000 (IDE chipset support)
- ✓ pdc202xx_old (IDE chipset support)
- ✓ opti621 (IDE chipset support)
- ✓ ns87415 (IDE chipset support)
- ✓ jmicron (IDE chipset support)
- ✓ it821x (IDE chipset support)
- ✓ hpt34x (IDE chipset support)

And, next, if we are carrying out the installation on a laptop computer, we will be asked if we wish to initiate the PC card services to use PCMCIA cards, to which we will answer selecting Yes:

Figure 5.12



Detect and mount CD-ROM

Please choose whether PC card services should be started in order to allow the use of PCMCIA cards.

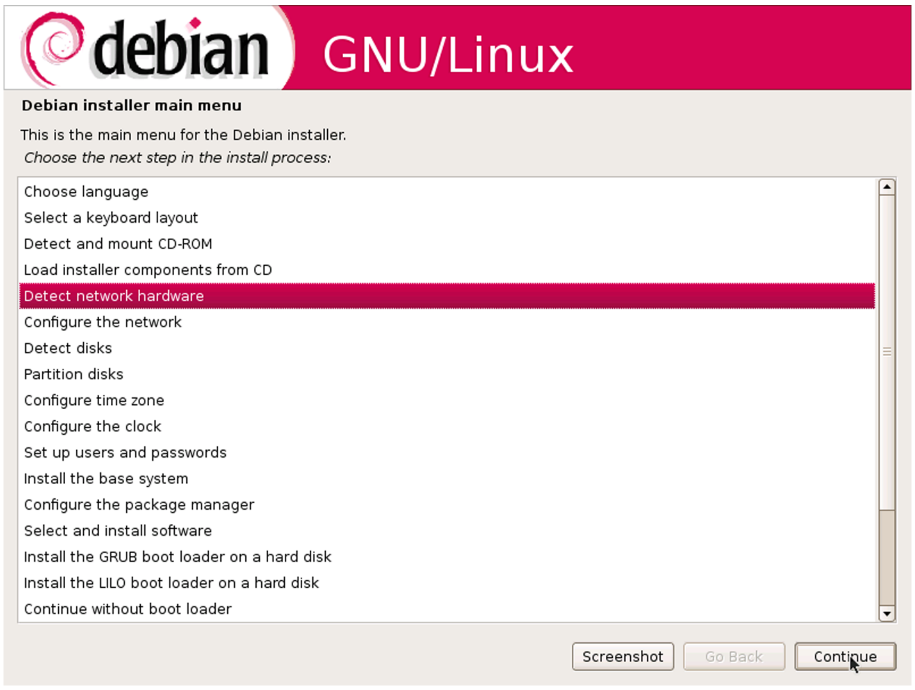
Start PC card services?

No [✓ Yes]

5.3.6. Network configuration

The next step of the installation process that we will carry out is to Detect network hardware:

Figure 5.13



Debian installer main menu

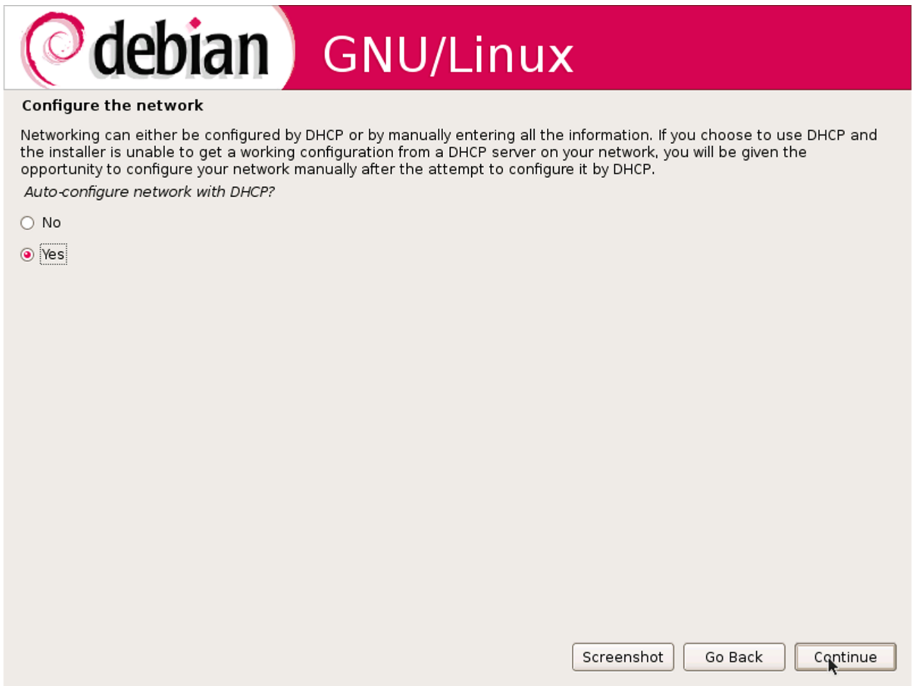
This is the main menu for the Debian installer.

Choose the next step in the install process:

- Choose language
- Select a keyboard layout
- Detect and mount CD-ROM
- Load installer components from CD
- Detect network hardware
- Configure the network
- Detect disks
- Partition disks
- Configure time zone
- Configure the clock
- Set up users and passwords
- Install the base system
- Configure the package manager
- Select and install software
- Install the GRUB boot loader on a hard disk
- Install the LILO boot loader on a hard disk
- Continue without boot loader

We can configure the network by DHCP or manually. In this case, we will select the option to configure the network by DHCP.

Figure 5.14



Configure the network

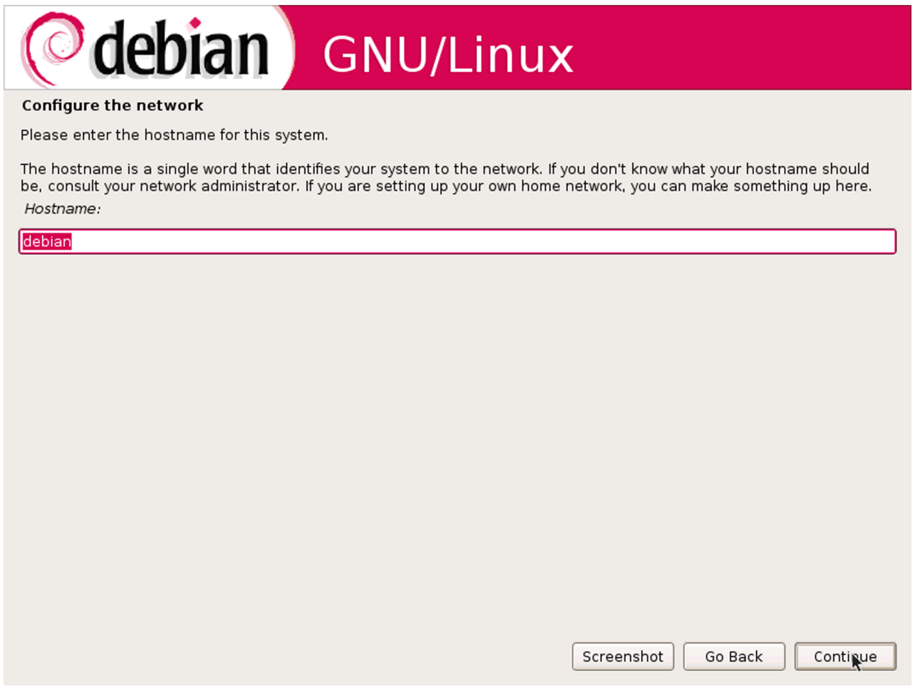
Networking can either be configured by DHCP or by manually entering all the information. If you choose to use DHCP and the installer is unable to get a working configuration from a DHCP server on your network, you will be given the opportunity to configure your network manually after the attempt to configure it by DHCP.

Auto-configure network with DHCP?

No [✓ Yes]

Once we have configured the network, it is time to name the system that we are installing, we are asked for the Name of the machine, which by default it suggests calling debian. We can delete this suggestion and give it a name that we prefer, consisting of just one word.

Figure 5.15



Configure the network

Please enter the hostname for this system.

The hostname is a single word that identifies your system to the network. If you don't know what your hostname should be, consult your network administrator. If you are setting up your own home network, you can make something up here.

Hostname:

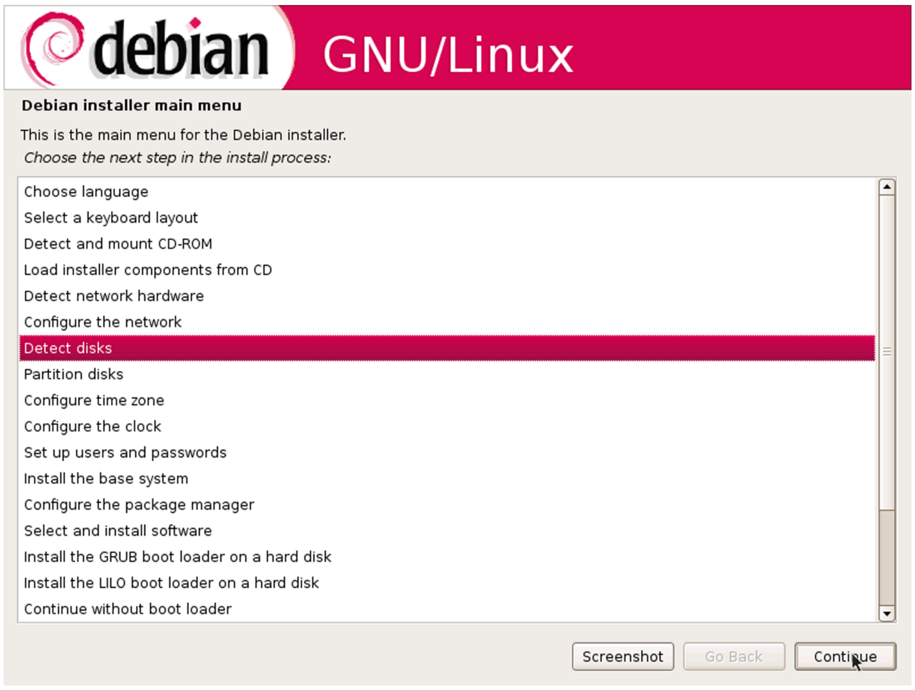
debian

Now, all we have to do is specify the system's local name. If it is going to form part of a network, and therefore, needs a domain, this will be specified on the following screen.

5.3.7. Hard disk partition

Once we have specified the name of the machine and its domain (where necessary) the installation process will suggest that we detect disks and then partition the disks.

Figure 5.16



Debian installer main menu

This is the main menu for the Debian installer.

Choose the next step in the install process:

```
Choose language
Select a keyboard layout
Detect and mount CD-ROM
Load installer components from CD
Detect network hardware
Configure the network
Detect disks
Partition disks
Configure time zone
Configure the clock
Set up users and passwords
Install the base system
Configure the package manager
Select and install software
Install the GRUB boot loader on a hard disk
Install the LILO boot loader on a hard disk
Continue without boot loader
```

When it comes to partitioning the disks, the size, characteristics, and number of partitions will depend to a great extent on the type of use and available amount of the hard disk space. Since this is an installation for educational purposes, information is given next on the partitions to be created assuming that we are working on a space of between five and fifteen gigabytes designed for the new installation.

We need to create two partitions at least: a first one for mounting the system and a second one as swap (it is true that we can perform installations without the swap, but, as already mentioned, it is highly inadvisable). To increase the system's efficiency, we will create six partitions.

The tool that Debian provides for disk partitioning is `cfdisk` and it functions based on using the cursors to move through the disk partitions (upper part of the screen, with the Up and Down cursors), as well as to select the potential operations that can be carried out at each time (lower part of the screen, with Left and Right), since these will vary according to the status of the selected partition. To create a new partition, we need to select the [New] option; next, if we can still create primary partitions (the maximum number is four), we will be asked whether we wish to create it as a primary or a logical partition; then we need to specify the size of the partition, and finally, its physical location on the disk (it is advisable to make a sketch of how we want to do this before starting to partition the disk, and to create the partitions on this basis, so that we can always answer this question with [Beginning]).

The first partition is the one designed to host the root (/); it does not have to be very big and for this reason we will allocate it at least ten percent of the hard disk, preferably in a primary partition, but if we do not have it, we can create it as a logical partition without it mattering. We will specify that the partition goes at the beginning of the free space and that the type of file system chosen for the partition is ext3.

The second will be used for the swap partition (swap). It should have at least the same size as the RAM, 512Mb, 1024Mb etc., with this size we will make sure, save for exceptional cases, that it will never become saturated; it is also preferable for this to be a primary partition, but if it has to be logical, it will not affect the system's performance either. If we have more than one GNU/Linux installation on the same computer, we can use the same swap partition for all of them, since the information that can be stored on it during the system's functioning is totally volatile. The file system for the swap partition will be of the swap type (swap area).

The third partition will be for the usr directory (`/usr`); we need to bear in mind that this partition will host most of the software to be installed, meaning that its size will have to be significant, approximately forty percent of the disk. Its file system will be ext3.

The fourth partition will be used for the var directory (`/var`), where libraries, log files etc., are kept. As in the previous case, it will also be ext3.

The fifth partition we will use for users' home directories (`/home`), designed to store users' data, and depending on the size of the hard disk, we can allocate between ten and twenty percent of the hard disk to it, depending on the number of users and use to be made of the system. This partition will also be ext3.

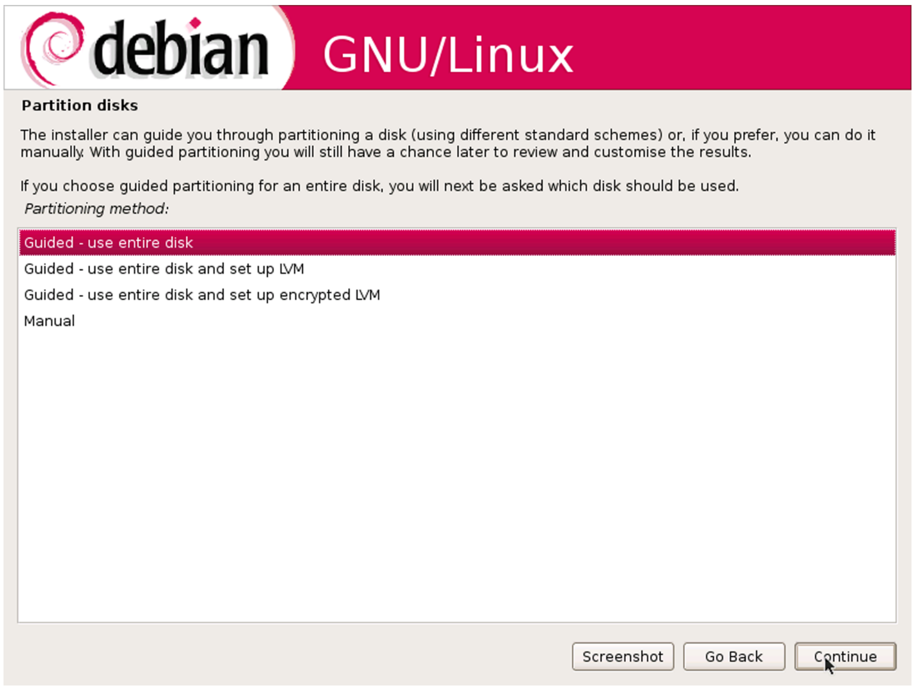
The remaining space, the sixth partition, will be used for the var directory (`/tmp`) and its file system will also be ext3.

The distribution of partitions above is merely a suggestion, with two objectives: on the one hand, to improve the performance in relation to an installation based on just one or two partitions, and on the other hand to make the system more robust. Among other advantages, having the data distributed over different partitions means that the corruption of one of them does not automatically imply the loss of all the system's information. Obviously, we can create other partitions or omit some of the ones we have proposed (the order of the partitions does not affect the system's behaviour).

Therefore, having detected the disks, we can select the partition method, which can be either guided or manual. If it is manual, we can create the partitions we wish, with the size and type of file system, as we have explained above, for example.

And if we select the guided option, it will suggest the partitioning system for us. Because this is our first installation, we will select the guided option.

Figure 5.17



Partition disks

The installer can guide you through partitioning a disk (using different standard schemes) or, if you prefer, you can do it manually. With guided partitioning you will still have a chance later to review and customise the results.

If you choose guided partitioning for an entire disk, you will next be asked which disk should be used.

Partitioning method:

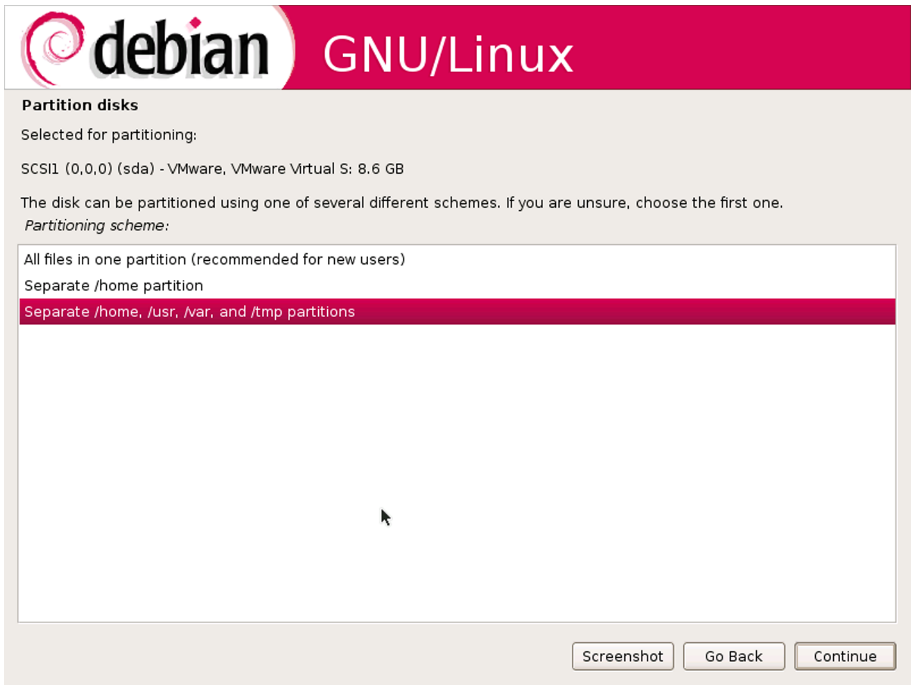
Guided - use entire disk
Guided - use entire disk and set up LVM
Guided - use entire disk and set up encrypted LVM
Manual

Next, we will be shown the choice of devices to be partitioned. If we have more than one hard disk, we can use the cursors to select which one we wish to operate on. Having made the selection, we press Continue.

Debian suggests various partitioning schemes: all the files on the same partition, adding a partition for user data, or separating the main partitions. As we have seen beforehand, it is not advisable to do the installation on the same partition.

Since we have an option that allows us to separate the partitions automatically `/home`, `/usr/`, `/var` and `/tmp`, we will choose this option for our first installation of a Debian GNU/Linux system.

Figure 5.18



Partition disks

Selected for partitioning:

SCSI1 (0,0,0) (sda) - VMware, VMware Virtual S: 8.6 GB

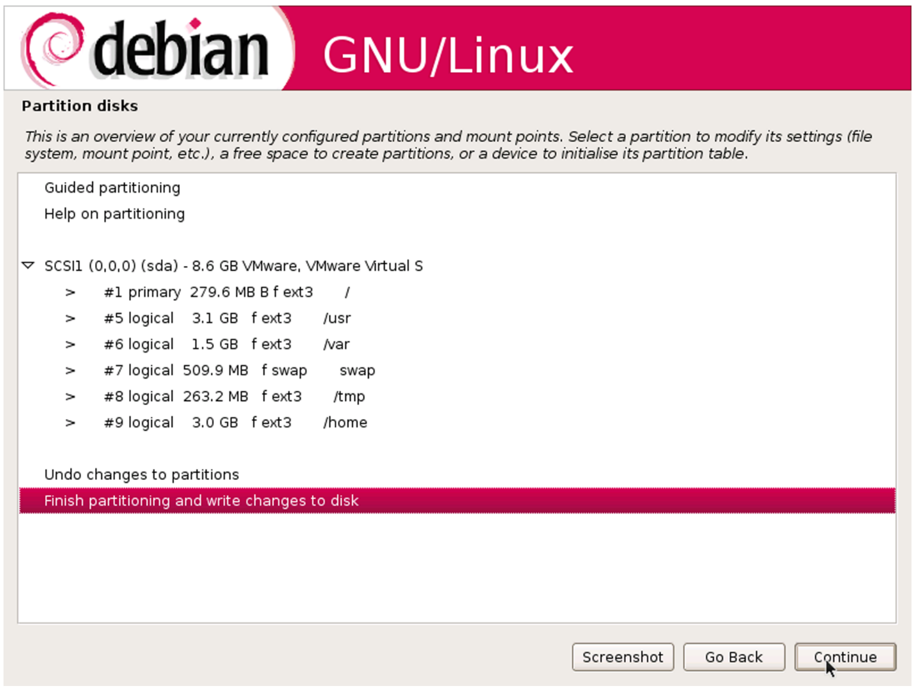
The disk can be partitioned using one of several different schemes. If you are unsure, choose the first one.

Partitioning scheme:

All files in one partition (recommended for new users)
Separate /home partition
Separate /home, /usr, /var, and /tmp partitions

So we will follow the previous partitioning scheme, obtaining the following partitions table:

Figure 5.19



Partition disks

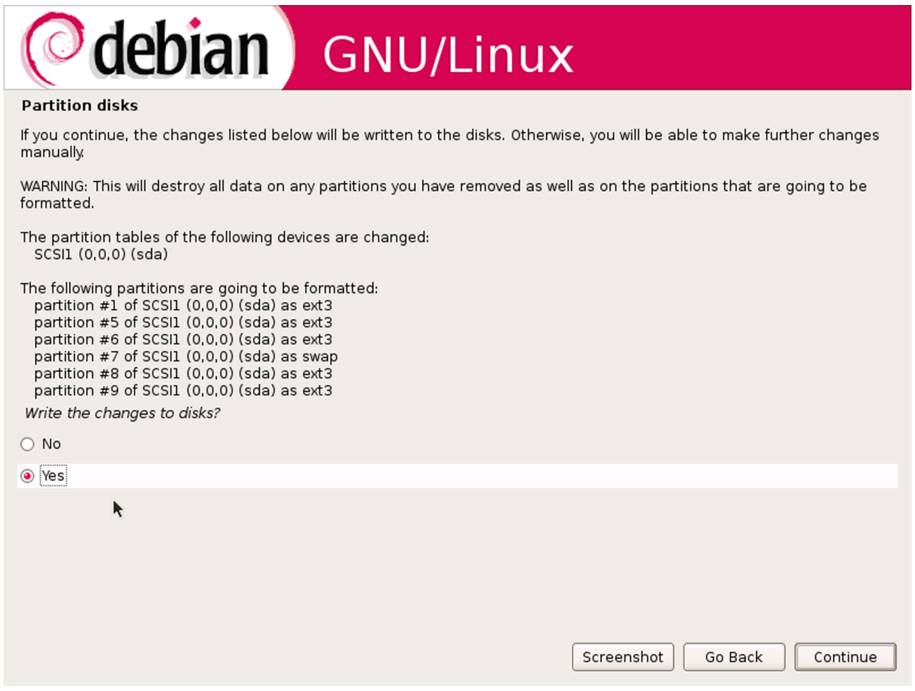
This is an overview of your currently configured partitions and mount points. Select a partition to modify its settings (file system, mount point, etc.), a free space to create partitions, or a device to initialise its partition table.

```
Guided partitioning
Help on partitioning

SCSI1 (0,0,0) (sda) - 8.6 GB VMware, VMware Virtual S
    >   #1 primary  279.6 MB B f ext3     /
    >   #5 logical    3.1 GB   f ext3     /usr
    >   #6 logical    1.5 GB   f ext3     /var
    >   #7 logical  509.9 MB   f swap     swap
    >   #8 logical  263.2 MB   f ext3     /tmp
    >   #9 logical    3.0 GB   f ext3     /home

Undo changes to partitions
Finish partitioning and write changes to disk
```

As we can see, this is the proposed partitioning that the application automatically suggests. If we wish to modify a partition (its size, position or filesystem), to add or even delete any of the proposed partitions, we can select the partition in question and make the relevant changes, or go to the Undo changes to partitions option and repartition.

If, however, we wish to apply this proposed partitioning, we just need to select the Finish partitioning and write the changes to disk option and Continue. Now this information will be written on the MBR and the changes will become effective. We quit the `cfdisk` application and continue with the installation.

Figure 5.20



Partition disks

If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.

WARNING: This will destroy all data on any partitions you have removed as well as on the partitions that are going to be formatted.

The partition tables of the following devices are changed:
SCSI1 (0,0,0) (sda)

The following partitions are going to be formatted:
partition #1 of SCSI1 (0,0,0) (sda) as ext3
partition #5 of SCSI1 (0,0,0) (sda) as ext3
partition #6 of SCSI1 (0,0,0) (sda) as ext3
partition #7 of SCSI1 (0,0,0) (sda) as swap
partition #8 of SCSI1 (0,0,0) (sda) as ext3
partition #9 of SCSI1 (0,0,0) (sda) as ext3

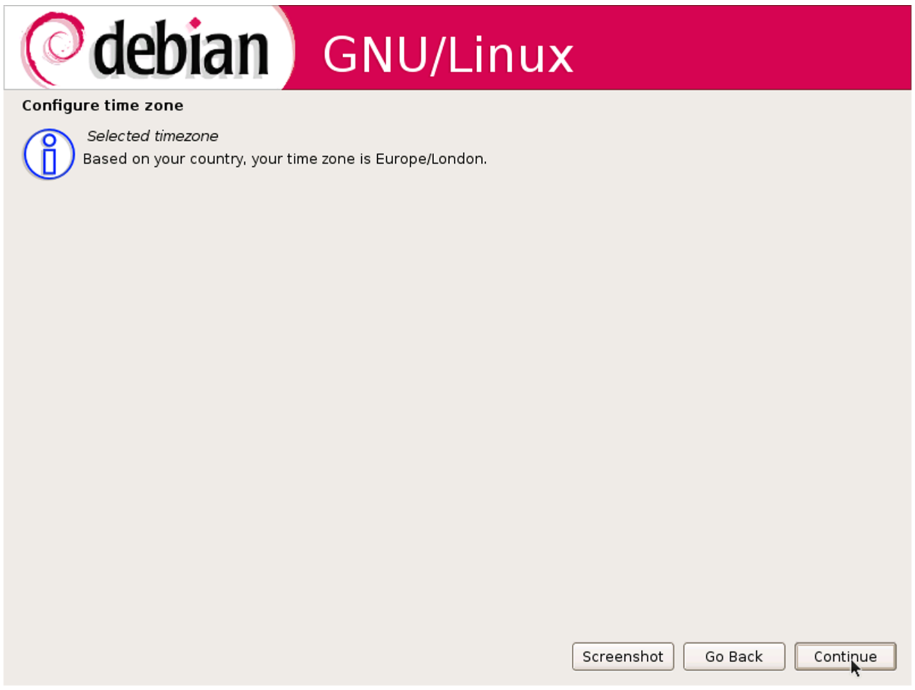
Write the changes to disks?

No [✓ Yes]

5.3.8. Time configuration

Immediately after partitioning the hard disk, we enter the system's time configuration.

Figure 5.21

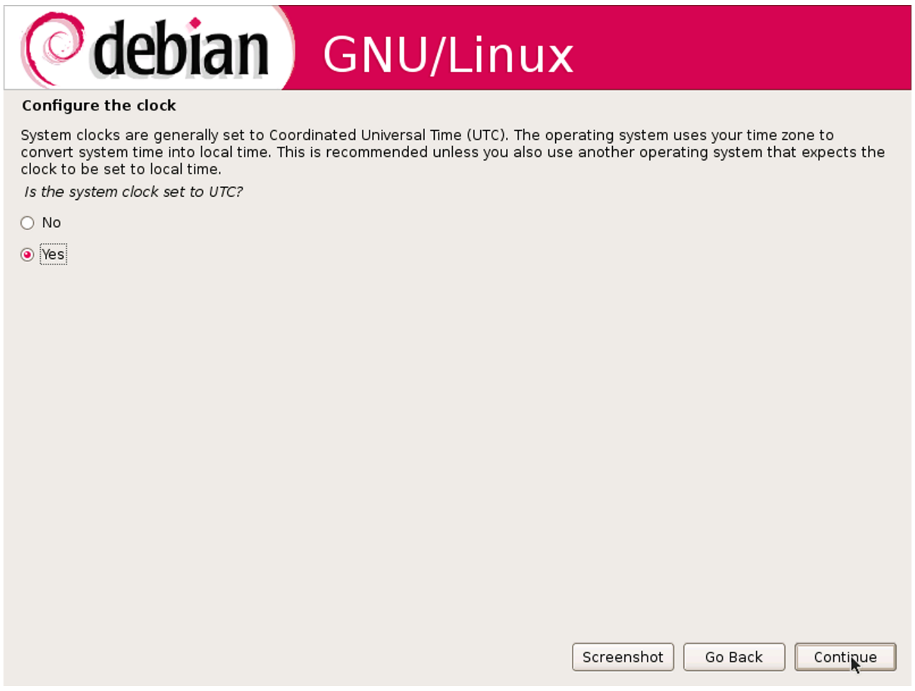


Configure time zone

Selected timezone
Based on your country, your time zone is Europe/London.

Next, we select UTC time for the clock, universal time coordinated.

Figure 5.22



Configure the clock

System clocks are generally set to Coordinated Universal Time (UTC). The operating system uses your time zone to convert system time into local time. This is recommended unless you also use another operating system that expects the clock to be set to local time.

Is the system clock set to UTC?

No [✓ Yes]

Bibliography

For more information on universal time coordinated, visit the "Universal Time Coordinated" entry in Wikipedia.

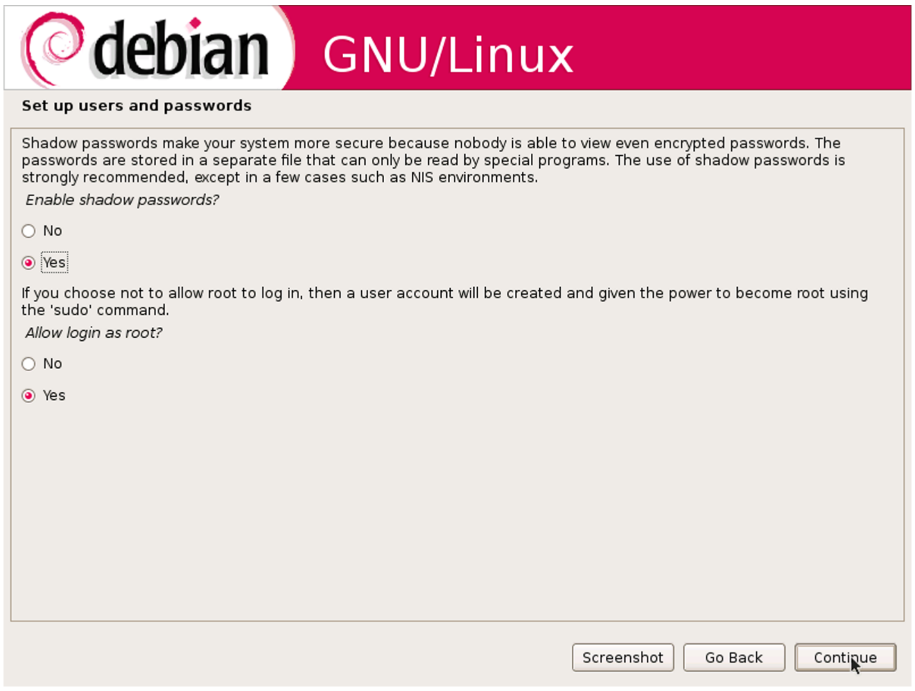
5.3.9. Configuring users and passwords

Once we have completed the procedures for configuring the system's time, it is time to configure users and passwords.

First of all, the system will ask us if we want to enable the encrypted passwords option, so that only the root user has access to the file containing the passwords (`/etc/shadow`). This represents an additional security measure for our system.

The second question is to decide whether we will allow the root user access to the system or whether we want to create a user with permissions to be the administrator using the `sudo` command. Here, we will leave the reader to decide how he wants to work when carrying out administration tasks; in this first installation, we will allow the root user access to the system.

Figure 5.23



Set up users and passwords

Shadow passwords make your system more secure because nobody is able to view even encrypted passwords. The passwords are stored in a separate file that can only be read by special programs. The use of shadow passwords is strongly recommended, except in a few cases such as NIS environments.

Enable shadow passwords?

No [✓ Yes]

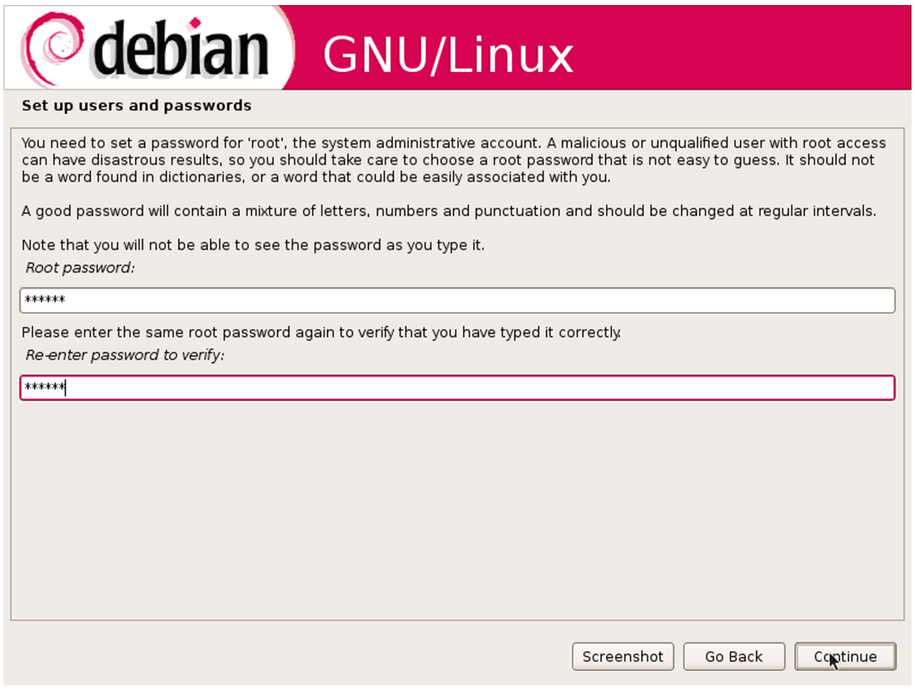
If you choose not to allow root to log in, then a user account will be created and given the power to become root using the 'sudo' command.

Allow login as root?

No [✓ Yes]

Now we need to write the password of the root. It is important to follow the recommendations in selecting this password and to remember it. In order to confirm that the required password has been entered and avoid possible typing errors, we are asked to retype the password as confirmation.

Figure 5.24



Set up users and passwords

You need to set a password for 'root', the system administrative account. A malicious or unqualified user with root access can have disastrous results, so you should take care to choose a root password that is not easy to guess. It should not be a word found in dictionaries, or a word that could be easily associated with you.

A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.

Note that you will not be able to see the password as you type it.

Root password:

******

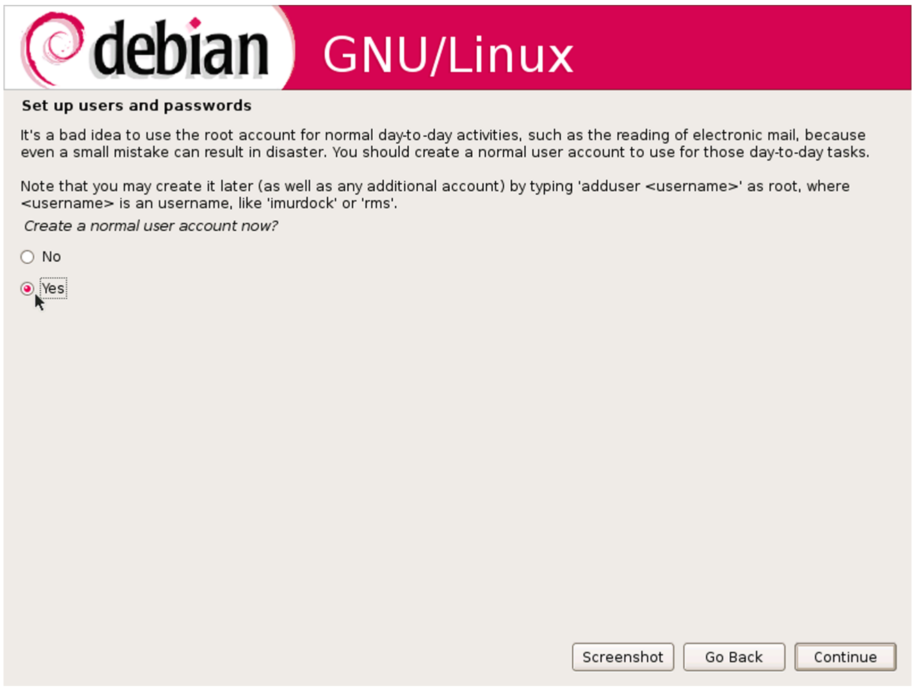
Please enter the same root password again to verify that you have typed it correctly.

Re-enter password to verify:

******

As mentioned, always working as home is bad policy for various reasons. Therefore, the system advises us to create a normal user account.

Figure 5.25



Set up users and passwords

It's a bad idea to use the root account for normal day-to-day activities, such as the reading of electronic mail, because even a small mistake can result in disaster. You should create a normal user account to use for those day-to-day tasks.

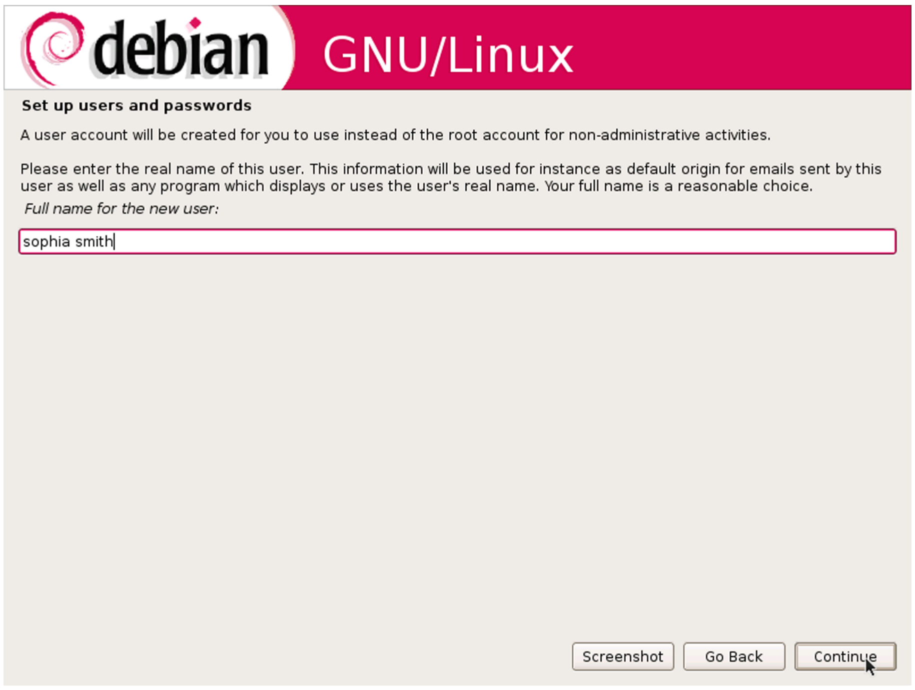
Note that you may create it later (as well as any additional account) by typing 'adduser <username>' as root, where <username> is an username, like 'imurdock' or 'rms'.

Create a normal user account now?

No [✓ Yes]

It will ask us for the full name of the new user:

Figure 5.26



Set up users and passwords

A user account will be created for you to use instead of the root account for non-administrative activities.

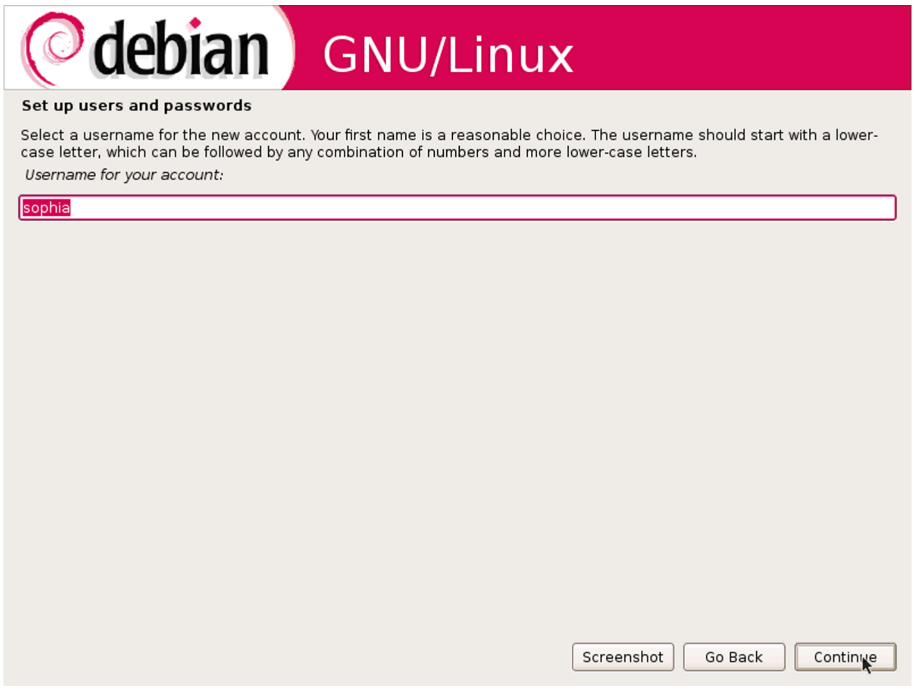
Please enter the real name of this user. This information will be used for instance as default origin for emails sent by this user as well as any program which displays or uses the user's real name. Your full name is a reasonable choice.

Full name for the new user:

sophia smith

and next, for the name of this account, in other words, the user to access the system:

Figure 5.27



Set up users and passwords

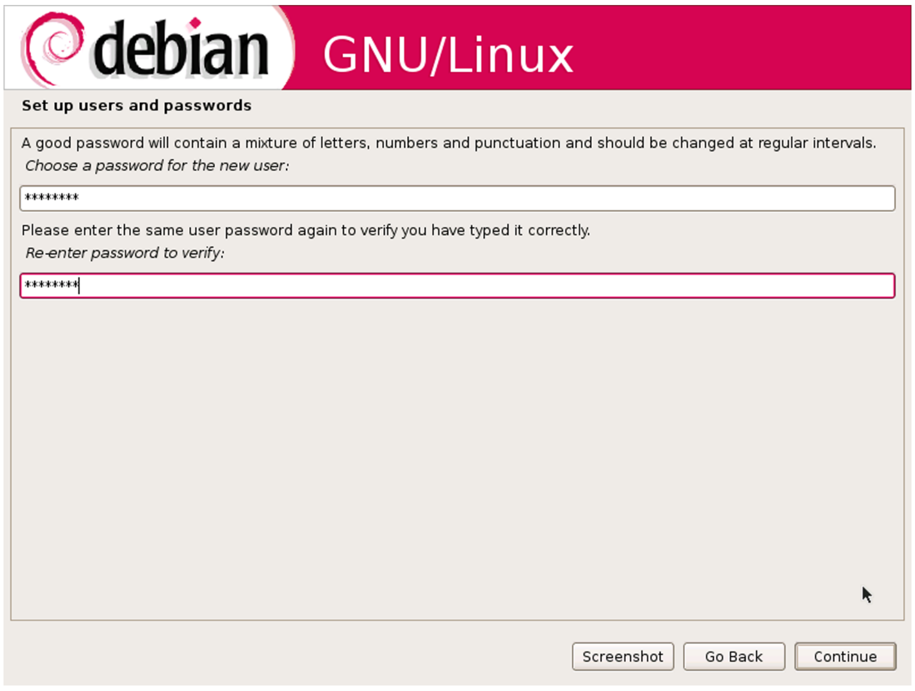
Select a username for the new account. Your first name is a reasonable choice. The username should start with a lower-case letter, which can be followed by any combination of numbers and more lower-case letters.

Username for your account:

sophia

and, finally, for the user's password, which for the same reason as before, needs to be written twice:

Figure 5.28



Set up users and passwords

A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.
Choose a password for the new user:

****

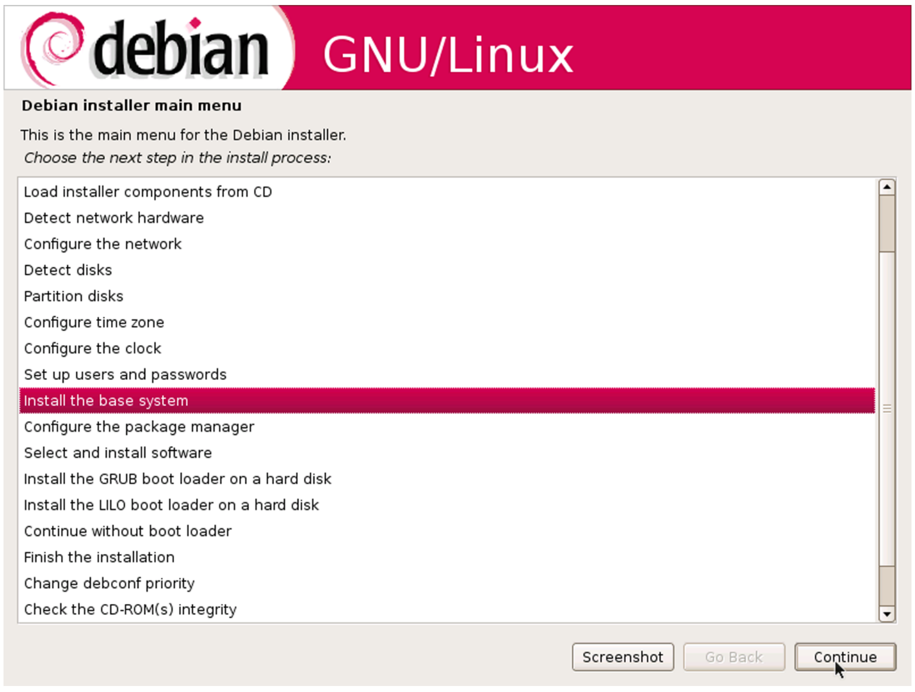
Please enter the same user password again to verify you have typed it correctly.
Re-enter password to verify:

****

5.3.10. Base system installation

With all of this information, the system starts installing the base system:

Figure 5.29



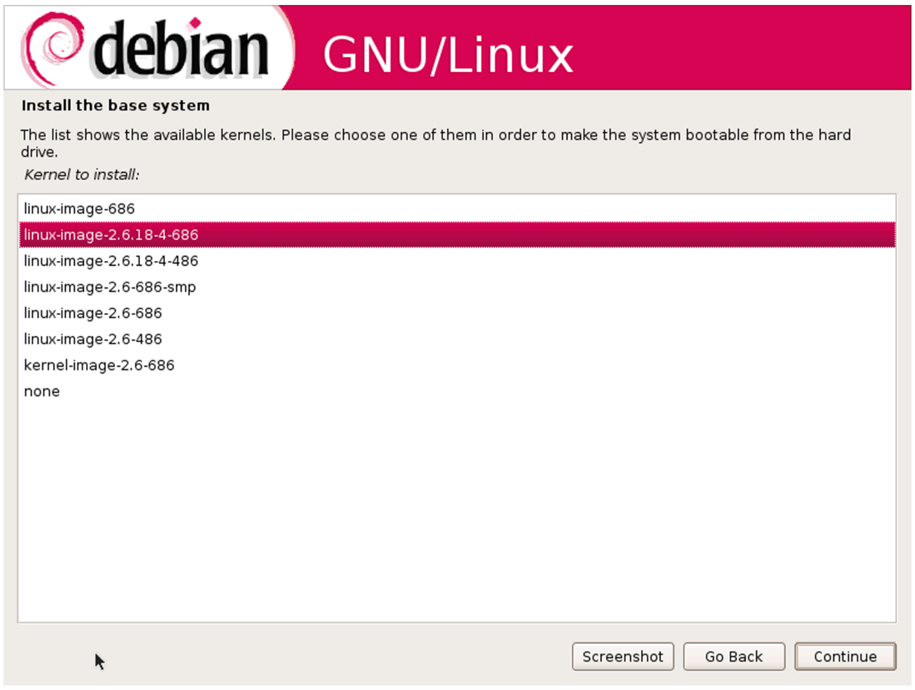
Debian installer main menu

This is the main menu for the Debian installer.
Choose the next step in the install process:

Load installer components from CD
Detect network hardware
Configure the network
Detect disks
Partition disks
Configure time zone
Configure the clock
Set up users and passwords
Install the base system
Configure the package manager
Select and install software
Install the GRUB boot loader on a hard disk
Install the LILO boot loader on a hard disk
Continue without boot loader
Finish the installation
Change debconf priority
Check the CD-ROM(s) integrity
Screenshot
Go Back
Continue

The first thing it will ask us for is the kernel we would like to install. As we can see from the following image, we can choose between four kernels. We will select the first option:

Figure 5.30



Install the base system

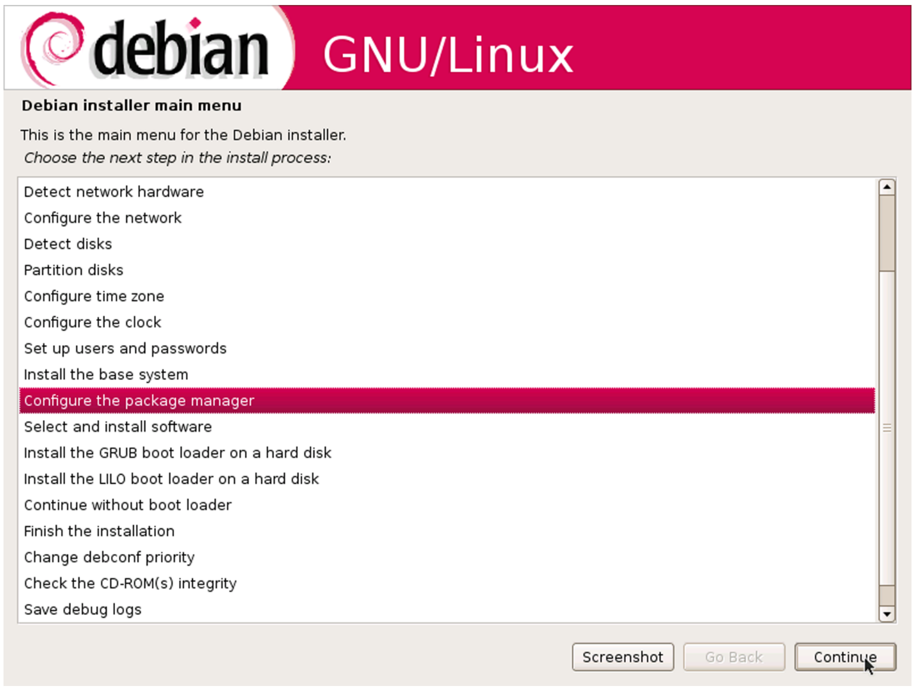
The list shows the available kernels. Please choose one of them in order to make the system bootable from the hard drive.

Kernel to install:

linux-image-686
linux-image-2.6.18-4-686
linux-image-2.6.18-4-486
linux-image-2.6-686-smp
linux-image-2.6-686
linux-image-2.6-486
kernel-image-2.6-686
none

5.3.11. Configuring the package manager

Figure 5.31



Debian installer main menu

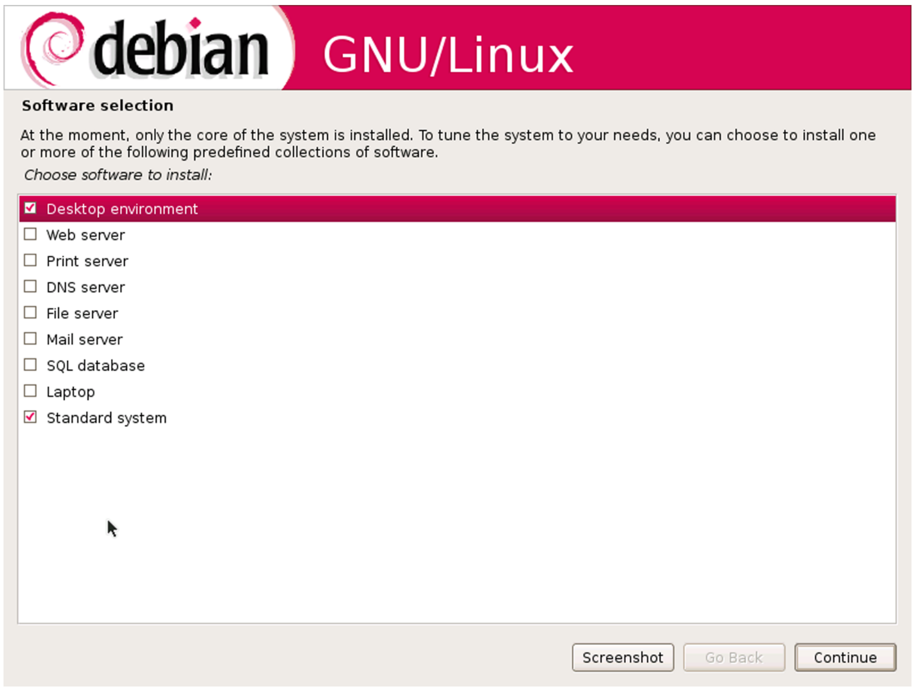
This is the main menu for the Debian installer.

Choose the next step in the install process:

Detect network hardware
Configure the network
Detect disks
Partition disks
Configure time zone
Configure the clock
Set up users and passwords
Install the base system
Configure the package manager
Select and install software
Install the GRUB boot loader on a hard disk
Install the LILO boot loader on a hard disk
Continue without boot loader
Finish the installation
Change debconf priority
Check the CD-ROM(s) integrity
Save debug logs

5.3.12. Selecting and installing programs

Figure 5.32



Software selection

At the moment, only the core of the system is installed. To tune the system to your needs, you can choose to install one or more of the following predefined collections of software.

Choose software to install:

- [✓] Desktop environment
- [ ] Web server
- [ ] Print server
- [ ] DNS server
- [ ] File server
- [ ] Mail server
- [ ] SQL database
- [ ] Laptop
- [✓] Standard system

5.3.13. GRUB installation

Next, the package starts installing `GRUB boot loader`. It analyses our hard disk and looks for all installed operating systems so that once the installation of Debian is finished when we restart our computer, we can choose between the installed operating systems. It will ask us for confirmation to install it on the MBR, and we will reply affirmatively.

5.3.14. Reinitiating the system

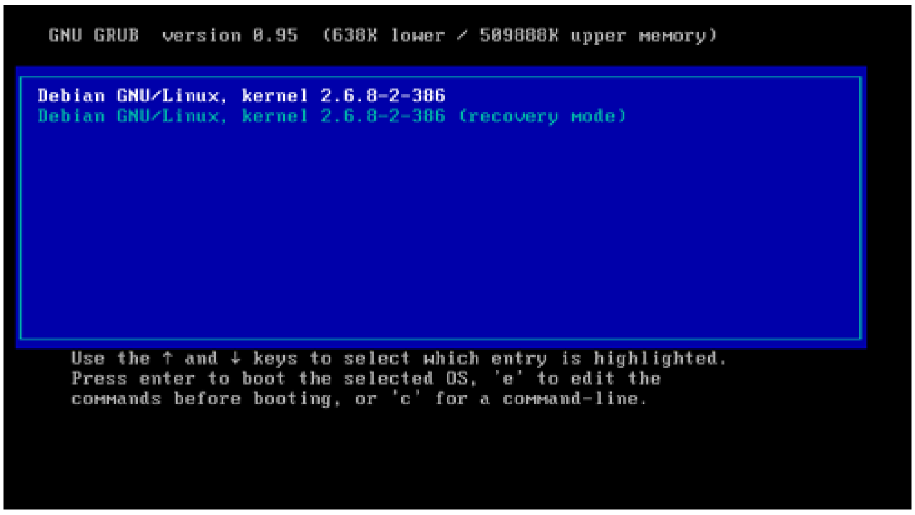
We have reached the end of the base system installation, and this is the displayed screen:

It is time to reinitiate the system to boot the base system that we have installed on our hard disk, and from there to start customising our first GNU/Linux installation. Therefore, we will remove the DVD-ROM and click the Continue option shown on the previous image.

5.3.15. Base system boot

If everything has gone as it should, when we reinitiate we will see that the system allows us to choose the Debian GNU/Linux operating system:

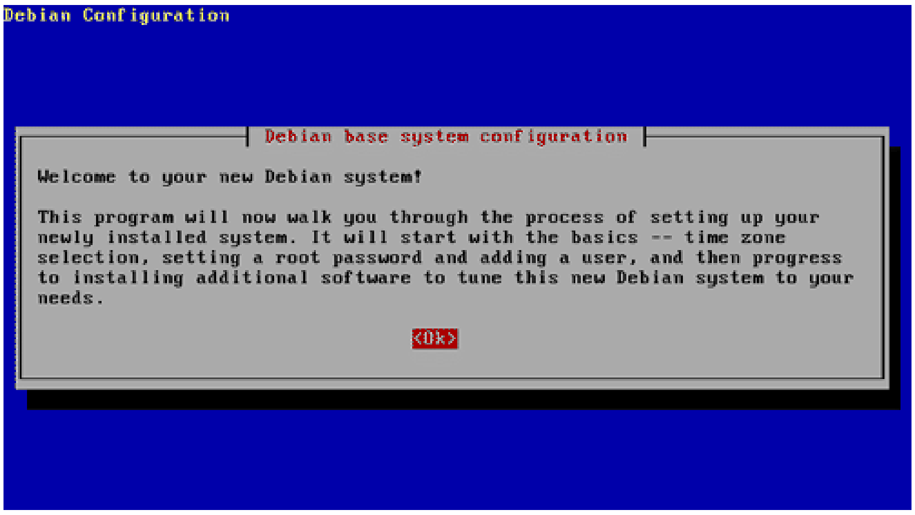
Figure 5.34



```
  GNU GRUB  version 0.95  (638K lower / 509888K upper memory)

 Debian GNU/Linux, kernel 2.6.8-2-386
 Debian GNU/Linux, kernel 2.6.8-2-386 (recovery mode)

     Use the ↑ and ↓ keys to select which entry is highlighted.
     Press enter to boot the selected OS, 'e' to edit the
     commands before booting, or 'c' for a command-line.
```

After a few moments, during which the screen will display the results of various processes executed during boot, the system will display a welcome screen and invite us to continue with the installation, reminding us that we can repeat this process whenever we want by executing the `base-config` command as root. We click OK and continue:

Figure 5.35

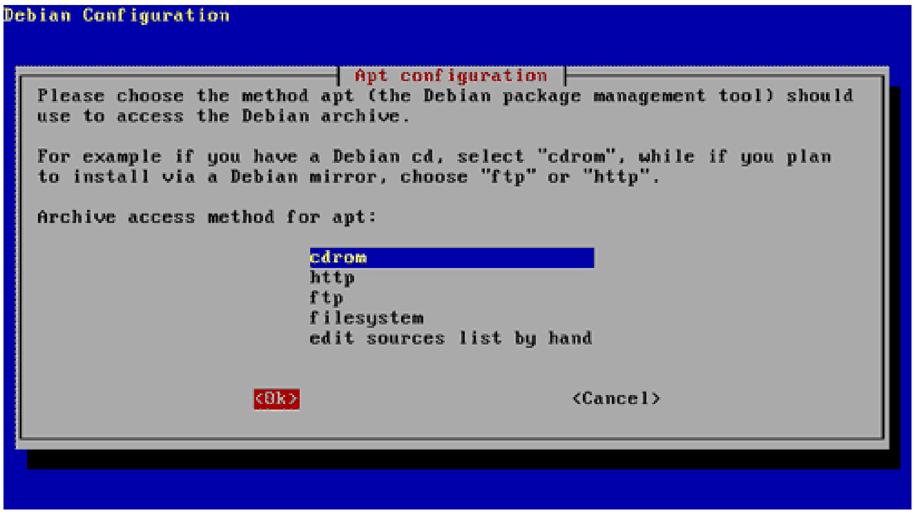


```
Debian Configuration

            Debian base system configuration

 Welcome to your new Debian system!

 This program will now walk you through the process of setting up your
 newly installed system. It will start with the basics -- time zone
 selection, setting a root password and adding a user, and then progress
 to installing additional software to tune this new Debian system to your
 needs.

                              <Ok>
```

5.3.16. apt configuration

In this section we are asked about the sources where `apt` must go to find information for building its database over the packages (dependencies, installation status etc.). Since we are carrying out an installation fully based on DVD-ROM, we will select the first option, cdrom, and click on OK:

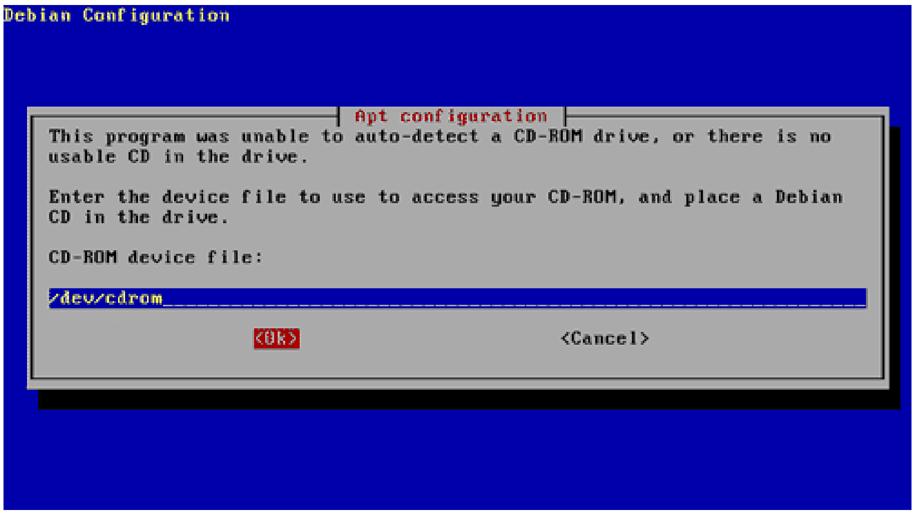
Figure 5.36



```
Debian Configuration

                         ┤ Apt configuration ├
  Please choose the method apt (the Debian package management tool) should
  use to access the Debian archive.

  For example if you have a Debian cd, select "cdrom", while if you plan
  to install via a Debian mirror, choose "ftp" or "http".

  Archive access method for apt:

                        cdrom
                        http
                        ftp
                        filesystem
                        edit sources list by hand

                    <Ok>                         <Cancel>
```

On the following screen, we will click OK again:

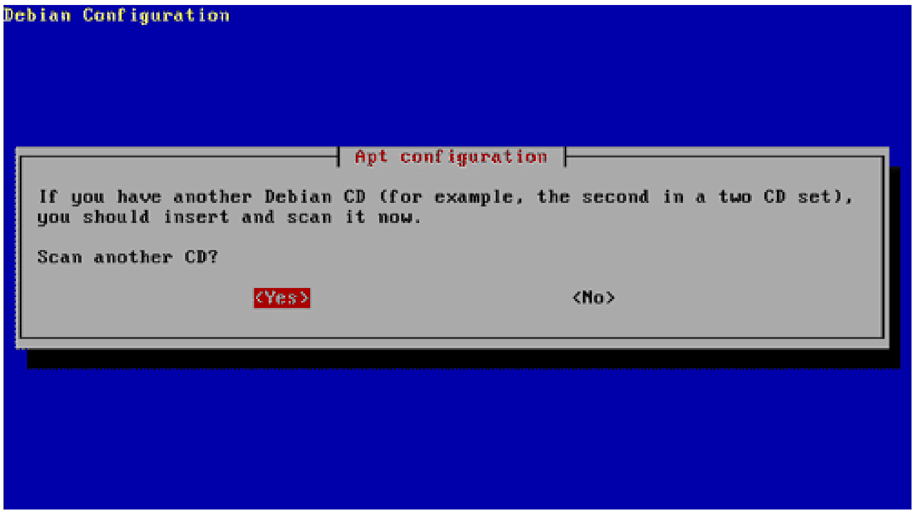
Figure 5.37



```
Debian Configuration

                         ┤ Apt configuration ├
  This program was unable to auto-detect a CD-ROM drive, or there is no
  usable CD in the drive.

  Enter the device file to use to access your CD-ROM, and place a Debian
  CD in the drive.

  CD-ROM device file:

  /dev/cdrom_______________________________________________________________

                    <Ok>                        <Cancel>
```

And since we have inserted the DVD-ROM 1, after a few moments during which `apt` extracts the information regarding the packages that this DVD contains, we will be asked if we wish to add the content of another DVD to the database. Once we have inserted the DVD-ROM 2 in the reader, we will answer this question with yes:

Figure 5.38



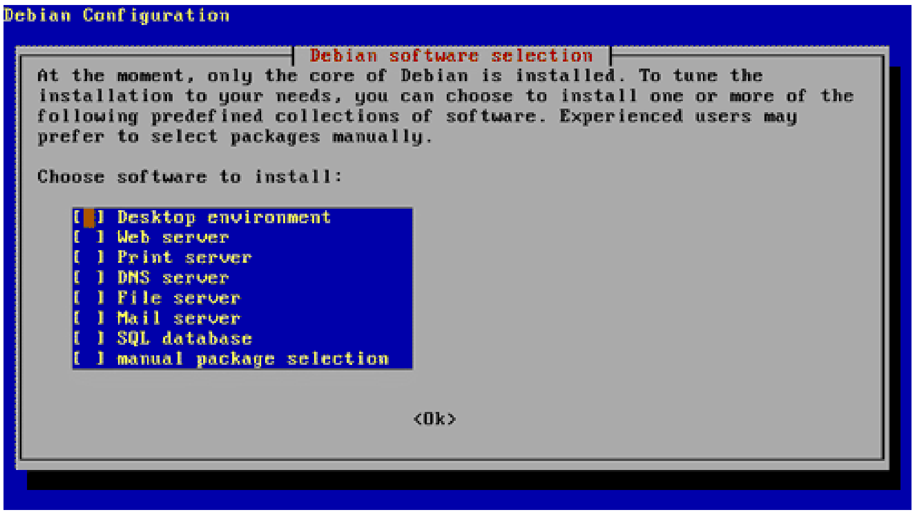
```
Debian Configuration

Apt configuration

If you have another Debian CD (for example, the second in a two CD set),
you should insert and scan it now.

Scan another CD?

<Yes>                    <No>
```

We do the same with the third DVD-ROM. Once the content of the three DVD-ROMs has been scanned, we will be asked if we wish to add any other source from which apt can obtain packages, and for the time being we will answer no.

5.3.17. Tasksel

Next, the package selection program tasksel will run:

Figure 5.39

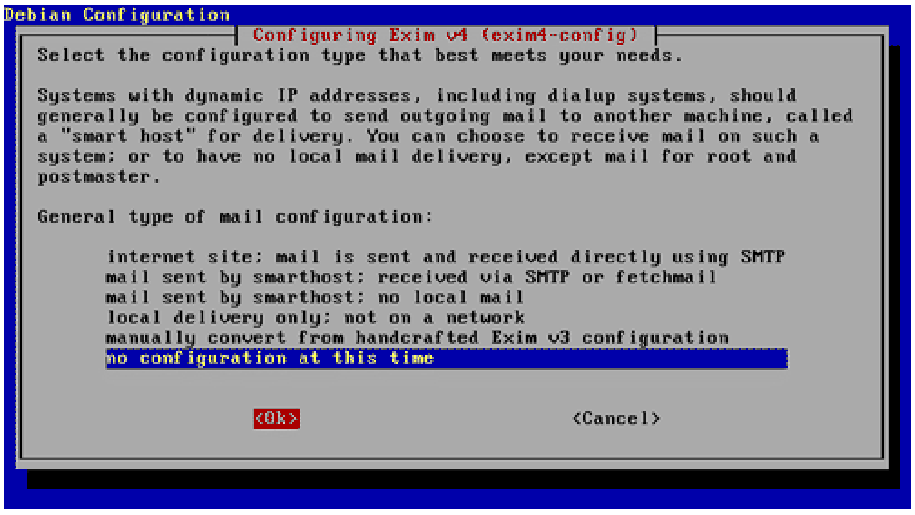


```
Debian Configuration

Debian software selection
At the moment, only the core of Debian is installed. To tune the
installation to your needs, you can choose to install one or more of the
following predefined collections of software. Experienced users may
prefer to select packages manually.

Choose software to install:

  [ ] Desktop environment
  [ ] Web server
  [ ] Print server
  [ ] DNS server
  [ ] File server
  [ ] Mail server
  [ ] SQL database
  [ ] manual package selection

                <Ok>
```

We will not mark any option, since we will also be able to run this program from the command line once we have completed the installation. As root we will simply have to execute the tasksel command. There is also another package selection program, dselect.

Next, we will be asked for the Exim configuration; since this workshop is not dedicated to configuring this mail management program, we will select option 6 to quit the program:

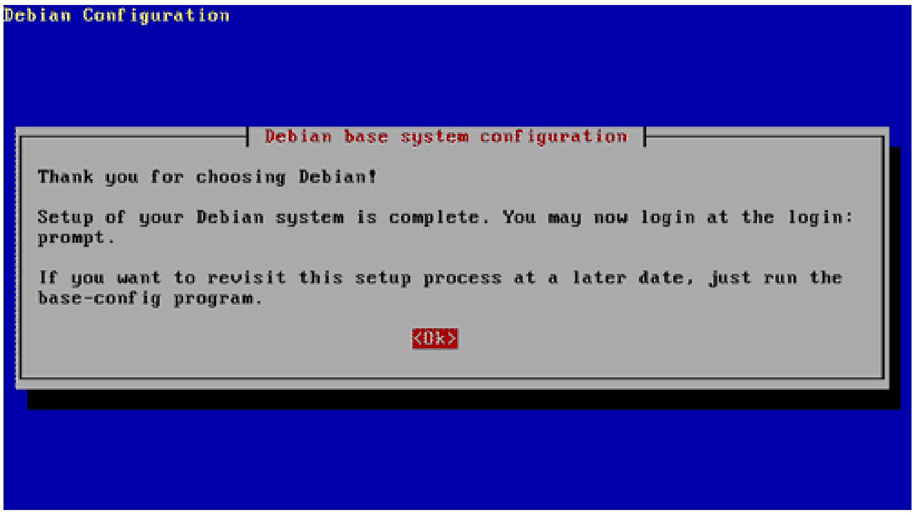
Figure 5.40



```
Debian Configuration
              ┤ Configuring Exim v4 (exim4-config) ├
  Select the configuration type that best meets your needs.

  Systems with dynamic IP addresses, including dialup systems, should
  generally be configured to send outgoing mail to another machine, called
  a "smart host" for delivery. You can choose to receive mail on such a
  system; or to have no local mail delivery, except mail for root and
  postmaster.

  General type of mail configuration:

        internet site; mail is sent and received directly using SMTP
        mail sent by smarthost; received via SMTP or fetchmail
        mail sent by smarthost; no local mail
        local delivery only; not on a network
        manually convert from handcrafted Exim v3 configuration
        no configuration at this time

                  <Ok>                         <Cancel>
```

And this is where the configuration of the Debian base system ends:

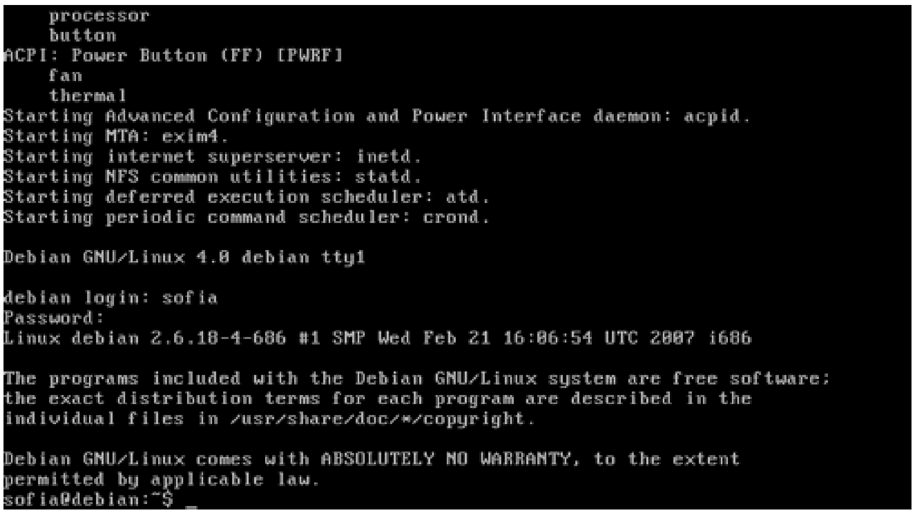
Figure 5.41



```
Debian Configuration

               ┤ Debian base system configuration ├
  Thank you for choosing Debian!

  Setup of your Debian system is complete. You may now login at the login:
  prompt.

  If you want to revisit this setup process at a later date, just run the
  base-config program.

                               <Ok>
```

Finally, after a few moments the screen will display `debian login` and next we will key in the user that we have created during the installation process, for example *sofia*, and after hitting Enter, we will write the password to initiate the session in the Debian GNU/Linux system that we have just installed:

Figure 5.42



```
    processor
    button
ACPI: Power Button (FF) [PWRF]
    fan
    thermal
Starting Advanced Configuration and Power Interface daemon: acpid.
Starting MTA: exim4.
Starting internet superserver: inetd.
Starting NFS common utilities: statd.
Starting deferred execution scheduler: atd.
Starting periodic command scheduler: crond.

Debian GNU/Linux 4.0 debian tty1

debian login: sofia
Password:
Linux debian 2.6.18-4-686 #1 SMP Wed Feb 21 16:06:54 UTC 2007 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
sofia@debian:~$ _
```

Now we have a standard basic system installed on our computer. Now begins the task of tailoring the system to our requirements. Having reached this point, it is very important to distinguish between two types of systems: development systems and production systems.

A development system is one designed for experimentation, where tests and experiments are carried out, and where stability and efficiency are not primordial. It is a system designed to acquiring knowledge about the system itself. However, stability and efficiency are the main concerns for a system designed for production. Therefore, we need to make sure that it contains solely and exclusively the strictly required packages, since having unnecessary packages installed will be detrimental for the system's efficiency. The strategy to be applied before mounting a production system (a web server, applications server etc.) always includes working first on a development system, where different tactics can be rehearsed and tested so that we can draw conclusions before mounting the production system.

Our first system will obviously be a development system, both because we have not conceived it to cover a particular need, and because it is our first experience of mounting a system, for the main purpose of acquiring knowledge. We will install and uninstall various packages, test different configurations etc., and of course this will seriously affect the system's stability and efficiency. Hence, we encourage students once they have completed this section and have a global idea of what installing a system involves, to reinstall the entire system again starting from scratch, to adapt it strictly to their own requirements.

5.4. Network installation of Debian Etch

Many of the steps for carrying out a network installation are the same as for a DVD-ROM installation, so we will highlight and focus only on the aspects that differ from the process described above.

5.4.1. Particularities of a network installation

The basic difference between an installation made from a set of DVD-ROMs and from a network is the location of the packages when it comes to installing them. Whereas for an installation made using DVD-ROMs we will have to insert at least one of them in the disk drive each time we install a new package so that we can extract the necessary data from it, with a network installation the packages are obtained remotely, which allows us firstly, to access their latest version, and update all the ones we have installed with just a single command line (`apt-get upgrade`).

At the same time, to carry out a network installation, we generally need just one CD-ROM or single DVD-ROM containing the necessary information for booting a basic operating system from which we can make the installation program run (which is already included on the CD-ROM or DVD-ROM, together with the modules that may be necessary for configuring the network access), and from here onwards, the rest of the information will be obtained remotely.

5.4.2. Aspects common to the different methods of installation

As already mentioned, many of the steps that we need to follow to carry out a network installation are similar to those for carrying out an installation using the distribution's set of DVD-ROMs. Therefore, we can launch the installation program in the same way as before ([5.3.1.] and [5.3.2.]) and initially follow the same steps to configure the installation language, the keyboard, to partition the hard disk and to activate the partitions ([5.3.3.], [5.3.5.], [5.3.6.] y [5.3.7.]).

At this point, normally the kernel will have recognised our network card and, if not, we will proceed to configure the relevant module in order to make it operative. In many cases the configuration, the passing of parameters, can be done automatically through modprobe, a program that can be launched from the installation interface itself after selecting a module. Having done this, we will need to configure the Network (through DHCP is the most convenient), specify from which site we will obtain the sources, and from here on, follow the same steps as with the CDROM to finish the installation.

5.4.3. Network module installation

This is a key point for carrying out the network installation, since this is where, if the driver of our network card has not been compiled within the kernel, we will have to select the required module in order to obtain access to it. In the first place, we need to find out whether our network card has already been detected during the boot process and whether its corresponding driver has been loaded. To do this, we will access the second terminal (Alt+F2) and execute the `dmesg` command. Now we need to search, from the various lines returned by this command, whether any one refers to our network card. As an example, for a RTL-8029 card (Realtek Semiconductors) we should get:

```
sofia@debian:~$ dmesg . . . ne2k-pci.c:v1.02 10/19/2000 D. Becker/P. Gortmaker
http://www.scyld.com/network/ne2k-pci.html PCI: Found IRQ 11 for device 00:0b.0
PCI: Sharing IRQ 11 with 00:07.2 eth0: RealTek RTL-8029 found at 0xe400, IRQ 11,
52:54:00:DB:FB:D4. . . .
```

If the search has been unsuccessful, firstly we need to determine what network card we have. To do this, the best thing is to turn to the documentation that comes with it or to inspect it visually. If this is not possible, there are other strategies for determining what our card is, like pressing Alt+F2 to access the console and investigating the content of the `/proc/pci` file (using `cat`, for example), or we can refer to the information that another operating system we have installed on the computer may be able to provide.

Once we know what type of network card we have, we need to find out what module will allow us to access the card. The safest strategy for this purpose is to use any search engine, for example Google, enter key words for our card, (reference of the card NIC linux module, for example) and read some of the pages that we find. We can also go to the pages of the main linux distributions and enter the card reference in their searches. As a last resort, we can go to the documentation of kernel network modules, where it specifies the corresponding module for all supported cards.

It is also good to know whether the manufacturer has developed its own module. Finding a module for a card can be a fairly complicated task, it can even be impossible, since there might not be support for that card or maybe advanced methods are needed in order to configure it; for this reason, we always advise using the most standard possible cards.

Once we have found out what module we need, after installing the kernel, we should select the suggestion given to us by the Configure Device Driver Modules main menu. After a warning screen, reminding us that many drivers are already included in the kernel, we will enter the Select Category module selection screen, (we can access this interface at any moment by executing the `modconf` command. This command serves as a front-end for administrating drivers that have been compiled in a modular fashion with the kernel) and using the cursors we will select the kernel/drivers/net option. Once inside the screen for selecting network card modules, Select Kernel/drivers/net modules, we will select the module we need, again using the cursors. After answering yes to whether we really want to install the module, we can let the autoprobe configure the module for us, unless we need to pass a particular parameter to the module in question. After a few seconds, we will get a message telling us whether the module has been installed correctly or not.

5.4.4. Network configuration

Once we know that the network card is operative, from the installation main menu we will follow the proposed Configure the Network step in order to configure the network. In the first place, we will need to configure the host name (before Debian was suggested, for example), without entering the domain. Next, we will need to enter the IP, network mask, gateway, domain name, and DNS servers (up to three, separated by spaces).

5.4.5. apt configuration

Once we have configured the network, the following steps we should follow are identical to those with the CD-ROM installation, until we reach the Apt Configuration. At this point, instead of choosing the CD-ROM option, we will select the most appropriate network option. For practical purposes it is the same to select the http protocol as the ftp. After making the selection we will be asked whether we want to use non-US type packages; in principle, unless there are legal problems, we will answer yes.

Regarding the answer to the following question, referring to the use of non-free packages, students are allowed to make their own choices according to their own ethical principles. Next, we will be asked what status the mirror will have from which `apt` will obtain the packages; we should always choose the one most readily accessible, which tends to be the one nearest to us geographically speaking. Having selected the status, we will be asked to select a specific server (the `netselect` application is designed to facilitate the choice of package servers based on the criterion of access speed). When this issue has been resolved, we will be shown the proxy access configuration screen; if we do not need to use this service, we will leave the line blank.

It is important for security reasons to obtain critical packages from secure servers. For this reason, we advise obtaining them specifically from http://www.debian.org/security/. Having completed all these steps, `apt` will connect with the mirror that we have specified in order to configure its database. From here onwards, for the rest of the installation we will follow the same steps as we did for the CD-ROM installation.

5.5. Conclusion

In this workshop we have learned how to install GNU/Linux on our computer. Although for now our system is very basic, the workshop's objective has been totally fulfilled, since we have laid the groundwork for starting to take advantage of the flexibility and potency of this operating system. In the next workshop we will learn how to configure the system and how to install new applications so that we can adapt it and equip it with all the tools we consider necessary in order to cover our needs.

6. Basic configurations

6.1. The login system

If we have still not configured the graphic environment, when we boot a GNU/ Linux system a login screen will appear asking the user to identify himself or herself before starting to use the system. In fact, most distributions launch several consoles that we can access through Alt+F1, Alt+F2 etc.

This allows us to work simultaneously with several accounts at the same time, having various sessions open to run different programs etc. The program responsible for managing each of these consoles is `getty`. The only thing that this program does is to open a connection with the appropriate device (in the case of the screen consoles, it is `/dev/tty`*`X`*, where the *`X`* is the number of the console) and to launch the login application. This mechanism gives us a lot of flexibility, since the `getty` program itself allows communication with different devices meaning that we could connect a terminal to the computer's serial port, mount a console using a telephone line and a modem etc.

Before launching the de login application, a welcome message appears on screen. We can configure this message in the `/etc/issue` file, writing whatever we want. In this same file we can also show some of the system's variables referring to them as:

Table 6.1

`\d`	current date
`\s`	name of the operating system
`\l`	console number
`\m`	computer architecture
`\n`	name of the computer
`\o`	name of the domain
`\r`	operating system version
`\t`	current time
`\u`	number of active users on the system

Once we have entered the system, the `login` program takes care of displaying the day's message. This message is what is written in the `/etc/motd` file, which we can also change. This mechanism is very useful for informing all users of a specific event, warning them of any problem etc. If a user wish-es to suppress this message, he can do so by creating an empty file called `.hushlogin` in his home directory. After displaying this message, the login process launches the shell configured for the user by default. The first thing the commands interpreter will do is to execute the content of the `.profile` file (which should be in the user's home directory). This file serves to execute the configured instructions every time the user enters the system. In addition to this one`~/.profile`, we also have the `/etc/profile`, which is executed for all system users and should be configured generically for all users with the options we want without having to put the instructions in each user `.profile`.

6.2. Exploring the bash

Although the `.profile` file is executed irrespective of the shell that we use, the files `.bashrc` or `.bashprofile` tend only to execute when we use the interpreter or bash shell (although it can be configured from the user's `.profile`, which is where the execution of this file is called). Let's look at a few of the instructions that we can find in these files:

```
#BASIC CONFIGURATIONS mesg n umask 022 #PATH export PATH= /usr/local/sbin:/usr/local/bin:
/usr/ sbin:/usr/bin:/sbin:/bin:/usr/bin/X11 #PROMPT export PS1='\h:\w\$ ' #USER ALIAS
alias l='ls - - colour=auto' alias ll='ls - - color=auto -al' alias
rm='rm -i' alias cp='cp -i' alias mv='mv -i' alias v='vim'
```

As we can see, in this file we can define whatever we want. The first two instructions of the file annul other users' incoming messages and configure the permissions that the new files that we create will have. The following instruction is the definition of the **PATH**. The PATH is the directories where we have the commands, programs, applications etc., that we want to be able to call from any place within the file system hierarchy without having to write the full path [we separate every PATH directory with a colon (:)]. The following declaration is the system's **prompt**. The prompt is the line that appears in the shell before the "#" character (for the root user) or "$" (for all other users). We can configure this prompt for ourselves as we wish using the following system variables:

Table 6.2

\d	system date
\h	name of the machine
\s	shell that we use
\u	user name
\v	bash version
\w	current directory

\!	history number command
\$	the "#" appears if we are the root user and "$" appears for all other users

PATH variable

If we wish to execute the programs from the directory that we are in without having to place "./" at the beginning, we could add this entry to the PATH declaration. Likewise, if the PATH does not have the program that we need to execute, we can specify its full path in the command line. In any case, it is not advisable to add "./" to the PATH because it can become a security hole.

Finally, we have the user **alias**. The alias[2] are synonyms, generally for the commands that we use most often (to avoid having to write them in full). For example, in one of the alias that we had in the example we defined that when we write `lls --colour=auto should execute.`. This way, we can use long commands without having to write everything every time we use them.

(2)We can see all the aliases defined from the same `alias` command.

In defining both the `PATH` and the prompt we have used the `export` command. This command allows us to define what we call an **environmental variable**. The shell uses these variables to carry out certain operations, save certain types of information etc. We can see all the ones that are declared using the same `export` command. With `set` and `unset` we can also handle other attributes of the command interpreter.

echo command

With `echo $VariableName` we can see the content of these variables and their attributes.

Some of these variables and attributes that the bash has by default, are:

- `PWD`: current directory.
- `BASH VERSION`: bash version that we are using.
- `RANDOM`: generates a different random number every time we display its content.
- `SECONDS`: number of seconds that have passed since we opened the shell.
- `HOSTNAME`: system name.
- `OSTYPE`: type of operating system that we are using.
- `MACHTYPE`: computer architecture.
- `HOME`: user's home directory.
- `HISTFILESIZE`: size of the history file (number of commands that are saved).
- `HISTCMD`: number of the current command in the history.
- `HISTFILE`: file where the command history is saved (in general, `.bash history` the user's home directory).

By handling these variables we can customise our commands interpreter much more adapting it better to our tastes and needs.

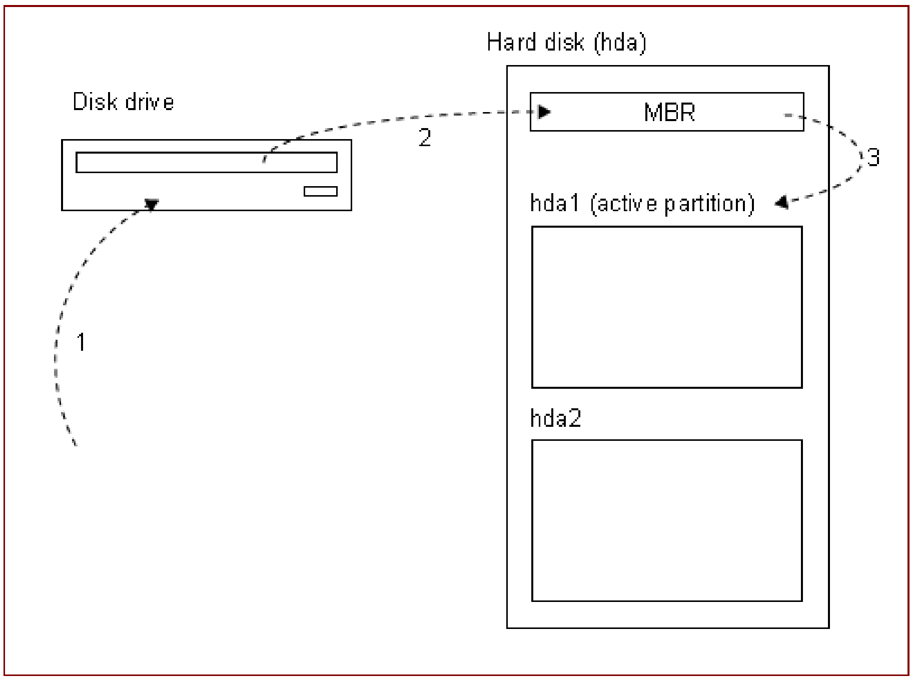
6.3. The boot system

Although with the installation of the operating system a boot system is already configured and installed, in this subsection we will look in more detail at what options we are given and how we can customise them and adapt them to our requirements. Although there are many, Lilo and Grub are the most common ones in GNU/Linux environments, so we will only focus on these two.

Before going into detail about the configuration of these two programs, we will explain how a standard PC's boot system works. As we already know, we can configure a computer's boot sequence from the BIOS or EFI. Generally, this sequence tends to start searching in the disk drive continuing with the CD/DVD drive and the hard drive. Although we can install Lilo or Grub on a disk or boot sector of a CD, it is more usual to install it on the hard disk so as not to have to insert the disk every time we boot our computer.

When the computer's boot system searches for the hard drive, the first thing it will inspect is whether the MBR (Master Boot Record) of the first hard drive (master of the first IDE channel or first disk of the SCSI channel) contains any indication of the system to be loaded. The MBR is the first track of the hard disk, where the information regarding configured partitions is stored, and optionally, the program responsible for initiating the operating system. If this program is not found here, the boot sector of the disk's active partition will be inspected. Whenever we wish to install a program in the computer's boot system we must place it in one of these zones. In the following figure we can see this entire process when the boot sequence is first the disk drive and then the disk:

Figure 6.1



Whenever we install a boot system, we need to remember that the order in which this sequence is carried out is important: if we install one on the MBR and another on the active partition, the MBR's will be executed because this is the area that the BIOS or EFI inspects first. If we do not have an active partition, we need to place the boot program in the MBR. At all events, the most advisable is to always install the program on the MBR because it is the first thing to be inspected. Although we may have other operating systems installed on other disks, we should install Lilo or Grub in one of these zones. In the program configuration we will tell it where the operating system that we wish to load is located.

Normally, in modern computers the boot sequence is: CD/DVD, hard disk, USB, network etc.

6.3.1. Grub

Grub allows us many different configurations to carry out almost any action with the computer's boot system. Grub also serves to install a program in the computer's boot zone that we wish.

It offers many more possibilities that make it very versatile: it provides a small commands interpreter when we boot the computer, gives us access to the disk partition files without loading any operating system etc. As in the previous case, in this subsection we will just look at its basic configuration. If we wish to go into further depth about its use, we can study its manual or the corresponding HOWTO.

The Grub[3] boot system is loaded in two phases. Generally, with the installation of the package, two files corresponding to these two phases are incorporated. When we start up Grub if we do not want to be shown any menu for selecting the operating system we wish to load, all we have to do is execute the Grub program and execute in the commands interpreter that it shows us:

(3)Grub (Grand Unified Bootloader) is the boot program of the GNU project.

Boot system

The boot system of a GNU/Linux or similar system allows us, with the same file system, to load various different kernels. This way, we can work with different kernels without having to install the system anew. To configure it we will just need to specify two local sections, placing which kernel we will use in each one.

```
$ install (hd0,0)/PATH/stage1 d (hd0) (hd0,0)/PATH/stage2
```

This instruction installs Grub on the MBR of the master disk of the first IDE channel. The way of referencing the disks varies very little from how it is done in GNU/Linux and with Lilo. In `hd`*X* the *X*, instead of `a`, `b`,... , is `0`, `1` etc. For partitions, we also start with number `0` to name the first and unlike `hda1`, we should write `(hd0,0)` and so on for the rest. Reading the instruction in this way let's look at how the first parameter serves to designate where the file of the first phase of Grub is (we specify the corresponding partition, directory

PATH and file stage1). In general, when we install the Grub package we also add these two files for each one of the loading phases (they are normally found in /usr/share/grub/i386-pc/). The d parameter (hd0) indicates that the first phase of the Grub will be installed on the MBR of the first disk. The last option indicates where the file is located for the second phase of loading, which is executed by the first.

Passing parameters to the Linux kernel

With Grub we can pass parameters to the Linux kernel at the moment of booting. This is very useful when we wish to carry out a specific operation on the system; for example, by passing single or 1 the system would initiate in runlevel 1, with root=/dev/hda3 we would specify the file system root etc.

With this configuration, when we restart the computer, the Grub commands interpreter will appear by default. With it we can handle many aspects of the disk, boot the operating system that we want etc. If we wish to boot a GNU/Linux system we will write the following instructions:

```
$ kernel (hd0,0)/vmlinuz root=/dev/hda1 $ boot
```

With the first we are specifying where the kernel image is located (with the parameters we want) and with the second we initiate the process of loading the operating system. If we choose a selection menu so that we do not have to write these commands every time we start the computer, we can generate a menu file like the following one (the comments begin with "#"):

```
#Specification of the operating system to be loaded by #default. This number corresponds
to the order of the #systems on the sections local to the #operating systems. default 0
```

Help for Grub

To see all of the commands that are available in the Grub shell we can hit the Tab key. Help is also included for a full reference of all commands.

```
#Let’s specify that it should wait 10 seconds before loading the system #configured by default.
timeout 10 #Boot configuration for a GNU/Linux system title Debian GNU/Linux kernel
(hd0,0)/vmlinuz root=/dev/hda1 #Boot configuration for a Windows XP title Windows XP system
root (hd0,2) makeactive
```

To install[4] Grub with this boot menu we must execute the same instruction as above but adding the p (hd0,0)/PATH/menu.lst parameter with the disk, path, and menu file. To protect the boot system we can place the password order in the global section of the configuration file. This way, when we wish to enter the Grub shell from the menu we will be asked for the password. If we use this instruction it is very important that only the root can read this configuration file (although in this case there is also the option of placing the password encrypted with MD5).

(4) Another way of installing grub is using the grub-install program.

There is also another type of boot that uses an image called RAM Disk (`ini-trd`). This other type of boot serves for making a modular configuration of the Linux kernel. It is frequently used when we need a kernel with some sort of special configuration, to include modules in the same kernel, to make a boot image for a live CD, to have the same image for all the computers of a laboratory stored on a single server etc. In any case, the standard installations of the operating system almost never use this type of boot. If we wish to create an image of this type, we can find out more in the `initrd` manual and in the manual for the `mkinitrd program`.

6.4. Access to other partitions and devices

UNIX-type systems treat all of the computer's devices as if they were files. This gives us a lot of flexibility since we can take advantage of all the mechanisms and functions that we used with the files and apply them to devices. In the `/dev/` directory we have all the devices recognised by the system. If the system does not recognise a device correctly, or if we wish to create a special one, the `mknod` command allows us to carry out this type of operation, although it is important to know exactly what we want to do before using it, since defective use could damage parts of the system.

Disk drive and CD/DVD directories

For the disk drive and CD/DVD many distributions create a default directory where they can be mounted (`/floppy/` or `/mnt/floppy/` and `/CD-ROM/` or `/mnt/CD-ROM/`). The `/mnt/` and `/media/` directories are also normally provided and in these we can create directories for other devices that we have on the system.

For the storage units, the system offers us another type of operation to access its file systems, the **mount** operation. For this operation, we will use the `mount` and `umount` commands, which mount or unmount all of the file system of a specific device/unit in an existing system directory. The basic way to use the command is `mount` *`directory device`*, where *`device`* can reference any device on the IDE or SCSI channel (`/dev/hd`*`XX`*, `/dev/sd`*`XX`*, the disk drive `/dev/fd`*`X`*, backup tapes, USB memories etc. and `directory` is the location where we will mount the file structure of the device. It is helpful if the directory where we mount these devices is empty, as it is not possible to access the devices when this directory is used as a mount point.

To unmount one of these devices, we can use `umount` *`directory`*, where *directory* must be the mount point used. If we mount devices such as a diskette or CD, it is important not to remove the device from the medium before we have told the system, so that it can update the device's file system cache. Likewise, we cannot unmount the device if any user or application is using any of its files or directories (the system would display an error message if we tried this).

Notice that we are not specifying the type of file system used in the unit that we are mounting at any point in the operation; this means that it has to be determined automatically. If we do want to specify the type of file system manually, we can use the `mount` command with the `-t`*`type`* parameter where *`type`* could be any one of the ones on the following table (see the `mount` manual for the full list):

Table 6.3

Type	System
Ext	GNU/Linux (versions of the kernel preceding version 2.1)
ext2	GNU/Linux (versions of the kernel after version 2.1)
ext3	GNU/Linux (versions of the kernel after versions 2.2 or 2.4)
swap	Swap system in GNU/Linux
sysv	UNIX type systems
minix	MINIX
iso9660	File system that most CDs use
nfs	Remote file system (Network File System)
smbfs	Remote file system in Windows TM networks (Samba File System)
ntfs	Tree of WindowsNTTM
msdos	MS-DOSTM
vfat	Tree of Windows95TM

Mounting and unmounting file systems

By default, superuser privileges are needed to mount/unmount file systems. What usually happens is that a user is defined as an administrator in the `sudoers` file so that the `sudoer` order in front of the command can carry out administration tasks, permitted from this `sudoers` file.

Apart from entering the type of file system used by the unit that we want to mount, we can also indicate other options that can be very useful in certain situations (always preceded by `-o` and separated by commas, if we want to enter more than one):

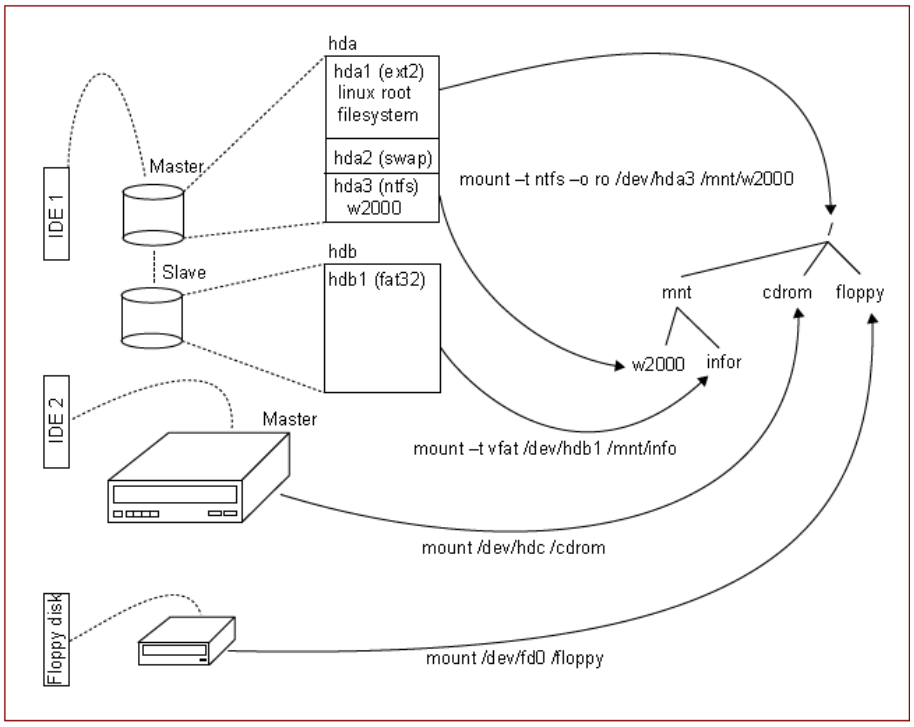
Table 6.4

Meaning of the option	Allow	Don't allow
Running binaries	`exec`	`noexec`
Use of the bit of SetUserId	`suid`	`nosuid`
Read-only files	`ro`	`rw`
Synchronised system (use of disk cache)	`sync`	`async`
Interpretation of special blocks or characters	`dev`	`nodev`
Permission for any user to mount or unmount the device	`user`	`nouser`
Swap type system	`sw`	

(If we entered `defaults`, we would use the `rw`, , `dev`, `exec`, `auto`, `nouser` and `async` options.)

The following figure shows an example of how to use this command to mount various different devices:

Figure 6.2



Apart from these commands that will mount and unmount units, the operating system provides us with another way of doing the same thing and always keeping the same configuration for each unit. In the `/etc/fstab` file we can save this information so that each line indicates a unit with its respective mount directory and the options that we want to configure. The syntax of each of these lines will be:

```
<dev> <dir> <typeSystem> <options> <dump> <order>
```

In the first field, we must specify the device just as we did with the `mount` command and the second field will contain the directory in which we want to mount the unit in question. In the `typeSystem`, we can specify the file system that the unit uses or `auto` so that it detects it automatically. In `options`, we can write the same things that we did when we used the `mount` command, separated by commas, if we want to put in more than one. A useful option in this field is the configuration of `auto` or `noauto`, with which we can tell the system to automatically mount (or not) the unit that has to be booted up. The `dump` field indicates whether we want to make backup copies (more information in the `dump` manual). If we do not use this system, we can enter a `0`. The last field can be used to specify the order in which the units should be mounted. If we enter `0`, we specify that the order is not important. The root file system is the first thing that has to be mounted; therefore, there should be a `1` in this field.

An entry that we always see in this file and that can be surprising is the `/proc/` directory, which has a special meaning. In reality, there are no files in this directory; instead, we have the value of many of the variables that the system's kernel uses. As was the case with the operating system, where it must be possible to reference everything as a file, in the `/proc/` directory, we can also determine certain internal variables of the kernel, as if they were files.

Although this design for mounting the units in the hierarchical structure of the directories is very powerful and provides a lot of flexibility, in some cases it is not too practical. For example, each time we want to copy a file to a disk, we must mount the unit, copy it and unmount it again. Consequently, there are a few other applications that make the whole process easier and save us having to take certain steps. One of these is `mtools`, which is a package with various applications that allow us to copy files to and from a disk directly, as well as a few other interesting tools. There is also a package called `autofs`, which automatically detects any device that is added to the system and mounts it without the administrator having to write any commands whatsoever.

`mount to` command

The `mount to` command is used to mount all the devices that have the auto option enabled (which is the default setting) for this configuration file.

6.5. Device configuration

Although this was not the case initially in GNU/Linux, currently more and more manufacturers provide drivers for their special devices for GNU/Linux. Before trying to configure them, we have to find information on them in the actual system manuals, the modules, Internet etc. to save us some problems when we start them up. Although most devices currently have HOWTOs, manuals or other types of documentation, it is important to find out whether there is any kind of driver available for a device before buying a new one.

Although we will only show how to configure some of the devices that are most frequently used in personal computers in this subsection, the process can vary significantly from distribution to distribution. Although we will only look at a system that uses Debian GNU/Linux in this subsection, if you learn how the basic process works from the start, you shouldn't have problems understanding the configuration used in other distributions.

6.5.1. The keyboard

Configuring the keyboard properly is very important for solving any problem that may arise with it. Firstly, we must know that, when the system boots up, a character map corresponding to the one configured during the installation process will be loaded. This character map is usually located in `/etc/console/boottime.kmap.gz` or in some other directory in `/etc/`. If we were to change the keyboard, all we would have to do would be to change this file with the one corresponding to the new keyboard and reboot the system, as the new map would then be loaded. All existing character maps are usually located in the `/usr/share/keymaps/` directory, sorted by computer architectures and countries. Specifically, the character map used in Spain with computers based on the i386 architecture and a standard keyboard, would be found in `i386/qwerty/es.kmap.gz`.

Although all these character maps are compressed in gzip format, we can decompress them and change some of the entries, to adapt them to our needs. In each line, we will find the `keycode` directive, which indicates that we are defining a key, indicated in the number that follows the command (we can find out which number corresponds to each of the keys on the keyboard using the `showkey` command). After making this definition, we need to define what happens when the key is pressed on its own, or with the Shift key etc.

Let's take a look at an example:

```
keycode 53 = minus underscore control keycode 53 = Delete
```

In the first line, we indicate the corresponding character when we press the key on its own (`minus`) or with the Shift key (`underscore`). The second line shows the function that will be performed if we press the key along with the Ctrl key (the following character would be eliminated). All the information you may need to configure a file of this type correctly will be found in the manual called `keymaps`, which is very useful if problems arise when using special keyboards or keyboards made in other countries. If we do not want to reboot the system when changing this map file, we can use the `loadkeys` command (`dumpkeys` shows us the configured ones).

Terminal font

Using the `consolechars` program, we can load the font that we want in the terminal. We need to differentiate clearly between a font and a character map: the map determines the meaning of each key, whereas the font is simply the graphical representation that we give to the key. All font configurations for the character are usually found in `etc/console-tools/config`.

Another aspect related to the keyboard is the question of umlauts, accents etc. We can configure all of this using the `/etc/inputrc` file (all possible directives for this file are specified in the `readline` manual). What can be more useful is `convert-meta`, which, if we disable `set convert-meta off`, allows us to use accents and umlauts. Finally, another important configuration (that is indirectly related to the keyboard) is that of the **locales**. With locales, we can configure the geographical region or regions in which we are so that we can use the keyboard's special keys, see the dates in the format that we are used to etc. This configuration is used by many of the system libraries; this means that this configuration will be used for many of the commands and applications in the system to adapt certain functions to our location. We can configure this in `/etc/locale.gen` and we can use the `locale-gen` and `locale` commands to see the configuration or update it.

Reconfiguring keymaps and locales

Another way of reconfiguring the keymap and the locales in Debian is to use `apt-reconfigure console-data` or `apt-reconfigure locales` respectively.

6.5.2. Network card (Ethernet type)

To configure a new network card (Ethernet type), the first thing we have to do is to add the necessary kernel module so that it can be properly recognised. Although with some cards we may not have to carry out this step because the kernel itself is already compiled to recognise the most common cards, we must make sure (before buying the card) that the necessary driver or module is available.

Network card module

We can find out which module our network card needs by using `discover -module Ethernet`. If we want to configure it so that it always loads, we have to overwrite it in `/etc/modules` (otherwise, we can insert it with `modprobe` or `insmode`).

Once the system has recognised the card, we can configure it however we want.

In the `/etc/network/interfaces` file, we can specify the whole configuration; we will also have the configurations of all the other interfaces in the system in this file. An **interface** is a (real or virtual) device linked to the network with which the system can communicate with other computers, offer certain services etc. The interface is the gateway that the system can use to communicate. This file should be used to specify the directives that are necessary to make the device work

Let's look at it with an example:

```
#loopback interface auto lo iface lo inet loopback #NIC auto eth0 iface eth0 inet static
address 192.168.0.10 netmask 255.255.255.0 network 192.168.0.0 #optional broadcast
192.168.0.255 #optional gateway 192.168.0.1 #optional
```

NIC

Network cards are also referred to as NICs (Network Interface Cards).

The first entry we will usually find in this file is for the loopback interface. This interface does not correspond to any real device or card in the computer, but is a mechanism in the operating system that makes it possible to use the communication protocols internally. In effect, if we are carrying out network functions without communicating with any other computer, it is not even necessary to have a network card installed. We will find the `auto` directive in all the entries, before specifying the configuration of the device. This directive indicates that the card can be mounted automatically when the system boots. The `iface` directive specifies the type of card and the protocol that will be used with it through the following syntax: `iface` *`device familyProtocol methodConfiguration`*. With the Ethernet cards, the device will be `eth`*`X`*, where the*`X`* will be a number starting with `0`, that indicates the number of the card installed in the computer. The family of the communication protocol using the card is usually any of the following:

- `inet`: IPv4, used in Internet and most local networks.
- `inet6`: IPv6, the new version of IPv4, which is gradually being implemented.
- `Ipx`: for Novell^{TM} networks.

Finally, the last field is used to indicate how the network card is configured (the address, the network in which it is, the gateway that has to be used etc.). The following table shows what these options are for the `inet` protocols family:

iproute2

Since version 2.2. of the Linux kernel, it is possible to use a new network infrastructure called iproute2. We can use this to handle all the aspects related to our NICs and the internal tables that the operating system uses to handle everything that is related to the network.

Table 6.5

Config	Options	Description
`loopback`		Method to define the `loopback` interface (it must be used with the `lo`).
`staticinterface`		Method for configuring a NIC with a static IP address
	`address`	IP address of the interface. Field must be filled in.
	`netmask`	Mask of the IP address Field must be filled in.
	`broadcast`	Broadcast address. If this is not specified, the system will calculate it automatically.
	`network`	Network ID address.
	`gateway`	IP address of the gateway that we use for this interface.
`dhcp`		Method for remotely configuring the IP of all the computers on a local network (Dynamic Host Configuration Protocol).
	`hostname`	DHCP server IP.
	`leasehours`	Time, in hours, that the IP is rented (this is renewed, once the time has elapsed).
	`leasetime`	Time, in seconds, that the IP is rented.
	`vendor`	Server type identifier (generally dhcpd).
	`client`	Client type identifier (generally dhcpd).
`bootp`		Method for remotely configuring the IP of all the computers on a local network (BOOT Protocol). Currently, DHCP is the more widely used protocol.
	`bootfile`	File that will be used when booting up.
	`server`	IP address of the BOOTP server.

Config	Options	Description
	hwaddr	MAC address of the BOOTP server.
ppp		Method used with the point to point protocol, which is used in various modems.
	provider	Service provider.

Although we will not go into further detail where computer networks are concerned in this subsection, it is important to note that we have many commands available for handling the network configuration of the operating system. The most widely used are `ifconfig`, with which we can see the configured devices, `ifdown` and `ifup`, which allow us to turn the interface that we want on or off and `route` which shows us the systems routing table.

6.5.3. WiFi card

A wireless network (wireless LAN), also known as WiFi, is a piece of technology that is based on the IEEE 802.11 standard. The most common definitions are amendments b, a and g of the original protocol. The 802.11b and 802.11g standards use spectrum band 2.4 GHz, which means that they can interfere with other devices that use the same band, such as wireless telephones. The 802.11a standard uses spectrum band 5 GHz.

The use of radio frequency spectrums is strongly regulated by governments. Most WLAN card devices cannot be distributed as open code. The part of the device that defines the frequencies of the radio waves is normally distributed as firmware, which, in principle, means that it cannot be modified.

Wireless communication networks are becoming more common, in domestic settings as well as in institutions, schools or businesses. To install these networks, it is necessary to have what are known as access points, which are the devices connected to the institution's physical network. These access points make it possible to connect any computer in the vicinity to the network, using WiFi cards (whether they are PCI, PCMCIA, integrated in the motherboard or in a USB adaptor). This greatly simplifies the cabling in buildings.

For our GNU/Linux system to detect and configure a wireless card properly, we must add the necessary modules to the system's kernel, which in many cases are already compiled in the same kernel. If we do not have any other Ethernet cable installed, it will be referred to as `eth0`, or `eth1` if we already have one etc. At this point, the only thing that we have yet to do so that the card will connect to the access point will be to edit the `/etc/network/interfaces` file and add the required configuration so that an IP address can be assigned. Naturally, this system interface can be treated like any other interface, using the firewall mechanisms of the system, with the `iproute2` applications etc. We will now look at an example of the configuration of the `eth1` card, which is WiFi:

```
# wireless network auto eth1 iface eth1 inet dhcp wireless_essid miwifi wireless_channel 6
wireless_mode managed wireless_keymode open wireless_key1 millavehexadecimal wireless_key2
s:millaveascii wireless_defaultkey 1
```

If we look at the entry in the manual for the `iwconfig` command, we will see the meaning of each one of the preceding commands.

ppp and pppconfig packets

The packets that contain the programs required for a modem to connect to the Internet are usually called ppp and `pppconfig`.

6.5.4. Modems

The `pppconfig` application is normally used to configure a modem; the application writes the configuration files that are necessary for the PPP system's daemon, which is the program in charge of establishing the Internet connection. Whenever using `pppconfig` (or similar applications), a series of steps must be taken, as explained below:

Observation

With `pppconfig`, we have a menu that allows us to add, modify or delete connections. The steps that we show here correspond to the creation of a new connection.

1) Name of the provider: the provider is the company with which we have contracted the Internet connection service. This name can be used to identify each connection that we configure individually.

2) Configuration of name servers: When we establish the contract with our provider, we are usually given the IP(s) of the name servers that must be used. If we have these IPs, we must indicate that we are using a static configuration, after which we will be asked for these IPs. We should only choose a dynamic configuration when our provider tells us that the DNS configuration is dynamic. With the third option, which tells us that the DNS will be treated by other means, we can use the file configuration `/etc/resolv.conf`.

3) Authentication method: the authentication method will be PAP or CHAP. Generally, the providers normally use PAP (Peer Authentication Protocol), although we should find out about this if it does not work for any reason.

4) User name and password: this is the information that the providers give us so that we can connect and access their services.

5) Modem speed: depending on the modem that we have, we may access Internet at a greater or lesser speed. Currently, the speed of all modems is 115200 bps, which means that the most advisable thing is to leave the value at 115200. If we have a slower modem, this is usually detected and the value is automatically reconfigured when connecting.

6) Pulse or tone dialling: most telephones now work with tone dialling, although the old pulse dialling system is still used in certain rural areas.

7) Telephone number: this number should also be provided by the Internet service provider.

8) Communication port: the communication port is the port to which we connect the modem. If we ask the system to detect the modem automatically, it will check all the ports and configure them automatically. Otherwise, we can enter `/dev/ttyS`*X*, where the *X* is a `0` for COM1, a `1` for COM2 etc.

All these configurations are usually stored in the files located in the `/etc/ppp/` directory. Although it is also possible to edit these files and change the directives automatically, it is more advisable to use an automatic application, as configuring these files is quite complex. To establish the connection with our provider, we must start up the daemon by running `/etc/init.d/ppp start`. To stop it, we can use `/etc/init.d/ppp stop`.

6.5.5. ADSL and PPPoE

Asymmetric digital subscriber line (ADSL) is a communication technology that makes it possible to transmit data through telephone and television cables. Asymmetric communication implies that the download and upload time rates are different; generally, the upload rate is slower.

ADSL uses a cable modem or a concentrator depending on the method implemented by the Internet service provider. Both connect the respective devices to the computer's Ethernet interface (generally, the first one, `eth0` and sometimes to a USB port).

The ideal configuration for the Ethernet interface would be in DHCP mode, with the modem cable providing the IP address, DNS etc. as this means that everything configures itself automatically.

PPPoE (Point-to-Point Protocol over Ethernet) is a protocol that is very similar to PPP that uses an Ethernet interface instead of a modem. This type of protocol has been adopted by some ADSL Internet providers.

The ADSL connection is generally part of the telephone service. The signals (telephone and ADSL signals) are divided in a device called a splitter, which separates them to their respective frequency ranges. The ADSL signal then enters a device known as a concentrator that is connected to the computer's Ethernet interface with a twisted pair cable, with RJ45 connectors or to one of the computer's USB ports.

6.5.6. Soundcard

The soundcard needs a module of the system's kernel to be inserted in order to work properly. If we have installed the `discover` application, we can discover which module corresponds to our soundcard using the `discover --module sound` command. We can also determine the manufacturers and chipset using the `lspci | grep audio` command.

To install the module, we can use the `insmod` or `modprobe` commands, and if we want to leave it configured permanently, we should write it in the `/etc/modules` file. Although we can already use the soundcard properly once the corresponding module has been included, it is generally common to install the ALSA (Advanced Linux Sound Architecture) infrastructure as well. Generally, most distributions include this architecture by default; however, if this is not the case, it can be installed with the corresponding package.

ALSA

ALSA is a project that has developed a lot of software related to sound processing applications, new modules for the Linux kernel etc.

For a normal user to be able to use the sound devices, it is necessary for the user to be a member of the `audio` group.

```
addgroup audio user
```

This will mean that the `user` will be added to the list of the members of the `audio` user group in configuration the `/etc/group` file.

6.5.7. Printer

Printers can be configured using many different applications in GNU/Linux. Although `lpd` (Line Printer Daemon) was one of the first printer management programs that appeared in UNIX type systems, there are currently plenty of programs that are easy to configure and manage. We will discuss some of the most widely used programs below:

- `lpd`: one of the first printing daemons for UNIX type systems. It has to be configured manually.

- `Ipr`: the BSD version of `lpd`. It is very advisable to use some form of automatic filter such as `magicfilter` or `apsfilter` to configure the printers. This type of filter automatically detects the type of file that will be printed and prepares the print properly (it uses a filter called IFHP).

- `lprng`: applications based on `lpr` with the advantage of incorporating a configuration tool called `lprngtool`, which makes it possible to configure the printer graphically and easily.

- `gnulpr`: the GNU version of the printing system `lpr`. It also incorporates graphical configuration tools, service management tools etc.

- `CUPS`: stands for Common UNIX Printing Systems; this set of applications is compatible with the `lpr` commands and can also be used in Windows^{TM} networks. It uses a set of its own filters and supports most of the printers on the market.

Although all of these applications have their own configuration methods, all use the `/etc/printcap` file to save it. In general, they also use some form of daemon so that the printing system will be operative. The daemon can be configured so that the computer to which the printer is connected serves as the printing server. In this way, various computers on the same network will be able to use the same printer, which saves on resources. For printing clients, the same programs can be used, specifying, during configuration, that the printer is remote (the IP of the printing server and the printing queue will usually be required).

If we wish to configure a printing server for Windows^{TM} networks or configure a printer of a Windows^{TM} server from a GNU/Linux client, we must use other types of programs. `Samba`this is a set of GNU/Linux applications that use the protocols of the Windows^{TM} networks. Although there are many more functions apart from that of configuring a server or printing client, in order to use printers in Windows^{TM} we need to use this set of applications or those provided by `CUPS`.

Swat

Swat (Samba Web Administration Tool) is a tool that is very useful for configuring a samba server.

Secure configuration of the printer server

When configuring a printer server, it is important to define which machine/users are allowed to use the printer. Otherwise, a hostile user could take advantage of the vulnerability and take advantage of our resources, leaving the printer without paper etc.

7. Daemons and runlevels

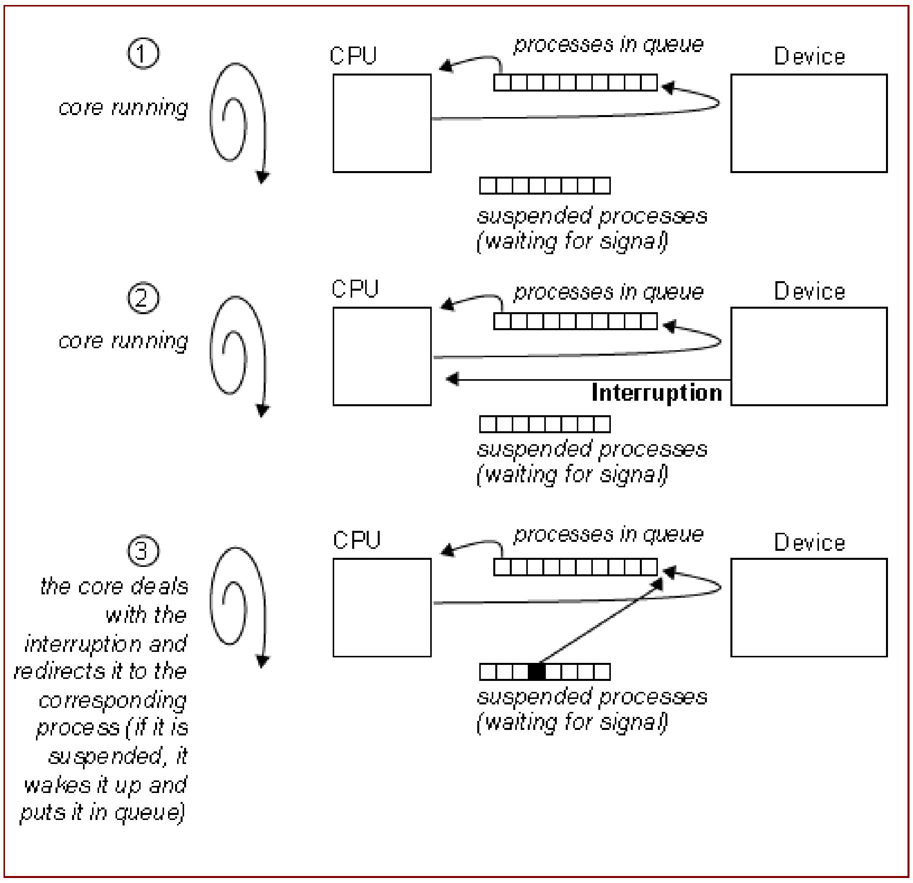
7.1. The daemons

As we know, GNU/Linux allows us to run as many processes as we want simultaneously, by sharing the CPU time equally among the processes. In fact, the mechanism for handling the processes must also take into account what are known as **interruptions**. An interruption is a signal that reaches the system's kernel from any of the devices that we have installed in our computer. These interruptions are usually linked to a specific process, which means that the kernel must wake up the process in question (if it is not running) and redirect the interruption so that it processes it properly. A typical example of an interruption is when we press a key on the keyboard or move the mouse: when we do this, the device sends a signal that must be redirected to the right application so that it is properly treated.

In order to properly handle all the interruptions that occur, the kernel does not listen permanently to the devices in the system, waiting for their signals. Instead, the system runs the operations of the processes in an execution queue and it is only when the interruption occurs, that the system attends to the device that has generated the interruption. This has to take place in this way due to the large differences in the speeds of the system's devices and the CPU. The way interruptions are treated is fundamental for any operating system, as this mechanism, among others, is the one that makes it possible to run as many processes as we want and, as soon as the interruptions arrive, wake up the processes that are waiting for it.

A **daemon** (disk and execution monitor) is a process that, generally, we have loaded in the memory, waiting for some signal (from an interruption in a device or the kernel itself) to wake up and execute the functions required to treat it. Although this definition can also apply to other processes in the system (launched by the users or the system itself), a daemon also tends to run in this manner (although it does not in certain special cases). In this way, the daemons that we have loaded do not take up CPU whilst they are not strictly necessary and we can always work on the computer without a problem no matter how many we have loaded in the memory.

Figure 7.1



Shell scripts of the daemons

The shell scripts of the daemons are nothing more than a tool for facilitating the whole boot up process, shut-down process etc. In some cases, we can also use the mechanism and the organisation of these daemons to execute certain operations that we are interested in (writing a shell script that executes when a determined runlevel is entered).

Although a daemon will be a process like any other that runs in the background, the way in which we organise and treat them is different to the rest of the commands and programs in the system. Generally, all the daemons have a shell script located in the `/etc/init.d/` directory that make it possible to start it, stop it or see its run status. In order to carry out one of these tasks, we must run the shell script corresponding to the daemon that we want to deal with, by specifying some of the following parameters:

- `Start`: to start the daemon. If it is already running, an error message will appear.
- `stop`: to stop the daemon. If it is not already running, an error message will appear.
- `restart`: restarts the daemon. This can be used to reread its configuration files.

- `reload`: although not all the daemons allow this, this parameter can be used to reload the configuration files, without having to stop it.

Most of these scripts use a program called `startstop-daemon` which is provided by the operating system and is used for handling these processes. When we are administrating a server, we will often have to design our own daemons in order to carry out a specific task. The directory containing all the shell scripts of the daemons also usually contains an example of one (`/etc/init.d/skeleton`) so that we can use it when we need to configure a new one that is not already in the distribution They are generally programmed as follows:

```
#!/bin/sh PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin: /usr/sbin:/usr/bin
DAEMON=/usr/sbin/daemon NAME=daemon DESC="some daemon" test -x $DAEMON || exit 0 set -e
case "$1" in start) echo -n "Starting $DESC: $NAME" start-stop-daemon --start --quiet
--pidfile \ /var/run/$NAME.pid --exec $DAEMON echo "." ;; stop) echo -n "Stopping $DESC:
$NAME " start-stop-daemon --stop --quiet --pidfile \ /var/run/$NAME.pid --exec $DAEMON
echo "." ;; restart|force-reload) echo -n "Restarting $DESC: $NAME" start-stop-daemon
--stop --quiet --pidfile \ /var/run/$NAME.pid --exec $DAEMON sleep 1 start-stop-daemon
--start --quiet --pidfile \ /var/run/$NAME.pid --exec $DAEMON echo "." ;; *)
N=/etc/init.d/$NAME echo " Usage: $N {start|stop| \ restart|force-reload}" >&2
exit 1 ;; esac exit 0
```

Running a daemon

In order to run a daemon, we have to call it up with its complete path `/etc/init.d/`*`nameDaemon`* and enter the parameter in question. Some distributions have the `service` command, which makes it possible to do the same thing without having to specify the complete path.

As we have seen, in the variables stated at the beginning of the shell script, we have to specify which `PATH` is necessary for processing the daemon, the program that we will run (`DAEMON`), the name that we will give it (`NAME`, which must be the same as the name of the shell script) and its description (`DESC`). The only thing that the code does when booting the daemon is to write a file in the `/var/run/` directory with the PID of the process. When it is stopped, this PID will be searched and the stop signal will be sent to the corresponding process. Naturally, we will find shell scripts prepared for carrying out many operations with the daemon that we will handle, although, at the least, they must all have this structure.

Programming the daemons

Although the daemons are programs like any other program, programming them is somewhat different to programming the user applications because we need to include functions for suspending them and having them wait for the signals to wake them up etc.

7.2. The runlevels

The daemons that we have running at a determined moment, show us the services that the operating system is offering and/or receiving. The fact that we can have so many different daemons does mean that we need to organise them properly. We understand a runlevel (**or execution level**) as the execution of specific daemons, which, in turn, provide specific services. When installing a server, it is common to design a configuration so that certain services can be offered at certain moments and not at other moments. To allow this kind of function, the operating system offers us different runlevels that we can adapt to our needs.

Although we can configure the number of runlevels that we want and the functionalities of each one, UNIX-like systems generally provide us with 6 different runlevels with the following properties:

Table 7.1

Level	Functionality
0	Runlevel 0 is configured to stop the system.
1	This level is called single user, as it only allows the root user to enter the system. Only a minimal number of daemons are started and it is used for maintenance tasks.
2-5	Runlevels from 2 to 5 are there to be configured according to the requirements of each installation process. When installing the system, they are all the same, by default. These levels are also called multi-user levels as, by default, they allow more than one user to work on the system.
6	The last runlevel is prepared for rebooting the system. It is very similar to runlevel 0 but it has the rebooting function.

The command necessary for changing the runlevel is `init` (we enter the runlevel that we want as the parameter) and `runlevel` can be used to see our current runlevel.

The only thing that the `halt`, `reboot`, `shutdown` or `poweroff` commands do is to call up runlevel 0 or 6 after performing a specific operation (see the manual for more information). Only the system's home user can use all of these commands.

The way in which these daemons are organised in each runlevel is very simple. Each runlevel has a directory located in `/etc/rcX.d/` where *X* is the number of the level. In these directories, we will find symbolic links to the shell scripts of the daemons located in `/etc/init.d/`, which we can use to tell the system whether we want to start up or stop the daemon that they point to. The name of the link itself identifies the operation that we will perform: if the link begins with an s (start) we tell the system that we want to start the daemon, whereas if it starts with a k (kill) we tell the system that we want to stop or kill it. If the name does not begin with either of these letters, the system will not do anything. After this letter, a number with 2 figures, between 00 and 99, must be entered, to indicate the order in which they must start or stop. This order is important, because some daemons need others to be running before they can start up.

When changing the runlevel, the system will inspect the daemons of the corresponding directory and will begin, firstly, by stopping the indicated daemons and will then start up the others. The only thing we do here is call up the daemon entering the `start` or `stop` parameters so that if we stop any that are not running at the time, there will be no problem because the shell script itself takes this into account. This can be used to change the runlevel without taking into account the previous level that we were in. The following table shows an example of how to configure 3 runlevels:

Table 7.2

Runlevel 2		**Daemons running**

K50sshd
K51apache-ssl
K52tftpd
K53telnet
	S10sysklogd
	S12kerneld
	S20dhcpd
	S50proftpd
	S90apache
		`sysklogd`
		`kerneld`
		`dhcpd`
		`proftpd`
		`apache`

Runlevel 3		

K50dhcpd
K51proftpd
K52apache
K53tftpd
K53telnet
	S10sysklogd
	S12kerneld
	S20sshd
	S50apache-ssl
		`sysklog`
		`kerneld`
		`sshd`
		`apache-ssl`

Runlevel 4		

K50dhcpd
K51proftpd
K52apache
K53tftpd
K53telnet
	S10syslogd
	S12kerneld
	S20tftpd
	S50telnet
		`sysklogd`
		`kerneld`
		`tftpd`
		`telnet`

The `/etc/inittab` file contains all the definitions of the configurations of the runlevels: the default runlevel, the number of consoles available for each of them etc. Each line of the file is a directive with the syntax: `<id> :<runlevels> : <action> : <process>`. The first field is the identifier of the directive; the next shows the runlevel in which that directive is valid, the operation that we will perform and the process that will be launched. The following example explains how to configure these directives:

```
# The default runlevel (in this case, 2) id:2:initdefault: # Scripts to be run when the
system boots up (before # entering the default runlevel) si::sysinit:/etc/init.d/rcS # Program
that is called up when entering runlevel # single user (the wait operation indicates that the
# process is launched and nothing else happens) ~~:S:wait:/sbin/sulogin # Configuration of the
different runlevels # available in the system l0:0:wait:/etc/init.d/rc 0 l1:1:wait:/etc/init.d/rc
1 l2:2:wait:/etc/init.d/rc 2 l3:3:wait:/etc/init.d/rc 3 l4:4:wait:/etc/init.d/rc 4
```

```
l5:5:wait:/etc/init.d/rc 5 l6:6:wait:/etc/init.d/rc 6 # Command that runs when pressing
Ctrl+Alt+Del ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now # Definition of the consoles open
on each # runlevel (the respawn action indicates # that when the process finishes running # getty
will launch again) 1:2345:respawn:/sbin/getty 38400 tty1 2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3 4:23:respawn:/sbin/getty 38400 tty4 5:23:respawn:/sbin/getty
38400 tty5 6:23:respawn:/sbin/getty 38400 tty6
```

Observation

We could also use this file to configure a terminal that communicates with the system through a modem and another computer with the directive `T1:23:respawn:/ sbin/ mgetty -x0 -s 57600 ttyS1`. In this way, we could have a console of the system in another terminal, communicating with a telephone line.

As we can see, this file is used to configure everything related to the runlevels in a very flexible manner, which means that we can change whatever we want, to better adapt it to our needs. Note that, although we are defining the default runlevel here, we could also specify it when booting up the system with Grub. This is very useful, for example, when we have severe problems in the system that we cannot solve properly; if we boot up with the first runlevel (entering `1` or `single` in Grub), only the most essential functions will load and we will be able to access the system to fix whatever is necessary.

7.3. Booting up the system

The parent process of all the others is `init`. This process is in charge of starting up the others that are available in the configured runlevel. However, before entering this runlevel, all the shell scripts of `/etc/rcS.d/` (configured in `/etc/inittab`), that may be daemons like those of the other runlevels or simply shell scripts necessary for the system (loading the character map, loading the kernel's modules etc.), will run. If we wish to eliminate any of them, we must know exactly what we are doing, as they are generally essential for the operating system to work properly. Once these daemons have booted up (or the shell scripts), we enter a runlevel configured by default, stopping and starting the daemons specified there. Once we have learnt how all this is organised, we can start adapting the system's booting up process to our needs by creating and placing the daemons that we want in any of the places we have discussed.

7.4. Basic daemons

Depending on the GNU/Linux distribution that we use, the same installation process already configures certain daemons. Nevertheless, all distributions tend to incorporate the daemon for the logs system and the one for the regular and delayed running of the applications (although the configurations of these can vary somewhat). In this subsection, we will look at how these three basic daemons work and how we can configure them. It is important to know how to handle these basic daemons because they can help us a lot with some of our administration tasks.

7.4.1. System logs (sysklogd)

The system logs are traceable files that a daemon of the operating system generates so that any action performed in the system can be traced back. The daemon in charge of these tasks is `sysklogd`, which is configured in `/etc/syslog.conf`. Each line in this file consists of a rule with two fields: the selector and the action. The selector is used to configure the service that we wish to log and the level of priority. The action is used to indicate where we want to redirect the logs (to a file, a console etc.). The following tables show the options that are valid for these fields.

Generally, all the log files of the system are saved in the `/var/log/` directory. Although most of the log files are in plain text format and we can see them with any editor, there are also some special ones that do not save the data in this format. Generally, they are the `/var/log/wtmp` and `/var/log/btmp` files, which are the user entry logs in the system and the invalid entries, respectively. To see these files, we can use the `last` and `lastb.` commands. If we have these logs configured in any other file, we could also see them by entering the `-f` *`file`* parameter with the `last` command.

klogd

The system's kernel also launches a daemon to manage its logs called `klogd`.

lastlog

We can also see the last user entry logs by using the `lastlog` command.

Table 7.3

Selector			
Service	**meaning**	**Priority**	**meaning**
authpriv	Permission messages or security aspects.	emerg	The system is unusable.
cron	Daemon `crond` and `atd`.	alert	The action must be performed immediately.
daemon	Daemons of the system without log options.	crit	Critical conditions.
ftp	Daemon of the FTP server (File Transfer Protocol).	err	Error conditions.
kern	System kernel messages.	warning	Emergency conditions.
lpr	Printing subsystem messages.	notice	Normal, but important, notices
mail	Mail subsystem messages (if we have configured it).	info	Information messages
news	News subsystem messages (if we have configured it).	debug	*Debugging* messages.
syslog	Logs generated by the daemon itself `syslogd`.		

Selector			
Ser-vice	**meaning**	**Prior-ity**	**meaning**
user	Logs of user-level applications.		
uucp	Messages generated by the UUCP system (Unix-To-Unix Copy Protocol).		
local0~7	Reserved for local use.		
(The services and priorities can be combined however we want).			

Table 7.4

Action	
Destination	**Explanation**
Regular file	The complete path of the file is specified. If we put a hyphen (-) in front, it is not necessary for the file to be synchronised each time we write in it (although the log will be lost if the power fails).
Named pipe	This system makes it possible to redirect the messages to a pipe created before starting up the daemon `sysklogd` with the `mkfifo` command. It is indicated by putting the vertical line character (|) before the file name. It is very useful for program debugging operations.
Console	By specifying `/dev/tty`*X* where *X* is a console number or `/dev/console`, the logs will be redirected to the specified screen.
Remote machine	To specify that the logs must be redirected to a remote machine, we must precede the name of the remote host with the commercial at symbol (@).
Users	If we specify the name of the user or users (separated by commas), the corresponding logs will be redirected to these.
Users online	With an asterisk (*) we will specify that the logs must be redirected to all of the users who are in the system when the log occurs. This is used to warn the users that some critical action has occurred in the system.
(The actions can be put in all the selectors that we may want).	

This way of treating the logs provides a lot of flexibility if we want to configure them when we install a server, which is a very important task if we need to control the aspects that most interest us in the system . Nevertheless, if we had to save all the logs generated in a server, we would certainly end up saturating the disk due to the growing size of these files. To avoid this, there is a logs rotation system, which consists of regularly compressing these files and only saving those that have been around for a certain period. Although they are generally compressed once a week and only those dating back one or two months are saved, we can configure all of this using the `/etc/logrotate.conf` file. The logs of certain servers and/or applications can also be configured explicitly so that they can be controlled properly. The personalised configuration of the logs for these applications are usually located in `/etc/logrotate.d/`. Internally, the system uses some programs to handle this whole log system in a better way. We can use logger to write in the system's log system. `savelog` and `logrotate` can also be used to save and, as an option, compress some of the log files that we have (with the latter, we can configure more options than with the former). These commands can also be used to create our own log files or, if necessary, to manually handle those of the system (for more information on how to treat and handle these files, please see the manuals).

Observation

If we want to configure a console in the system so that we can see all the logs that are generated, we can add the line `*.* /dev/ttySX` (where *X* is the console where we want to see the logs) to the `/etc/syslog.conf` file and reboot the `sysklogd` daemon.

Legal aspects of administration

Depending on the server that we are administrating, we have to take into account the legislation in force (according to the country), which, in some cases, requires that we preserve the files containing some types of logs for a specific period of time.

7.4.2. Regular executions (cron)

Many of the administration tasks for a server must be performed regularly. There are also lots of actions, such as updating the logs or the internal databases that use certain commands, which need to be performed regularly for the system to work properly. For this reason, it is very important for the operating system to provide us with some kind of tool for configuring all these regular executions efficiently.

The `cron` daemon is the one that handles the whole system for all regular executions. It is very simply organised: the `/etc/crontab` file is where the internal configuration of the daemon is saved; the `/etc/cron.daily/`, `/etc/cron.weekly/` and `/etc/monthly/` directories, contain the shell scripts of the programs that will run every day, week or month, respectively. There is also `/etc/cron.d/`, where we can place the files in a special format for configuring the execution of certain programs in a more flexible manner.

Configuring cron

Many of the applications of the system require some form of regular update, generally configured using `cron`. If the computer is switched on all day, it is important to configure `cron` properly, so that the operations take place at a time that we know the computer will be switched on.

Generally, we will find the following directives in `/etc/crontab`:

```
SHELL=/bin/sh PATH=/usr/local/sbin:/usr/local/bin: /sbin:/bin:/usr/sbin:/usr/bin #m h dom
mon dow user command 25 6 * * * root test -e /usr/sbin/anacron || run-parts --report
/etc/cron.daily 47 6 * * 7 root test -e /usr/sbin/anacron || run-parts --report
/etc/cron.weekly 52 6 1 * * root test -e /usr/sbin/anacron || run-parts --report
/etc/cron.monthly
```

The definition of the variables SHELL and PATH, which are used to tell the daemon to interpret the commands that are to be used and what the PATH will be. The following three lines are formed by the fields: <minute> <hour> <dayMonth> <month> <dayWeek> <user> <command>. The first five indicate when the corresponding command should run (they must be coherent) and the sixth shows the user account that will be used to run the command specified in the last one. Note how in the configuration file, the commands that run once a day, once a week or once a month, are in charge of launching the shell scripts that are in the specified directories. If the anacron program exists, they will run with it; otherwise, run-parts will be used, which, whilst it does not have as many functions as anacron, can also be used to run all the shell scripts that are in a specific directory. This configuration is the one that allows us to have the whole directory structure that we discussed above. If we wanted to, we could change the directives of this file to adapt them to our needs.

anacron program

If we install GNU/Linux in a computer that is not switched on all day, it is advisable to install the anacron program because it will run the scripts configured with cron appropriately (even if at different times than expected).

Configuration of regular executions

If we use the /etc/crontab file to configure our own regular executions, each time that we modify it, we have to reboot the cron daemon. This will not be necessary if we use its directory structure.

Although we could also use this same configuration file for entering our own commands, it is better to use the directories structure provided by the daemon itself. The only one that does not appear in this configuration is /etc/cron.d/, which the daemon already takes into account automatically. We can put files with exactly the same syntax as /etc/crontab in this directory so as to schedule personalised executions. This means that we have total flexibility.

Whilst it is advisable to use the whole of this structure for regular administration tasks in the system, when it is the users that configure some regular task, it is more usual to use particular files for each one of them. The crontab command can be used to enter the -u USER -e parameters and the particular configuration file for the specified user will be edited. The particular user files are saved in the directory /var/spool/cron/crontabs/ (in fact, the /etc/crontab file is the same as the particular file of the root user). In order to limit which users can use this daemon, we can edit the /etc/cron.allow and /etc/ cron.deny files, where we can specify, respectively, the list of users that we allow to use cron and those that we don't.

Observation

The crontab command, used when saving the file, checks that the syntax is correct.

7.4.3. Delayed executions (at and batch)

Although cron makes it possible for certain operations to take place regularly, every certain period of time, the atd daemon makes it possible to execute a command or run an application at a given moment. As occurs with the preceding daemon, we can configure which users can use it or not, using the /etc/at.allow and /etc/at.deny files. In this case, we do not have an explicit configuration file for the daemon, so it is the at command that we use to specify the moment at which we want a certain operation with the syntax to occur: at f *file TIME*. Generally, the file is a program or shell script created by the same user, where the instructions that need to be executed are written. Specifying the *TIME* can be a very complex procedure, as it can determine an *HOUR* with the format hh:mm, a time following the time of execution with now

System load

The system load is a parameter that tells us the amount of activity taking place in the computer. We can use the top command to see this load in an interactive format.

`+ XX minutes` etc. (all the possible formats are specified in the corresponding manual). With `atq` we can see the operations that are delayed and `atrm` can be used to delete any of the ones that are in the queue. Finally, if we want to run all the jobs in the `at` queue, we can use the `atrun` command. This allows us to enter the `-l LOADAVERAGE` parameter where `LOADAVERAGE` must be a number that indicates the moment at which the delayed commands may execute when the system is loading. This links directly with the `batch` command, which does exactly the same thing as `at` and follows the same syntax, but with which we need to specify the exact execution time. The operations configured in this queue are finished when the system load goes below 1.5.

Mechanisms of the run queue

The whole system of `at` and `batch` works with certain execution queue mechanisms (one for each of the two). Although we can configure the system further, generally, these will be enough to carry out any type of delayed execution in the system.

In this way, when we need to run a specific command at a specific time, we need to use the `at` command, whilst, for other operations that we wish to perform without them getting in the way of the normal working of the computer, we must use the `batch` command. The `/var/spool/cron/atjobs/` and `/var/spool/cron/atspool/` directories are where the files corresponding to all the delayed tasks are stored.

8. Installation of applications

8.1. Introduction

Managing and handling the packages is an essential aspect of any GNU/Linux distribution. A package is one or various programs, libraries or software components that are bundled into one single file, which is prepared to be installed and integrated in the operating system. When designing any distribution, it is very important to provide the tools that are necessary for installing and managing the packages properly. Tools should also be provided, especially for software developers, so that other new packages can be created. These packages usually include the program's executables and its dependencies and conflicts with other applications. When a package is installed, the dependencies indicate whether other programs are needed for the application to work properly, whereas the conflicts tell us about any incompatibilities between the installed programs and the ones that we want to install. The package systems are designed so that new applications can be installed easily, as some libraries are used by more than one program and it would not make sense to have the applications that use them install them again.

Currently, most distributions use one of the two package formats that are most widely used in the GNU/Linux world: *`deb`* or *`rpm`*. On the one hand, the `deb` packages are the ones that the Debian GNU/Linux distribution uses in its distribution, whereas `rpm` (Redhat Package Manager) is native to RedHat. The distributions based on one of these two generally adopt the corresponding package system, although most other distributions have also adopted one of the two systems, as the great majority of the programs are packaged using these formats. On the other hand, the programs with a GPL license or similar, are also usually distributed with their source code (packaged and compressed with some standard format, such as tar). We can also install the program in our operating system using this source code, by compiling it and placing the executables in the corresponding location.

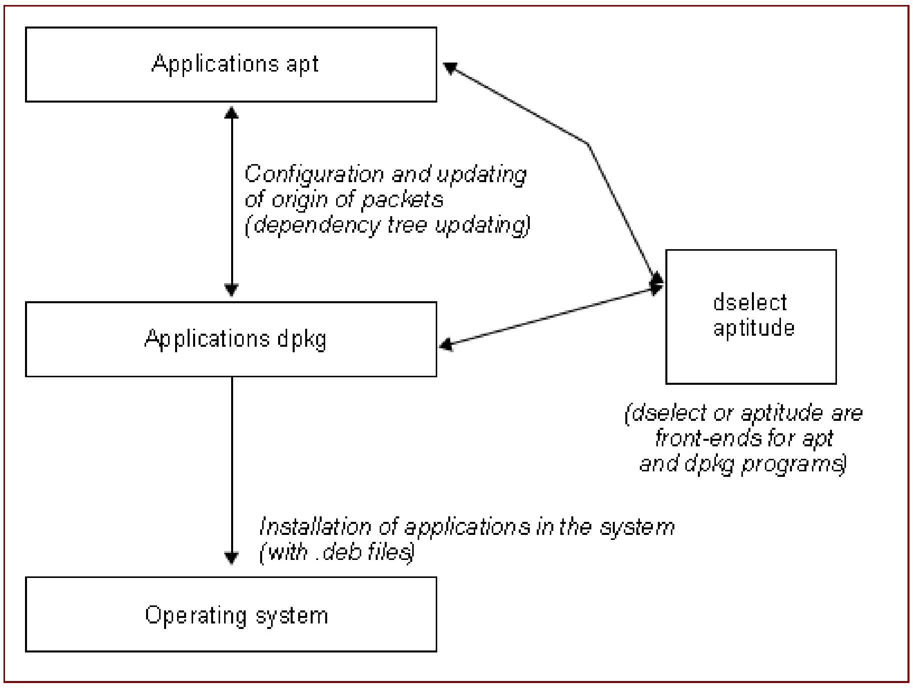
In this section, we will look at how the package system of the distribution is organised Debian[5] due to the large amount of tools that are provided and the flexibility of its configuration. In the last subsection, we will learn how to install a program using its source code, as in some cases we might find that the program that we need is not packaged. This installation method was the one that was always used before the first packaging systems appeared, which were created to make the whole process easier.

(5)Debian GNU/Linux was the first distribution to create a package system.

8.2. Debian packaging system

The applications that are used to manipulate the Debian GNU/Linux packaging system are basically one of two types: `apt` (Advanced Packaging Tool) and `dpkg programs` (Debian Package). The `apt` set of applications is used to configure where we obtain the packages from and which packages we want and it resolves any dependencies and conflicts with others. The `dpkg` programs can be used to install packages, configure them, find out which ones have been installed etc. There are other applications, such as `dselect` or `aptitude`, which can be used to manipulate the `apt` and `dpkg` programs, providing, in one single environment, interactive tools. The following diagram shows the schema:

Figure 8.1


Applications apt
Configuration and updating of origin of packets (dependency tree updating)
Applications dpkg
dselect aptitude
(dselect or aptitude are front-ends for apt and dpkg programs)
Installation of applications in the system (with .deb files)
Operating system

The programs that, in the last instance, install the applications are the `dpkg` programs. These programs decompress the `.deb` file and install the program. The `apt` applications help us to locate the latest versions of the programs that we need, copy the files from the sources from where we have extracted them (FTP, CD-ROMs, Internet etc.) to the disk and check for any dependencies and conflicts between the old and new packages so that they can be installed properly. The main `apt` applications are the following:

- `apt-config`: this application is used to configure some of the `apt` options (the architecture of our system, the directory where we save the files, etc).
- `apt-setup`: this application is used to configure the sources of the packages (from where we obtain them).

- `apt-cache`: management of the packages cache (directory where the `.deb` files are saved before they are installed).
- `apt-cdrom`: application for managing the CD-ROMs that contain packages.
- `apt-get`: updating, installing or downloading the packages.

The whole configuration of `apt` is located in the `/etc/apt/` directory. The `/etc/apt/sources.list` file is where the configuration of the sources (origins) of the packages is saved. From these sources, a list of the available packages is generated, which we can consult and install whenever we want to. Generally, the format of this file uses the following syntax:

```
deb http://site.http.org/debian distribution section1 section2 section3 deb-src
http://site.http.org/debian distribution section1 section2 section3
```

dpkg programs

Although the `apt`applications can also be used to install packages, the only thing that they do is call up the programs called `dpkg`.

The first field in each line indicates the type of file that we are referring to: binaries (`deb`) or source code (`deb-src`). Next, we will find the reference to the source of the packages, which may be a CD-Rom, an Internet address etc. The distribution field tells `apt` which version of Debian GNU/Linux we are using. This field is important because each version of the distribution has its own packages. We can use the last fields to specify what types of packages we wish to use.

Packages from the source code

The systems of packages also make it possible to create packages with the source code of the applications. If all we want to do is use the application, it is not necessary to download the source code packages.

If we were to change this file manually, we could use the `apt-get update` command to update all the packages available in the system. In order to insert the packages of a CD-ROM in the list of available packages, we could use `apt-cdrom add`, with which we would explore the inserted CD and update the list of packages in the system. If any of the sources were to contain identical packages, on installation, the `apt` application itself will detect which of the packages is the most recent or which would take the least time to download and then download it from the corresponding source. In addition, we can use the `netselect` program to configure the whole of the download system in more detail.

Another very interesting option that most distributions provide is that of updating the packages in which any vulnerability or bug has been discovered. With Debian, all we have to do is add the following line in the `/etc/apt/sources.list` file:

```
deb http://security.debian.org/ stable/updates main contrib
non-free
```

As critical packages are detected, they are placed in this source, so that by running `apt-get update`, we can find out what new updates have to be performed in the system and the required packages can be reinstalled.

Although the `dpkg` programs can be used to change any aspect of the packages installed in the system, create new packages, modify the installed ones etc., we will only look at the most important ones at the user-level in this course, so that we can carry out all the basic operations. The main `dpkg` programs are as follows:

- `dpkg-divert`: can be used to set the place where we will install some of the installation packages in the system. It is very useful for avoiding problems with dependencies.

- `dpkg-reconfigure`: the `deb` package itself very often includes some mechanism to configure some of the options of the application interactively. With this application, we can reconfigure the package that we want using the same mechanisms used during installation.

- `dpkg-scanpackages`: this program can be used to scan a specific directory of the system that contains `.deb` files, so that an index file is generated. We can use this index file to include the directory as another source of `apt`. It is very useful when we download unofficial programs of the distribution.

- `dpkg-scansources`: this is an application with the same functions as the preceding one but for packages of source code.

- `dpkg-split`: this program is used to divide and unify a package in various different files.

With these programs, we can manipulate our packages in any way we wish to. The main application, `dpkg`, is the one that allows us to install, list or delete the packages in the system. In order to list the available packages, we can enter the `-l` parameter which will show a complete list of the packages and their installation status (installed, installed but not configured etc.). If we wanted to see all the information of a specific package, we could use the `-p` parameter followed by the name of the package, which would show all the dependencies, conflicts with other packages, version number, description etc.

To install new packages, we could use the `-i` parameter followed by the name of the file. If there are problems with the dependencies, we can ignore them with `--ignoredepends=`*X*, where the *X* indicates the dependency, although we must be very careful with the way in which we use this parameter because, when we ignore dependencies, the installed program might not work properly. If all we want to do is decompress the `.deb` file to see what it contains, we could also use `-x`. In order to delete the packages, we must enter `-r`, followed by the name of the package, which would delete them from the system, but saves the configuration files (we can use `-P` to delete everything).

The `dpkg` utility

With `dpkg`, we can also use templates to select, install, delete etc. the packages in the system.

Another very interesting parameter is `--force-things` *X* (where *X* is one of the following options), which can help us if any of the following situations arise:

- `auto-select`: this automatically selects the packages that must be installed or uninstalled with the new package that we choose.
- `downgrade`: this installs the package even if there are more recent versions of it.
- `remove-essential`: although this package is considered essential for the system, this will delete it.
- `depends`: this does not take the dependencies into account, but considers them as alerts.
- `depends-version`: this does not take any of the dependencies of the version of packages into account.
- `conflicts`: this installs the package, even if it is in conflict with any other package in the system.
- etc.

Although all the programs that we have discussed in this subsection have many options and there are many more programs, the ones we have described should be enough (with the package system of the distribution that we are using) to carry out almost any task that is necessary. Although we have already mentioned that programs such as `dselect` or `aptitude` can be used to carry out the basic installation and deletion procedures on the packages, it is important to learn about these other commands properly because they can be very useful for performing specific operations or automating the selection and installation processes.

8.3. Compilation of new programs

When administrating any server, we will probably find that in some cases we have to use some program that our distribution does not have or that we need the latest version of an applications server that has not yet been packaged properly etc. In these cases, we can always download the source of the program and compile it manually. It is important to understand the difference between compiling a program to obtain its executable and directly downloading the binary. When we compile a program, this uses the libraries available in the system, whereas if we download it directly, what will most probably happen is that it will not work properly because it will try to use some library that is not exactly the same as the one installed in the system. For this reason, the most advisable thing to do when we need to install a new program and we do not have the corresponding package, is to recompile it.

When we download the sources of an application, we will obtain a file that has been packaged and compressed with `tar` and `gzip` or similar. There will usually be an added file called `README`, which explains each of the steps that have to be taken in order to compile the program properly. Although reading it is advisable, in most cases the installation process is the same.

The first thing that we must do to compile the new program is to decompress it and extract it from the package. Once we have done this, we should have the source code structured into various directories. In its root, we can find (or not) a file called `Makefile`. This file tells the compiler which libraries are used, how the code files must be compiled etc. If we have this `Makefile`, we can already compile the program by running `make`. It is not necessary to enter any parameter because, by default, it already looks for the `makefile` file and executes the actions specified therein. If the process does not result in an error message, we can then move the executable that has been generated so as to put it in one of the directories of the configured `PATH`; by doing this, we will not have to write the complete path every time we want to run it. Many `makefiles` also provide instructions so that we do not have to carry out the last step. Generally, by running `make install`, the same program will be in charge of putting the binaries and the documentation files, if applicable, in the right place. Finally, if we do not want to save the program's source code, we can delete all of the contents of the directories we have created.

Compiling a program

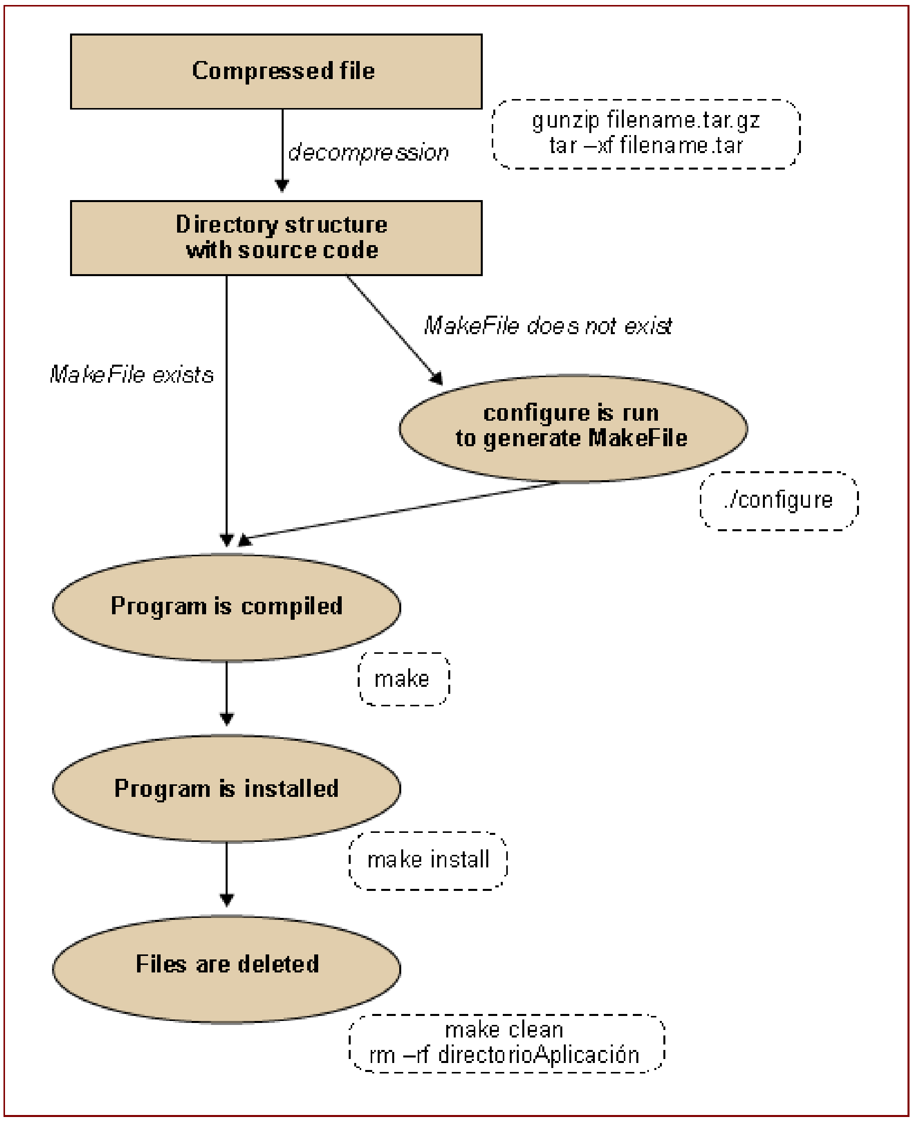
When we compile a program, we will most likely have to have the sources or headers of the libraries that are used in the system installed. Generally, these packages usually have the same name as the library, but with an added "-dev" (which stands for development) at the end.

Compiling time

The compilation process for a program can take from seconds to hours, depending on the application in question. For example, compiling the Linux kernel will take anything from 5 or 10 minutes to 2 hours (depending on the version of the kernel and the power of the computer in question).

If the program does not incorporate the `makefile`, it usually includes some form of shell script so that the file can be generated automatically (generally, this script is usually called `configure`). When we run this shell script, the system will be checked to ensure that all the libraries that are necessary for a proper compilation have been installed and an alert message will be generated if any library is not present. Once this shell script has run properly, we will have a `makefile`, which means that the process that we should then follow is as described above. The following figure shows how this process occurs.

Figure 8.2



Although most of the source code of the programs is organised as we have explained here, we might come across other types of installation. There are some that incorporate menus, environments in X or other more user-friendly methods. Recently, there are some applications that make it possible to carry out the whole installation process by downloading the code directly from the Internet.

9. Workshop on basic configurations

9.1. Introduction

In the second workshop, we have learnt how to install a basic system so that, on this basis, we can start to assemble a system tailored to our needs. This will be the subject of these last two workshops. In the first workshop, after specifying certain aspects concerning the configuration of the package installation system, we will set the foundations necessary for installing the required applications. We will then learn how to use the different tools that Debian provides for package management, in other words, how to install, uninstall, update etc. the applications and we will configure some of the most frequently used, which are necessary in most UNIX-type systems. The last workshop will cover the graphical system. In the workshop, we will learn how to install it, configure it, adapt it to our preferences and, finally, how to install and use the applications that make it possible for the system to work.

9.2. Bootloader

Firstly, we must make sure we install a bootloader system that is able to manage all the possible operating systems that we have installed in our computer without any problems. This question was already covered superficially in the preceding workshop, during the installation process. If we have already installed the bootloader and checked that it works properly, we can go on to the following subsection; however if this is not the case, in other words, if we need the backup disk we created during the installation every time we want to boot up our operating system, it is time to install a startup management system or bootloader in the hard drive to avoid the tedious procedure of using the disk every time we start the system up. In any case, it is always in our interests to install a bootloader, as it will allow us to start up different kernels, among many other things.

As we have explained, there are different bootloaders, including Lilo and Grub. Grub is a powerful bootloader from the GNU project, which is known for being able to manage the booting up process of any operating system that we have properly installed in our computer; nevertheless, using and configuring this bootloader is somewhat complicated.

In any case, if we do not have a bootloader and we want to install one, the first thing that we must do is start up the operating system that we have and check whether the application is installed; to do this, we can use the `man` command for the application, for example.

9.2.1. Installing Grub

Currently, Grub is the default bootloader. In any case, if we need to reinstall it, we should read the following subsections to learn about the package managing system, its configuration and how to use it; or, simply, if we have entered the contents of all the CDs/DVDs into the database during installation or we have decided to carry out the installation using the Internet, we should run the following line:

```
brau:~# apt-get install grub grub-doc
```

If we do not want to install the documentation package, we can omit the last parameter from the line above.

One advisable way of working with Grub, is to use, before writing in the MRB, the tests that we think may be appropriate for the disk and check that it works properly when booting from it. To do this, we must first copy the necessary files in `/boot/`:

```
brau:~# cp /usr/lib/grub/i386-pc/stage* /boot/ brau:~# cp /usr/share/doc/grub/examples/menu.lst
/boot/
```

Once we have done this, we will edit Grub's configuration file, `/boot/menu.lst`, and we will tailor it to our needs. This operation can be somewhat complex due to, among other factors, the change in the nomenclature where the hard disks are concerned; therefore, in order to identify the device in which a specific file is located, we can use the `find` command, once we have entered Grub's command mode, using `grub`. Therefore, for example, if we want to find a file, whose secure location we will have to specify in Grub's configuration file, such as `/vmlinuz`, we should proceed as follows:

```
brau:~# grub GRUB version 0.91 (640K lower / 3072K upper memory) [ Minimal BASH-like line
editing is supported. For the first word, TAB lists possible command completions. Anywhere
else TAB lists the possible completions of a device/filename. ] grub> find /vmlinuz (hd2,0)
```

Grub has many more commands, for a list of some of these, simply press the Tab key in the command line to obtain the following:

```
grub> Possible commands are: blocklist boot cat chainloader cmp color configfile deb ug
device displayapm displaymem embed find fstest geometry halt help hide impsp robe initrd
install ioprobe kernel lock makeactive map md5crypt module moduleno unzip partnew parttype
password pause quit read reboot root rootnoverify savede fault serial setkey setup terminal
```

```
testload testvbe unhide uppermem vbeprobe grub>
```

To obtain help on any specific command, all we have to do is enter `help` on the keyboard followed by the name of the command. A good way of working on our Grub configuration file, `/boot/menu.lst` is to open a new session in a different tty and to gradually modify the file as we are working on Grub's commands interface.

Once we have finished editing the configuration file, and entered a blank disk in the disk drive, we should use the keyboard to key in the command, substituting the *X* with the number of the hard disk, and *Y* with the corresponding partition with the following command line:

```
grub> install (hdX,Y)/boot/stage1 d (fd0) (hdX,Y)/boot/stage2 p (hdX,Y)/boot/menu.lst
```

With the preceding line, we have transferred the information of the MRB (`stage1`), Grub and its command line interface (`stage2`) and the boot up menu (`/boot/menu.lst`) that we have configured in the file to the disk (`fd0`). We should then be able to reboot the machine, boot up using the disk that we have just created and check whether our configuration is correct.

It is very interesting to have a Grub diskette, as, using the command line interface, we can try to start up the different operating systems installed in the computer directly, so as to perform tests to debug the contents of the `/boot/menu.lst` file. As an example, the procedure that has to be followed in order to boot up a Microsoft^{TM} system installed as a `(hd0,0)` is explained here and we will then discuss the commands necessary to boot up a GNU/Linux system installed as `(hd1,0)`:

For a Microsoft system:

```
grub> rootnoverify (hd0,0) grub> makeactive grub> chainloader +1 grub> boot
```

For a GNU/Linux system:

```
grub> kernel (hd1,0)/vmlinuz root=/dev/hdc1 grub> boot
```

Once we have the definitive bootloader that we wish to implement in our diskette, all we have to do is transfer this same configuration to the MBR of the booting hard disk; using the same command line of the diskette, we should key in:

```
grub> install (hdX,Y)/boot/stage1 d (hd0,0) \ (hdX,Y)/boot/stage2 p (hdX,Y)/boot/menu.lst
```

As we have seen, using Grub is somewhat complex, which is why we think it is very advisable to read the man files and documents that have been installed with the `Grub-doc` package, which can also be seen at http://www.gnu.org/software/grub/sto, before attempting to install and configure it. .

9.3. The package system

We will now analyse and learn how to use Debian's package system, which is probably the most solid and reliable of those existing in the world of GNU.

In the following subsections, we will learn how to configure its database, how to use it, install different packages, update them etc. There are often many different ways of achieving the same result. Some of these methods will be explained here, with two main objectives: the first is for readers to be able to choose which one they want to use, and the second is for readers to always know more than one solution to the same problem, in case any of these methods does not work.

Activity

9.1 For a comprehensive understanding of how Debian's package system works, we recommend reading:

- APT HOWTO: http://www.de.debian.org/doc/manuals/apt-howto/index.en.html
- http://www.de.debian.org/doc/manuals/ debian-faq/ch-pkg basics.en.html
- http://www.de.debian.org/doc/manuals/ debian-faq/ch-pkgtools.en.html
- The man of: `apt`, `apt-cache`, `apt-get`, `sources.list`, `dpkg` and `dselect`.

In order to make reading easier, we could wait until we have configured the printer and learnt how to print the man.

9.3.1. /etc/apt/sources.list

The file `/etc/apt/sources.list`is the core of Debian's package system configuration. As it is a text file, like most configuration files in UNIX-type systems, we can edit it manually or using some of the tools that the system has for this purpose. The content of this file will depend to a large extent on the speed of our Internet connection, supposing we have a connection. However, we should never forget to run the following command as the root, once we have modified the file:

```
brau:/etc/apt# apt-get update
```

If we neglected to do this, the database of the package system would not be updated and therefore, none of the changes that we had made would have any effect. In an installation in which the packages are obtained remotely, this command must be run regularly, so that the database is kept updated; for this reason, it is not a bad idea to include it in the `cron system`.

Each line of this file refers to a source of packages and the fields are separated by a space. Firstly, we must specify whether the source is binary packages `deb` or if it is source code `deb-src`. The second field is used to specify how these packages are accessed: `CD-ROMs`, `http`, `ftp` etc. followed by the address. The remaining fields refer to the type of packages that we wish to access using this line.

In the most ideal case, we should have a fast Internet connection. This will probably already have allowed us to carry out the installation of the basic system using the Internet, as well as provided us with the latest versions of the packages. In addition, it is the most comfortable way of working with the packages, as we do not even have to worry about inserting the correct CD in order to carry out the installation.

However, what we must do, before doing anything else, is to ensure that the content of `/etc/apt/sources.list` is correct. For example:

```
deb http://ftp2.de.debian.org/debian/ stable main non-free contrib deb-src
http://ftp2.de.debian.org/debian/ stable main non-free \ contrib deb
http://non-us.debian.org/debian-non-US stable/non-US main \ contrib non-free deb-src
http://non-us.debian.org/debian-non-US stable/non-US main \ contrib non-free deb
http://security.debian.org/ stable/updates main contrib non-free
```

All the addresses given above are official; in other words, they are recognised by Debian. Apart from these sources, we can also use unofficial packages; just because they are unofficial does not mean that their quality is not good enough to include them in our configuration file. A good address where information on the location of these packages can be found is http://www.aptget.org.

Once we have edited and saved the file, we should run the `apt-get update` command; after a few moments, during which the package system will reconfigure its database and display the different accesses performed on the screen, we should have access to the new packages.

If we do not have an Internet connection or the one that we have is too slow, we shouldn't hesitate to use the set of CDs/DVDs of the distribution to install the different packages. If we have not inserted the CDs or DVDs during the installation process up to this point, now is the time to do so. We should insert the first CD-ROM/DVD in the reader and use the `apt-cdrom add` command to include its contents in the database:

```
brau:/etc/apt# apt-cdrom add Using CD-ROM mount point /CD-ROM/ . . . Repeat this process
for the rest of the CD in your set.
```

Having reached this point, we should repeat the same process for each and every one of the distribution's CDs/DVDs. Likewise, we can use the same procedure to incorporate data from unofficial CDs/DVDs. Once we have configured our Internet connection, if necessary, we can include the sources of remote access packages. In order to do this, we should edit the file called `/etc/apt/sources.list` and then run `apt-get update`, to have the new packages.

9.3.2. apts available.

`apt` is an abbreviation for Advanced Package Tool, which, as we have discussed on various occasions, is the basic system in charge of administrating the packages of distributions based on Debian. `apt` essentially provides us with two tools: `atp-get` and `apt-cache`. The first command can only be used by the root user of the system, as it is the package management tool: installing, uninstalling, updating etc.; whereas, the second, as it is a command used to find information in the database, whether it concerns installed packages or packages that have not been installed yet, can be used by any of the users.

In order to make it easier to manage the packages, other applications have been developed; these run on a level above `apt`, such as the `dpkg` middle-end or the `dselect` or `aptitude` front-ends. But before looking at the different package management systems, we should learn some concepts about these systems and how the system and the management system relate to each other.

Types of package, according to their priority

Within the package system, there are five distinct types of packages, which are defined by their degree of dependence on the system itself. In their decreasing order of priority, they are classified as:

- `Required`. These are packages that are essential for the system itself to work properly.
- `Important`. These are the packages that should be present in any UNIX-type system.
- `Standard`. These are packages that are commonly found in a GNU/Linux system. Generally, they are small applications, which are no longer essential for the system.
- `Optional`. These are packages that may or may not be present in a GNU/Linux system. Among others, this group includes all the packages referring to the graphical system, as it is not considered essential. In reality, in many servers, the graphical environment is discarded, with a view to increasing the system's performance levels.

- `Extra`. These are packages that either present conflicts with packages with a higher priority or that require special configurations that render them inapt for integrating them as optional packages.

We can determine which group a specific package belongs to, using, for example, the sentence `apt-cache show` *`<name of the package>`* and checking the contents of the Priority field.

The degree of dependency between

`apt` packages is characterised by the large amount of consistency when it comes to managing the dependencies between packages. For example, it is possible that a given application that we want to install depends on a library and, consequently, on another package that we have not installed. In this case, `apt` will tell us about this dependency and will ask us whether we want to install the package that contains that library, along with the application that we are installing. The relationships between packages are classified as follows:

- `depends`. The package that we want to install depends on these packages and, consequently, if we want this package to work properly, we must allow `apt` to install the rest of them.

- `recommends`. The person in charge of maintaining the package has decided that the users that are going to install this package will normally also use the ones that are recommended.

- `suggests`. The suggested packages make it possible to obtain a better performance than that obtained from the package that we were going to install.

- `conflicts`. The package that we will install will not work properly if these other packages are present in the system.

- `replaces`. The installation of this package means uninstalling other packages.

- `provides`. The package that we will install incorporates all the content of the abovementioned packages.

We can determine the dependencies of a package with, for example, the sentence `apt-cache depends` *`<name of the package>`*.

Actions performed on the packages

Using the `dpkg` or `dselect` flags, we will obtain information on what the user intends to do with those packages.

- `unknown`. No reference has ever been made to the package in question.
- `install`. The package has to be installed or updated.
- `remove`. The package is going to be uninstalled, but without deleting its configuration files (which are usually located in `/etc/`).
- `purge`. The package will be completely uninstalled, and the configuration files deleted.
- `hold`. This is used to indicate that no operation should be performed on this package (in other words, its version and configuration should be maintained until such an operation is explicitly commanded).

Status of the package installation

Within the package system, we will find:

- `installed`. The package has been installed and configured correctly.
- `half-installed`. The installation of the package began but, for some reason, it was not completed.
- `not-installed`. The package has not been installed in the system.
- `unpacked`. The package has been unpacked, but not configured.
- `config-files`. Only the configuration files of the package in question exist in the system.

apt-cache

As we have mentioned, `apt-cache` is a command used to analyse the package system and, therefore, as it is not potentially dangerous for the system, all users can access it. The parameters that are most often used for this command are as follows:

- `search pattern`. This searches the database for the packages whose name contains pattern or packages that contain the word pattern in their description (if the result is a very long list, given that patterns are very common, we can use pipes and `grep`
- `show package`. Provides information on the package.
- `policy package`. Provides information on the installation status, the version and revision number of the package, as well as its origin.

- `depends package`. This lists the package's dependencies.
- `show package`. It shows the direct and reverse dependencies of the package.

apt-get

`apt-get` is the command used to manage the system's packages. This is why it can only be used by the system's root user. The parameters that are most often used for this command are as follows:

- `install package`. This installs the package. If this depends on packages that are not found on the system, `apt` will report this to us and will ask if we want to install the packages that the package that we want to install depends on and that are not installed, at the same time; generally, it is in our interest to follow the advice provided by `apt`.
- `update`. This updates the `apt` database. This command must run every time that the file is modified. `/etc/apt/sources.list`.
- `upgrade`. Forces the upgrading of the packages installed to the latest available version.
- `remove package`. This deletes the package, without deleting the configuration files, in case any reinstallations need to be performed.
- `remove -purge package`. This completely deletes the package, including the corresponding configuration files.
- `clean`. This eliminates the expired copies of the packages that have been gradually installed; during the process, a copy of the unpackaged package is saved automatically in `/var/cache/apt/archives` whenever a package is installed. This is a very useful command for freeing up space on the hard drive occupied by files that, most probably, will never be used again.
- `autoclean`. This eliminates all the unpackaged copies of the packages, regardless of the expiry date.

9.3.3. dpkg

`dpkg` stands for Debian Package Manager and it was conceived as a `apt` backend. The most commonly used parameters are the following:

- `-l`. To list the packages in the database and their installation status (this option is generally combined with grep).

- `-L package`. To list the files contained in the package.
- `-r package`. This has the same effect as `apt-get remove package`.
- `-P package`. This has the same effect as `apt-get remove -purge package`.
- `-p package`. This has the same effect as `apt-get show package`.
- `-s package`. Describes the package's installation status.
- `-S file`. Searches for the packages to which the file belongs.

9.3.4. dselect

`dselect` is a GUI (Graphical User Interface) that runs on `apt`. To enter it, all we have to do is key in the command `dselect` and, using the interface's menus, go selecting the different packages that we wish to operate on and specify the type of operation that we wish to carry out.

9.3.5. aptitude

`aptitude` is another GUI that runs over `apt`. It is not installed by default, which means that we have to install it before we can use it.

```
brau:/etc/apt# apt-get install aptitude
```

Once it has been installed, we can launch it using the `aptitude` command and we will immediately see that using it is just as simple or even simpler than using `dselect`, as it has drop-down menus that can be called up pressing F10.

9.4. locales: regional configuration

Although it may appear that our keyboard is working properly, as we can use accents, umlauts and other characters that are not English, as we adapt the system to our needs, and especially when we want to install the graphical interface and make the applications run in it, as we will in the next workshop, we will observe that this is not the case. At this point, we could proceed to configure these aspects properly so that we do not have to do so later.

Firstly, we should check whether the `locales` package has been installed:

```
brau:/# dpkg -l| grep locales ii locales 2.2.5-11.2 GNU C Library: National Language (locale)
results in
```

If we do not obtain the last line, we must proceed to install and configure the package:

```
brau:/# apt-get install locales
```

And if we do have it, we should key in the following line to reconfigure it:

```
brau:/# dpkg-reconfigure locales
```

Of the many options that we are offered, we choose [*] `es ES UTF8`, in other words, we highlight the option and press the space bar. We use the Tab key to highlight `OK` and press Enter. In the next screen, we select C. Back in the command line, we edit the file `/etc/environment` so that it is left as follows:

```
LC_ALL=es_ES LANGUAGE=en_US LC_TYPE=es_ES LC_MESSAGES=UTF8 LANG=C
```

To make the change effective, all we have to do is key in the `locale-gen` command and we will come out of all the sessions that we may have opened to load the new configuration.

9.5. The main booting file, /etc/inittab

Although the process for booting up a GNU/Linux system is complex, this section only works on one of the main files of this process: `/etc/inittab`. During the booting process, this file indicates, among other elements, the runlevel at which we will start and define the processes that will launch automatically during the booting procedure. To learn which runlevel we are in, all we have to do is key in the `runlevel` command. To change the runlevel, to home, we use the `init <`*`destination runlevel`*`>` instruction.

Additional reading

To learn more about `locales`, we recommend visiting the website:

http://www.uniulm.de/ssmasch/locale/.

It is interesting to open this file and become familiar with its contents, as this will provide us with a better understanding of the process for booting a GNU/Linux system.

9.6. Assembly of devices, /etc/fstab

`/etc/fstab` is the file that contains the information on the partitions and devices that will be set up automatically during the booting process and the ones that can be set up subsequently, as well as information on who is permitted to carry out these procedures. We will now show the possible contents of this file and analyse them.

```
# /etc/fstab: static file system information. # # <file system> <mount point> <type>
<options> <dump> <pass> /dev/hdg1 / ext3 errors=remount-ro 0 1 /dev/hdg2
none swap sw 0 0 proc /proc proc defaults 0 0 /dev/fd0 /floppy auto user,noauto 0 0 /dev/hdg5
```

```
/usr ext3 defaults 0 2 /dev/hdg6 /var ext3 defaults 0 2 /dev/hdg7 /home ext3 defaults 0 2
/dev/CD-ROM /CD-ROM iso9660 ro,user,noauto 0 0 /dev/hdc1 /mnt/hdc1 ntfs ro,user,noauto,
gid=windows,umask=0007,utf8 0 0 /dev/hde1 /mnt/hde1 ntfs ro,user,noauto,gid=windows,
umask=0007,utf8 0 0 /dev/hde5 /mnt/hde5 ntfs ro,user,noauto,gid=windows, umask=0007,utf8 0 0
/dev/hde6 /mnt/hde6 vfat utf8,user,noauto 0 0 /dev/hde7 /mnt/hde7 vfat utf8,user,noauto 0 0
```

The first lines were automatically generated by the installation process and in them we can see how the different directories are distributed within the GNU/Linux structure. Perhaps the line that will most call our attention will be `proc /proc proc defaults 0 0`, which is in charge of setting up the virtual `proc` directory, which we already discussed in the first workshop.

The most interesting lines are the ones like `/dev/hdc1 /mnt/hdc1 ntfs utf8,ro,noauto,user,gid=windows,umask=0007,utf8 0 0`. These specify the point of origin and the point of assembly of the partitions belonging to the Windows2000^{TM} operating system; in other words, that of an ntfs type. We cannot write in these partitions from GNU/Linux, although we can read the contents; this is reflected in the options `ro,noauto,user,gid=windows, umask=0007,utf8` (it is essential not to leave any space blank in the options, as this character is the one used to separate the fields in this file). The first indicates that it is a read-only partition; the second, that it will not be set up automatically during the system booting process; the third indicates that this partition can be set up by any user; the fourth indicates that only members belonging to the Windows group will be able to access it (as defined in the file `/etc/group`); the second-to-last option establishes the antimask of the setup and the last, the table of codes that will be used.

The last lines of the preceding file are used to mount FAT32 partitions, on which it is possible to write from GNU/Linux. For this reason, it is a good idea to always have a small partition with this type of format, as it will be accessible both from GNU/Linux and from Windows^{TM}. Although it is true that it is possible to mount a file system from the command line, as, for example, would happen if we were mounting the CD-ROM,

```
brau:/etc/apt# mount /dev/CD-ROM /CD-ROM -t iso9660 ro
```

it is a lot more comfortable to have the information entered in the `/etc/fstab` file, as this will allow us to do the same thing simply by keying in:

```
brau:/etc/apt# mount /CD-ROM
```

9.7. Device configuration

Having addressed the basics of package administration, we can cover the task of starting to configure the system so as to tailor it to our specific needs. The process basically consists of two parts: configuring the different hardware devices that we have installed in the computer, and installing the software that we will use.

Configuring the system's hardware is usually the part that requires most effort, in general, because each computer will have different devices and therefore each computer can be a world unto itself. Generally, in GNU/Linux systems, we can configure any device, however unusual it is, although this will be either more or less complicated depending on how standardised the devices are. However, it is also true that these are the processes during which we can learn the most, as, generally, configuring a device requires certain previous tasks, such as finding out exactly what type of device we have, reading all the documentation on these types of devices included in GNU/Linux systems, how the device in question integrates with the system etc.

Given that not all hardware manufacturers support GNU/Linux systems, and there are some that do not even provide the information needed so that the developers of the GNU/Linux community can write the code that is necessary for integrating these devices in the operating system, it is recommended that, whenever purchasing new hardware, we should find out exactly which product we need (this information should generally be much more precise than that usually provided by the suppliers of the hardware themselves, which, sometimes, they do not even have), whether it is fully supported, what information we will need to integrate the new product in our system etc. In general, when purchasing new hardware, the things that should be considered are the degree of standardisation and the quality of the product. Where standardisation is concerned, the more standardised the product, the higher the number of users who have it and, therefore, it is much more lkely that it is fully supported. Where the quality of the product is concerned, a few years ago, hardware manufacturers began to replace functions that would be implemented with hardware with functions implemented by software in order to reduce costs (the most famous example of this practice is possibly that of the modems known as winmodems). This results, on the one hand, in reduced system performance, as it requires burdening the CPU with new tasks, many of which were not even been designed to carry out, and, on the other hand, in the need to have the software that replaces the removed hardware and that, generally, has only been developed for certain types of operating system; for this reason, it is advisable to avoid any product that is specifically designed for a particular operating system.

Throughout each subsection covering system configuration, we will discuss some of the aspects of the different devices that we may come across and the implicit problems.

Before we start configuring the different devices in our system, we must remember some strategies that may be useful for this purpose. Firstly, the `lspci` command can be used to obtain a lot of information on how these devices are recognised by the system during the booting process. If this information is insufficient, we can always resort to the virtual `/proc/` directory, where all the information on the system's hardware, among other elements, will be recorded. Likewise, the log files, located in `/var/log/` can be very useful (an interesting method for seeing how the contents of these files evolve is to use the `tail` command with the `-f` parameter and redirect its output to a tty that we are not using; as an example, `tail –f /var/log/messages > / dev/ tty10`).

9.7.1. Configuring the mouse

As was the case in the workshop on KNOPPIX, the deamon that will be in charge of administrating the mouse will be `gpm`. We should then install the package:

```
brau:/etc/apt# apt-get install gpm
```

Upon completion of the installation, a script will launch automatically to help us configure the mouse; The parameters that we must enter are essentially the same ones that we entered during the workshop on KNOPPIX, but if anything were to fail, we could always re-launch the configuration program using the `gpmconfig` command (generally, the configuration `-m /dev/ psaux -t ps2` should be valid for most PS2 mice with three buttons). The configuration that `gpm` will use every time the system is booted is saved in `/etc/gpm.conf`. It is helpful for the mouse to have three buttons, as functions will be assigned to all three, especially in graphical environments. For this reason, if we have a mouse that only has two buttons, we can press both buttons at the same time, for the functions that the third button would usually carry out.

Starting up and stopping the `gpm`

As we have mentioned, the program in charge of administrating how the mouse works is a deamon. Therefore, we will act as we would with any other normal deamon in order to start it up or stop it. This subsection will provide an example of how to start up or stop the daemons.

Both starting and stopping a deamon are performed using a script that is located in /etc/init.d/. Generally, if we invoke this script without entering any parameter, it will show us a help line that will assist us by telling us how to use it. Let us therefore proceed to stop the gpm deamon:

```
brau:/etc/init.d# ./gpm stop Stopping mouse interface server: gpm
```

Using ps aux, we can check whether, effectively, there is no other process called gpm running, and besides, we can see that if we move the mouse, nothing happens on the screen. Now we can proceed to start it up:

```
brau:/etc/init.d# ./gpm stop Starting mouse interface server: gpm
```

Now, when we move the mouse, we will see the arrow appear on the screen. Now, let us proceed to analyse whether this deamon will launch automatically when we boot up the computer. The /etc/inittab file will tell us the runlevel in which the operating system will boot up: by default, it will be 2; therefore, in this file, we should find a line similar to the following:

```
# The default runlevel. id:2:initdefault:
```

Let us therefore check whether the directory /etc/rc2.d/ contains a symbolic link to /etc/init.d/gpm:

```
brau:/etc/init.d# ls -l ../rc2.d/ | grep gpm lrwxrwxrwx 1 root root 13 feb 21 13:03
S20gpm -> ../init.d/gpm
```

If this symbolic link does not exist, and we want gpm to launch automatically during the booting process, we should create it manually using ls –s. If, on the other hand, the link exists and we do not want gpm to launch during the booting process, all we would have to do is delete this symbolic link; however, it is not advisable to delete the scripts from /etc/init.d, as they are very useful for booting up and stopping daemons.

9.7.2. Configuring modems

As occurs with all other hardware, modems can be configured completely manually, but this practice is generally being forgotten, as over time, tools that are sufficiently powerful and reliable have been developed, which can help us to avoid the tedious task of configuring a modem manually. One of these tools is pppconfig, which is the one we suggest here to configure our modem.

However, before we start configuring our modem, we have to state that not all the devices that are spoken of or sold as such are actually modems. As we have discussed, many manufacturers, in order to reduce their costs, have started replacing physical components with software, and historically, modems were probably the fist devices to suffer from this practice. We have to be especially careful with internal modems, as in reality, very few of these incorporate all the hardware that these devices should have. Real modems are easily recognisable by the difference in price compared to fake modems. For this reason, many of these devices have been reduced to mere slaves of the software.

Bibliography

For more information on these devices and how they integrate in GNU/Linux, please go to:

http://www.tldp.org/HOWTO/Winmodems-and-Linux-HOWTO.html
http://www.tldp.org/HOWTO/Linmodem-HOWTO.html
http://www.idir.net/gromitkc/winmodem.html

For this reason, it is generally recommended that we use external modems, whenever this is possible. Regardless of whether we have a real modem or not, we recommend reading http://www.tldp.org/HOWTO/Modem-HOWTO.html. Given that we will use `pppconfig` to configure our modem, unless we installed it during the system installation procedure (you can check this by trying to launch the application directly, in other words, entering `pppconfig` on the keyboard or using `dpkg -l | grep pppconfig`), the first thing we must do is install the application:

```
brau:~# apt-get install ppp pppconfig
```

Once we are on the main screen of the installation interface, we must select Create a connection and on the following screen, we will enter the name with which we will refer to this configuration, as it is possible to configure and manage more than one connection.

After assigning a name to the new configuration, we must configure the DNS access: we can choose the default option, static DNS assignment, or a dynamic DNS assignment, if we are sure that our ISP will provide us with the DNS addresses during the connection process. If we choose the default option, we will be asked to enter the IPs of the DNS that we want to use and they will be stored in the `/etc/ppp/resolv/`*`nameofconfiguration`* file).

On the following screen, we must choose the authentication method for establishing the configuration. Generally, unless our case is exceptional, we will choose the first option, `PAP`. Next, we will enter the user name and the password for the connection and choose the connection speed; the default speed is `115200`, and with most connections, this should work without a problem. After specifying whether our telephone line is dial-based or tone-based (currently, most are tone-based), we must enter the number that should be rung, which our ISP should have given us.

Having reached this point, `pppconfig` will ask us whether we want the modem to be automatically detected. With the modem working, we can let the program itself detect the port to which the modem is connected, or we can do it ourselves manually (always remembering the correspondence: first serial port, COM1, `/dev/ttyS0`, second serial port, COM2, `/dev/ttyS1`, etc).

Once we have entered the ttyS to which the modem is connected, we will access a screen summarising the data that we have entered, which also allows us to establish certain advanced options (these are generally not necessary). If the data are correct, we can choose the `Finished Write files and return to main menu` option, and after confirming the operation, for returning to the interface's main menu, if we do not want to configure another connection, we should choose the `Quit Exit this utility` option to return to the command line. Once back in the command line, we can check whether these data have been saved properly in `/etc/ppp/peers/nameofconfiguration`. For users of PPP connections, it might also be interesting to install the pppstatus package to monitor the traffic over the connection, among other elements.

Establishing and terminating a connection pon, poff

In order to establish a connection, all we have to do is key in the `Pon` command followed by the name of the connection that we wish to use; if we have only configured one connection, it will not be necessary to specify its name. In principle, if its use is not restricted, `Pon` can be run by any of the users.

To terminate the connection, all we have to do is run the `Poff` command.

9.7.3. Configuring DSL modems

As we did in the preceding section for configuring traditional modems, here we will use a tool for configuring DSL modems: pppoeconf. But before beginning, we recommend reading http://www.tldp.org/HOWTO/DSL-HOWTO and http://www.tldp.org/HOWTO/ADSL-BandwidthManagement-HOWTO.

PPPoE (Point-to-Point Protocol over Ethernet), which is a protocol that is very similar to PPP that uses an Ethernet interface instead of a modem. This type of protocol has been adopted by some ADSL Internet providers.

The ADSL connection is generally part of the telephone service. The signals (telephone and ADSL signals) are divided in a device called a splitter, which separates them to their respective frequency ranges. The ADSL signal then goes into a device called a concentrator that is connected to the computer's Ethernet interface with a twisted pair cable with RJ45 connectors.

To configure the connection:

```
# apt-get install pppoeconf
```

When running pppoeconf, the tool will scan all the network interfaces until it finds one that accesses a concentrator. We will choose to configure PPPoE in this interface.

In ENTER USERNAME, we enter the name of the user that the provider has assigned to us and then the password. In USE PEER DNS we will choose the automatic modification option, /etc/resolv.conf. We choose to have PPPoE run when booting and then select the connection. After a few seconds you will see something similar to this on the screen,

```
Every 2s: /sbin/ifconfig ppp0 Sun May 13 12:11:56 2007 ppp0 Link encap:Point-to-Point
Protocol inet addr:200.89.50.138 P-t-P:10.52.0.3 Mask:255.255.255.255 UP POINTOPOINT
RUNNING NOARP MULTICAST MTU:1492 Metric:1 RX packets:5 errors:0 dropped:0 overruns:0
frame:0 TX packets:7 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:3
RX bytes:190 (190.0 b) TX bytes:199 (199.0 b)
```

9.7.4. Configuration of network cards

If, for any reason, we have not configured the network card during the installation process, now is the time to do so. Although practically all network cards are supported in GNU/Linux systems, as they are key components for the network-oriented operating system, again, we would recommend using the most standardised hardware possible, to avoid complications when it comes to configuring the system. It is possible that our card is supported in two different ways: the first is that its driver has been directly compiled from the kernel itself, and the second is that the driver has been compiled in a modular way so that it can be loaded subsequently. The easiest way of finding out whether the driver of our network card has been compiled within the kernel itself is by analysing the dmesg return message, as we did in section 5.4.3.

If, after analysing the output of the dmesg command in detail, we reach the conclusion that the driver for our card has not been loaded, we can run the modconf command, which loads modules into the kernel that have been compiled by it, and checks whether the driver for this appears in the kernel/drivers/net subsection. If this is the case, all we have to do is select it to load it into the kernel.

If our network card is not supported by default, we must resort to recompiling the kernel.

Once we have done this, we must edit the file /etc/network/interfaces to enter the parameters corresponding to our network for the card. One possible configuration would be:

```
# /etc/network/interfaces -configuration file for ifup(8), #ifdown(8) # The loopback interface
auto lo iface lo inet loopback # The first network card this entry was created during the
#Debian installation (network, broadcast and gateway are #optional) auto eth0 iface eth0 inet
static address 158.109.69.132 netmask 255.255.0.0 network 158.109.0.0 broadcast 158.109.255.255
gateway 158.109.0.3
```

We recommend reading http://www.tldp.org/HOWTO/NetworkingOverview-HOWTO.html, as well as http://www.fokus.gmd.de/linux/HOWTO/Net-HOWTO and the man on interfaces. If we do not have the network card and we need to carry out tests, we can always resort to the `dummy`; module; in this case, the device, instead of being called `eth0`, would be called `dummy0`.

If we wanted to configure more than one network card in the same computer (which is very common when using gateways, as well as in other cases), it is necessary to enter the parameters corresponding to the kernel during the booting process, to avoid conflicts between devices.

Starting and stopping network services: ifup, ifdown

All the network services (those of `/etc/network/interfaces`) can be restarted using the `/etc/init.d/networking` script with the `restart` parameter. `ifup`, which starts up the network services of a specific interface, and `ifdown` to stop them. Therefore, for the abovementioned configuration, if we wanted to stop and restart the eth0 services, we would proceed as follows (the `ifconfig` command can be used to check the results):

```
brau:~# ifconfig eth0 Link encap:Ethernet HWaddr 00:01:02:B4:3A:61 inet addr:158.109.69.132
Bcast:158.109.255.255 Mask:255.255.0.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX
packets:36409683 errors:0 dropped:0 overruns:221 frame:0 TX packets:35938 errors:0 dropped:0
overruns:0 carrier:0 collisions:0 txqueuelen:100 RX bytes:1489273710 (1.3 GiB) TX
bytes:20116974 (19.1 MiB) Interrupt:5 Base address:0x9400 lo Link encap:Local Loopback inet
addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:823 errors:0
dropped:0 overruns:0 frame:0 TX packets:823 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0 RX bytes:3169619 (3.0 MiB) TX bytes:3169619 (3.0 MiB) brau:~#
ifdown eth0 brau:~# ifconfig lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:823 errors:0 dropped:0 overruns:0 frame:0
TX packets:823 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX
bytes:3169619 (3.0 MiB) TX bytes:3169619 (3.0 MiB) brau:~# ifup eth0 brau:~# ifconfig eth0
Link encap:Ethernet HWaddr 00:01:02:B4:3A:61 inet addr:158.109.69.132 Bcast:158.109.255.255
Mask:255.255.0.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:36420981 errors:0
dropped:0 overruns:221 frame:0 TX packets:35965 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100 RX bytes:1490867554 (1.3 GiB) TX bytes:20118868 (19.1 MiB)
Interrupt:5 Base address:0x9400 lo Link encap:Local Loopback inet addr:127.0.0.1
Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:823 errors:0 dropped:0
overruns:0 frame:0 TX packets:823 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
```

```
   txqueuelen:0 RX bytes:3169619 (3.0 MiB) TX bytes:3169619 (3.0 MiB)
```

Wireless network card

If you have a WiFi network card, you may check the following link:
http://ftp.cl.debian.org/man-es/wlan.html.

9.7.5. Configuring printers

Having the printer configured can be very useful, as it will allow us, among other things, to print out the `man` files, the configuration files etc. so that we can study them better in paper format.

Printing in Linux is based on a queue system that is handled by a daemon that runs when booting. The printers may be connected directly to the parallel port or USB connection of the computer (local printer) or connected to the network (network printer).

Where the preferable type of printer is concerned, in a domestic environment, the best thing would be to use the parallel port (`/dev/lpX`, normally `/dev/lp0`) so as to guarantee that the printer does not depend on its software, as most of the new printers that are appearing on the market, especially those that connect through the USB port, are printers that are designed specifically for a determined operating system. These printers are popularly known as *winprinters* (and they are similar in this respect to what we call *winmodems*). In a professional environment, the best thing would be to have a printer that incorporates its own network interface, making it, simply, another node in the network. For more information on printers and how they integrate in GNU/Linux operating systems, we recommend visiting the site http://www.linuxprinting.org, where you will find an exhaustive list of the printers available on the market and information on the degree to which they are supported.

CUPS (Common Unix Printing System) is a modular printing system that uses IPP (Internet Printing Protocol) to handle the queues. This is a modern and sophisticated system, which can be used in a desktop environment such as GNOME or KDE. Furthermore, it comes with the option of maintaining the `lpr` traditional system of BSD command lines.

The general topology of a printing system in GNU/Linux is that of client-server. The server, CUPS, is a daemon-type server and we will therefore treat it as such. The server's configuration file is `/etc/cups/cupsd.conf`. In this file, we will be able to configure as many printers as we wish.

To install the cups system, we must run the command:

```
# apt-get install cupsys cupsys-client cupsys-bsd printconf foomatic-filters-ppds
```

This last package contains the PPDs (Postscript Printer Definitions) of Adobe, which is used by CUPS to define the printers.

CUPS-PDF is a virtual printer for printing documents to a PDF file. To install it in our system, we would run the command:

```
# apt-get install cups-pdf
```

In CUPS, there are two ways of configuring the printers. One of these is from the utility provided by the GNOME or KDE desktop. The other is from the website . In both cases, we need administrator privileges in order to configure the printers.

Printing text files

Formatters are programs that are used, mainly, to transcribe files in text format to PostScript format (Postscript language, historically, has been more widely implemented in the field of printing). These programs can be very useful, as they allow us to print the help commands, configuration files etc. on paper, so that they are easier to read. One of these programs is `mpager` (within the package with the same name), or another would be `enscrip` (also packaged with the same name; this one is more powerful than the first one). A few lines are given below as an example of how we would use it:

```
man man | mpage -2 -o |lpr
```

Using this line, we redirect the output of the man to mpage, which formats it so that there are two columns per sheet, without margins, and we redirect its output to the printer client `lpr`:

```
endscript /etc/fstab -B -fTimes-Roman7 -r
```

With this line, we will print out the contents of the file `/etc/fstab` without a header, using the Times-Roman font, in size 7 and a wide format.

9.7.6. Configuring soundcards

Due to the large number of soundcards that there are on the market, it is almost impossible to provide a description of how all of them should be configured. We recommend reading http://www.tldp.org/HOWTO/Sound-HOWTO/ and visiting the websites of the most important sound projects under GNU/Linux: http://www.opensound.com/ and http://www.alsa-project.org/.

Advanced Linux Sound Architecture (ALSA) is a project that works under a GNU license, to provide Linux with audio and MIDI devices.

We will now discuss how we should proceed if we need to configure a fairly common soundcard: SoundBlasterPCI (chipset ES1371). For these types of card, the `lspci` command will show us a line like the following:

```
00:0d.0 Multimedia audio controller: Ensoniq 5880 AudioPCI (rev 02)
```

Firstly, we will load the module corresponding to this soundcard, using the `modconf, kernel/drivers/ sound, es1371` command.

Next, we will create the `audio` group in `/etc/group` and we will include all the users that will require access to the sound device in the group (if we want all the users to have access to it, we can omit this step and provide all the permissions to the files `/dev/dsp` and `/dev/mixer`); and then associate the files `/dev/dsp` and `/dev/mixer` to the new group we have created.

With this, we will have configured the soundcard. We can now check that this is the case by directing an audio file directly to `/dev/dsp`, as suggested in http://www.tldp.org/HOWTO/Sound-HOWTO/ or using the audio applications that run on this. We will have to wait until we have the graphical environment configured, before we can install it.

9.8. Conclusion

In this workshop, we have learnt how to work with Debian's package system, which is essential, as it allows us to learn how to install, uninstall and generally manage the applications. We have also learnt how to configure different hardware devices and, with this, a basic point: that GNU/Linux is not as simple as other operating systems, as it requires, generally, more in-depth knowledge of the actual device as well as the system itself. But, to compensate this aspect, we can guarantee that once the device has been configured, it will work properly and we will not have to worry about it again. And we should not forget that we have not even configured the graphical environment yet. Nevertheless, to encourage those of you who may be beginning to think that stepping into the world of GNU/Linux may not have been such a good idea, a failed attempt and a waste of time, we would like to quote one paragraph from the book Debian GNU/Linux 2.1 by Mario Camou, John Goerzen and Aaron Van CouWenberghe, part I, chapter 1, Why Linux Is Better:

> "Windows NT, however, learned to speak the language of the Internet just a few years ago. It is not refined by any stretch imagination, but plagued with bugs and inconsistences. Bussinesses need a solution that they can put online and expect to remain available 100% of the time, but Windows NT cannot meet this need. On the contrary, a Windows NT administrator's most common job is crash recovery; he wastes too much time doing busywork, such as booting servers that have gone down[...]".
>
> Aaron Van CouWenberghe

10. X-Window architecture

10.1. What is X-Window?

X-Window is an architecture of windows designed in the mid-1980s to provide a graphical environment in workstations. Unlike other windows environments, the X-Window architecture was designed to be independent of the platform, so that it could be installed in any computer that worked on a UNIX-type system. Although the X-Window windows architecture has an extensive history in which different types of licenses, various implementations and many different teams of developers have been involved, it currently uses, mainly, the implementation developed by the X.Org. project. X.Org is an implementation of open code of the X Window System that arose as a fork of the XFree86 project.

X-Window is designed with a client/server architecture. This type of architecture means that the software is structured into two parts that are totally independent (client and server), which communicate using a communication link. Although this means that the design and coding is slightly more complex, the architecture provides complete flexibility in the sense that the client and the server can be located in different places and use different platforms and/or operating systems. In addition, we can take advantage of the same client to a much greater extent, as the client will be able to provide the service to more than one server at a time. In this way, the server computers can work with one graphical environment and the client's resources. Naturally, this architecture also allows us to work with XWindow locally on the machine where the client is located, although this is not essential.

Windows architecture.

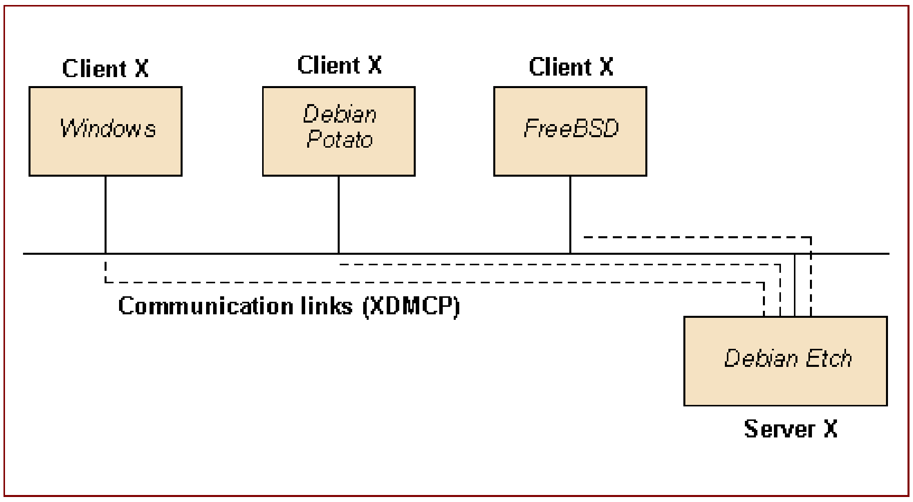
A windows architecture (or windows system) is a graphical environment that allows us to have different applications located in different regions of the screen, generally limited by some form of window. These environments usually provide mechanisms for moving and handling these windows so that the work is more interactive and easier to perform.

The components that comprise X-Window are: client, server and communication link. Client and server are designed to be independent of the platform and the communication link is designed to be independent of the network protocol.

Consequently, we can use X-Window in any type of scenario: for example, we could have the server installed in a computer with Windows^{TM}, connecting to a client with GNU/Linux and use Internet as the communication channel (protocol IPv4). Although configuring each one of these components (especially the client) does depend, in a certain way, on the platform in which it is installed, the communication link makes it possible to isolate the components and give them their own language so that they understand each other.

This link uses its own protocol called XDMCP (X Display Manager Control Protocol), which is on a higher level than that of the communication network that is used (which is why it is independent of the network).

Figure 10.1



In this architecture, the server is programmed to register the events that occur as a result of the input devices, such as the keyboard, mouse etc. and report them to the client. The client processes these events and replies to the client, which shows the results in the output devices (usually a monitor). Although the first impression that this design might give is that the response time must be very slow, the XDMCP protocol is especially designed to provide a fast link between the client and the server, so that we can really work interactively. The only scenarios in which we might notice this inconvenience is in remote connections using slow communication networks.

To summarise, then, the main characteristics and functions of each one of the components of X-Window are as follows:

Table 10.1

Client	Manages different servers at the same time.
	Platform-dependant.
	Processing applications
Server	Control of user display
	Independent of platform
	Processing input devices
Link	Designed to work interactively.

	Designed to minimise traffic in the network
	Transparent (independent of the network)

As video cards have developed, more and more applications and games have started needing faster 2D or 3D processing. Although the X-Window windows architecture provides many advantages, when we need to use these types of application, the client/server design is not the most appropriate, as we are not taking advantage of all the extremely fast 2D and 3D processing operations provided by the video card installed in the server. In order to solve this problem, a new technology called DRI (Direct Rendering Infrastructure) appeared in 1998; this makes it possible to take advantage of the cards' processing chips to save the X-Window client work. In this way, we continue having all the advantages of X-Window and take advantage of the specific elements of the video cards.

Unlike other operating systems where the graphical environment is intimately integrated with the rest of the functions, the X-Window architecture is completely independent of the operating system and does not limit us to any one GUI (Graphic User Interface). In fact, the architecture only provides us with low-level graphical tools to manipulate the monitor's output. These tools are included in the **Xlib** library and mainly consist of functions for creating and manipulating windows, operations with character fonts, detection of user events and graphical operations. With these functions, we can provide our applications with the look and feel that we want, create new GUIs etc. In fact, this created additional work for the first developers of applications in XWindow, as they now, apart from programming the application, had to develop their own libraries for the creation of menus, icons etc. As X-Window continued to grow, what we call **toolkits**, which are libraries that are generally implemented with Xlib and that provide a particular GUI, began to appear.

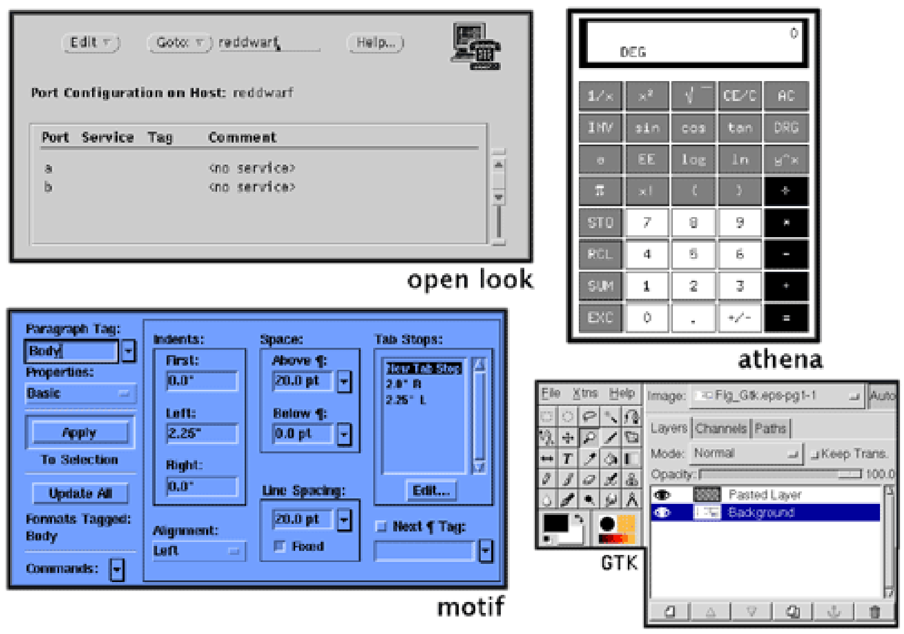
Look and feel

The look and feel is the design used for the buttons, the scroll bars, the menus etc., in a graphical interface or application.

In this way, when we design an application, we can use some of these toolkits, which already provide standard tools for creating menus, buttons, managing the cut and paste operations etc. and we can focus on programming the application itself. Not defining a particular look and feel has been another of the keys to the success of the X-Window architecture, as each software manufacturer or software developer has been able to design their own, which sets it apart from all the others.

Although there are many different toolkits, the following figure shows some of the most popular that have been used throughout the history of X-Window:

Figure 10.2



The `window manager` is a special X-Window server that is in charge of managing all the windows, the desktops, the virtual screens etc. Naturally, all the applications can work with any window manager, as all it does is manage the window where the program is located. Although programming a window manager is very different to programming an application, particular toolkits that provide a specific look and feel are often used. Currently, there are dozens of different window managers (wmaker, sawmill, olvwm etc.), and the users themselves can choose which one they like the most.

Another type of software that is strongly linked to X-Window is the one in charge of providing an environment comprising the applications, the desktop, the system administration tools etc. The most popular that currently exist are KDE (the K Desktop Environment) and GNOME (GNU Object Model Environment). Both provide a particular toolkit, a desktop environment with very many different functionalities and configurations and a list of integrated applications that is constantly growing. Most GNU/Linux and UNIX distributions provide one of these two desktop environments as they are very user-friendly and provide their own tools and software, which are of a high quality and which help the user to configure their own system and desktop to a large extent. The two may work with any window manager that has a series of basic features. The following figures show the appearance of the two:

Figure 10.3

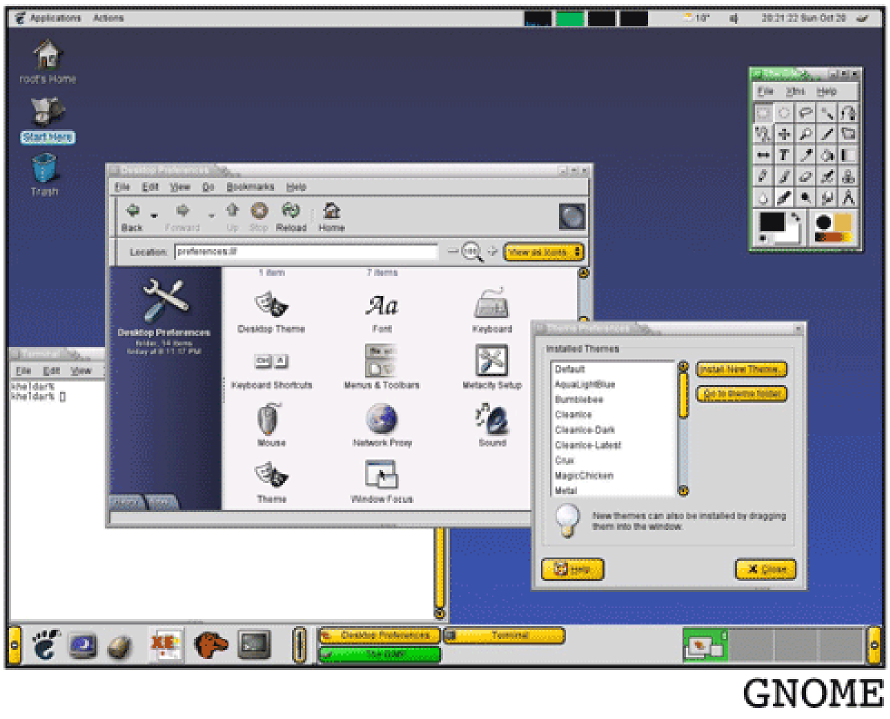
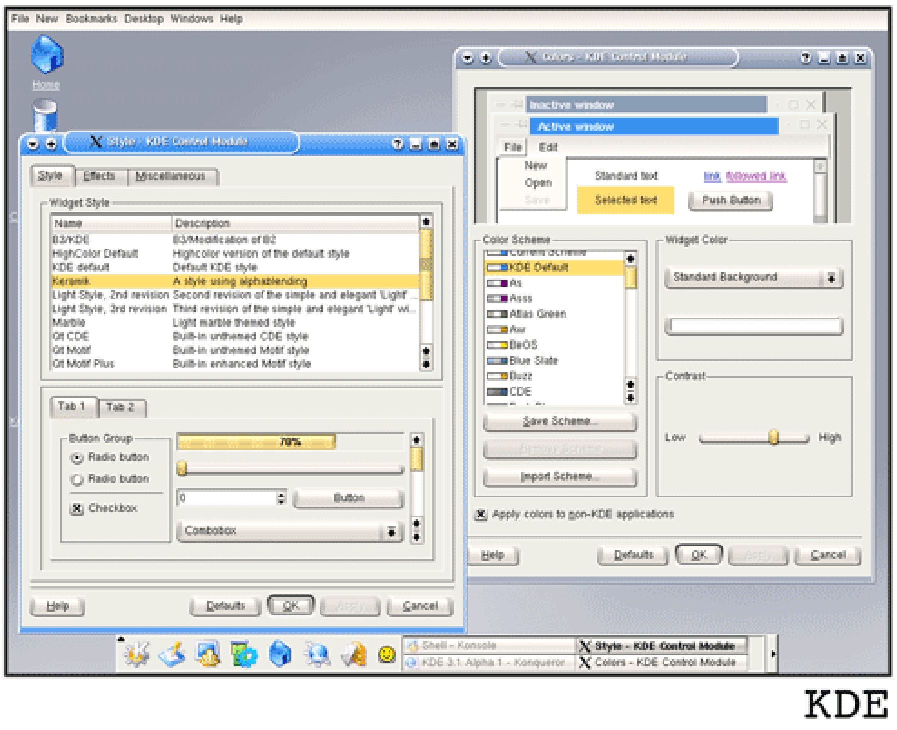


Figure 10.4



Finally, another application that is used in X-Window is the session manager, which consists of a series of programs that make it possible to save the configuration of a determined user session so that the applications that have been configured will load whenever X-Window is started up again. Generally, these tools are automatically provided in integrated environments; if this is not the case, we can resort to the one provided by X-Window itself: `xsm`.

Activity

10.1 Read the history of X-Window in the article: http://www.linux-mag.com/2001-12/xfree86 01.html.

Activity

10.2 See some examples of the window manager and desktop environments that exist at: http://www.xwinman.org.

10.2. Configuration

Most video cards on the market are currently supported, but sometimes, when a new card appears on the market, it takes a few weeks or months before it is supported in XWindow. In any case, the number of manufacturers that support GNU/Linux is gradually increasing, and in some cases, they are already providing their own drivers for this operating system. Nevertheless, before buying a video card, it is important to check that it has some kind of driver that is adequate to the distribution that we use. In order to install XOrg on our computer, we must first download the packages that contain the basic tools and the software for the client and the server. Generally, these packages are called `xorg`, `xserver-xorg` etc. and they come with various source dependencies and some basic utilities for working on X-Window. Once we have installed these packages, we must configure the devices that we have properly, so that we can boot up correctly with the X-Window server and client. Depending on the distribution we are using, we can use one program or another or, in certain cases, the installation of the packages itself launches a small configuring application. However, this configuration must always follow certain steps, which we will describe below, classified according to the type of device that has to be configured:

- Video card
  - Driver: the different families of video cards come with specific microprocessors and use determined functions in order to carry out their operations. This is why we must find out the appropriate driver for our card. If we do not know which it is, we could install some form of application that automatically detects the hardware; if we were to use, for example, discover, we would be able to tell which driver our card would need with the `discover --xdriver video` command.

  - *Identifier*: the card's identifier could be any name that we wish to use to refer to our card. This identifier is used internally to correctly reference the cards that we have installed in the system.

  - *Amountofmemory*: depending on the card's amount of memory, we will be able to start up the graphics with a higher or lower resolution and with higher or lower levels of colour depth. Although it is not essential to enter this amount (the system will detect it automatically), it is advisable to specify it in the configuration.

  – *Use of the kernel's framebuffer*: The kernel's framebuffer is a special Linux driver that makes it possible to perform certain operations on X-Window. Although it is not obligatory to use it, it is generally used so that the X-Window server can communicate directly with the system's kernel. In any case, if it creates any problems, we can always disable it.

- Keyboard
  – *Rule XKB*: for the X-Window server to handle the keyboard correctly, it needs to know what rules apply to it. For most standard PC keyboards, the rule "xfree86" is used and the "sun" rule is usually used for Sun workstations.

  – *Keyboard model*: the keyboard's model is usually identified by the number of keys it comes with. Standard PC keyboards that have menu and logo keys usually have 104 keys (we identify them with the name "pc104"). Keyboards that do not have these keys are identified as keyboards with 101 keys ("pc101").

  – *Keyboard layout*: in this section, we must identify the country of origin of the keyboard using the ISO 3166 standard. The code for a Spanish keyboard is "es", "fr" is for a French keyboard etc.

  – *Keyboard options*: option for personalising some of the keys on the keyboard.

- Mouse
  – *Port*: the mouse's port is the connection that it uses to communicate with the computer. When we buy a mouse, it will always indicate whether it is PS/2-type, serial-type, USB etc. If the mouse is a PS/2-type mouse, the port will be `/dev/psaux`; for a serial mouse the port will be `/dev/ttyS0` (COM1), `/dev/ttyS1` (COM2) and so forth, consecutively.

  – *Type*: A list is usually provided, so that we can specify the type of mouse, from which we must choose the one with the model and manufacturer most similar to our mouse. Generally, once we know the model of the mouse, we can choose the right option.

  – *Emulation of 3 buttons*: if our mouse were only to have 2 buttons, we would be provided with the possibility of emulating the third (the one in the middle) by pressing the two buttons at the same time. If our mouse does not have a central button, we should enable this option, because some X-Window programs need a mouse with 3 buttons.

- Monitor
  - *Identifier*: as occurs with the video card, the monitor identifier is used so that the system can reference it internally. We can assign any name we want to it.
  - *LCD-type monitors*: in most configuring processes, we will be asked whether or not our monitor is an LCD-type monitor (TFT screen). It is important to answer this question correctly because how one or another monitor is used varies considerably.
  - *Features*: when configuring the features, we will be asked about the maximum resolution of our monitor, the refresh rate etc. Although more or less questions of this nature will arise depending on the program used to configure X-Window, it is important to have the monitor information available and to answer the questions properly to take advantage of all the features.
  - *Available resolutions*: at this point, we must enter the resolution at which we want our monitor to operate when we start up X-Window. It is also normal for the system to ask us to define the colour depth that we wish to use by default; the most advisable thing is to use a high one (16 or 24 bits) so that we can see all the colours sharply. Once we have answered these questions, of which there may be more or less depending on the program that we use, the whole configuration is saved in the file `/etc/X11/XF86Config-4`.

All the different sections that we have looked at are organised in this file and if we look at the manual, we will see that we have very many more possibilities that provide us with complete flexibility for configuring our X-Window how we want it to be. To check whether they really work, we can run `X`, at which point a screen with very small black and white boxes should appear and the mouse should appear as an "X" (to come out of this screen, press Ctrl+Alt+Backspace).

xsession

When we use `startx` or `xinit`, the instructions of file `/etc/X11/Xsession will run` . If the home directory of the user that starts up X-Window contains an `.xsession` file, it will be the commands in this file that will run, instead of those in the other file.

If we have installed some form of window manager, what we will usually do when starting up X-Window is to use one of the shell scripts `xinit` or `startx`. These will launch the configured window manager and will carry out some other actions required to start X-Window up properly. Once the screen is in graphic mode, we can change the resolution using the Ctrl+Alt++ and Ctrl+Alt+- keys, or return to the text consoles with Ctrl+Alt+F1, Ctrl+Alt+F2 etc. (with Ctrl+Alt+F7, we would return to the graphical environment).

Another important feature in the configuration of X-Window is the configuration of the **Xwrappers**. The Xwrappers allow us to control which users can start their sessions with X-Window. The file `/etc/X11/Xwrapper.config` contains the `allowed users` command, with which we will specify who is allowed to start up X-Window with various values:

- `console`: any user that is in the local console can start up X-Window.
- `rootonly`: only the root can start up X-Window.
- `anybody`: any user of the system can start up X-Window (even if they are not connected locally).

This is very useful, especially for administrating a server in which users are not generally allowed to work with the graphical environment due to the amount of resources that this consumes.

10.3. X display manager

In the previous subsection, we saw how to configure X-Window locally. As we have mentioned throughout the section, the architecture of the X-Window windowing system allows the client and server to be installed in different computers. In order to configure our computer so that it carries out the functions of an X-Window client, we must install some kind of X display manager. These programs open a communications port with which the clients can communicate with the client and work with X-Window remotely. Although there are many programs of this type, one of the first to appear and on which many of the others are based, is `xdm`.

The X display managers can work both locally and remotely. Among other functions, these show the screen (in a graphical environment) so that the users can identify themselves with their login and password. They work like any other daemon of the system, so that we can configure them to start and stop as we choose (using the runlevels that the system provides). We must remember that if we configure it to work locally, when we boot up the system we will find ourselves on the graphical login screen and not with the consoles that we are used to (although they will still be available). With this configuration, we will no longer be able to use `startx` or `xinit` to start up X-Window, as it will be running by default.

When we install `xdm`, all its configuration files will be placed in directory `/etc/X11/xdm`. We will now go over what each of these files contains:

Table 10.2

`xdm-config`	Location of the configuration files of `xdm`.

`xdm.options`	Global configuration options.
`Xaccess`	Definition of the remote equipment that we no longer access.
`Xservers`	Local servers of `xdm`.
`Xresources`	Configuration of the login screen: colours, sources, sizes etc.
`Xsetup`	Script that will run when we start up `xdm`.
`Xstartup`	Script that will run when a user enters XWindow. There will usually be actions related to the `xdm`.
`Xsession`	Script that will run when we enter a user session. There are usually special actions for the users, although the file's execution is usually called up. `/etc/X11/Xsession`.
`Xreset`	Script that will run when a user's sessions ends.

We will find the configuration of the local servers in the file `Xservers`. If we wanted to disable the local server, we could comment all the lines in this file. In this way, even if we had an X-Window client installed, by default, it would not start up in the local machine. If, on the other hand, we wished to install more than one, we could edit the file and add directives such as the following:

```
:0 local /usr/X11R6/bin/X :0 vt7 :1 local /usr/X11R6/bin/X :1 vt8
```

These two directives indicate that we want to have two instances of X-Window, one in console 7 (`vt7`) and the other in console 8 (`vt8`), accessible with Ctrl+Alt+F7 and Ctrl+Alt+F8 respectively. Let us examine how each directive includes a `:0` or `:1`, depending on the instance of X-Window referred to. By default, we will always use 0, but we must reference it in this way if we want more than one local server. At the end of each one of these lines, we could add special parameters for each server of X-Window (we will find all the possible ones in `man X`), such as the colour depth that we want for each one, the screen resolution etc. In this way, we could work with different open sessions of X-Window, as we did with the consoles.

Security of X-Window

If the file `Xaccess` contains a line with the character "*", this indicates that we will allow any server to connect to the server's X-Window. Using X-Window remotely without any type of encryption could create a hole in the security, which is why it is very advisable that users inform themselves adequately on this point, before using it.

Generally, the default setting of `xdm` does not allow remote connections for security reasons. If we wished to enable these connections, we could edit the file `Xaccess` and, using the syntax that is indicated, add the servers to which we wish to provide this service. We should also comment on the line `DisplayManager. requestPort: 0` of file `xdm-config`, which, by default, disables all the connections that are received. Having performed these changes, if we reboot the daemon of `xdm`, the client would already be prepared to serve X-Window to any server that requested it.

For machines in which we only wish to install the XWindow server, we must install X-Window just as we saw in the preceding section and use the `X -query IP` command, where the IP should be that of the client. In the same way as when we had more than one X-Window server on a local machine, if we already had another instance of X-Window running on the machine, we would have to use `X -query IP :1` for the second instance, `:2`, for the third and so on, consecutively.

Xnest

Xnest is an X-Window server that allows us to open another instance of X-Window in a window.

11. X-Window workshop

11.1. Introduction

In the second workshop, we discussed the basic aspects for manipulating and managing packages properly and we learnt how to configure some devices. Nevertheless, due to its complexity, we did not cover the configuration of the video card, or, rather, the installation of the X graphical environment. Due to the complexity of its structure and configuration, we have decided to dedicate one whole workshop to the X Window System. In this workshop, we will learn how to install, configure and personalise this system. But we will not try to provide an exhaustive examination of the whole system, as doing so would probably be impossible for many different reasons. What we will do is provide some basic knowledge so that each person should be able to configure their own system, depending on the type of video card that they have and their personal tastes and preferences. By the end of the workshop, we should be able to install an X environment and know how to take advantage of its incredible power.

Activity

11.1 Due to the complexity of the X system, it is helpful to read the following documents to get an idea of the basic concepts, before working on the system. In addition, these documents will provide additional knowledge, which we will be able to put into practice during the workshop.
a) XWindow-Overview-HOWTO. This is a simple document that can be read to assimilate the basic concepts related to the system.
b) XWindow-User-Howto. This is a document with more advanced contents than the previous one, but it is also helpful.

11.2. Installation and configuration of X server.

As we have mentioned, the X Window System is a very complex system that integrates very many libraries and applications, some of which are essential for it to work, although most only have to be installed if we really need them, due to our particular necessities. This is one of the reasons why, in Debian, the system is distributed in many different packages, of which we will only install the ones we need.

11.2.1. Different strategies for installing the packages

Due to the fact that there are strong interdependencies between the different packages, we can take advantage of this so that the actual package management system installs all those that it believes to be necessary for a high-level application to work, among which it will find, obviously, all of the system's basic packages. Consequently, we could use `dselect` or `apt-get` to install one of these applications. But this is not a good strategy, because it involves the loss of control of the packages that we are installing and it can also mean that some essential packages are omitted, because no reference has been made to them when the dependencies were calculated, for any reason. Therefore, we recommend building the system step-by-step, so that we know which packages are being installed during each stage and understand why.

11.2.2. Installation of basic packages

Chipset and device

The first step is to determine the chipset of the video card and the device that supports it. Run the `lspci` command in a terminal and search for the words `VGA compatible controller:`. The information that follows normally identifies at least the brand of the video card and possibly the device that you will need. For example,

```
$lspci 01:03.0 VGA compatible controller: ATI Technologies Inc ES1000 (rev 02)
```

The first column shows the number of the PCI bus to which the card is connected in the format `<bus>:<slot>:<func>`.

By running

```
$lspci -n -s bb:ss.f
```

we identify the PCI bus numbers to the `vendor` and `device ID` of the card. For example,

```
$lspci -n -s 01:03.0 01:03.0 0300: 1002:515e (rev 02)
```

The `vendor` and `device ID` have the format `vvvv:dddd` in hexadecimal numbers. In this case, the `vendor` and `device ID` of the card is `1002:515e`. With this information, we search the video cards (http://www.calel.org/pci-devices/xorg-device-list.html) for the device that we need to use. In this case, the device is `radeon`.

We also need to know the monitor's horizontal and vertical scan rates. The scans are usually found in the specifications section of the monitor's manual. These data are very important and incorrect values can cause the monitor to stop working properly and can even cause damage.

Installation of X.org

From the commands interpreter, from the root account, we key in:

```
#apt-get install xorg
```

This is a metapackage that, when using the dependencies system, installs a set of X11 packages, such as X.Org server, a set of basic tools and fonts.

We will now go over a step-by-step configuration of the `xserver-xorg` package, carried out with `debconf`. If we do something wrong, we can undo it with "Crtl+C" and reconfigure it with,

```
#dpkg-reconfigure xserver-xorg
```

X Device

We will choose the device that most suits the video card from the list, in accordance with how we identified the chipset at the beginning. For example, .

We will then give it a name, such as the default one, for example `Generic Video Card`.

We then identify the card in the PCI bus, for example, `PCI:1:0:0`.

It almost always automatically detects it. We should leave the amount of memory blank so that the server automatically detects it. We should choose not to use the kernel's “framebuffer” device.

Keyboard

We should set it so that the keyboard is not automatically detected. We select the keyboard's default language (`us` by default, `es` for Spanish). We select the keyboard rules `xorg` (by default). We select the type of keyboard (`pc104` by default). Other options are `pc101`, `pc102`, `pc105`. The last two are for European keyboards. We leave the variant and options of the keyboard blank.

Mouse

We select the mouse device, for example, `/dev/input/mice`.

If the mouse is connected to a serial communication port, `/dev/ttyS0` corresponds to port COM1, `/dev/ttyS1` to port COM2 etc. The unit `/dev/psaux` is for PS/2-type mouse and `/dev/input/mice` for USB-type mice.

We select the mouse's protocol, for example, `ImPS/2`.

We select the option for emulating a three-button mouse

Modules

We select the default modules

File path

We select the default configuration of the section `"Files"`.

Monitor

We choose to automatically detect the monitor. We will then give it a name, such as the default one, for example `Generic Monitor`.

Using the spacebar, we choose the resolutions that the video card and monitor support, typically 1280 x 1024 or 1024 x 768. We select the advanced mode for configuring the monitor. We enter the horizontal scan rate and then the vertical scan rate. These values or ranges are usually found in the technical specifications section of the monitor's manual. If we do not have this information, it is better to enter a higher range, such as 28-100 for the horizontal scan rate and 40-100 for the vertical. We select the colour depth, typically 24 bits.

Running scripts

After unpacking the packages, the configuration scripts of several of them will run automatically. It is always possible to interrupt the execution of these scripts by pressing Ctrl+C, and then restart the process, by running the previous command.

11.3. Configuring X

The file `/etc/X11/xorg.conf` contains the configuration of `X.Org` and is divided into sections:

```
Files # Path of the Module files # Dynamic modules InputDevice # Description of the peripheral
devices Device # Description of the video devices Monitor # Description of the Monitor Screen
# Configuration of the ServerLayout screen # Global DRI scheme # Configuration specific to DRI
```

Each section begins with the `Section` command followed by the name of the section in inverted commas and ends with `EndSection`. There are commands that are specific to each section. We will proceed step-by-step, not necessarily in the order of the file, describing the sections and explaining the meaning of each command, so that we can then modify the configuration using a text editor (`nano` for instance) with the parameters that are appropriate for the hardware.

In order to make the changes to /etc/X11/xorg.conf effective, we must reboot the display manager using

```
#/etc/init.d/gdm restart
```

11.3.1. “Device” section

This section is the one that defines and configures the video card device.

The Identifier entry is simply a name to identify each section.

```
Section "Device" Identifier "Generic Video Card" Driver "sis" BusID "PCI:1:0:0" EndSection
```

The Driver entry specifies the device. In this example, the device sis supports many types of video cards with the SiS chipset. Other common devices are: cirrus, ati, r128 (ATI Rage 128), radeon (ATI Radeon), s3virge, savage, trident, tseng. We can use the method described at the beginning to determine what type of device we are using.

The BusID entry identifies the video card in the PCI bus, which we can determine with lspci.

11.3.2. "Monitor" section

This section defines the monitor.

```
Section "Monitor" Identifier "Generic Monitor" Option "DPMS" HorizSync 28-100 VertRefresh
40-100 EndSection
```

The Identifier entry is similar to that of the "Device" section. There may be various "Monitor" sections in the file, each with a different identification. This is convenient if, for example, we have various monitors, say one at home and another at work. In this way, we can easily define a configuration for when the computer is in the home and another for when it is at the workplace.

The HorizSync entry specifies the monitor's horizontal scan rate in kHz units. It may be a fixed rate [31.5], multiple fixed rates, [31.5,35.2], a range, [30-64], or ranges, [15-25,30-64]. The VertRefresh entry specifies the vertical refresh rate in Hz units and the values have the same format as HorizSync.

11.3.3. "**Screen section**"

This section defines the screen, combining the configuration of the device and the monitor.

```
Section "Screen" Identifier "Default Screan" Device "Generic Video Card" Monitor
 "Generic Monitor" DefaultDepth 24 Subsection "Display" Depth 1 Modes "1024x768"
 "800x600" "640x480" EndSubsection Subsection "Display" Depth 4 Modes "1024x768"
 "800x600" "640x480" EndSubsection Subsection "Display" Depth 8 Modes "1024x768"
 "800x600" "640x480" EndSubsection Subsection "Display" Depth 15 Modes "1024x768"
 "800x600" "640x480" EndSubsection Subsection "Display" Depth 16 Modes "1024x768"
 "800x600" EndSubsection Subsection "Display" Depth 24 Modes "1024x768" EndSubsection
 EndSection
```

We should note that the `Device` and `Monitor` entries have the same value as the `Identifier` entry in the `"Device"` and `"Monitor" sections`. This means that the configurations of the different sections are interlinked.

The `DefaultDepth` entry specifies the default colour depth, which in this example is 24 bits. There may be various `"Display" subsections`. The `Depth` entry specifies the colour depth of the subsection. The possible values are `8` (256 colours), `16` (64K colours) and `24` bits. This means that the `DefaultDepth` will specify the default `"Display"` subsection.

The `Modes` entry specifies the resolutions. A resolution of `["640x480"]` or a list of resolutions `["1600x1200" "1280x960" "1152x864" "1024x768" "800x600" "640x480"], may be specified.`]. Normally, the first in the list is used, if it is supported; if not, the next one, and so on.

Generally, the greater the colour depth, the less maximum resolution will be available. We can reduce the colour depth if the image on the screen is bad for a certain resolution, or we can maintain the depth by reducing the resolution.

11.3.4. "**InputDevice**" **section**

This is the section that configures the peripheral devices such as the keyboard, mouse, trackpad, touchscreen etc. The most common input devices are keyboards and mice; obviously, each one is separate.

Configuring the keyboard:

```
Section "InputDevice" Identifier "Generic Keyboard" Driver "kbd" Option "CoreKeyboard"
Option "XkbRules" "xorg" Option "XkbModel" "pc105" Option "XkbLayout" "es" EndSection
```

All keyboards have a common device called kbd. The Option "CoreKeyboard" entry indicates that the keyboard defined by the section is the main keyboard. The Option "XkbModel" entry specifies the type of keyboard. The most common values are "pc101", "pc102", "pc104", "pc105" or "microsoft" for the Microsoft Natural keyboard. The Option "XkbLayout" entry defines the keyboard's language: "es"for a Spanish keyboard "us"for an English keyboard

Configuring the mouse:

```
Section "InputDevice" Identifier "Configured Mouse" Driver "mouse" Option "CorePointer"
Option "Device" "/dev/input/mice" Option "Protocol" "ImPS/2" Option "Emulate3Buttons"
"true" Option "Buttons" "5" Option "ZAxisMapping" "4 5" EndSection
```

The common mouse device will support four types of mouse: serial, Bus, PS/2 and USB. In the first example, we configure a USB IntelliMouse (with wheel).

The Option "CorePointer" entry indicates that the mouse defined by the section is the main mouse.

The Option "Device" entry specifies the mouse's device unit. Some examples are /dev/ttyS0 (serial), /dev/psaux (PS/2) and /dev/input/mice (USB). Generally, there is a symbolic link /dev/mouse that points to this device.

The Option "Protocol" entry defines the type of mouse. Other protocols are:

```
"MouseMan" "Mousesystems" "IntelliMouse" "ExplorerPS/2" "ThinkingMouse" "ThinkingMousePS/2"
"NetScrollPS/2" "NetMousePS/2" "GlidePoint" "GlidePointPS/2" "MouseManPlusPS/2"
```

Old serial mice with two or three buttons are normally supported by the protocol "Microsoft" or "MouseMan". Serial mice with a wheel are supported by protocol "IntelliMouse" and PS/2 ones by "ImPS/2". The "auto" protocol sometimes helps if the hardware is able to automatically detect the mouse.

The Option "Emulate3Buttons" entry emulates the middle button if we press both the left and right buttons at the same time. We can also use this system in mice with three buttons.

11.3.5. "ServerLayout" section

This section is the one that relates the screen with the peripheral devices.

```
Section "ServerLayout" Identifier "Default Layout" Screen "Default Screen" InputDevice
"Generic Keyboard" InputDevice "Configured Mouse" EndSection
```

We should particularly note how this section combines everything using the identifiers of each section defined with `Identifier`.

11.3.6. "DRI" Section

Some modern cards have what is called a direct rendering infrastructure (DRI). To use this, you must load modules `"glx"` and `"dri"` in the `"Modules" section`.

The DRI section, by default, is

```
Section "DRI" Mode 0666 EndSection
```

Although cards that do not support DRI may make X hang.

11.3.7. "Files" section

This section defines the file path necessary for X. In particular, it defines the path of the fonts in the `"FontPath" entries`.

11.3.8. Starting up the server

The time has come to check whether the server's configuration file is correct and, consequently, whether the server starts up as it should. In order to do this, all we have to do is run the `startx` command.

If everything works properly, after a few moments, a grey mesh with a cross in the middle will appear on our screen. This represents a large step forward, as configuring the server for startup is the most difficult thing about the X environment. Now it is just a question of the time required to finish giving the environment the appearance that we want it to have. If we move the mouse, the cross will move and we can explore some more possibilities in this rather rudimentary graphical environment by pressing the middle and left buttons. To exit this screen and continue configuring the system, we have to press the left button of the mouse, highlight the `Exit` option `Yes, reallyquit`, or simply press the Ctrl+Alt+Backspace keys together.

If, on the other hand, we return to the alphanumeric console after a few moments, this means that the server was not able to startup properly. We should now study the log file (`/var/log/Xorg.0.log`) in detail and try to detect possible sources of the errors. The most common are usually: bad choice of driver that has to be loaded (if we have left the selection process to the script, we must check the page mentioned above to ensure that the driver that the script has chosen is the right one), in the directive `UseFBDev` the `true` option has been selected, whereas it should be `false`, resolutions or scan rates that are higher than those supported by the cards have been used etc.

Having reached this point, we can use the following command to avoid entering the graphical mode every time we boot up the computer:

```
brau:~# rm /etc/rc2.d/S99xdm
```

This will be useful, whilst we have not yet configured this environment completely. Once we have finished, it will be down to each individual user whether or not they start up the computer in the graphic mode. To do this, all we have to do is create the symbolic link that we deleted again, using the preceding command:

```
brau:~# cd /etc/rc2.d brau:/etc/rc2.d# ln -s ../init.d/xdm S99xdm
```

11.3.9. The log file

Although the process for starting up the server was successful, that should not stop us from analysing the contents of the main log file of the X system, `/var/log/Xorg.0.log`, as this will help us to solve any bugs and minor errors that have not interrupted the server's booting process, but that might reduce its performance. Some typical examples of these are: delete the line of the configuration file that refers to the Cyrillic fonts, if we are never going to use them, as we had not even installed them. delete everything that refers to the Generic Mouse from the same file, as it makes it incompatible with the one we have configured etc.

11.4. Window managers

The window managers are client programs (in reality, they are called meta-clients), in charge of managing the different windows that run on the graphical environment and their presentation, as well as launching other clients (applications). By this stage, we already have a window manager installed, `twm`, as a complete installation of an X system requires at least one window manager, even if it is not part of the server, for it to run on. As we have seen, `twm` is very rudimentary, which means it might be in our interests to install one of the more complex ones, such as WindowMaker, BlackBox, qvwm etc. We will install some of these and try them out. The final objective, in accordance with the GNU philosophy, is for every user to end up using the software that they want. Effectively, this means that, once students have learnt about certain window managers, they will be able to choose which one they want to use themselves. Obviously, it is possible to have more than one window manager installed, although only one can run during each session. (However, we can, as an example of the flexibility of the X system, have two window managers running on the same computer, each in a different terminal).

Firstly, we should install, as an example, `qvwm`. It is, as we shall see when we launch it, a window manager that simulates an environment we are probably familiar with. To do this, all we have to do is run the command:

```
brau:~# apt-get install qvwm
```

By now, we will have two window managers installed: `twm` and `qvwm`. To run one or another, all we have to do is indicate the complete path of the location of the window manager that we wish to use, after the `startx` command (we should remember that the `whereis` command can be very useful if we need to locate a file). Therefore, to run the window manager that we wish to install, all we have to run is:

```
brau:~# startx /usr/bin/X11/qvwm
```

To use the twm, we must have specified its complete path, as follows:

```
brau:~# startx /usr/bin/X11/twm
```

Having to specify which window manager we wish to use, once we have decided on one particular one, can be rather tiring. To specify which window manager should be used if after the `startx` command, no particular one has been specified, we will create the file `.xsession` in the root user directory with the following contents, in case we need the default window manager to be `twm`, for example:

```
#˜/.xsession exec twm
```

If we wanted `qvwm` to be the default window manager, we would simply change `twm` for `qvwm`. Running the different processes involved in the booting of the graphical environment and the configuration files that will be read during the process are strongly determined. Effectively, if we create the preceding file, what we have done is to edit one of the last files (those that are in the user root directory) that are read before entering the graphical environment. This file, therefore, allows us to modify some of the aspects that are determined by default within the system and that are defined in the files in `/etc/X11` and their subdirectories. To end this subsection, we will install a window manager that is very widely used in the GNU/Linux world and that is famous for its versatility and the fact that it consumes very few resources: the WindowMaker:

```
brau:~# apt-get install wmaker
```

We have already installed three window managers, and we will surely install more. Apart form the method described above for pre-establishing which one we want to run by default, we can also use the menu of the `update-alternatives` command to establish it:

```
brau:~# update-alternatives x-window-manager
```

Additional reading

We would urge students to become a little familiar with this window manager and to visit its website to obtain more information: http://www.windowmaker.org.

11.5. X Session manager

The session managers are programs that can run on a graphic session and that allow us to establish and modify its parameters. `xms` is the session manager that comes by default with the installation that we have made of the graphic server. We may launch it from an X terminal (to launch an xterm, we should press the mouse's middle button and select Programs/Xshells/ Xterm), using the `xsm` command. Having launched `xsm` using `Checkpoint`, we can save the configuration of the current session (essentially referring to the applications that we have running), manage the processes that are running using Client List, check the log of the session or close the session whilst saving the current configuration. Apart from `xsm`, there are other *session managers*. These tend to be another part of the desktop managers, and the degree of integration is so high that recognising their actions can be quite difficult. A typical example of this is the question that we are asked with regard to whether we want to save the session when we close KDE.

11.6. X Display manager

When we finished installing Xserver, we suggested deleting the symbolic link `/etc/rc2.d/S99xdm` to avoid, when rebooting the system to enter *runlevel* 2 having to have the `xdm`, which stands for *X display manager*, run again. This is the default display manager that the X-Window-System installs. The *display managers* are the programs in charge of managing which users can enter the graphical environment and from where and how. To launch it, we will proceed as we would with any other daemon:

```
brau:?# /etc/init.d/xdm start
```

To stop it, we would also use the same procedure as we would follow with any other daemon, with the exception that we would have to press the keys Ctrl+Alt+F1, to come out of the graphical environment and use tty1, for example, instead of using the combination that is used to change ttys in alphanumeric environments.

```
brau:?# /etc/init.d/xdm stop
```

As we have seen, the display manager asks us for a login and a *password*, which are the same as the ones we use in order to access the system with the ttys, unless we have imposed some form of restriction. After we have authenticated our account, we enter the graphic mode in the same way as we did using the `startx` command. The difference is that, when we end the graphic session, the server will not stop, but continue running `xdm`.

One of the inconveniences of `xdm` is that it does not allow us to select which window manager we want to work with. But there are other *display managers*, such as `wdm` (by WindowMaker), `gmd` (by the GNOME project) or `kdm` (by the KDE project), which do allow this.

We can install `wdm`, to see its appearance and to get to know a different display manager:

```
brau:?# apt-get install wdm
```

When the post-installation script runs, we will be asked which display manager we wish to use, `xdm`, which we had already installed, or `wdm`. We will select the latter in order to create the link required to launch `wdm` com display manager during the system boot up process (if the file `/etc/rc2.d/S99xdm` exists, it is better to delete it to avoid *warning* messages when the system boots up). If we do not want any display manager to launch automatically when we boot up the system, all we have to do is delete the required links, in other words, the file `/etc/rc2.d/wdm`.

Once an X session has started from the display manager, in other words, once we have launched the corresponding window maker, it can be interesting to run the `pstree` command to see the dependencies and relationships between the different processes that are running at that moment, along with the information that we obtain from the line `ps aux`.

11.7. Desktop managers

The appearance of the different toolkits, along with the development of various projects that developed or used libraries for the graphic interface, resulted in the creation of projects that had the aim of unifying all of these operations. At this point, a new concept appeared in the X environment: the desktop manager. The desktop managers are projects that are designed to lay the foundations for the unification and standardisation of the presentation and the policies on programming and developing applications. One of the first to appear was the CDE (Common Desktop Manager), although the two most important projects currently existing in this area are: GNOME and KDE, which we will cover individually in separate subsections, due to their high degree of implementation and development. Before that, however, we can name some other desktop managers, such as: GNUStep, ROX, GTK+Xfce or UDE.

11.7.1. GNOME

GNOME is a project that is part of GNU, which is characterised by the fact that it does not strictly need a specific window manager, although it is advisable to use one to ensure that it works properly (a GNOME-compliant window manager) such as: IceWM or Sawfish. Nevertheless, in order to guarantee the preferences and freedoms of the user, GNOME's control panel always has a window manager selector that allows us to choose which window manager we want to use. GNOME is based on Gtk toolkit, the actual libraries developed within the project, known as specific gnome-libs.

Like all desktop managers, GNOME has its own panel, for managing Nautilus files and its control panel. GNOME Control Panel. In order to carry out a basic installation of GNOME, we will install the following package along with its dependencies:

```
brau:~# apt-get install gnome-session
```

As we have mentioned, although GNOME does not require users to use any specific window manager, it is recommended that the one used should be a GNOME-compliant window manager. We will now install Sawfish, which was developed solely to fulfil this requirement. We will install the package and all its dependencies:

```
brau:~# apt-get install sawfish-gnome
```

Consequently, we now have another window maker installed. We will stop the display manager and relaunch it so that this new window maker is integrated into it (GNOME also has its own display manager, `gdm`, which we can install if we want). We now have two options for achieving our objective, which is to: run GNOME. The first is to launch Sawfish, from the display manager or using `startx` and, once in the program, launching `gnome-session` from an X terminal. The second consists of carrying out the same operation but the other way around; in other words, launching GNOME with the same procedures as Sawfish, and then launching `sawfish` from an X terminal. We recommend proceeding as follows if we want Sawfish to run next time we start GNOME up (the session manager of GNOME itself will be in charge of carrying out and logging the changes that are necessary for this to occur).

Once we have familiarised ourselves with GNOME, what we can do is install some packages that can be useful, specifically `gnome-help` and `gnome-terminal`; the first provides us with an interface where we can read the manuals (mans), the text files in a graphic environment, and the second installs the GNOME's own xterm.

11.7.2. KDE

Unlike GNOME, KDE does need a specific window manager, which is `kwm`, based on Qt toolkit and the actual kdelibs. It also has its own launcher panel: `kpanel`, for its own file manager: `Konqueror` and its own configuration utility: `Control Panel`. Obviously, KDE can be installed in the same system in which we have installed GNOME and there are even applications belonging to a desktop manager that can run in the other system. Besides, KDE also has its own display manager (`kdm`) along with very many applications. Again, we recommend readers to visit the respective websites to learn about their possibilities: http://www.kde.org. Likewise, we can run the following line to see how KDE integrates in Debian:

```
brau:~# apt-cache search kde
```

The basic KDE packages are in the `kdebase package`. This, therefore, will be the first that we install:

```
brau:~# apt-get install kdebase
```

Again, we must reboot our window manager in order to access the desktop manager that has just been installed. Once we have done this, we can proceed to install the file manager, the package called konqueror. Using the package `kde-i18n-es`, we can install the files required for KDE to work in Spanish.

As of this point, each user will be able to install the different packages of the project that they want. As we have seen, in order to pre-establish the default window maker, we will use the menu of the `update-alternatives` command to select the session manager:

```
brau:~# update-alternatives x-session-manager
```

11.8. Personalising some of the aspects

The design of the X graphic environment is due to the ambitious aim of getting the best performance possible from the available hardware, using the least resources possible, whilst providing the greatest flexibility. The structure of the client server on which this system is based makes it possible to achieve these objectives, and the apparent difficulties that inexperienced users will come across will disappear quickly with a bit of practice, allowing the many advantages that this design provides to flourish. The aim of this subsection is only to provide a cursory idea of the power of this system, which is completely evident when working on a network, although it is somewhat overshadowed when working on a stand alone system, such as the one on which this course is based. In any case, we will discuss some of the concepts that may be useful when working on a network.

11.8.1. Personalisation of local aspects

Generally, the configuration files are located in the directory `/etc/X11/` or in one of its subdirectories. Each user can personalise and redefine the configuration parameters and add new ones by creating or editing the files in their home directory that have the same name as those of a general configuration, but preceded with a ".". It will be possible to redefine or establish all those parameters that do not require superuser permissions, as the home files will be processed after those of the general configuration, and the parameters will always take the last value that are assigned to them.

Xsession

`/etc/X11/Xsession` is a script that runs when a user starts a session.

This script is the one that governs the whole of the process of starting a session until we can start to work, and it is also in charge of managing the error messages that may occur during this process, which are logged in `$HOME/.xsession-errors`.

In `$HOME/.xsession`, we can personalise the booting process for a particular user. So, if we want the window manager to be `blackbox`, and we want `bbkeys` to launch automatically in the background when we begin the session, this will contain the following lines:

```

bbkeys blackbox
```

Xresources

In the file `$HOME/.Xresources`, we can personalise the appearance of the different applications. The syntax is `application*parameter: value`. Therefore, if we want to invert the colours of the application `xterm`, we will add the following line in the file:

```
Xterm*reverseVideo: true
```

The xrdb command is in charge of managing the database of Xresources. Using xrdb -query, we can learn about all the established properties and their value, and, using the -display parameter, we will obtain a list of all the parameters that the command accepts. If we enter the location of a file as a parameter, this will read all the parameter definitions from here.

Xmodmap

The graphic server uses the table of character codes to convert the signals from the keyboard (server-independent) to system symbols (server-dependent). The conversion table that has to be used will have been selected during the keyboard configuration process, but the xmodmap command will allow us to modify its contents. An example of how it would be used follows:

Assignment of symbols

These commands were used for a very long time due to the investment of the symbol assignments in the conversion tables; now, however, this problem has been solved.

```
brau:~# xmosmap -e "keycode 127 = Delete" brau:~# xmosmap -e "keycode 22 = BackSpace"
```

The -pk, xmodmap parameters will return all the contents of the conversion table that is being used.

11.8.2. Personalisation of aspects of the network

The aspects discussed here are also interesting for a stand alone system, as, like the whole operating system, the X system always uses a network-oriented design.

$DISPLAY

The variable $DISPLAY can be used to tell the client which server it should communicate with. Its syntax is as follows: hostname:display number.screen number. Therefore, if we had defined another graphic terminal to system X, adding the following line to /etc/X11/xdm/Xservers:

```
:1 local /usr/X11R6/bin/X vt8
```

we could launch a graphic application from an xterm of a graphic terminal to another by defining the appropriate variable. For these reasons, if we wanted to launch xeyes from the first graphic terminal, using xterm, and to display it on the second, we would proceed as follows:

```
brau:~$ set DISPLAY :0.1; export DISPLAY brau:~$ xeyes
```

If we enter a graphic session, open an xterm, change the user using the su command and try to launch a graphic application, we will receive an error message stating that a connection may be established with the server. A strategy to avoid this problem is to use the -p parameter to export the whole set of variables of the environment, thereby avoiding the server rejecting our connection request. This practice may be very useful for launching configuration programs that require root permissions, as this will allow us to avoid having to enter the graphic environment as a root user (a practice that is not very advisable and that, although it is allowed by default, is often restricted manually).

xhost and xauth

The `xhost` command makes it possible to establish which computers can access the graphic server remotely, in other words, which client machines can launch an application that can be displayed on the server. Its syntax is as follows: `xhost +hostname`. If we do not specify a hostname, any machine will be able to launch applications on the server. By default, no connection is allowed from any remote machine. The `xauth` command can be used to determine which users can launch applications on the graphic server. Consequently, using these two commands together allows us to establish the security policy with regard to accessing server X, in a fairly reasonable way. `xhost` + for stand alone systems.

11.9. Configuring printers

Configuring printers may be easier from the graphic environment. There are very many applications that can be used to configure the native printing system and others that replace this system for their own, commonly also based on the client-server structure. In order to install CUPS (Common Linux Printing System), we will have to install the package of the printing server, `cupsys`; and it is advisable to install, along with this package, the printing clients package, `cupsys-client`. We can also install the package called `cupsys-bsd` so that we have the commands usually found in the BSD printing system. For example, we could install the printing system using the command:

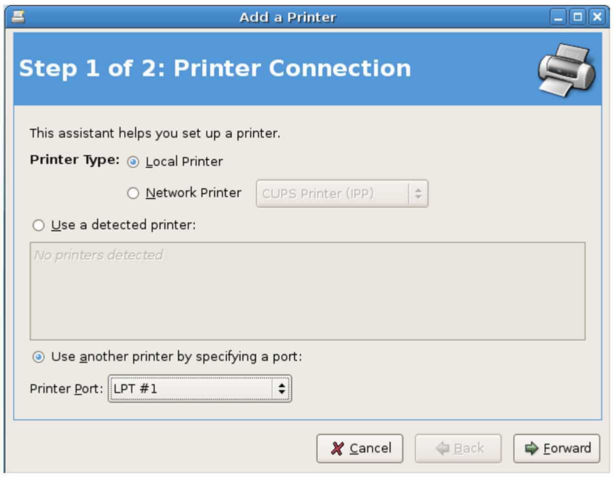
```
# apt-get install cupsys cupsys-client cupsys-bsd printconf \ foomatic-filters-ppds
```

This last package contains the PPDs (Postscript Printer Definitions) of Adobe, which is used by CUPS to define the printers.

In GNOME we can go to "System/Administration/Printers ". We double-click on New Printer, enter the root password and follow the steps for defining the printer, depending on whether it is a local or network printer. The second step consists of finding out the name of the manufacturers and the model of the printer. Once we have defined the printer, we should highlight it using the mouse's right button and go to Properties. We then highlight the option Become the administrator and modify the configuration if necessary.

The following sequence of images shows the process for installing a printer using the assistant `gnome-cups`:

Figure 11.1

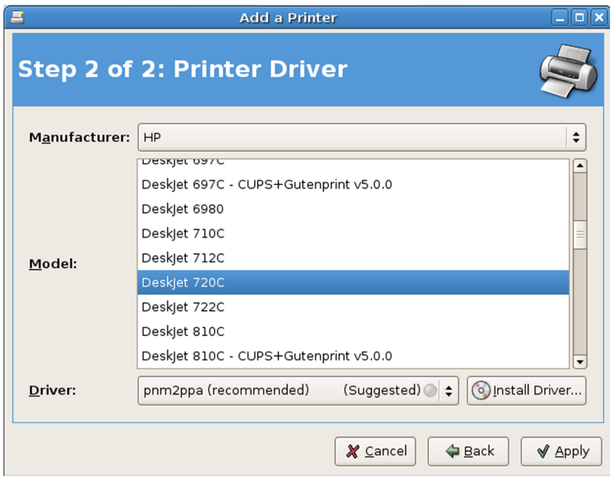


Step 1 of 2: Printer Connection

This assistant helps you set up a printer.

Printer Type: [✓ Local Printer]
Network Printer CUPS Printer (IPP)
Use a detected printer:
No printers detected
[✓ Use another printer by specifying a port:]
Printer Port: LPT #1

Figure 11.2



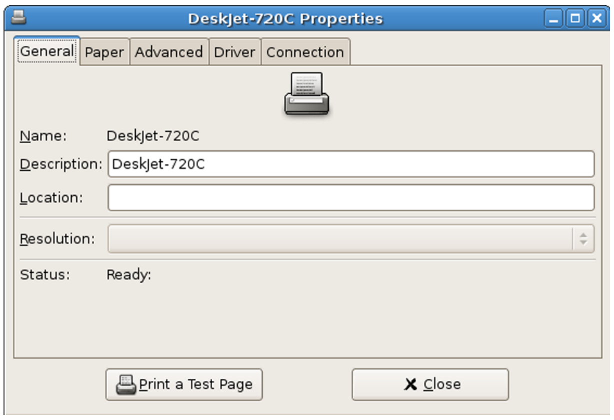
Step 2 of 2: Printer Driver

Manufacturer: HP

Model:
DeskJet 697C
DeskJet 697C - CUPS+Gutenprint v5.0.0
DeskJet 6980
DeskJet 710C
DeskJet 712C
[✓ DeskJet 720C]
DeskJet 722C
DeskJet 810C
DeskJet 810C - CUPS+Gutenprint v5.0.0

Driver: pnm2ppa (recommended) (Suggested)

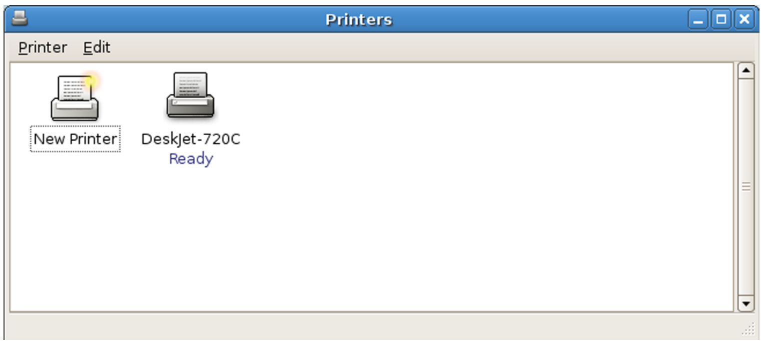
Figure 11.3



Name: DeskJet-720C
Description: DeskJet-720C
Location:
Resolution:
Status: Ready:

Once we have installed the printer, a window will appear; this window corresponds to the tool `gnome-cups`:

Figure 11.4



If the printer uses the HP JetDirect system, it is better to install the package `hplip`,

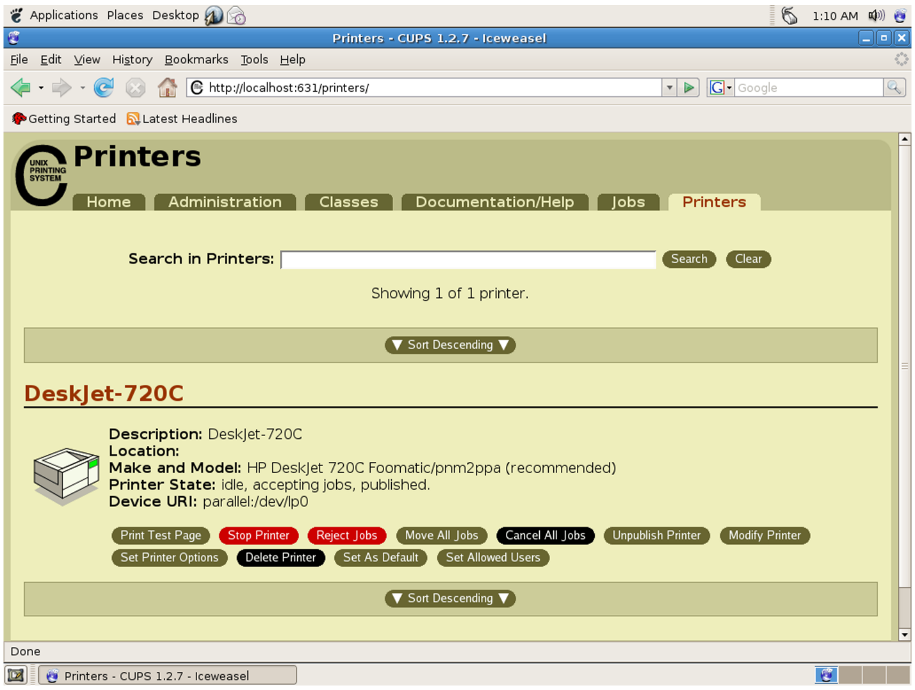
```
  # apt-get install hplip
```

and use the tool `hp-setup` to define the printer.

Another way of configuring a printer using CUPS is through the website of the CUPS service. All we have to do is open our browser and enter the URL: http://localhost:630, which is the port that our server listens to.

The following screenshots show the appearance of this administrative utility:

Figure 11.5

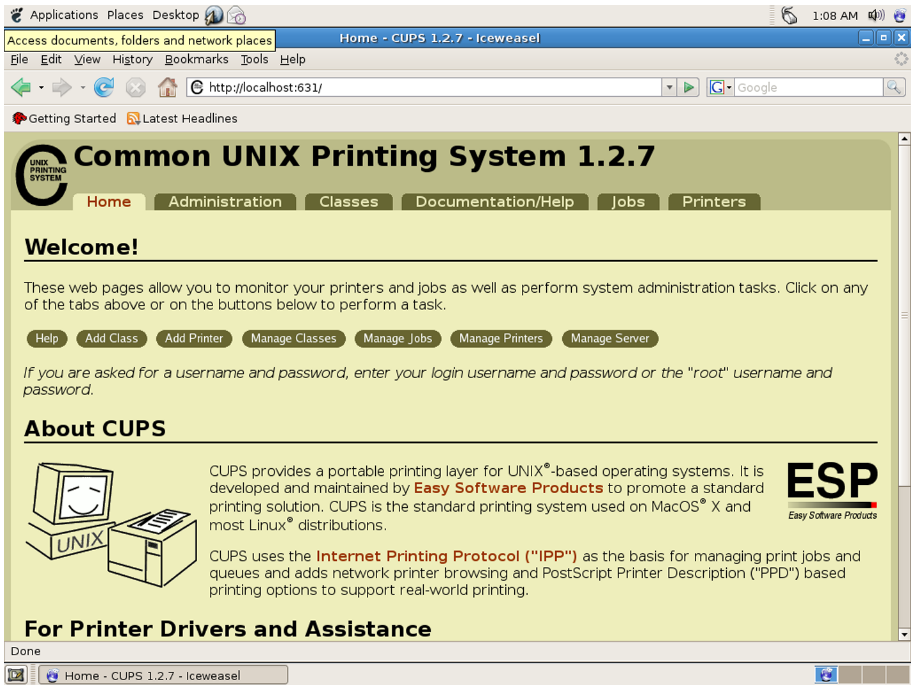

Applications Places Desktop
1:10 AM
Printers - CUPS 1.2.7 - Iceweasel
File Edit View History Bookmarks Tools Help
http://localhost:631/printers/
Google
Getting Started Latest Headlines
Printers
Home Administration Classes Documentation/Help Jobs Printers
Search in Printers:
Search Clear
▼ Sort Descending ▼
Print Test Page Stop Printer Reject Jobs Move All Jobs Cancel All Jobs Unpublish Printer Modify Printer
Set Printer Options Delete Printer Set As Default Set Allowed Users
▼ Sort Descending ▼
Done
Printers - CUPS 1.2.7 - Iceweasel

Showing 1 of 1 printer.

DeskJet-720C

Description: DeskJet-720C
Location:
Make and Model: HP DeskJet 720C Foomatic/pnm2ppa (recommended)
Printer State: idle, accepting jobs, published.
Device URI: parallel:/dev/lp0

Figure 11.6



Common UNIX Printing System 1.2.7

Welcome!

These web pages allow you to monitor your printers and jobs as well as perform system administration tasks. Click on any of the tabs above or on the buttons below to perform a task.

If you are asked for a username and password, enter your login username and password or the "root" username and password.

About CUPS

CUPS provides a portable printing layer for UNIX®-based operating systems. It is developed and maintained by Easy Software Products to promote a standard printing solution. CUPS is the standard printing system used on MacOS® X and most Linux® distributions.

CUPS uses the Internet Printing Protocol ("IPP") as the basis for managing print jobs and queues and adds network printer browsing and PostScript Printer Description ("PPD") based printing options to support real-world printing.

For Printer Drivers and Assistance

11.10. OpenOffice

In this subsection, although the subject is not within the scope of the main aims of this course, we will present an office suite, which can be extraordinarily useful for those of us who are used to working with software of this type. The element in question is OpenOffice.org, a project derived from StarOffice, by Sun MicroSystems. We must point out that this project is multiplatform and that it can therefore be implemented in other operating systems that are not Unix-type.

To install the suite, we can use the following command (all on one line):

```
brau:~# apt-get install openoffice.org openoffice.org-base openoffice.org-calc
openoffice.org-common openoffice.org-core openoffice.org-draw openoffice.org- evolution
openoffice.org-gnome openoffice.org-gtk openoffice.org-help-en openoffice.org-help-es
openoffice.org-impress openoffice.org-java-common openoffice.org-l10n-es
openoffice.org-math openoffice.org-writer
```

All we have to do during the installation process (which should be carried out in a graphic environment) is to remember that we have to install it for the network. In this way, it will only be necessary for each user's home directory to contain one small directory, in which their personalised configurations are saved.

Once we have installed the program for the first time, each user must run the following program to create their own directory:

```
/usr/lib/openoffice/program/setup
```

Once we have done this and we have answered a few questions, the `openoffice` command will open the suite.

Currently the OpenOffice version in Debian Etch is 2.0.4.

11.11. Conclusion

With this workshop, we come to the end of the didactic material of the second module of the Masters. In the workshop, we have learnt how to install the graphic environment in the system, which, as we saw at the time, is not an essential part within the operating system. But it is evidently very useful in many cases and many people say that the operating system that we have studied in this module can be a serious alternative to other systems. For all of these reasons, we, the authors, would like to express our total conviction that GNU/Linux is an extraordinary operating system, not only because of its graphic interface, which is perhaps what most catches our attention when we first look at the system, but due to many other countless arguments, of which we would highlight the philosophy behind the system, its reliability, adaptability, power, potential levels of security etc. We are convinced that this operating system is an investment for the future, from which we can only expect positive developments, although it has already proven that it is a serious competitor in the world of operating systems,. We hope that we have provided enough knowledge and transmitted the enthusiasm that is needed for readers to begin their own journey into the world of GNU/LINUX, as well as known how to open the doors to a community in which everyone is welcome and respected.