Libranet Basics

Covers Libranet GNU/Linux 2.8.1

Libranet Basics: Covers Libranet GNU/Linux 2.8.1

Table of Contents

List of Tables

I. Getting started

Chapter 1. About this book

About this book

This book aims to provide an introduction to Libranet GNU/Linux. It addresses people who have little or no GNU/Linux experience. It aims to cover the Libranet GNU/Linux installation, basic GNU/Linux commands and the configuration of Libranet GNU/Linux. As you can see the book is still work in progress, but the first bits are released in the “release early, release often” spirit.

This book is written and maintained by members of the Libranet community, and is available freely under the terms of the GNU Free Documentation License. It is continually under development, not just to keep up with the latest Libranet versions, but also to refine the documentation, and extend it where it is deemed necessary. The latest version can be found at http://www.libranet-basics.org/

We wish everybody a good time with Libranet Linux, and we hope this book is useful for you.

The authors

- **Daniël de Kok** <danieldk at pobox dot com>

  Daniël coordinates the “Libranet Basics” book project, and wrote a large share of the chapters.

- **Robert K. Naylor**

  Robert contributed to the “Running Windows programs on Libranet” chapter.

Conventions

This section gives a short summary of the conventions in this book.

File names

File or directory names are printed as: `/path/to/file`. For example: `/etc/fstab`

Commands

Commands are printed as bold text. For example: **ls -l**

Screen output

Screen output is printed as this:

```
Hello world!
```

If commands are being entered in the screen output the commands will be printed as bold text:

```
$ command
Output
```

If a command is executed as root, the shell will be displayed as “#”. If a command is executed as a normal non-privileged user, the shell will be displayed as “$”.

Notes

Some sections contain extra notes. It is not necessary to know the information in notes, but notes may provide valuable information, or pointers to information. Notes are printed like this:

> **Note:** This is a note.

Chapter 2. An introduction to Libranet GNU/Linux

What is Linux?

Linux is a Unix-like kernel which is written by Linus Torvalds and other developers, who communicate using the Internet. Linux runs on many different architectures, for example on many IA32, IA64, AMD64, Alpha, m68k, SPARC and PowerPC machines. The latest kernel and more information can be found at: http://www.kernel.org

Linux is often confused with the GNU/Linux system. Linux is only a kernel, not a complete operating system. GNU/Linux consists of the GNU operating system with the Linux kernel. Please read the following section for a more detailed explanation of GNU/Linux.

What is GNU/Linux?

At the beginning of the eighties Richard Stallman started an ambitious project with the goal to write a free Unix-like operating system. The name of this system is GNU (GNU is Not Unix). At the beginning of the nineties most important components of the GNU operating system were written, except for the kernel, which is still under development under the name HURD. HURD consists of some servers which provide Unix-like kernel functionality. In turn these servers run under the Mach micro-kernel. At the beginning at the nineties the HURD team still had to wait till the Mach sources were released as free software. In the meanwhile Linus started filling the gap with the Linux kernel. GNU/Linux thus refers to the GNU system running on the Linux kernel. Right now the HURD kernel is also in a usable state and can be downloaded in the form of the GNU/HURD operating system. The Debian (http://www.debian.org/) project has even developed a version of the GNU operating system which works with the NetBSD (http://www.netbsd.org/) kernel. We should call "Linux distributions" "GNU/Linux distributions", because GNU is a substantial part of most distributions.

What is Libranet GNU/Linux?

Libranet is a GNU/Linux distribution that is based on Debian GNU/Linux. Debian provides a huge package collection and an excellent package management tool named APT. Libranet builds on the reliable Debian base and adds an excellent installation program and a configuration tool named AdminMenu. Besides that Libranet includes a carefully selected collection of software. Distinguishing features that are often mentioned are:

- A user friendly installer.
- The AdminMenu configuration tool.
- A wide selection of packages.
- Good support.
- A friendly and helpful community.

Getting Libranet GNU/Linux

Libranet can be purchased on the Libranet site as a download, or on CD-ROM. Besides that there are many software and Linux vendors around the world that provide Libranet on CD-ROM. The official Libranet site can be found at http://www.libranet.com/

Besides offering the latest Libranet version on CD-ROM or as a paid download, Libranet Computer Systems Ltd. also offers an older downloadable version for evaluating Libranet. Have a look at the Libranet site for more information.

Chapter 3. Sources of help

On your system

Linux HOWTO's

The famous Linux HOWTOs are a collection of documents which cover specific parts of a GNU/Linux system. Most HOWTOs are distribution independent, and therefore very useful for using them with Libranet GNU/Linux.

Manual pages

Most Unix-like commands are covered by the a traditional *nix help system called the manual pages. You can read the manual page of a program by using the **man** command. Executing **man** with the name of the command as a parameter shows the manual page for that command. For example:

```
$ man ls
```

If you do not know the exact name of a manual page or command, you can search a page using the keyword (*-k*) parameter:

```
$ man -k routing
NETLINK_ROUTE (7)    - Linux IPv4 routing socket
netstat (8)          - Print network connections, routing tables, interface statistics,
route (8)            - show / manipulate the IP routing table
rtnetlink (7)        - Linux IPv4 routing socket
NETLINK_ROUTE (7)    - Linux IPv4 routing socket
netstat (8)          - Print network connections, routing tables, interface statistics,
route (8)            - show / manipulate the IP routing table
rtnetlink (7)        - Linux IPv4 routing socket
```

We have to add that there are also manual pages that cover other things than commands. These sections of manual pages are available:

```
1   Executable programs or shell commands
2   System calls (functions provided by the kernel)
3   Library calls (functions within program libraries)
4   Special files (usually found in /dev)
5   File formats and conventions eg /etc/passwd
6   Games
7   Miscellaneous  (including  macro  packages and conven-
    tions), e.g. man(7), groff(7)
8   System administration commands (usually only for root)
9   Kernel routines [Non standard]
```

On the Internet

The Libranet support solutions database

The Libranet support solutions database has a collection of short HOWTO’s covering a wide range of topics. The Libranet support solutions database can be found at: http://libranet.com/support/

The Libranet forum

The Libranet forum is well known for its friendly and helpful community. Besides being a place for solving problems it is an excellent place to just hang out and talk about Libranet and GNU/Linux. The Libranet forum can be found at: http://forum.libranet.com/

The libranet-users mailinglist

The libranet-users mailinglist can be used to ask questions about Libranet. Information about subscribing to the list can be found on the Libranet support page at http://www.libranet.com/support.html

The Libranet IRC channel

Libranet has its own IRC channel, #libranet, which can be found on the irc.debian.org server. Fire up an IRC client, like XChat, BitchX or Epic, connect to the irc.debian.org server and enter "/join #libranet" to join the Libranet channel.

Chapter 4. General concepts

This chapter gives an introduction to some general Unix and GNU/Linux concepts. It is a good idea to read this chapter thoroughly if you do not have any UNIX or GNU/Linux experience. Many concepts covered in this chapter are used in this book and in GNU/Linux.

Multitasking

Introduction

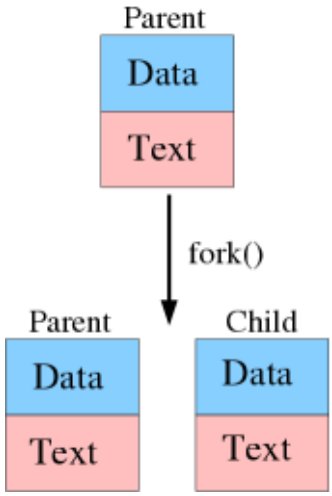
One of Linux' strengths is multi-tasking. Multi-tasking means that multiple processes can run at the same time. You might wonder why this is important to you, because most people are using one application at a time. Besides the more obvious reason that it is just handy to browse while you have a word processor running in the background, multi-tasking is a bare necessity for Unix-like systems. Even if you have launched no applications there are a bunch of processes running in the background. Some processes might provide network services, others sit there showing a login prompt on other consoles, and there is even a process that executes scheduled tasks. These processes that are running in the background are often called *daemon* (not to be confused with the word demon, a daemon is a protective angel). At a later stage we are going to look at how you can move processes to the background yourself (see Chapter 10).

> **Note:** Note that on single-processor systems processes are not really running simultaneously. In reality a smart scheduler in the kernel is dividing CPU time between processes, giving the illusion that processes are running simultaneously.

Processes and threads

Applications run as one or more processes. To see what a process actually is we need to know what it consists of. Every process basically has two areas; an area that is named *text* and an area that is named *data*. The *text* area is the actual program code; it is used to tell the computer what to do. The *data* area is used to store information that the process has to keep. The operating system makes sure that every process gets its time to execute. New processes can be created by duplicating a running process with a system call named *fork*. Figure 4-1 shows a fork in action schematically. The parent process issues a *fork()* call, and as a result a new process is created.

Figure 4-1. Forking a process



The problem with processes is that they can get quite large, and that is not very efficient for computers with more than one processor. This problem is solved by duplicating the *text* area of a process. So, a threaded process is basically a program that has multiple instances of its executing code running, but these instances share the data area of the process (unlike a fork). These threads can be divided over multiple CPUs, making it possible to run one process on more than one CPU simultaneously.

Filesystem hierarchy

Structure

Operating systems store data in filesystems. A filesystem is basically a collection of directories that hold files, like the operating system, user programs and user data. In GNU/Linux there is only one filesystem hierarchy, this means GNU/Linux doesn’t have multiple drives (e.g. A:, C:, D:), like Windows. The filesystem looks like a tree, with a root directory (`/`) which has no parent directory, branches, and leaves (directories with no subdirectories). Directories are separated using the “/” character.

Figure 4-2. The filesystem structure

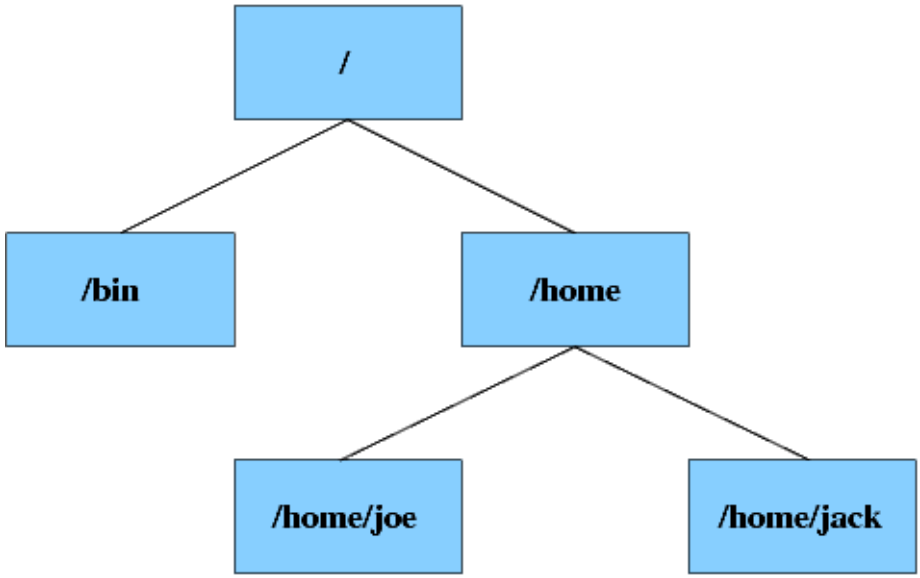


Figure 4-2 shows the structure of a filesystem. You can see that the root directory (`/`) has two child directories; `bin` and `home`. The `home` directory has two child directories, `joe` and `jack`. The diagram shows the full pathname of each directory. The same notation is used for files. Suppose that there is a file named `memo.txt` in the `/home/jack` directory, the full path of the file is `/home/jack/memo.txt`.

Each directory has to special entries, "`.`", and "`..`". "`.`" refers to the same directory, "`..`" to the parent directory. These entries can be used for making relative paths. Suppose that you are working in the `jack` directory. From this directory you can reference to the `joe` directory with `../joe`.

Mounting

You might have started to wonder how it is possible to access other devices or partitions than the hard disk partition which holds the root filesystem. Linux uses the same approach as UNIX for accessing other mediums. Linux allows the system administrator to connect a device to any directory in the filesystem structure. This process is named "mounting". For example, one could mount the CD-ROM drive to the `/cdrom` directory. If the mount was correct, the files on the CD-ROM can be accessed through this directory. The mounting process is described in detail in the Section called *Mounting filesystems* in Chapter 9.

Common directories

The Filesystem Hierarchy Standard Group has attempted to create a standard that describes which directories should be available on a GNU/Linux system. Nowadays most major distributions use the Filesystem Hierarchy Standard as a guideline. This section describes some mandatory directories on GNU/Linux systems.

Please note that GNU/Linux does not have a separate directory for each application (like Windows), files are ordered by function and type. For example, the binaries for most common user programs are stored in `/usr/bin`, and their libraries in `/usr/lib`. This is a short overview of important directories:

- **/bin**: essential user binaries that should still be available in case the `/usr` is not mounted.
- **/dev**: device files. These are special files used to access certain devices.
- **/etc**: the `/etc` directory contains all important configuration files.
- **/home**: contains home directories for individual users.
- **/lib**: essential system libraries (like glibc), and kernel modules.
- **/root**: home directory for the *root* user.
- **/sbin**: essential binaries that are used for system administration.
- **/tmp**: a world-writable directory for temporary files.
- **X11R6**: the X Window System.
- **/usr/bin**: stores the majority of the user binaries.
- **/usr/lib**: libraries that are not essential for the system to boot.
- **/usr/sbin**: non-essential system administration binaries.
- **/var**: variable data files, like logs.

Devices

Introduction

In UNIX and Linux almost everything is represented as a file, including devices. Each GNU/Linux system has a a directory with special files, named `/dev`. Each file in the `/dev` directory represents a device. You might wonder how this is done; a device file is a special file because it has two special numbers, the *major* and the *minor* number. The kernel knows which device a device file represents by these numbers. The following example shows these numbers for a device:

```
$ ls -l /dev/zero
crw-rw-rw-    1 root     root       1,   5 Apr 22  2003 /dev/zero
```

The **ls** lists files and information about files. In this example information about the `/dev/zero` device is listed. This particular device has *1* as the major device number, and *5* as the minor device number.

> **Note:** If you have the kernel sources unpacked after installing Libranet, you can find a comprehensive list of all major devices with their minor and major numbers in `/usr/src/linux/Documentation/devices.txt`. An up-to-date list is also available online at: ftp://ftp.kernel.org/pub/linux/docs/device-list/

For the Linux kernel there are two types of devices: *character* and *block* devices. Character devices can be read byte by byte, block devices can not. Block devices are read per block (for example 4096 bytes at a time). Whether a device is a character or block device is determined by the nature of the device. For example, most storage media are block devices, and most input devices are character devices. Block devices have one distinctive advantage, namely that they can be cached. This means that commonly read or written blocks are stored in a special area of the system memory, named the cache. Memory is much faster than most storage media, so it is a huge performance benefit to

perform read and write operations on commonly used blocks in memory. Of course, eventually changes have to be written to the storage medium.

ATA and SCSI devices

There are two categories of devices that we are going to look into in detail, because understanding the naming of these devices can be crucial for partitioning a hard disk, or for mounting. Almost all modern computers use ATA hard disks and CD-ROMs. Under Linux these devices are named in the following way:

```
/dev/hda - master device on the first ATA channel
/dev/hdb - slave device on the first ATA channel
/dev/hdc - master device on the second ATA channel
/dev/hdd - slave device on the second ATA channel
```

On most computers the hard disk is the master device on the first ATA channel (`/dev/hda`), and the CD-ROM the master device on the second ATA channel. Hard disk partitions have the device name plus a number. For example, `/dev/hda1` is the first partition on the `/dev/hda` disk.

Most average PCs do not have SCSI hard disks or CD-ROM drives, but SCSI is often used for USB drives. For SCSI drives the following notation is used:

```
/dev/sda - First SCSI disk
/dev/sdb - Second SCSI disk
/dev/sdc - Third SCSI disk
/dev/scd0 - First CD-ROM
/dev/scd1 - Second CD-ROM
/dev/scd2 - Third CD-ROM
```

Partitions are notated in the same way as ATA disks; `/dev/sda1` is the first partition on the first SCSI disk.

Chapter 5. Installation

This is a placeholder for the future installation chapter. In the meanwhile, check out Libra Computer System Ltd.’s own excellent Libranet Install Guide at: http://libranet.com/guide/

II. Getting your work done

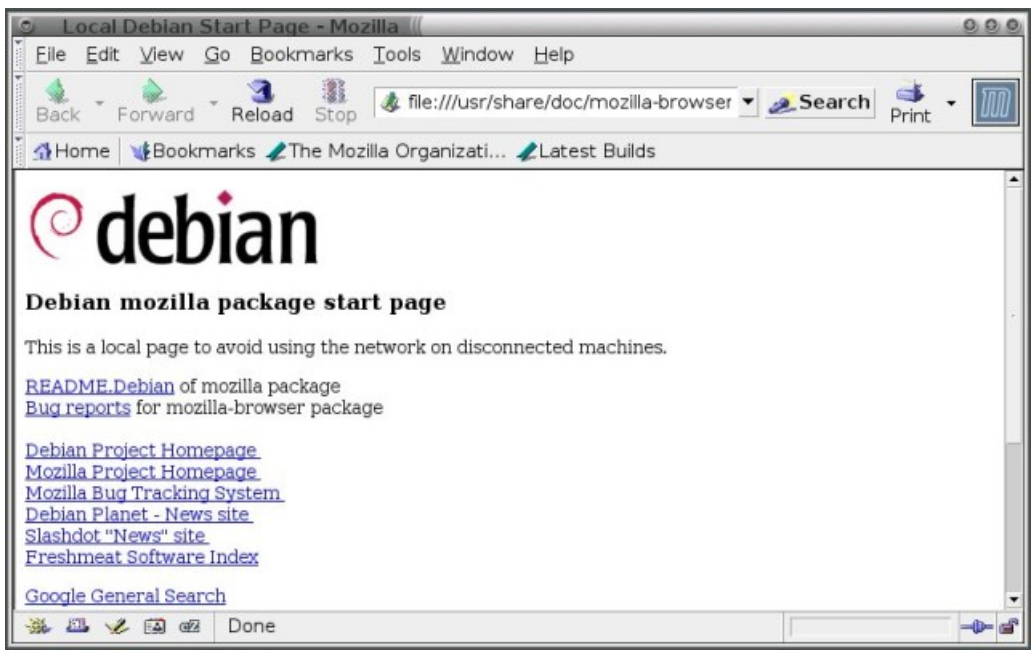
Chapter 6. Using the Internet with Libranet

Browsing the web with Mozilla

Mozilla is one of the most popular GNU/Linux browsers, and is rapidly gaining market share on the Windows platform. Many browsers are based on the Mozilla rendering engine (named Gecko), for example FireFox, Epiphany and Galeon. Mozilla is a complete suite, containing a web browser, an e-mail program, a USENET client and a IRC client. This section is a short introduction to Mozilla, and provides some hints that might be helpful.

Libranet includes Mozilla, and it can be installed via the conventional Libranet package management tools. Mozilla can be started by launching “Apps/Net/Mozilla Navigator”. The default Libranet graphical environment also includes a Mozilla launcher in the main menu.

Figure 6-1. The Mozilla browser



Plugins

Some sites require plugins like Flash or Java. Realplayer, Flash, Java and the MozPlugger plugins can be installed via AdminMenu. These options are available under the “Browser Plugins” tab in AdminMenu.

Tabs

Figure 6-2. Mozilla browser tabs

Mozilla provides tabs, which means you do not have to open a new window if you want to visit several sites simultaneously. You can just create multiple tabs within one channel, which have different active sites. Try pressing the <Ctrl> + <t> keys, as you can see a new tab is opened in which you can start browsing. You can browse through the tags with the <Ctrl> + <Page Up> and <Ctrl> + <Page Down> keys. You can also open a link in a new tab by clicking on the link with the right mouse button and selecting “Open Link in New Tab”.

Popup blocking

Many Internet sites display popup ads, which can become a bit annoying, fortunately Mozilla has a function that blocks popup windows. This option can be found in the preferences window, which is accessible from the “Edit” menu. The popup window settings are located under the “Privacy & Security” section.

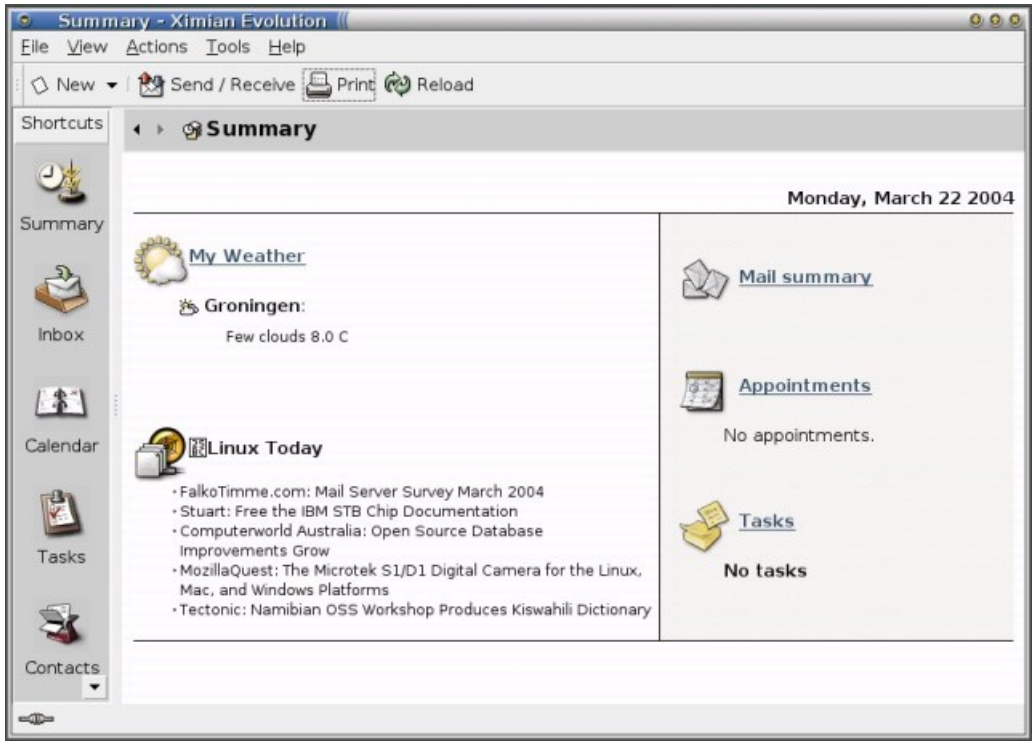
Check the “Block unrequested popup windows” to block popup Windows by default. There might be some sites that you regularly visit, which have popup windows which you do not want to block. For these sites an exception can be made by clicking the “Allowed Sites” button. A new window will appear, which allows you to add sites that should be excluded from the popup killer.

Ximian Evolution, the ultimate e-mail client

Ximian Evolution is an e-mail client that offers alot of personal information management (PIM) features, that is being developed by Novell. It features:

- A summary page that can display the weather, newsfeeds, appointments, tasks and other information, a calendar, an addressbook and much more.
- A calendar, which can be used for appointments and other stuff.
- An addressbook, which allows detailed configuration of contacts.
- PGP/GnuPG, SASL and SSL/TSL message encryption and/or signing.
- Support for LDAP directory servers.
- Much more.

Figure 6-3. Ximian Evolution



The summary screen

Evolution has a very nice summary screen, which can display the local weather, news feeds appointments and tasks. This screen is easily customizable. We will have a look at some options in this section.

My weather

The “My Weather” functionality allows you to display information about the weather at one or more locations. To change the weather options, open up the settings dialog by clicking on the “Tools -> Settings...” menu item. Go to the “Summary Preferences”, and click on the “Weather” tab. The first list on the left shows all available locations for which weather information can be shown. The right list shows which for which locations weather is currently shown. You can add and remove locations using the “Add” and “Remove” buttons. After you have customized the “Shown” list you can choose to show the weather in Celsius instead of Fahrenheit, by activating the “Celsius” option. Click “OK” when you are finished, and the “My weather” overview will show the weather at the locations you selected.

News feeds

Evolution accepts news feeds in RSS or RDF formats. This is a standard for supplying news feeds, and many news sites provide a RSS or RDF news feed. The selection of news feeds can be customized in the settings dialog. Click on the “Tools -> Settings...” menu item to open the setting dialog. Select on the “Summary Preferences” icon, and click the “News Feeds” tab to open the news feeds tab. This tab works like the weather tab; a list of news feeds is listed on the left side, the right list shows which news feeds are currently displayed. You can remove or add items with the “Add”

and “Remove” buttons. It is also possible to add an unlisted news feed, by clicking the “New Feed” button. Evolution will ask you to input the name of the news feed, and the URL where the RDF/RSS news feed can be found. Suppose that you would like to add the OSNews (http://www.osnews.com/) news feed, you could enter *http://www.osnews.com/files/recent.rdf* as the URL.

Setting up e-mail accounts

The next thing you will probably want to do is to setup some e-mail accounts. The first time Evolution is started it will ask the user to setup an account, so there is a good change that you have already set up an e-mail account. The account settings are located under the “Mail Accounts” icon in the Evolution settings dialog, which can be launched by clicking the “Tools -> Settings...” menu item. In the account summary you can add, edit and remove accounts.

Click the “Add” button to add an e-mail account. This will launch an assistant that will help you setting up an account. Just click “Forward” at the welcome screen. The assistant will ask you for information about your identity. Here you can fill in the name and e-mail address that will displayed in the “From” headers. If the address that users should apply to differs from the e-mail address, fill in the optional “Reply-To” field. If the reply address is identical to the e-mail address, this field can be ignored. Optionally you can fill in the “Organization” field. Press “Forward” after completing this form.

After setting the identity you can specify which mail server should be used for sending mail. The “None” option is useful if you just want to add an additional identity, but no other full blown e-mail account. Suppose that you have two e-mail addresses, *sales@mycompany.com* and *ceo@mycompany.com*, and e-mails to both addresses are delivered to the same account. In this case you could set up an extra identity, which will allow you to choose the appropriate “From” address. If you would like to receive e-mail using this account, there are several server types you can choose from. The most commonly used types are POP and IMAP. This information should be supplied when you registered your e-mail account. It is safe to test which works if you do not know which type of server is used. After selecting the IMAP or POP server types you can enter the name of the host or IP address of the e-mail server, and the user name for the account. If you would like to store the password for this account after the first time you entered it, check the “Remember this password” check-box. Clock “Forward” when you are finished. The assistant will then ask you to tune the settings a bit, for example, you can set your account up to check for new mail automatically. Click “Forward” to go to the “Sending E-Mail” configuration.

In the “Sending E-Mail” screen you can set up how Evolution can send mail. Unless you have set up a local mail server on the machine Evolution is worked on you should select “SMTP” as the server type. You should also fill in the name or IP address of the SMTP server. If your SMTP server requires authentication you can check the “Server requires authentication” check-box, and fill in the user name. Click “Forward” to continue.

The final setup screen will ask you to supply the name of this e-mail account. You can fill in anything you like. Check the “Make this my default account” check-box if you would like to make this the default account. This means this account will be used by default when you compose a new e-mail. After clicking “Forward” the assistant will congratulate you with finishing the account configuration.

GAIM, the universal instant messenger

GAIM is one of those killer applications. Once you got used to it changes are you never want to look

back at other messengers. GAIM supports AIM, ICQ, MSN Messenger, Yahoo, IRC, Jabber, Gadu-Gadu and Zephyr. The nice thing about GAIM is that you can use these services simultaneously.

Figure 6-4. The GAIM messenger

Some first advice

There have been many problems in the past with supporting all messengers, because many messaging protocols are closed. The Jabber protocol is a big exception; it is an open protocol that is an official standard. The Jabber user-base is quite large and surpassed ICQ in user numbers. There are many Jabber clients available, for most commonly used platforms. Try to convince yourself and friends to switch to Jabber. More information about Jabber can be found at: http://www.jabber.org/

Configuring accounts

When you launch GAIM for the first time you can setup accounts using the accounts button. After you have installed GAIM you can setup accounts with the *Tools -> Accounts* menu item. This will present a window with the accounts that are currently configured. You can use the *Online* check-box next to an account name to go online or off-line. If the *Auto-login* check-box is checked GAIM will automatically log in the next time GAIM is launched. Adding, modifying or deleting accounts is easy with the buttons on the bottom side of the window. Adding an account should be fairly straightforward, make sure you select the right protocol.

If GAIM does not work (well) with a supported protocol, you should probably update your GAIM version. It often happens that the protocols are changed slightly, which breaks third party clients like GAIM.

Chapter 7. The OpenOffice suite

Introduction

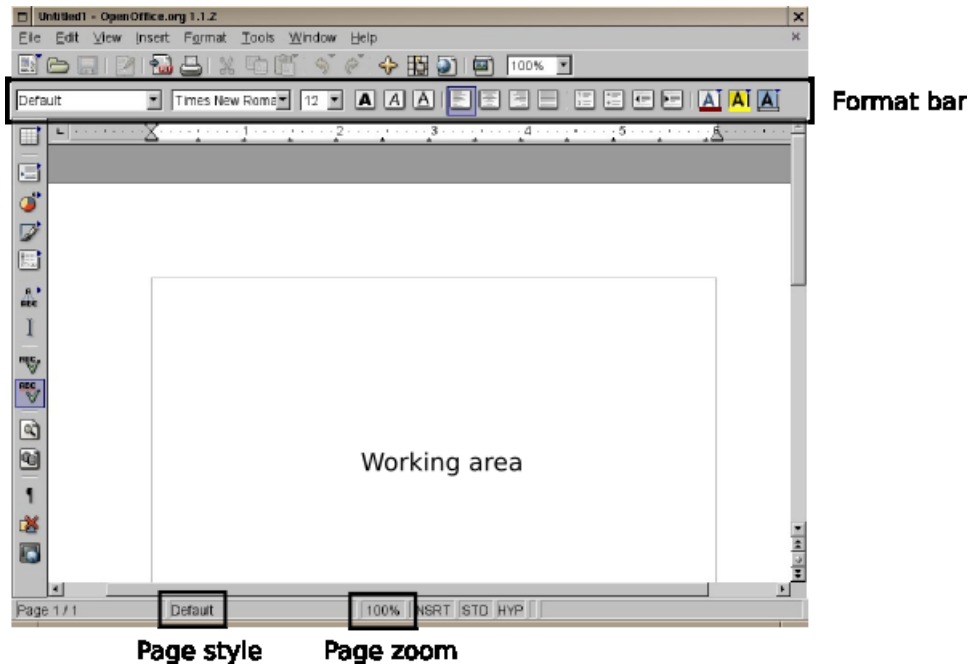
This chapter gives a short introduction to the OpenOffice office suite (officially named OpenOffice.org). OpenOffice is a fairly complete replacement for other office suites, like Microsoft Office. It offers the most commonly used functionality, an can open most major word processing, presentation and spreadsheet formats. OpenOffice features the following functionality:

- **Writer**: Word processor
- **Calc**: Spreadsheet
- **Impress**: Presentation software
- **Draw**: Vector-based drawing program

OpenOffice Writer

The Writer interface

Figure 7-1. The OpenOffice writer



As you can see in Figure 7-1, the OpenOffice Writer actually looks a lot like other word processors. So, migrating to OpenOffice should not be too difficult for most users. Let's start to have a look at some GUI elements:

- *Working area*

The working area is where most of the action will take place. As you can see, the working area shows the paper you are virtually typing on, and the thin grey lines mark the border of the area you can add text to. When you type the text will be inserted at the point the blinking cursor was at, and the cursor will be moved forward as you type text, indicating the current typing position. You can also add other elements, such as images, to the working area. You probably have noticed that the working area is truly WYSIWYG (What You See Is What You Get).

- *Format bar*

  The format bar is used to change the appearance of your text, or parts of your texts. There are multiple buttons and other elements that you can select, you can see the meaning of each element by placing the mouse pointer on a specific element. After a short time a tip will show, with a description of the element. If you have no text selected the appearance will change for the text that you will type after making the change. You can also change the appearance of text that has already been typed, by selecting the text, and making the necessary changes. You will see that the text will be altered directly. Be aware that you can alway undo changes by using the <Ctrl> + <z> keys, or selecting the *Edit -> Undo* menu item.

- *Page zoom*

  The page zoom indicator shows the zooming percentage of the document you are currently using. If you have trouble reading smaller fonts, you can zoom in to be able to read these fonts more comfortably. You can also zoom out to get a better overview of a page or multiple pages. Select the *View -> Zoom...* menu item to change the zooming percentage. There are also three special options, most notably *Optimal*, which will zoom to an ideal percentage for editing.

- *Page style*

  Each page can have a page style. Page styles define options for a page, such as page size and margins. Page styles can be changed through the *Format -> Page* menu item.

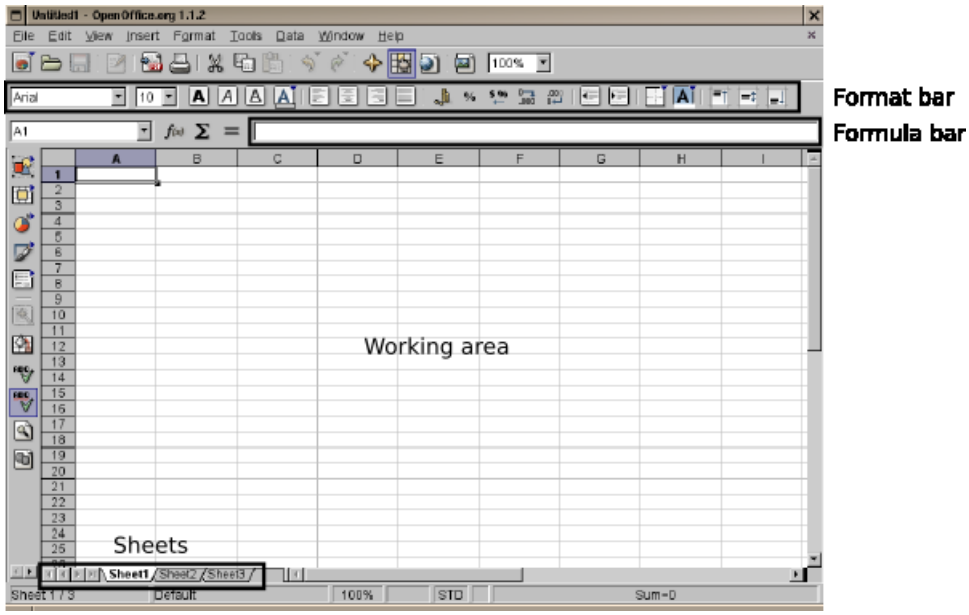
Creating PDF files

The PDF format is currently a very popular format for sharing documents for printing or viewing. Since version 1.1 has integrated support for creating PDF files. To make a PDF of the document you are currently editing, select the *File -> Export as PDF* menu option. This will open a file dialog, select the directory you would like to safe the PDF to, and enter the filename of the document that should be created (e.g. `document.pdf`). Finally, press the *Export...* button. Another dialog will appear that allows you to refine some options, and the range of pages you would like to save to the PDF file. Select *Export* to create the PDF file.

OpenOffice Calc

The Calc interface

Figure 7-2. OpenOffice Calc



OpenOffice Calc is the OpenOffice spreadsheet. Figure 7-2 shows what Calc looks like after launching it. These are the basic interface elements:

- *Working area*

  The working area is divided in columns (vertical), and rows (horizontal). Columns are referenced by letters of the alphabet (A, B, C, ...), rows are referenced by number (1, 2, 3, ...). You can reference to a particular cell by the column letter and row number, like A1 or C3. The contents of a cell can be changed by selecting the cell and starting to type a number or word.

- *Format bar*

  The formatting and appearance of information in cells can be changed. You can, for example change the appearance of a font or align the contents in a cell another way. You can get a description of each icon on the format bar by slowly moving the mouse over each icon.

- *Formula bar*

  You can see the contents or formula of a selected cell in the formula bar. This bar can also be used to alter the contents of a cell.

- *Sheets*

  You can have multiple sheets open at one time, the sheet tabs helps you to switch quickly between sheets. The arrow keys can be used to browse available sheets.

Using formulas

One of the main tasks for a spreadsheet is making calculations. Besides holding data a cell can contain a formula. When cells that are used in a formula are changed, the result of a formula will automatically be updated. A formula starts with the “=” character, followed by the formula. The formula can contain cell names as variables, and common mathematical operators. Besides that the spreadsheet also offers some functions, like SUM. These are some common math operators:

Table 7-1. Mathematical operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division

You can also add parentheses to structure formula. For example, in *= A1 + (A2 / A3)*, the expression between the parentheses, *A2 / A3* will be processed first, and then *A1 + result of (A2 / A3)*.

The result of the calculation will be displayed in the cell, but the formula bar will display the formula.

III. Linux Basics

Chapter 8. The Bash shell

Introduction

The shell is the traditional interface used by UNIX and GNU/Linux. In contrast to the X Window System it is an interface that works with commands. In the beginning this can be a bit awkward, but the shell is very powerful. Even in these days the shell is almost unavoidable ;).

The default shell on Libranet GNU/Linux is Bash. Bash means “Bourne-Again SHell”, which is a pun on the name of one of the traditional UNIX shells, the “Bourne Shell”. Libranet provides some other shells, but Bash is the main topic of this chapter.

Starting the shell

The procedure for starting the shell depends on whether you use a graphical or text-mode login. If you are logging on in text-mode the shell is immediately started after entering the (correct) password. If you are using a graphical login manager like gdm, log on as you would normally, and look in your window manager or desktop environment menu for an entry named "XTerm". XTerm is a terminal emulator, after the terminal emulator is started the shell comes up.

The shell might remind some people of MS-DOS. Be happy, it has nothing to do with DOS, the only similarity is that you can enter commands ;).

Shell basics

This chapter might be a difficult to read for the first time, because you might not know any shell commands. Many important commands are described in the next chapters, but those chapters are not really useful without any knowledge of the shell. So, it is not a bad idea to browse quickly through this chapter, and the next few chapters, to get an idea what this shell thing is all about. After that quick overview you should be able to understand this chapter.

Executing commands

The most important job for the shell is to execute your commands. Let’s look at a simple example. Most UNIX variants have a command named **whoami**, which shows as which user you are logged in. Try typing **whoami**, and press the <Enter> after that. The <Enter> tells the shell that it should execute the command that you have typed on the current line. The output looks like this:

```
daniel@tazzy:~$ whoami
daniel
daniel@tazzy:~$
```

As you can see the control is handed back to the shell after the command is finished.

Browsing through shell commands

It often happens that you have to execute commands that you executed earlier. Fortunately, you do not have to type them all over again. You can browse through the history of executed commands with the up and down arrows. Besides that it is also possible to search for a command. Press <Control> + <r> and start typing the command you want to execute. You will notice that bash will display the first match it can find. If this is not the match you were looking for you can continue typing the command (till it is unique and a match appears), or press <Control> + <r> once more to get the next match. When you have found the command you were looking for, you can execute it by pressing <Enter>.

Completion

Completion is one of the most useful functionalities of Unix-like shells. Suppose that you have a directory with two files named `websites` and `recipe`. And suppose you want to **cat** the file `websites` (**cat** shows the contents of a file), by specifying `websites` as a parameter to cat. Normally you would type “cat websites”, and execute the command. Try typing “cat w”, and hit the <Tab> key. Bash will automatically expand what you typed to “cat websites”.

But what happens if you have files that start with the same letter? Suppose that you have the `recipe1.txt` and `recipe2.txt` files. Type “cat r” and hit <Tab>, Bash will complete the filename as far as it can. It would leave you with “cat recipe”. Try hitting <Tab> again, and Bash will show you a list of filenames that start with “recipe”, in this case both recipe files. At this point you have to help Bash by typing the next character of the file you need. Suppose you want to **cat** `recipe2`, you can push the <2> key. After that there are no problems completing the filename, and hitting <Tab> completes the command to “cat recipe2.txt”.

It is worth noting that completion also works with commands. Most GNU/Linux commands are quite short, so it will not be of much use most of the time.

It is a good idea to practice a bit with completion, it can save alot of keystrokes if you can handle completion well. You can make some empty files to practive with using the **touch** command. For example, to make a file named `recipe3.txt`, execute **touch recipe3.txt**.

Wildcards

Most shells, including Bash, support wildcards. Wildcards are special characters that can be used to do pattern matching. The table listed below displays some commonly used wildcards. We are going to look at several examples to give a general idea how wildcards work.

Table 8-1. Bash wildcards

Wildcard	Matches
*	A string of characters
?	A single character
[]	A character in an array of characters

Matching a string of characters

As you can see in the table above the “*” character matches a string of characters. For example, **.html* matches everything ending with *.html*, *d*.html* matches everything starting with a *d* and ending with *.html*.

Suppose that you would like to list all files in the current directory with the *.html* extension, the following command will do the job:

```
$ ls *.html
book.html        installation.html     pkgmgmt.html  usermgmt.html
filesystem.html  internet.html         printer.html  xfree86.html
gfdl.html        introduction.html     proc.html
help.html        libranet-basics.html  shell.html
```

Likewise we could remove all files starting with an *in*:

```
$ rm in*
```

Matching single characters

The "?" wildcard works as the "*" wildcard, but matches single characters. Suppose that we have three files, `file1.txt`, `file2.txt` and `file3.txt`. The string *file?.txt* matches all three of these files, but it does not match `file10.txt` ("10" are two characters).

Matching characters from a set

The "[]" wildcard matches every character between the brackets. Suppose we have the files from the previous example, `file1.txt`, `file2.txt` and `file3.txt`. The string *file[23].txt* matches `file2.txt` and `file3.txt`, but not `file1.txt`.

Redirections and pipes

One of the main features of Unix-like shells are redirections and pipes. Before we start to look at both techniques we have to look how most Unix-like commands work. When a command is not getting data from a file, it will open a special pseudo-file named *stdin*, and wait for data to appear on it. The same principle can be applied for command output, when there is no explicit reason for saving output to a file, the pseudo-file *stdout* will be opened for output of data. This principle is shown schematically in Figure 8-1

Figure 8-1. Standard input and output

You can see *stdin* and *stdout* in action with the **cat** command. If cat is started without any parameters it will just wait for input on *stdin* and output the same data on *stdout*. If no redirection is used keyboard input will be used for *stdin*, and *stdout* output will be printed to the terminal:

```
$ cat
Hello world!
Hello world!
```

As you can see cat will print data to *stdout* after inputting data to *stdin* using the keyboard.

Redirection

The shell allows you to take use of *stdin* and *stdout* using the "<" and ">". Data is redirected in which way the sharp bracket points. In the following example we will redirect the md5 summaries calculated for a set of files to a file named `md5sums`:

```
$ md5sum * > md5sums
$ cat md5sums
6be249ef5cacb10014740f61793734a8  test1
220d2cc4d5d5fed2aa52f0f48da38ebe  test2
631172a1cfca3c7cf9e8d0a16e6e8cfe  test3
```

As we can see in the **cat** output the output of the **md5sum *** output was redirected to the `md5sums` file. We can also use redirection to provide input to a command:

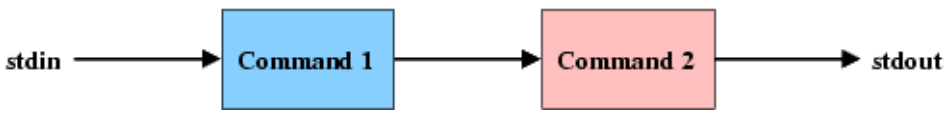
```
$ md5sum < test1
6be249ef5cacb10014740f61793734a8  -
```

This feeds the contents of the `test1` to **md5sum**.

pipes

You can also connect the input and output of commands using so-called *pipes*. A pipe between commands can be made with the "|" character. Two or more combined commands are called a *pipeline*. Figure 8-2 shows a schematic overview of a pipeline consisting of two commands.

Figure 8-2. A pipeline



The "syntax" of a pipe is: **command1 | command2 ... | commandn**. If you know how the most basic Unix-like commands work you can now let these commands work together. Let's look at a quick example:

```
$ cat /usr/share/dict/american-english | grep "aba" | wc -l
123
```

The first command, **cat**, reads the dictionary file `/usr/share/dict/american-english`. The output of the **cat** command is piped to **grep**, which prints out all files containing the phrase "aba". In turn, the output of "grep" is piped to **wc -l**, which counts the number of lines it receives from *stdin*. Finally, when the stream is finished **wc** prints the number of lines it counted. So, combined three commands to count the number of words containing the phrase "aba" in this particular dictionary.

There are hundreds of small utilities that handle specific tasks. As you can imagine, together these commands provide a very powerful toolbox by making combinations using pipes.

Chapter 9. Files and directories

Introduction

Unix-like operating systems use a hierarchical filesystem to store files and directories. Directories can contain files and other directories, the top directory (`/`) is named the root directory (not to be confused with the `/root` directory). Most filesystems also support file links (which provide alternative names for a file) and soft links. Other filesystems can be "connected" to an arbitrary directory. This process is named "mounting", and the directory in which the filesystem is mounted is named the "mount point".

This chapter covers the basic navigation of the filesystem, commands which are used to remove and create directories, filesystem permissions, links and mounting.

The basics

pwd

pwd(1) is a simple utility which shows the directory you are currently working in. The **pwd** does not require any parameters. This is an example output of **pwd**:

```
$ pwd
/home/danieldk
```

ls

ls is similar to the **dir** command in DOS and Windows. **ls** can be used to display files and directories located in specific directories. Running the **ls** command without any parameters shows the contents of the current directory:

```
$ ls
COPYING  CVS  Makefile  README  html  images  pdf  src  tex
```

Naturally it is also possible to show the contents of other directories. You can do this by specifying the path as a parameter tot the **ls** command:

```
$ ls /
bin   cdrom   dev  floppy  initrd  lost+found  opt   root  sys  usr  windows
boot  cdrom1  etc  home    lib     mnt         proc  sbin  tmp  var
```

A disadvantage of the default output is that it provides little information about files and directories. For example, it is not possible to see whether some entry is a file or directory, what size a file is, or who the owner of the file is. The **ls** has the "-l" parameter to show more information:

```
$ ls -l
total 52
-rw-r--r--    1 daniel   daniel      20398 Jul 16 14:28 COPYING
drwxr-xr-x    2 daniel   daniel       4096 Jul 16 14:28 CVS
-rw-r--r--    1 daniel   daniel        768 Jul 16 14:28 Makefile
```

```
-rw-r--r--    1 daniel   daniel        408 Jul 16 14:28 README
drwxr-xr-x    3 daniel   daniel       4096 Jul 16 14:28 html
drwxr-xr-x    3 daniel   daniel       4096 Jul 16 14:28 images
drwxr-xr-x    3 daniel   daniel       4096 Jul 16 14:28 pdf
drwxr-xr-x    3 daniel   daniel       4096 Jul 20 00:11 src
drwxr-xr-x    3 daniel   daniel       4096 Jul 16 14:28 tex
```

cd

Another important command is the **cd** command. It can be used to change the current working directory:

```
$ cd /home/danieldk/
```

With the **pwd** command you can see it worked:

```
$ pwd
/home/danieldk
```

mkdir

As you might have guessed, the **mkdir**(1) command can be used to create directories. For example:

```
$ pwd
/home/danieldk
$ mkdir test
$ cd test
$ pwd
/home/danieldk/test
```

It might happen that you want to create a directory in a parent directory which does not exist yet. For example, if you want to create the `test2/hello/` directory, but the `test2` directory does not yet exist. In this case you can make both directories with only one **mkdir** command:

```
$ mkdir -p test2/hello
```

cp

Files can be copied with the **cp**(1) command, the basic syntax is **cp source destination**. For example, suppose that we have a file named `memo` which we would like to copy to the `writings` directory. You can do this with the following command:

```
$ cp memo writings/
```

It is also possible to copy a file in the same directory. For example, if we would like to make a new memo based on `memo`, named `memo2`, we could execute the following command:

```
$ cp memo memo2
```

It is also possible to copy directories recursively, this can be done by adding the *-r*. The following command copies the `memos` directory, and all subdirectories, and (sub)files to the `writings` directory:

```
$ cp -r memos writings/
```

mv

The **mv**(1) command is comparable to **cp**, but it is used to move files. Suppose that we have the same situation as in the first **cp** example, but you would rather like to move `memo` to the `writings` directory. The following command would do that:

```
$ mv memo writings/
```

It is also possible to move directories. But, **mv** always works recursively. For example, the following command will move the `memos` directory to the `writings` directory:

```
$ mv memos writings/
```

rm

The **rm**(1) command is used to remove files and directories. Let's look at a simple example:

```
$ rm hello.c
```

This commands removes the file `hello.c`. Sometimes the **rm** asks for a confirmation. You can ignore this with the '-f' parameter:

```
$ rm -f *
```

This command removes all files in the current directory without asking for confirmation. It is also possible to delete directories or even whole directory trees. **rm** provides the '-r' parameter to do this. Suppose we want to delete the `ogle` directory and all subdirectories and files in that directory, this can be done in the following way:

```
$ rm -r -f ogle/
```

Many commands allow you to combine parameters. The following example is equivalent to the last one:

```
$ rm -rf ogle/
```

Permissions

A short introduction

Every file in GNU/Linux has permissions. As you might have noticed, file permissions can be shown with the **ls -l** command:

```
$ ls -l logo.jpg
-rw-r--r--    1 danieldk users        9253 Dec 23 19:12 logo.jpg
```

The permissions are shown in the first column. Permissions that can be set are read(r), write(w) and execute(x). These permissions can we set for three classes: owner(u), group(g) and others(o). The

permissions are visible in the second to ninth character in the first column. These nine characters are divided in three groups. The first three characters represent the permissions for the owner, the next three characters represent the permissions for the group, finally the last three characters represent the permissions for other users. Thus, the example file shown above can be written to by the owner and can be read by all three classes of users (owner, group and others).

Each GNU/Linux system has many distinct users (a list of users can be found in `/etc/passwd`), and a user can be a member of certain groups. This kind of user management provides makes it possible to manage detailed permissions for each file. In the example shown above *danieldk* is the owner of the file and group permissions apply to the group *users*. In this example groups rights do not differ from the rights of other users.

chown

The **chown**(1) command is used to set the file owner and to which group group permissions should apply to. Suppose we want to make *danieldk* the owner of the file `logo2.jpg`, this can be done with the **chown**:

```
$ chown danieldk logo2.jpg
```

Using the **ls** we can see that the owner is now *danieldk*:

```
$ ls -l logo2.jpg
-rw-r--r--    1 root     root        9253 Dec 29 11:35 logo2.jpg
$ chown danieldk logo2.jpg
$ ls -l logo2.jpg
-rw-r--r--    1 danieldk root        9253 Dec 29 11:35 logo2.jpg
```

But group permissions still apply for the *root* group. This group can be changed by adding a dot after the owner, followed by the name of the group (in this example the group is nedslackers):

```
$ chown danieldk.nedslackers logo2.jpg
$ ls -l logo2.jpg
-rw-r--r--    1 danieldk nedslackers        9253 Dec 29 11:35 logo2.jpg
```

It is also possible to change ownership recursively, this can be done with the recursive (-R) parameter:

```
$ chown -R danieldk.users oggs/
```

chmod

File permissions can be manipulated using the **chmod**(1) command. The most basic syntax of chmod is **chmod [u,g,o][+/-][r,w,x] filename**. The first parameter consists of the following elements: 1. which classes this manipulation permission applies to, 2. if the permissions should be added (+) or removed (-), and 3. which permissions should be manipulated. Suppose we want to make the file `memo` writable for the owner of the file and the groups for which the group permissions apply. This can be done with the following command:

```
$ chmod ug+w memo
```

As you can see below the `memo` is now writable for the file owner and group:

```
$ ls -l notes
```

```
-r--r--r--    1 daniel   users          12 Mar  9 16:28 memo
bash-2.05b$ chmod ug+w memo
bash-2.05b$ ls -l notes
-rw-rw-r--    1 daniel   users          12 Mar  9 16:28 momo
```

Just like the **chown** command it is also possible to do recursive (-R) operations. In the following example the `secret/`, including subdirectories and files in this directory, is made unreadable for the group set for this directory and other users:

```
$ chmod -R go-r secret/
```

Archives

Introduction

Sooner or later a GNU/Linux user will encounter tar archives, tar is the standard format for archiving files on GNU/Linux. It is often used in conjunction with **gzip** or **bzip2**. Both commands can compress files and archives. Table 9-1 lists frequently used archive extensions, and what they mean.

Table 9-1. Archive file extensions

Extension	Meaning
.tar	An uncompressed tar archive
.tar.gz	A tar archive compressed with gzip
.tgz	A tar archive compressed with gzip
.tar.bz2	A tar archive compressed with bzip2
.tbz	A tar archive compressed with bzip2

The difference between **bzip2** and **gzip** is that **bzip2** can find repeating information in larger blocks, resulting in better compression. But **bzip2** is also a lot slower, because it does more data analysis.

Extracting archives

Since many software and data in the GNU/Linux world is archived with **tar** it is important to get used to extracting tar archives. The first thing you will often want to do when you receive a tar archive is to list its contents. This can be achieved by using the “t” parameter. However, if we just execute **tar** with this parameter and the name of the archive it will just sit and wait until you enter something to the standard input:

```
$ tar t test.tar
```

This happens because **tar** reads data from its standard input. If you forgot how redirection works, it is a good idea to reread the Section called *Redirections and pipes* in Chapter 8. Let’s see what happens if we redirect our tar archive to tar:

```
$ tar t < test.tar
test/
test/test2
```

```
test/test1
```

That looks more like the output you probably expected. This archive seems to contain a directory `test`, which contains the files `test2` and `test2`. It is also possible to specify the archive file name as an parameter to **tar**, by using the “f” parameter:

```
$ tar tf test.tar
test/
test/test2
test/test1
```

This looks like an archive that contains useful files ;). We can now go ahead, and extract this archive by using the “x” parameter:

```
$ tar xf test.tar
```

We can now verify that tar really extracted the archive by listing the contents of the directory with **ls**:

```
$ ls test/
test1  test2
```

Extracting or listing files from a gzipped or bzipped archive is not much more difficult. This can be done by adding a “z” or “b” for respectively archives compressed with gzip or bzip. For example, we can list the contents of a gzipped archive with:

```
$ tar ztf archive2.tar.gz
```

And a bzipped archive can be extracted with:

```
$ tar bxf archive3.tar.bz2
```

Creating archives

You can create archives with the “c” parameter. Suppose that we have the directory `test` shown in the previous example. We can make an archive with the `test` directory and the files in this directory with:

```
$ tar cf important-files.tar test
```

This will create the `important-files.tar` archive (which is specified with the “f” parameter). We can now verify the archive:

```
$ tar tf important-files.tar
test/
test/test2
test/test1
```

Creating a gzipped or bzipped archive goes along the same lines as extracting compressed archives: add a “g” for gzipping an archive, or “b” for bzipping an archive. Suppose that we wanted to create a **gzip** compressed version of the archive created above. We can do this with:

```
tar zcf important-files.tar.gz test
```

Extended attributes

Introduction

Extended attributes (EAs) are relatively new on GNU/Linux. Extended attributes are a special kind of values that are associated with a file or directory. EAs provide the means to add extra attributes besides the common attributes (modification time, traditional file permissions, etc.). For example, one could add the attribute "Photographer" to a collection of JPEG files. Extended attributes are not physically stored in the file, but as meta-data in the filesystem.

Extended attributes are only supported by 2.6.x and newer 2.4.x kernels. Besides that they are not supported on all filesystems, the commonly used Ext2, Ext3 and XFS filesystems do support extended attributes.

Installing the necessary utilities

The extended attribute software is available from the the Debian archives as the *attr* package. You can install it through your favorite package manager, or APT:

```
# apt-get install attr
```

Showing extended attributes

Extended attributes can be queried using the **getfattr** command. Just using getfattr with a file as a parameter will show the attributes that are known for that particular file, without the values set for the attributes. For example:

```
$ getfattr note.txt
# file: note.txt
user.Author
```

This file has one extended attribute, *user.Author*. An attribute has the following form: *namespace.attribute*. There are four defined namespaces: *security*, *system*, *trusted*, and *user*. The role of these namespaces are described in the *attr*(5) manual page. The *user* namespace is of particular interest to us, because this namespace is used to assign arbitrary attributes to files.

The values associated with an attribute can be shown using the “-d” (dump) parameter. For example:

```
$ getfattr -d note.txt
# file: note.txt
user.Author="Daniel"
```

In this example the attribute *user.Author* has the value *Daniel* for the file `note.txt`.

Setting extended attributes

Attributes are set with the **setfattr** command. An attribute can be added to a file using the “-n” (name) parameter, be sure to specify the namespace and the attribute name, for example:

```
$ setfattr -n user.Author note2.txt
```

The value of the attribute can be set using the “-v” (value) parameter:

```
$ setfattr -n user.Author -v Mike note2.txt
```

But it is also possible to add an attribute and setting its value in one run, by specifying both the “-n” and “-v” parameters. For example, the following command adds the MD5 sum of the file as an extended attribute:

```
$ setfattr -n user.MD5 -v ‘md5sum note2.txt‘ note2.txt
$ getfattr -d note2.txt
# file: note2.txt
user.Author="Mike"
user.MD5="78be7a3148027ae7a897aad95e7d9c58"
```

Mounting filesystems

Introduction

Like most Unices Linux uses a technique named “mounting” to access filesystems. Mounting means that a filesystem is connected to a directory in the root filesystem. One could for example mount a CD-ROM drive to the `/cdrom` directory. Linux supports many kinds of filesystems, like Ext2, Ext3, ReiserFS, JFS, XFS, ISO9660 (used for CD-ROMs), UDF (used on some DVDs) and DOS/Windows filesystems, like FAT, FAT32 and NTFS. These filesystems can reside on many kinds of media, for example hard drives, CD-ROMs and Flash drives. This section explains how filesystems can be mounted and unmounted.

mount

The **mount**(8) is used to mount filesystems. The basic syntax is: **mount /dev/devname /mountpoint**. The device name can be any block device, like hard disks or CD-ROM drives. The mount point can be an arbitrary point in the root filesystem. Let’s look at an example:

```
# mount /dev/cdrom /cdrom
```

This mounts the `/dev/cdrom` on the `/cdrom` mountpoint. The `/dev/cdrom` device name is normally a link to the real CD-ROM device name (for example, `/dev/hdc`). As you can see, the concept is actually very simple, it just takes some time to learn the device names ;). Sometimes it is necessary to specify which kind of filesystem you are trying to mount. The filesystem type can be specified by adding the “-t” parameter:

```
# mount -t vfat /dev/sda1 /flash
```

This mounts the vfat filesystem on `/dev/sda1` to `/flash`.

umount

The **umount**(1) command is used to unmount filesystems. **umount** accepts two kinds of parameters, mount points or devices. For example:

```
# umount /cdrom
# umount /dev/sda1
```

The first command unmounts the filesystem that was mounted on `/cdrom`, the second commands unmounts the filesystem on `/dev/sda1`.

The fstab file

The GNU/Linux system has a special file, `/etc/fstab`, that specifies which filesystems should be mounted during the system boot. Let's look at an example:

```
/dev/hda5  /              ext3     defaults,errors=remount-ro   0      1
/dev/hda8  /home          ext3     defaults,errors=remount-ro   0      2
/dev/hda7  /tmp           ext3     defaults,errors=remount-ro   0      2
/dev/hda9  /usr           ext3     defaults,errors=remount-ro   0      2
/dev/hda6  /var           ext3     defaults,errors=remount-ro   0      2
/dev/hda1  /windows       vfat     defaults,gid=windows,umask=002 0      0
/dev/hda10 none           swap     sw                           0      0
proc       /proc          proc     defaults                     0      0
/dev/fd0   /floppy        vfat,auto defaults,user,noauto        0      0
usbfs      /proc/bus/usb  usbfs    rw,devmode=0660,devgid=432   0      0
/dev/hdc   /cdrom  udf,iso9660  defaults,user,noauto,ro          0      0
/dev/hdd   /cdrom1  udf,iso9660  defaults,user,noauto,ro         0      0
```

As you can see each entry in the `fstab` file has five entries: fs_spec, fs_file, fs_vfstype, fs_mntops, fs_freq, and fs_passno. We are now going to look at each entry.

fs_spec

The fs_spec option specifies the block device, or remote filesystem that should be mounted. As you can see in the example several `/dev/hda` partitions are specified, as well as the CD-ROM drive and floppy drive. When NFS volumes are mounted an IP address and directory can be specified, for example: `192.168.1.10:/exports/data`.

fs_file

fs_file specifies the mount point. This can be an arbitrary directory in the filesystem.

fs_vfstype

This option specifies what kind of filesystem the entry represents. For example this can be: ext2, ext3, reiserfs, xfs, nfs, vfat, or ntfs.

fs_mntops

The fs_mntops option specifies which parameters should be used for mounting the filesystem. The **mount**(8) manual page has an extensive description of the available options. These are the most interesting options:

- *noauto*: filesystems that are listed in `/etc/fstab` are normally mounted automatically. When the "noauto" option is specified, the filesystem will not be mounted during the system boot, but only

after issuing a **mount** command. When mounting such filesystem, only the mount point or device name has to be specified, for example: **mount /mnt/cdrom**

- *user*: adding the “user” option will allow normal users to mount the filesystem (normally only the superuser is allowed to mount filesystems).
- *owner*: the *owner* option will allow the owner of the specified device to mount the specified device. You can see the owner of a device using **ls**, e.g. **ls -l /dev/cdrom**.
- *noexec*: with this option enabled users can not run files from the mounted filesystem. This can be used to provide more security.
- *nosuid*: this option is comparable to the *noexec* option. With *nosuid* enabled SUID bits on files on the filesystem will not be allowed. SUID is used for certain binaries to provide a normal user to do something privileged. This is certainly a security threat, so this option should really be used for removable media, etc. A normal user mount will force the nosuid option, but a mount by the superuser will not!
- *unhide*: this option is only relevant for normal CD-ROMs with the ISO9660 filesystem. If *unhide* is specified hidden files will also be visible.

fs_freq

If the *fs_freq* is set to 1 or higher, it specifies after how many days a filesystem dump (backup) has to be made. This option is only used when **dump**(8) is set up correctly to handle this.

fs_passno

This field us used by **fsck**(8) to determine the order in which filesystems are checked during the system boot.

xvmount

Libranet has a nice tool named xvmount, which allows a user to mount and unmount filesystems using a X tool. Xvmount can be set up by launching AdminMenu and clicking the “xvmount” icon in the “Disk/CD/Floppy” menu. After setting up xvmount, launching the **xvmount** command will display a list of devices that can be mounted and unmounted.

Chapter 10. Process management

Introduction

Unix-like operating systems work with processes. A process is an unit the operating system schedules for CPU time and the memory manager manages memory for. Basically a process consists of program code (named text), data (used by a program) and a stack. The stack is used by the program to store variables. Programs are at least one process. A program/process can ask the system to create a new copy of itself, which is called a fork. For example, a web server could fork itself to let the new process handle a request.

A process can be parted in threads. The difference between forking a process and creating a thread is that different threads share the address space of the process. A forked process is a separate process with its own address space. Forking is more expensive in terms of memory requirement and CPU time.

A user can control a process by sending signals to the process. For example, the *SIGTERM* signal is used to terminate a process, and the *SIGHUP* signal to restart a process.

Process basics

This section describes some basic commands that are used for process management.

ps

The **ps**(1) command is used to report which processes are currently active. By running **ps** without any parameters you can see which processes are active in the current user session. Let's look at an example:

```
$ ps
  PID TTY          TIME CMD
 1191 pts/2    00:00:00 bash
 1216 pts/2    00:00:00 ps
```

In this example the **bash** and **ps** commands are running. As you can see each process has a process ID (PID). You will need the process number if you want to send a signal to a process, for example a kill signal. The **ps** has many parameters to modify the output. For example, the `x` shows all processes without a controlling tty:

```
$ ps x
  PID TTY      STAT   TIME COMMAND
 1044 tty1     S      0:00 -bash
 1089 tty1     S      0:00 /bin/sh /usr/X11R6/bin/startx
 1100 tty1     S      0:00 xinit /home/daniel/.xinitrc --
 1108 tty1     S      0:00 /usr/bin/wmaker
 1113 tty1     S      0:00 sylpheed
 1114 tty1     S      0:00 /bin/sh /opt/firefox/run-mozilla.sh /opt/firefox/fire
 1120 tty1     S      0:52 /opt/firefox/firefox-bin
 1125 tty1     S      0:00 /usr/libexec/gconfd-2 20
 1146 tty1     S      0:00 xchat
 1161 tty1     S      0:00 xterm -sb
```

```
 1163 pts/0    S      0:00 bash
 1170 pts/0    S      0:00 vi proc.xml
 1189 tty1     S      0:00 xterm -sb
 1191 pts/2    S      0:00 bash
 1275 pts/2    R      0:00 ps x
```

Have a look at the **ps**(1) manual page for a summary of available parameters.

kill

The **kill**(1) sends a signal to a process. If no signal is specified the *TERM* signal is send, which asks a process to exit gracefully. Let's have a look at the normal mode of execution:

```
$ ps ax | grep mc
 1045 tty4     S      0:00 /usr/bin/mc -P /tmp/mc-daniel/mc.pwd.756
$ kill 1045
$ ps ax | grep mc
$
```

As you can see the **ps** is used to look for the **mc** process. There is one occurrence of **mc** running with PID 1045. This process is killed, and the second **ps** command shows that the process is indeed terminated.

As we said earlier the **kill** command can also be used to send other signals. The **kill -l** displays a list of signals that can be sent:

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS
```

The *SIGKILL* signal is often used to kill processes that refuse to terminate with the default *SIGTERM* signal. The signal can be specified by using the number as a parameter, for example, the following command would send a *SIGKILL* signal to PID 1045:

```
$ kill -9 1045
```

It is also possible to specify the signal without the "SIG" letters as a parameter. In the following example the *SIGHUP* signal is sent to the **inetd** to restart it:

```
# ps ax | grep inetd
  727 ?        S      0:00 /usr/sbin/inetd
# kill -HUP 727
```

Advanced process management

Background processes

Normally a process takes over the screen and keyboard after it is started. It is also possible to start processes as a background process, this means that the shell starts the process, but keeps control over the terminal. In most shells a process can be started as a background process by placing an ampersand (&) after the command. For example:

```
$ rm -rf ~/bunch/of/files &
```

A process that runs in the background can be brought to the foreground using the **fg %<job ID>** command. You can see which jobs are running, with their job numbers, using the **jobs** command. For example:

```
$ sleep 1000 &
[1] 947
$ jobs
[1]+  Running                 sleep 1000 &
$ fg %1
sleep 1000
```

The first command, **sleep 1000 &**, starts sleep in the background. **sleep** is a command does nothing but waiting the number of seconds that are specified as a parameter. The output of the **jobs** command shows that **sleep** is indeed running, with Job ID *1*. Finally we move **sleep** to the foreground. As you can see, the shell will print which command is moved to the foreground.

Stopping processes

A process that is running can be stopped by pressing the <Control> and <z> keys simultaneously. Stopped processes can be moved to the foreground with the **fg** command. Running **fg** without any parameters moves the last process that was stopped to the foreground. Other processes can be moved to the foreground by specifying the job ID as a parameter to **fg**.

A stopped process can also be told to continue as a background process, by executing **bg <job ID>**. Executing **fg** without any parameter will move the last stopped process to the background.

Altering priorities

The Linux kernel allows a user to change the priority of a program. For example, suppose that you want to run a process that requires a lot of CPU time, but you do not want to hinder other users. In this case you can start the process with a low priority, the process will only get CPU time when there are not many other processes demanding CPU time. Or you can give processes that are important a higher priority.

GNU/Linux provides two commands to alter the priority of a process. The **nice**(1) command can be used to specify the priority when you are launching a process. With the **renice**(1) command you can alter the priority of a process that is already running. The priority is a numerical value from -20 (highest priority) to 19 (lowest priority). Let’s start with an example of **nice** in action:

```
$ nice -n 19 ./setiathome
```

As you can see the *-n* parameter is used to specify the priority value. In this case the **./setiathome** will have a very low priority. Be aware that only the superuser can use negative priority values. Thus, a normal user cannot give a process a higher priority, as illustrated by this example:

```
$ nice -n -1 nice
nice: cannot set priority: Permission denied
```

But it will work as the *root* user:

```
$ su -c "nice -n -1 nice"
Password:
-1
```

The **renice** command has a somewhat different syntax. The easiest way to use it is to specify the new priority and the process ID as parameters to the **renice** command, as is shown in the following example:

```
$ renice +5 5811
5811: old priority 0, new priority 5
```

As with **nice** a non-*root* user cannot set negative priority values:

```
$ renice -5 5811
renice: 5811: setpriority: Permission denied
```

> **Note:** A normal user can not increase the priority of the process beyond the default priority of 0. Such facilities could be misused by careless users. After all, the command **nice** is derived from being nice to the other users on the system ;^).

Chapter 11. User management

Introduction

GNU/Linux is a multi-user operating system. This means that multiple users can use the system, and they can use the system simultaneously. The GNU/Linux concepts for user management are quite simple. First of all, there are several user accounts on each system. Even on a single user system there are multiple user accounts, because GNU/Linux uses unique accounts for some tasks. Users can be members of groups. Groups are used for more fine grained permissions, for example, you could make a file readable by a certain group. There are a few reserved users and groups on each system. The most important of these is the *root* account. The *root* user is the system administrator. It is a good idea to avoid logging in as root, because this greatly enlarges security risks. You can just log in as a normal user, and perform system administration tasks using the **su** and **sudo** commands. Adminmenu will automatically ask you to enter the *root* password when you would like to administrate the system.

The available user accounts are specified in the `/etc/passwd`. You can have a look at this file to get an idea of which user account are mandatory. As you will probably notice, there are no passwords in this file. Passwords are kept in the separate `/etc/shadow` file, as an encrypted string. Information about groups is stored in `/etc/group`. It is generally speaking not a good idea to edit these files directly. There are some excellent tools that can help you with user and group administration. This chapter will describe some of these tools.

Administrating users via adminmenu

Adminmenu can be used to add and remove users, besides you can change a user password with adminmenu. These functions are located under the “users” section.

The “Add a user to your system” will start by asking a confirmation. After replying “y” you can enter the name of the user. Then you will be asked to enter the password for the account twice. Finally adminmenu will ask you whether you would like to add the user to the *windows* group or not. Adding a user to the *windows* groups will give the user access to Windows partitions.

The “Change a user’s password” option is really simple. Just enter the name of the user whose password you would like to change. If the user was found on the system, you will be asked to type the new password twice.

The “Remove a user from your system” option will start with asking for a confirmation. If you answered “y” the name of the user to be removed will be asked. Finally you can choose to leave the home directory of the user around(“n”), or to remove it (“y”).

Command-line tools

useradd

The **useradd** is used to add user accounts to the system. Running **useradd** with a user name as parameter will create the user on the system. For example:

```
# useradd bob
```

Creates the user account *bob*. Please be aware that this does not create a home directory for the user. Add the *-m* parameter to create a home directory. For example:

```
# useradd -m bob
```

This would add the user *bob* to the system, and create the `/home/bob` home directory for this user. Normally the user is made a member of the *users* group. Suppose that we would like to make *crew* the primary group for the user *bob*. This can be done using the *-g* parameter. For example:

```
# useradd -g crew -m bob
```

It is also possible to add this user to secondary groups during the creation of the account with the *-G*. Group names can be separated with a comma. The following command would create the user *bob*, which is a member of the *crew* group, and the *www-admins* and *ftp-admins* secondary groups:

```
# useradd -g crew -G www-admins,ftp-admins -m bob
```

By default the **useradd** only adds users, it does not set a password for the added user. Passwords can be set using the **passwd** command.

passwd

As you probably guessed the **passwd** command is used to set a password for a user. Running this command as a user without a parameter will change the password for this user. The password command will ask for the old password,once and twice for the new password:

```
$ passwd
Changing password for bob
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

The *root* user can set passwords for users by specifying the user name as a parameter. The **passwd** command will only ask for the new password. For example:

```
# passwd bob
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

userdel

Sometimes it is necessary to remove a user account from the system. GNU/Linux offers the **userdel** tool to do this. Just specify the username as a parameter to remove that user from the system. For example, the following command will remove the user account *bob* from the system:

```
# userdel bob
```

This will only remove the user account, not the user's home directory and mail spool. Just add the *-r* parameter to delete the user's home directory and mail spool too. For example:

```
# userdel -r bob
```

Avoiding root usage

It is a good idea to avoid logging in as *root*. There are many reasons for not doing this. Accidentally typing a wrong command could cause bad things to happen, and malicious programs can make a lot of damage when you are logged in as *root*. Still, there are many situations in which you need to have root access. For example, to do system administration, or to install new software. Fortunately the **su** can give you temporal root privileges.

Using **su** is very simple. Just executing **su** will ask you for the root password, and will start a shell with root privileges after the password is correctly entered:

```
$ whoami
bob
$ su
Password:
# whoami
root
# exit
exit
$ whoami
bob
```

In this example the user *bob* is logged on, the **whoami** output reflects this. The user executes su and enters the *root* password. **su** launches a shell with root privileges, this is confirmed by the **whoami** output. After exiting the *root* shell, control is returned to the original running shell running with the privileges of the user *bob*.

It is also possible to execute just one command as the *root* user with the *-c* parameter. The following example will run **update-grub**:

```
$ su -c update-grub
```

If you want to give parameters to the command you would like to run, use quotes (e.g. **su -c "ls -l /"**). Without quotes **su** cannot determine whether the parameters should be used by the specified command, or by **su** itself.

IV. System administration

Chapter 12. Package management

Introduction

The Debian distribution has a package manager since the early day of the distribution. This made software easily distributable and manageable. However, one part was still missing, a tool that could fetch packages from different locations (e.g. from CD-ROM and FTP), and that could automatically solve and get dependencies. APT, the Advanced Packaging Tool, was written to fill this gap.

Libranet has the advantage that it is based on Debian, so it inherited the APT tool. APT can be used on Libranet to get software from the Debian archives, or Libranet's own "safe update archive" (see the Section called *The Libranet update-safe archive*), a special archive with the goal of preventing breaking the system during updates or installation of new programs. This chapter gives a short introduction to the synaptic and aptitude package managers, and gives a short explanation about the configuration and usage of APT. For an extensive introduction to APT, read the APT HOWTO (http://www.debian.org/doc/manuals/apt-howto/index.en.html).

The Libranet update-safe archive

Introduction

On January 23, 2004 Libra Computer Systems Ltd. announced the "Libranet update-safe archive". This special archive was introduced to provide a safe upgrade method which is well-tested. Usage of the safe archive is free. Please note that somebody can still use the plain Debian archives if he or she wants. The Libranet archive just provides more safety, because it is a carefully selected collection of packages from all three Debian branches (plus some custom Libranet packages).

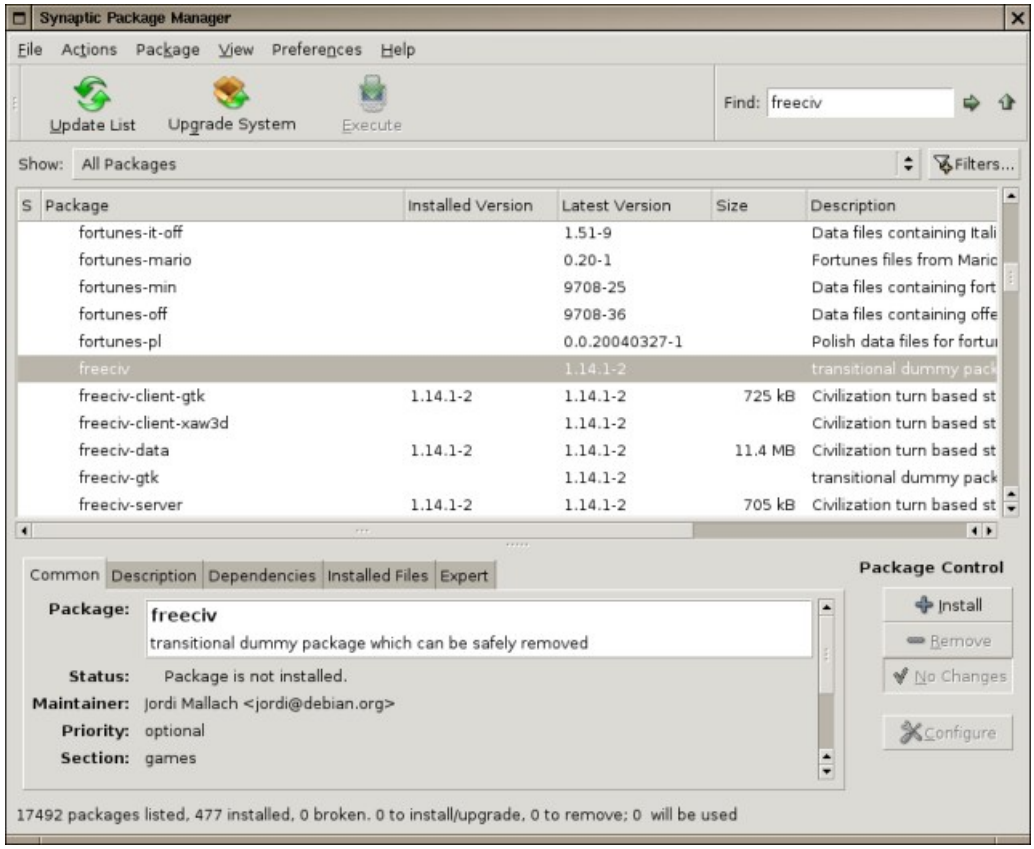
Using the archive

The update-safe archive was introduced after Libranet 2.8 and 2.8.1, so it is not used by default. Fortunately it is very easy to set up. Click "Update Adminmenu from the Internet" on the "Packages" tab in Libranet. This will update adminmenu. After updating adminmenu you will be asked whether you would like to use the update-safe archive or not. Choose "yes", after that the update procedure installs a new `sources.list` file which uses the archive. After re-launching adminmenu the "Packages" menu will show a new item, "Upgrade system from the Libranet Archive", which can be used to upgrade your system using the update-safe archive.

Synaptic

Synaptic is a graphical front-end for APT. It is included in Libranet, which can be launched using the Synaptic button which can be found on the "Packages" tab in Xadminmenu. It is very easy to use, and allows you to view detailed information about packages.

Figure 12-1. Synaptic



Updating the package indexes

APT uses package indexes to keep track of which packages are available. You can update the package indexes by clicking on the “Update List” button. A window will pop up, which shows the current progress in fetching the package indexes.

Upgrading the system

You can update your system after updating the package indexes by clicking the “Upgrade System” button. Synaptic will ask you whether you would like to upgrade the system with “dist-upgrade” or not. The difference between an “upgrade” and an “dist-upgrade” is explained in the Section called *Basic APT usage*.

Installing packages

By default the main screen lists all available packages in categories. You can install packages by double clicking on a package. You can execute the installation by clicking the “Execute” button.

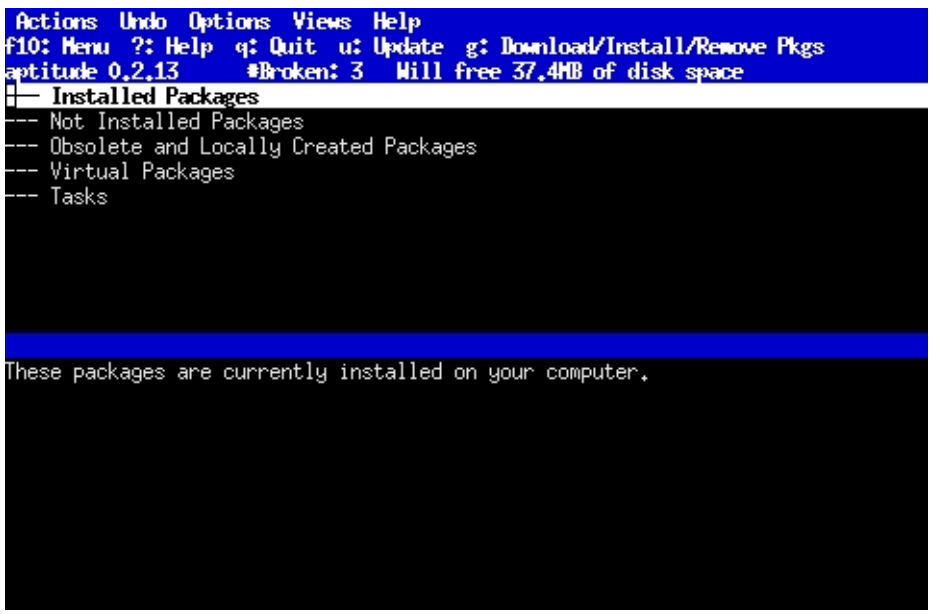
Removing packages

To remove packages it is a good idea to limit the listed packages to packages that are installed. This can be done by selecting “Installed” from the “Show:” drop-list. A package can be queued for removal by clicking on the package with the right mouse button, and selecting “remove”. This will not delete the configuration files of a program. You can remove these too, by selecting “Remove Including Configuration” instead. You can execute the removal of packages by clicking on the “Execute” button.

Aptitude

Aptitude is a popular APT front-end with a textual interface. It can be a bit more difficult to use than Synaptic, but Aptitude is really powerful. Aptitude can be launched using Adminmenu, or the **aptitude** command.

Figure 12-2. Aptitude



Updating the package indexes

The package indexes can be updated by pressing the “u” key, or by opening the menu with the F10 key, and selecting “Update package lists” from the “Actions” menu.

Upgrading the system

The system can be upgraded by selecting the “Install/remove packages” option from the “Actions” menu, after updating the package lists.

Installing and removing packages

Packages can be marked to be installed with the “+” key, the “-” key deselects a package, or selects it for removal. You can confirm your selection by using the “Install/remove packages” option from the “Actions” menu.

Basic APT usage

The sources.list file

The `/etc/apt/sources.list` is used to configure which package repositories should be used by APT. Each uncommented line specifies a repository, in the following form:

```
<Archive type> <Archive URL> <Distribution> <Component>
```

The archive type is either *deb* or *deb-src*. The first type (*deb*) specifies that the archive contains binary packages, the second type (*deb-src*) specifies that the archive contains sources. You will probably use *deb* most of the time, because it is rarely necessary to recompile packages. The archive URL (Universal Resource Locator, the standard which is also used in web browsers) specifies where the archive is located. The distribution parameter tells APT which distribution should be used from the archive, often this is one of the Debian distributions (for example *stable*. The last parameter specifies which component should be used, for example *main* or *non-free*. The naming conventions for distribution and component are fairly standard for most APT archives. Let’s have a look at an example line:

```
deb ftp://ftp.nl.debian.org/debian woody main contrib non-free
```

This line specifies that APT may use the archive located at *ftp://ftp.nl.debian.org/debian* to get binary *woody* distribution packages from the *main*, *contrib* and *non-free* components.

Updating the package index files

APT uses package index files to know which packages are available. These indexes are fetched from the locations specified in the `/etc/apt/sources.list` file. It is a good idea to update the package indexes regularly. This will allow you to upgrade installed packages and to install the latest available packages. Updating the package cache is very simple, the following command will do the job:

```
# apt-get update
```

Upgrading installed packages

There are two methods for upgrading installed packages: the *upgrade* and *dist-upgrade* methods. The *upgrade* will never remove installed packages, and it will not install packages that are not installed yet. This means that if an upgrade package is found for an installed package, but this upgrade depends on an package that is not installed, the package will not be upgraded. The *dist-upgrade* method removes and installs packages if necessary. This means it sometimes upgrades a package at the expense of a less important package. If you want to be on the safe side, use *upgrade*:

```
# apt-get upgrade
```

If you would rather like to upgrade your software as much as possible, use *dist-upgrade*:

```
# apt-get dist-upgrade
```

Installing packages

Packages can be installed with the *install* parameter of **apt-get**. You can just specify the packages you would like to install after the **apt-get install** command. For example:

```
# apt-get install gimp
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  aalib1 libgimp1.2 libgimpprint1 libgtkxmhtml1 libmpeg1
Suggested packages:
  gimp-nonfree freefont gimp-data-extras gimp-perl gimpprint-locales
The following NEW packages will be installed:
  aalib1 gimp libgimp1.2 libgimpprint1 libgtkxmhtml1 libmpeg1
0 upgraded, 6 newly installed, 0 to remove and 1 not upgraded.
Need to get 9204kB of archives.
After unpacking 29.2MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

As you can see in this example, we are requesting APT to install the GIMP. If there are other dependencies that have to be installed (like in the example above) APT will ask you for a confirmation.

Removing packages

Removing packages is as easy as installing packages. The packages specified after the **apt-get remove** command will be removed. For example:

```
# apt-get remove pppoe
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  pppoe pppoeconf
0 upgraded, 0 newly installed, 2 to remove and 1 not upgraded.
Need to get 0B of archives.
After unpacking 367kB disk space will be freed.
Do you want to continue? [Y/n]
```

Removing packages this way will remove the software, but will leave the configuration files around. Add the *--purge* parameter to remove the configuration files too. For example:

```
# apt-get --purge remove pppoe
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  pppoe* pppoeconf*
0 upgraded, 0 newly installed, 2 to remove and 1 not upgraded.
Need to get 0B of archives.
```

```
After unpacking 367kB disk space will be freed.
Do you want to continue? [Y/n]
```

As you can see a “*” is added to the package names, which means that the configuration files will be removed for these packages.

Searching for packages

Sometimes you might be looking for a program to do a certain task, but you do not know which programs can do the job. In this situation the *search* option of the **apt-cache** program can be of great help. The **apt-cache** utility can look for search terms in package descriptions. For example, suppose that you are looking for a program which can fetch mail for you from a HotMail account. Decide which keywords might be useful for the search, and add these keywords to the **apt-cache search** command. In this example the “hotmail” keyword might be specific enough. Let us have a look what results **apt-cache** gives:

```
$ apt-cache search hotmail
gotmail - Utility to download email from a Hotmail or MSN account
hotway - POP3 to Hotmail (HTTPmail) gateway
```

Two packages are found with the keyword “hotmail”, and as you probably guessed the “gotmail” package will do the job.

Showing information about a package

The **apt-cache** can also be used to show information about a package. The *show* parameter will show the package description, dependencies and other useful information. For example:

```
$ apt-cache show beneath-a-steel-sky
Package: beneath-a-steel-sky
Priority: optional
Section: games
Installed-Size: 70912
Maintainer: Tore Anderson <tore@debian.org>
Architecture: all
Version: 0.0368-2
Depends: scummvm (>= 0.5.1-1)
Filename: pool/main/b/beneath-a-steel-sky/beneath-a-steel-sky_0.0368-2_all.deb
Size: 69314378
MD5sum: 675257f6721d8b2f172a82dca01b45ed
Description: a science fiction adventure game
 A science-fiction thriller set in a bleak post-apocalyptic vision
 of the future, Beneath a Steel Sky revolves around "Union City",
 where selfishness, rivalry, and corruption by its citizens seems to
 be all too common, those who can afford it live underground, away
 from the pollution and social problems which are plaguing the city.
 .
 You take on the role of Robert Foster, an outcast of sorts from the
 city since a boy who was raised in a remote environment outside of
 Union City simply termed "the gap".  Robert’s mother took him away
 from Union City as a child on their way to "Hobart" but the helicopter
 crashed on its way, unfortunately Robert’s mother dies, but he
 survives and is left to be raised by a local tribe from the gap.
 .
```

```
Years later, Union City security drops by and abducts Robert, killing
his tribe in the process; upon reaching the city the helicopter taking
him there crashes with him escaping, high upon a tower block in the
middle of the city he sets out to discover the truth about his past,
and to seek vengeance for the killing of his tribe.
```

Useful tools

debfoster

Debfoster is a useful tool that helps you to keep your system clean. After starting the **debfoster** it will ask you for each package that is not a dependency for another package whether you would like to keep it or not. For example:

```
# debfoster

gdm2 is keeping the following 25 packages installed:
  bonobo-activation gconf2 gnome-mime-data libbonobo-activation4 libbonobo2-0
  libbonobo2-common libbonoboui2-0 libbonoboui2-common libfam0c102
  libgconf2-4 libgnome2-0 libgnome2-common libgnomecanvas2-0
  libgnomecanvas2-common libgnomeui-0 libgnomeui-common libgnomevfs2-0
  libgnomevfs2-common libgnutls5 libgsf-1 libidl0 liblinc1 libopencdk4
  liborbit2 librsvg2-2
Keep gdm2? [Ynpsiuqx?], [H]elp:
```

As you can see there are several option. “y” is the default option, and will keep a package installed. “n” will remove the package. Another useful choice is “p”, which will remove the package, and any dependencies that are not needed for other packages.

Be sure to read the **debfoster**(8) manual page before you start using debfoster. It is a very powerful tool, but you can remove important packages if you do not understand its behavior.

Chapter 13. Network configuration

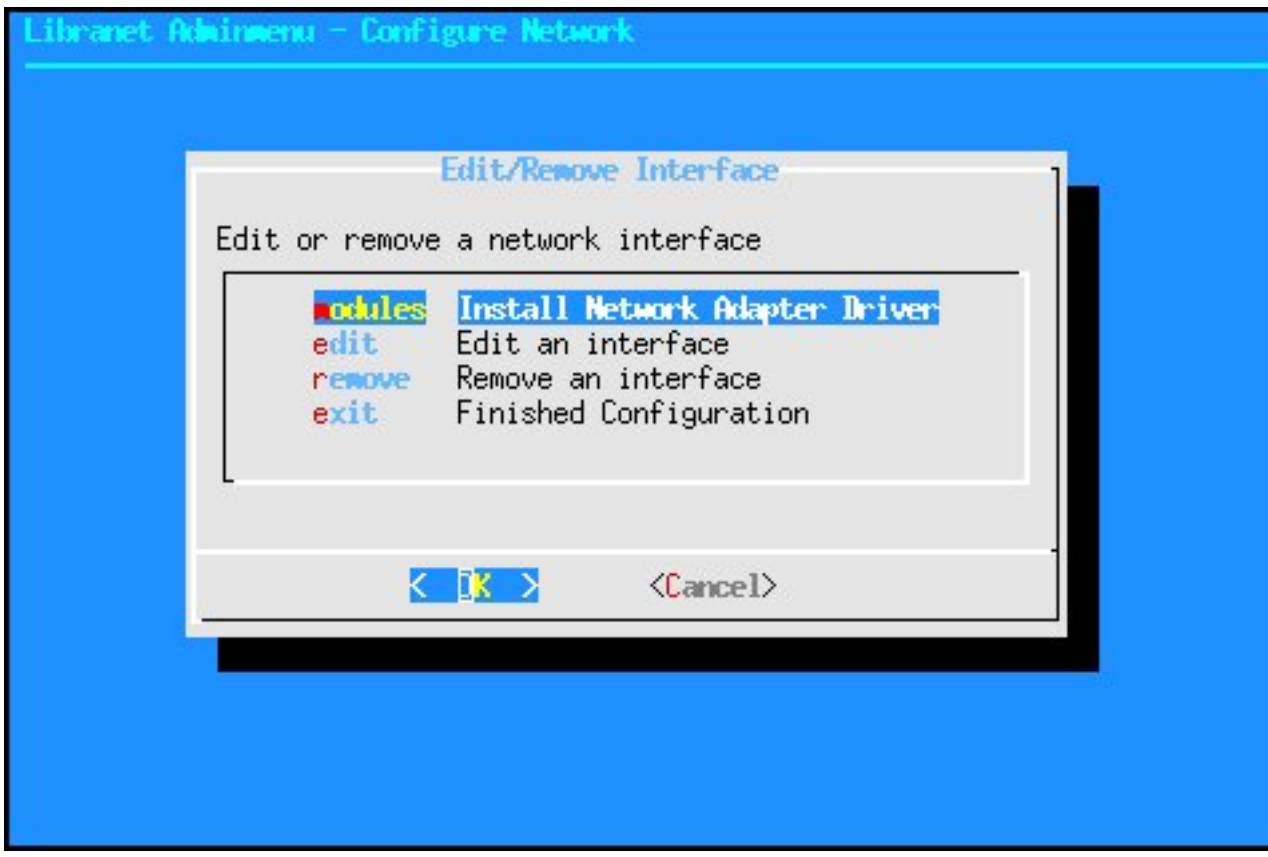
Adminmenu

Setting up ethernet cards

Ethernet cards can be configured by selecting “expert” and then “eth” in the network configuration. The resulting screen (Figure 13-1) provides several options:

- *modules*: This option is used to load modules (drivers) for the network adapters that you want to use.
- *edit*: You can configure the settings for an network interface, like the IP address, and default route with this option.
- *remove*: The *remove* option is used to remove the configuration for an interface.

Figure 13-1. Ethernet configuration menu

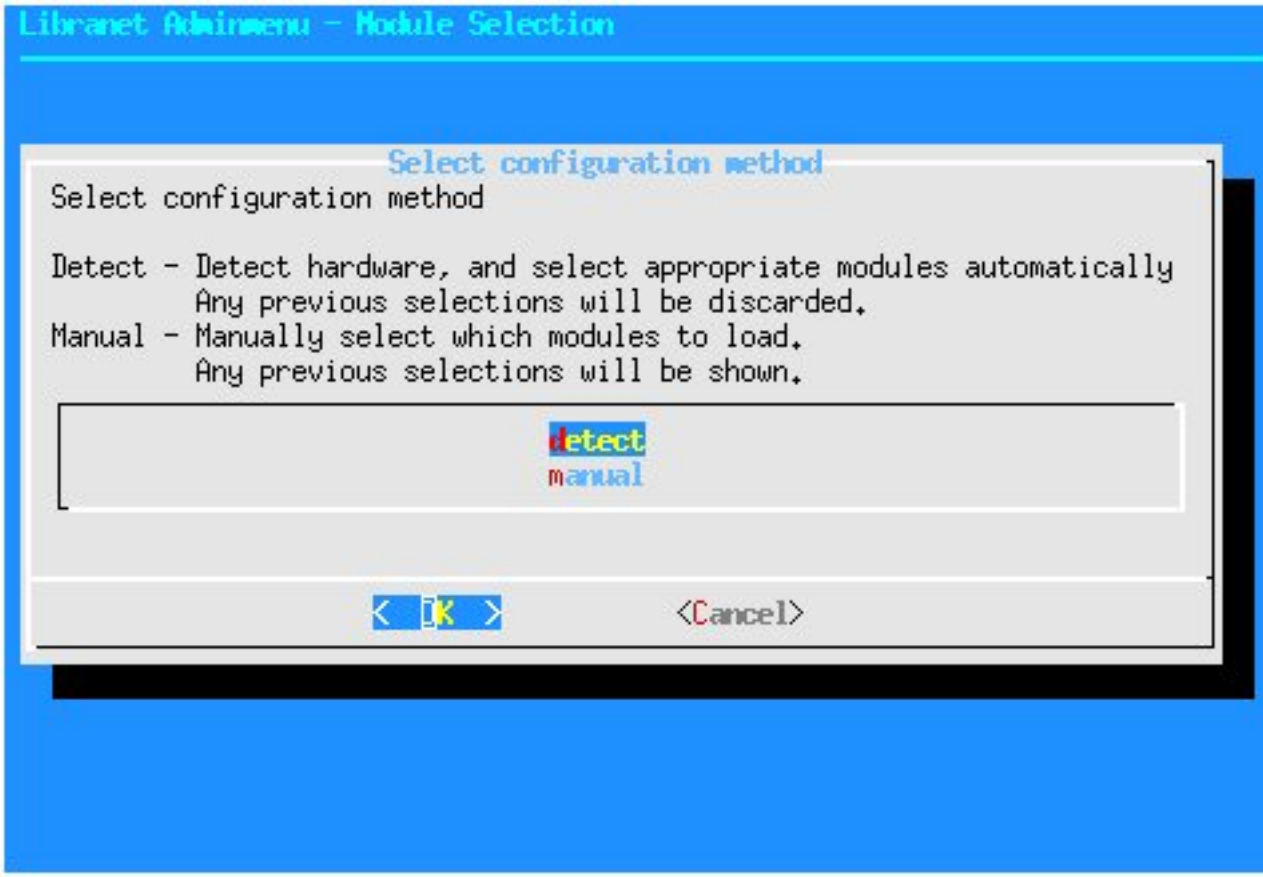


Interfaces are named in Linux, example names are *eth2*, *lo* and *ppp0*. Normal (ethernet) network interfaces are named *ethn*, in which *n* is a number starting with 0. For example, the first network card is named *eth0*, the second *eth1*, etc.

Installing the network card drivers

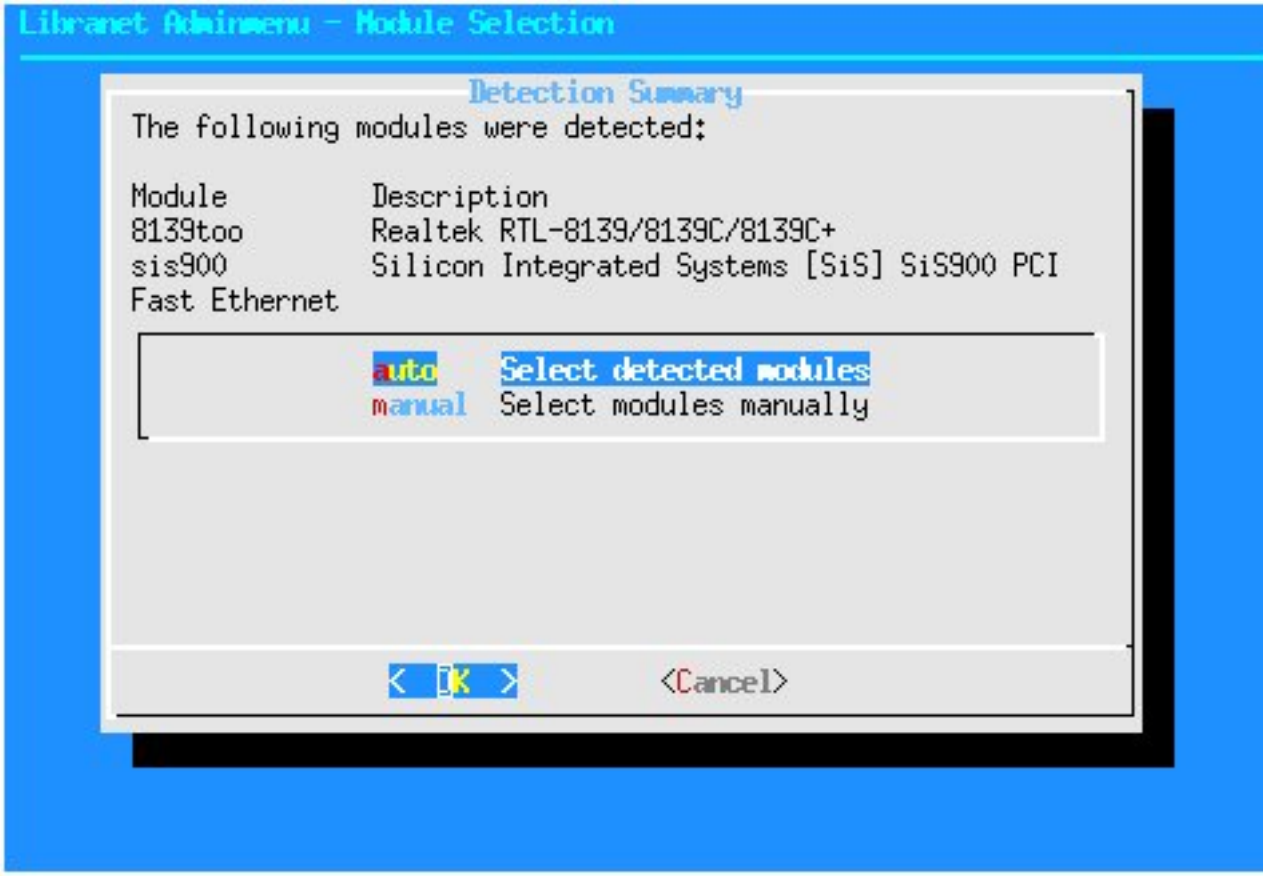
To install the drivers for the network adapters, select the *modules* option from the ethernet configuration menu described above. Adminmenu will then ask you whether you would like to let Adminmenu detect the network adapters automatically or not (Figure 13-2).

Figure 13-2. Selecting automatic detection or manual configuration



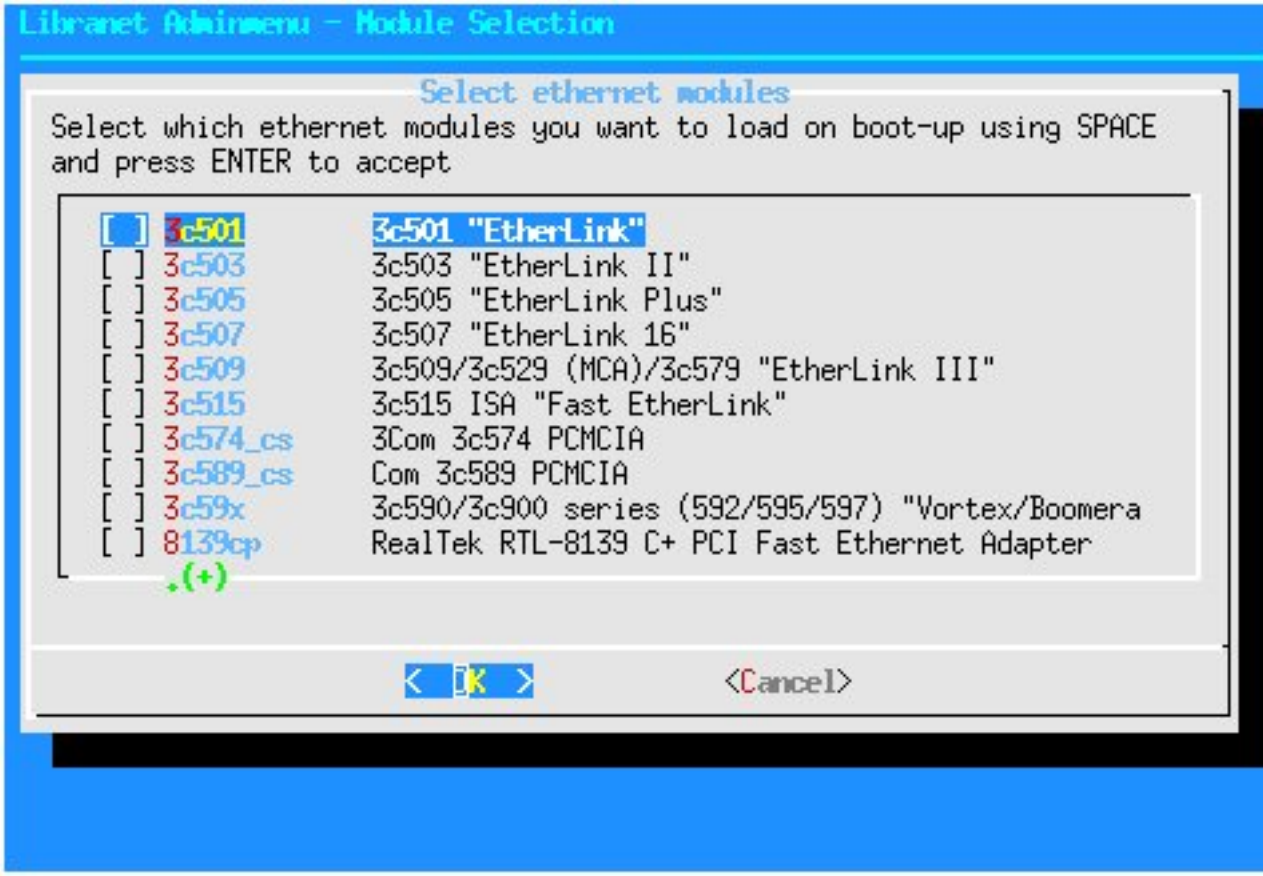
If you chose to let Adminmenu detect network adapters, Adminmenu will show a list of detected adapters when the detection is finished (Figure 13-3). Choose *auto* if you would like Libranet to use the drivers shown. At this point you can still choose to select the drivers manually by selecting *manual*.

Figure 13-3. Summary of detected network adapters



If you chose to configure manually which drivers should be used, Adminmenu will show all available drivers (Figure 13-4). You can select or deselect drivers with the <space> key, and confirm your selection with the <Enter> key.

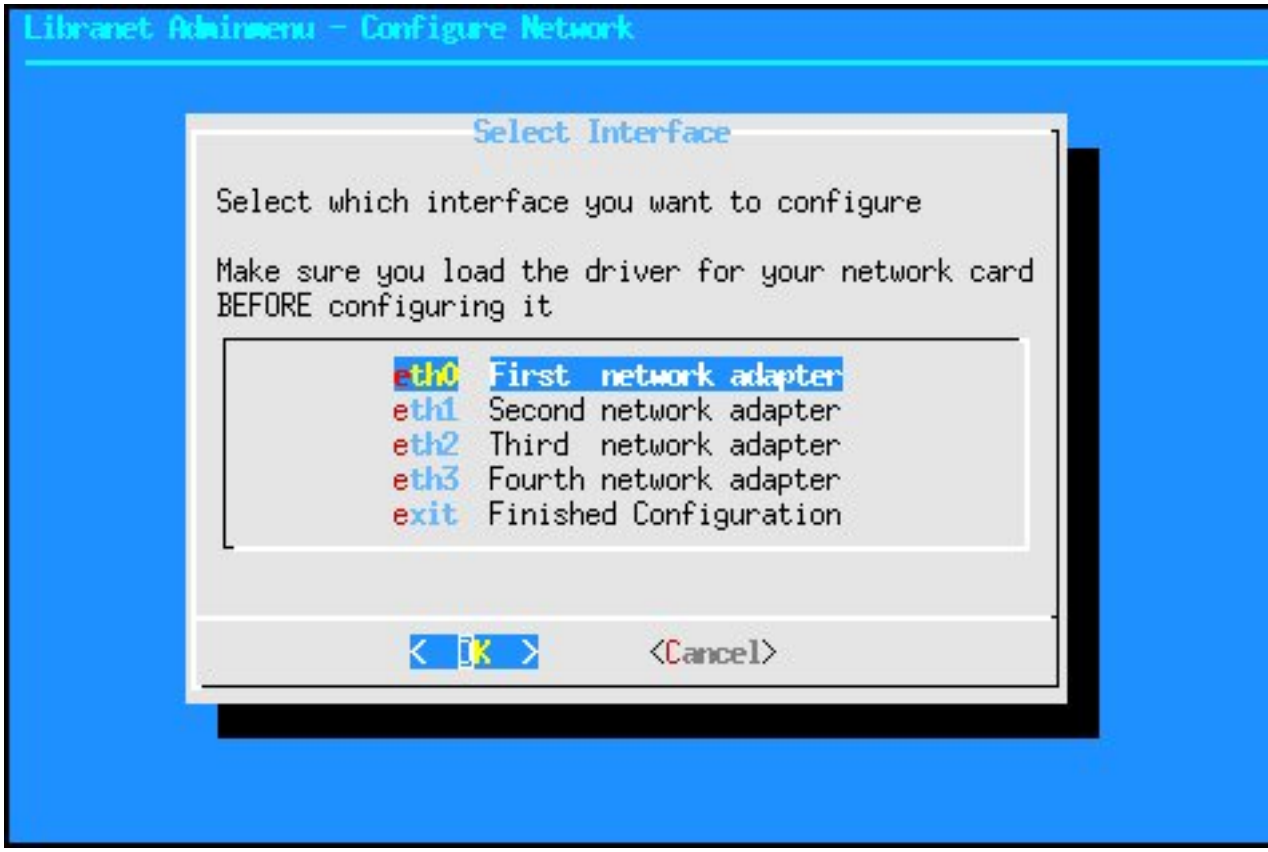
Figure 13-4. Summary of detected network adapters



Editing an interface

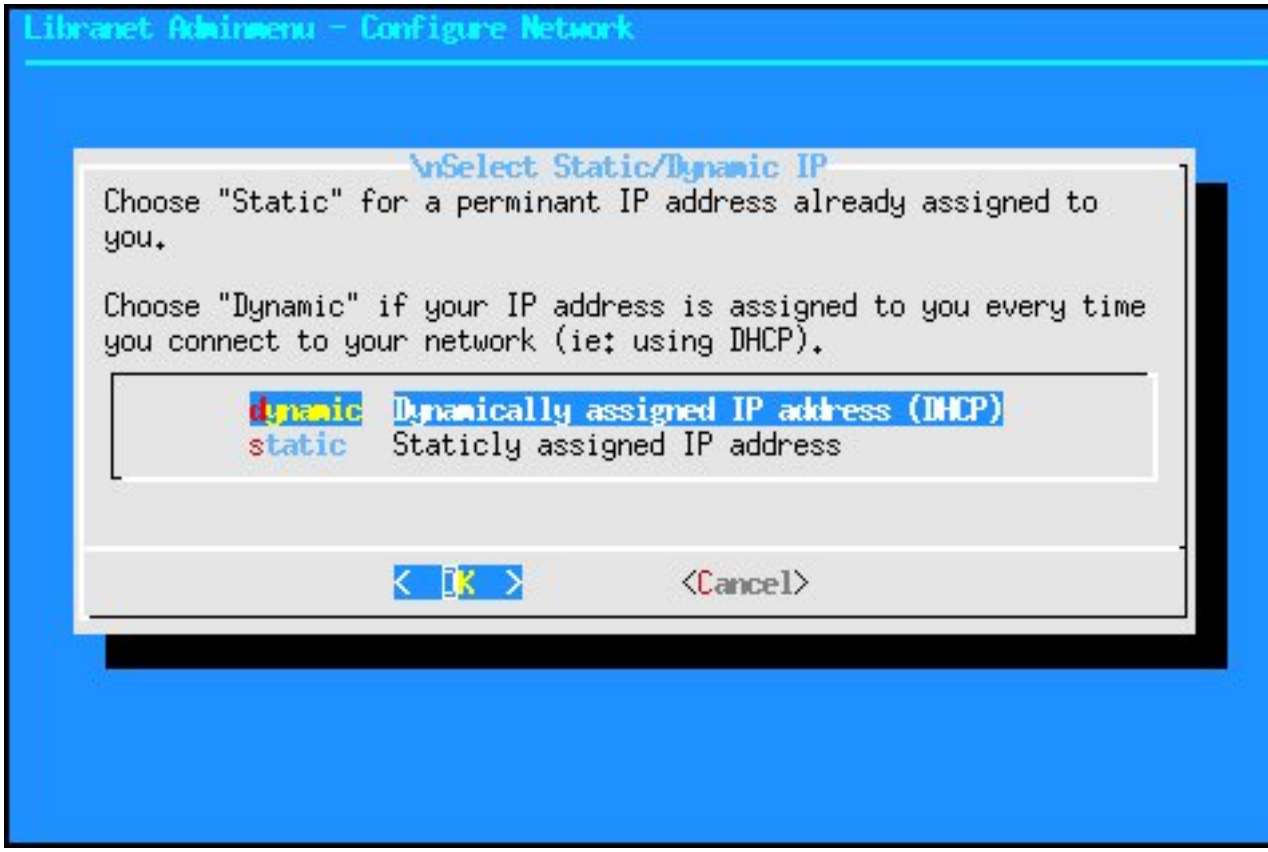
You can configure an interface by selecting the *edit* menu option described above. The next dialog will ask you which interface you would like to configure (Figure 13-5). Select the interface you would like to configure to proceed.

Figure 13-5. Selecting an interface to configure



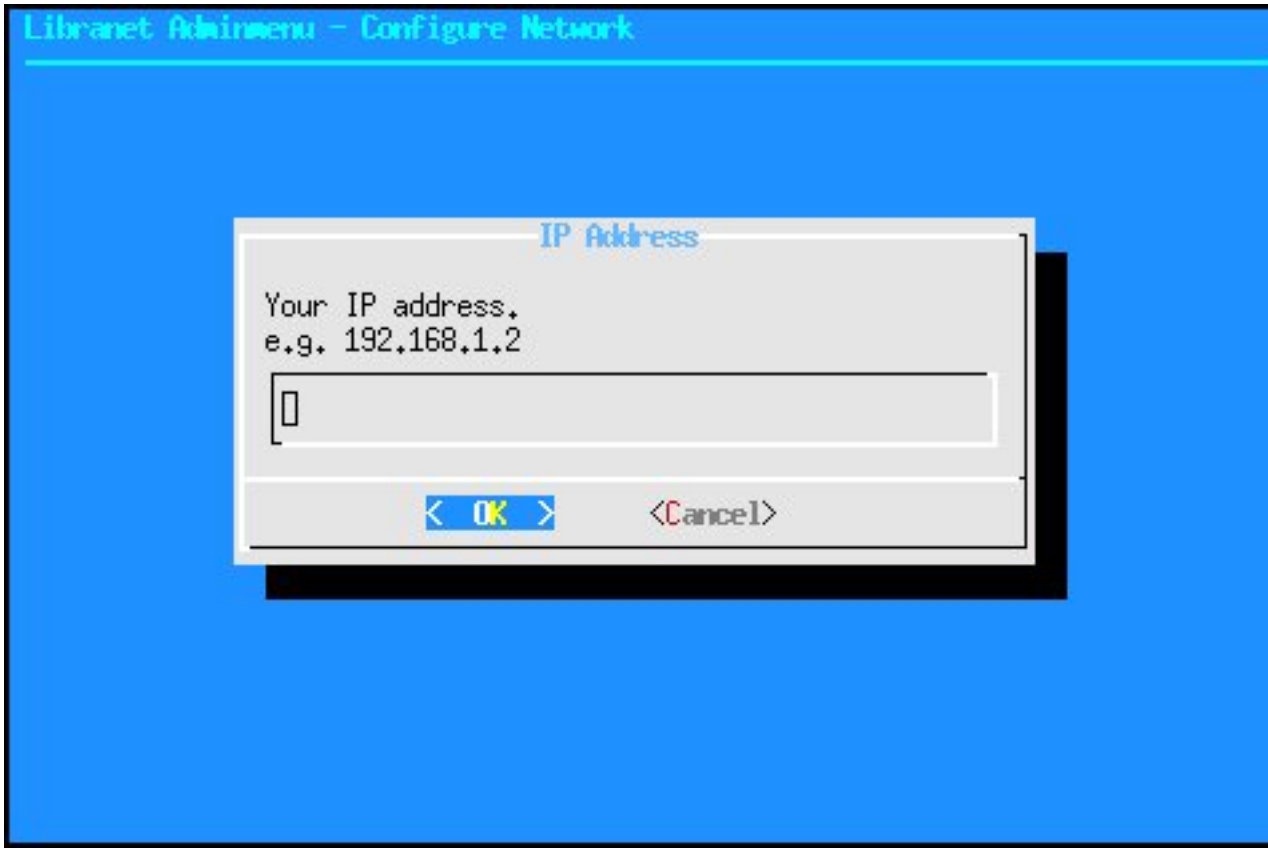
The first step of the configuration of an interface is choosing whether you want to use a dynamic IP address or not. If you want to use a DHCP server on the network, select *dynamic*. The rest of this section is based on choosing *static*, because using a dynamic IP address does not require further configuration except for setting an optional hostname.

Figure 13-6. Using a dynamic or static IP address



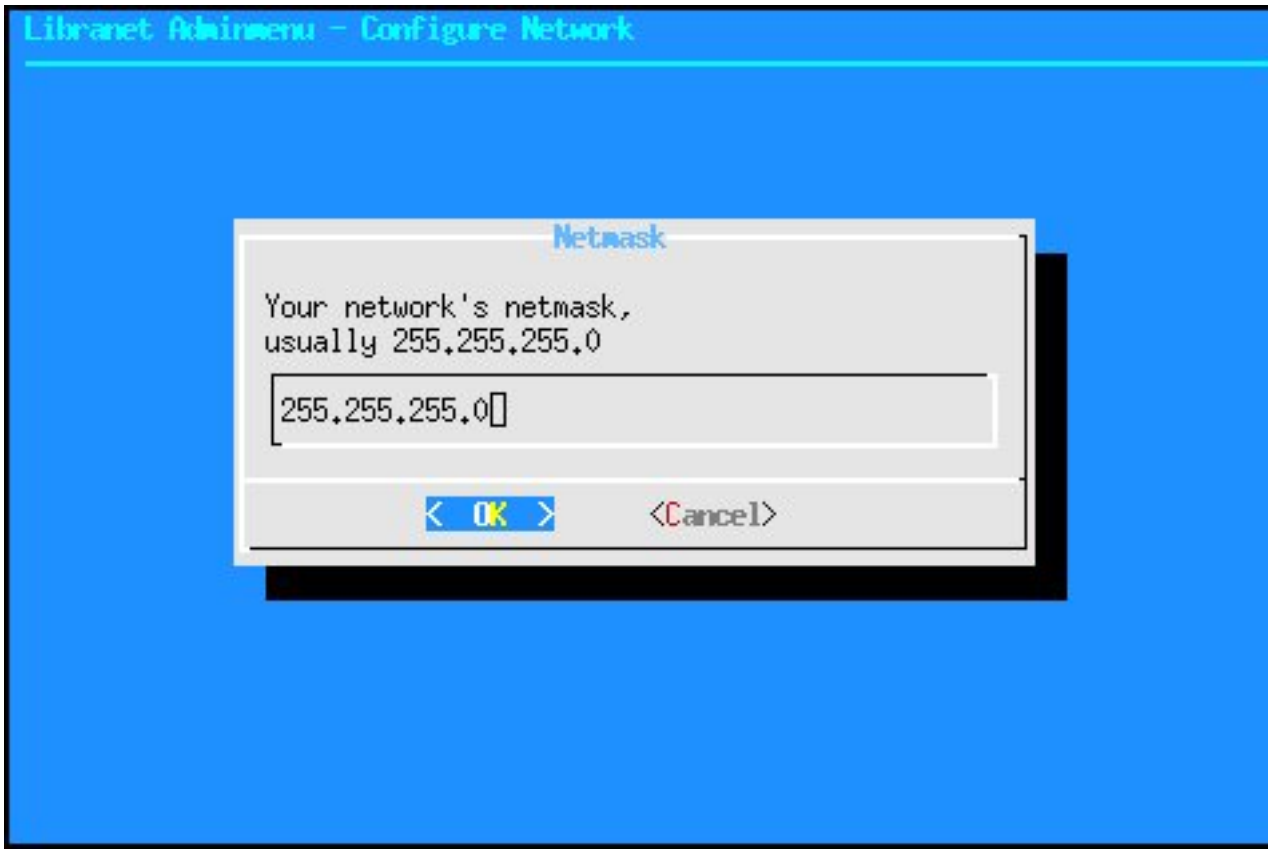
After choosing to use a static IP you can set the IP address of the interface. If you have a home network you can safely use the *192.168.1.xxx* range (where xxx is a number from 1 to 254). Please be aware that hosts on the same network should use the same IP range. If you are going to use Libranet in a company or institutional network, and you are not sure what IP address you should use, ask the network administrator.

Figure 13-7. Setting the interface IP address



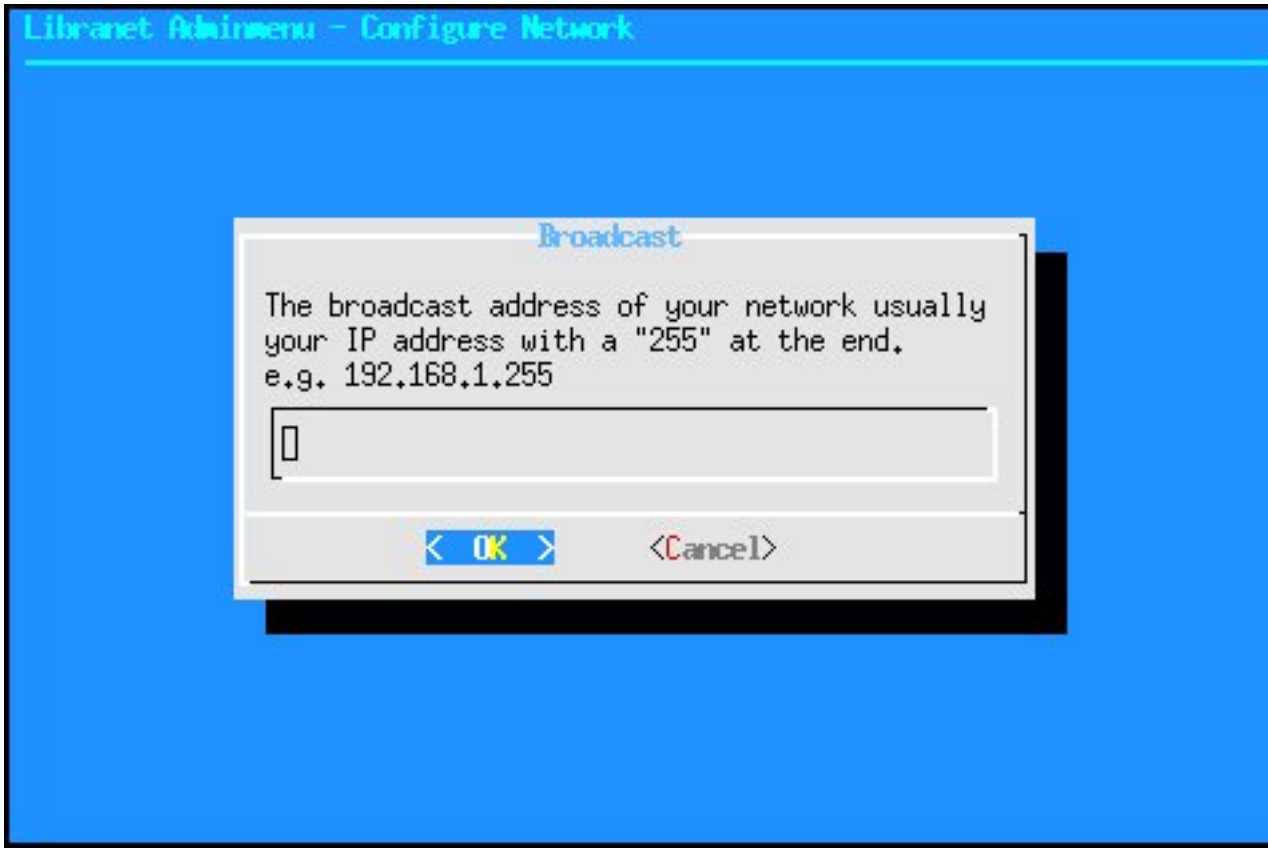
The next step is to set the netmask that should be used. On most smaller networks this can be set to *255.255.255.0*, for instance when all hosts are in the *192.168.1.xxx* range. Ask your network administrator if you are not sure about what netmask you should use.

Figure 13-8. Setting the interface netmask



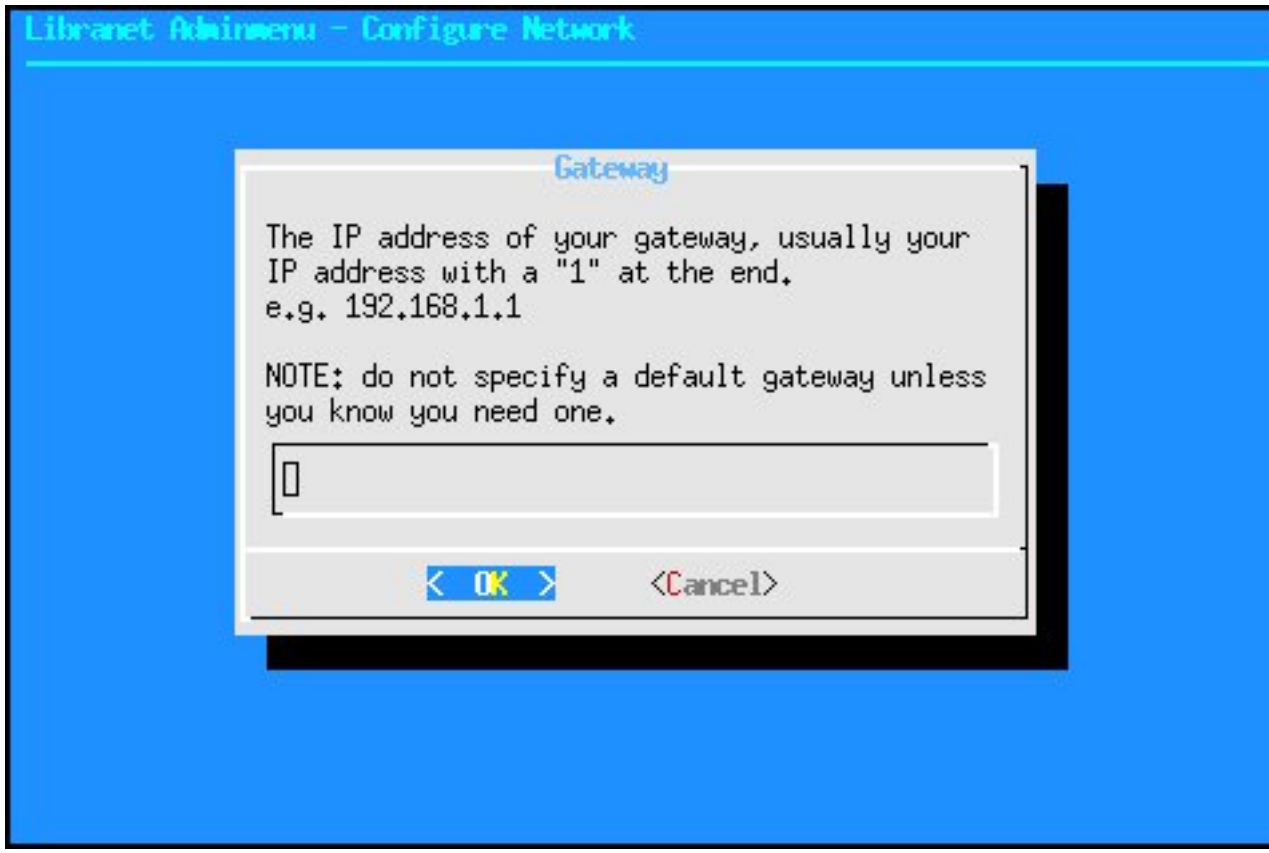
Adminmenu will also ask you to enter the broadcast address, aside from special situations you can leave this field blank. Linux will try to determine the broadcast address automatically.

Figure 13-9. Setting the broadcast address



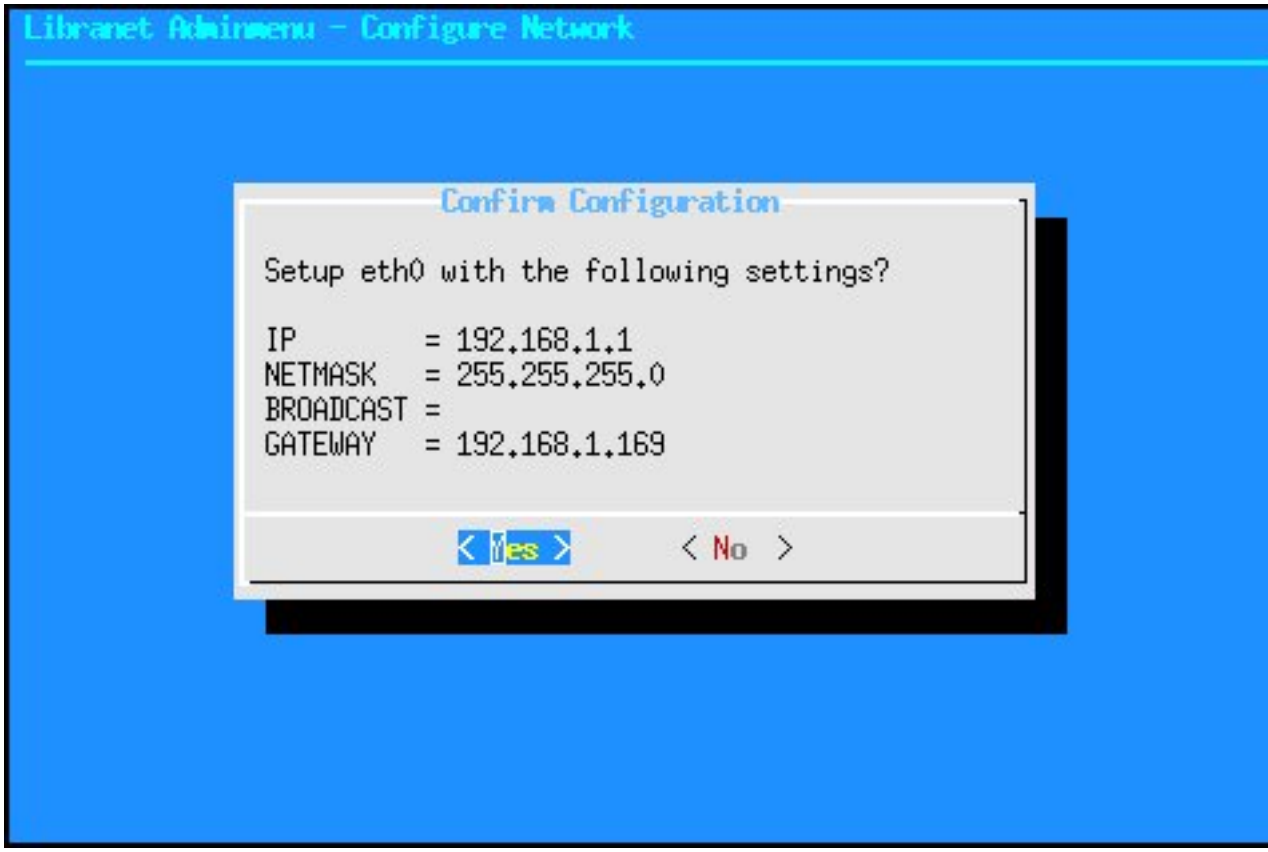
Finally you can set the default gateway. This is a host on the network that routes traffic to other networks, for example the internet. This is often called the default route. If you are using Libranet on a network that is not connected to any other networks or the internet, you can leave this setting blank.

Figure 13-10. Setting the default gateway



At this point all settings for the interface are completed. Adminmenu will ask you for a final confirmation on the selected values.

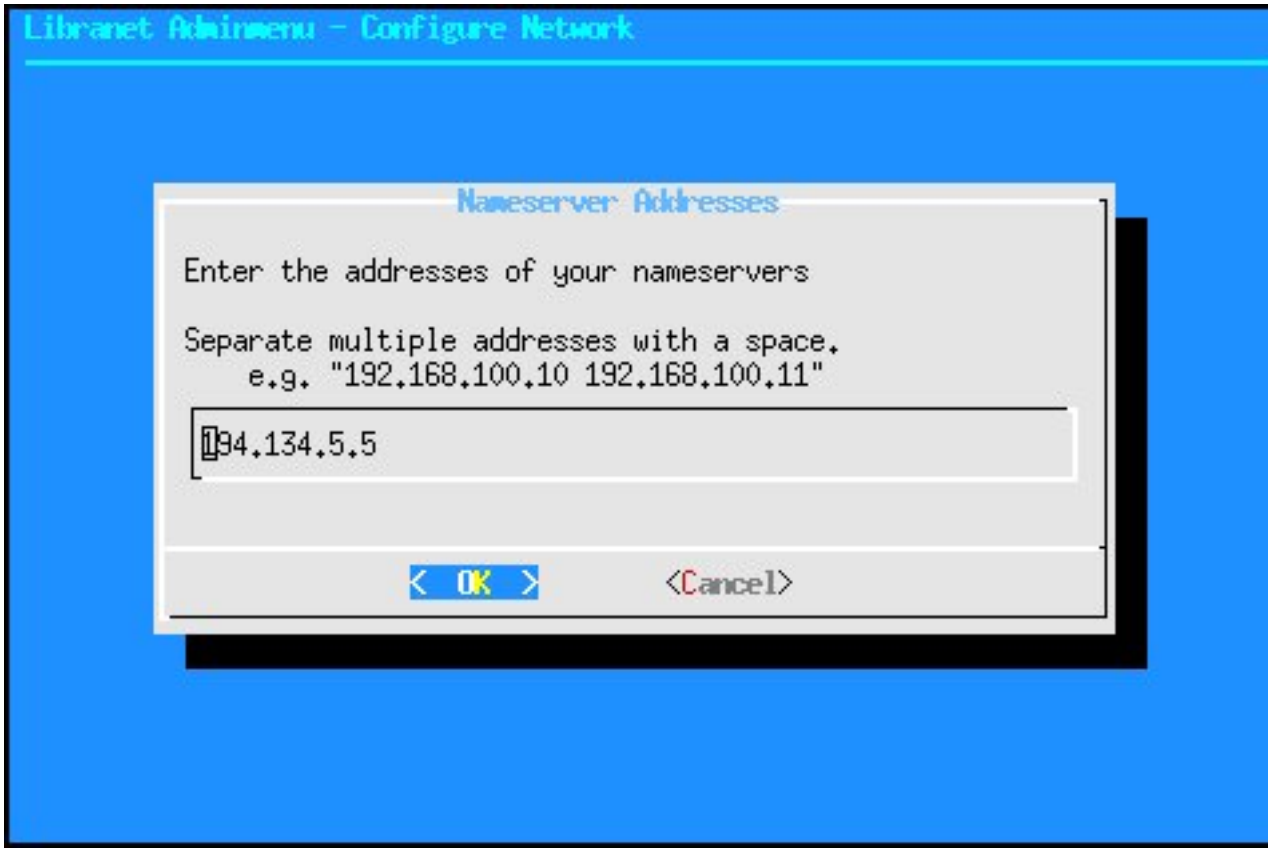
Figure 13-11. Selecting an interface to configure



Configuring DNS

DNS servers have a very special task in networking. They translate hostnames (for instance www.libranet.com) to IP addresses. To be able to use hostnames the system needs access to one or more DNS servers. Almost all Internet service providers provide DNS servers. DNS can be configured by selecting “expert” and then “nameserver” in the adminmenu network configuration. Adminmenu will then ask for the nameservers (Figure 13-12). You can enter the IP addresses in the input field, separating them with a space.

Figure 13-12. Configuring DNS



Manual configuration

This chapter explains how some common network-related configuration files work. Most things can be configured with adminmenu, but more experienced users might prefer configuring things by editing the relevant configuration files.

Configuring nameservers

The name resolver is configured using the `/etc/resolv.conf` file. Nameservers that should be used can be configured by adding a lines of the following form: *nameserver <IP address>*. For example:

```
nameserver 192.168.1.200
nameserver 192.168.1.201
```

Local hosts

Besides resolving using name servers specified in `/etc/resolv.conf`, GNU/Linux can also resolve hosts that are specified in `/etc/hosts`. This file is handy for adding names to local hosts,

without the hassle of configuring a local name server. Hosts are specified in the following form: <*IP address*> <*name1*> ... <*namen*>. This is an example of the `/etc/hosts` file:

```
127.0.0.1       localhost localhost.localdomain
192.168.1.1     host1.example.net
192.168.1.2 host2.example.net
```

Be sure that there is an entry for the host name that is used (you can display the host name with the **hostname** command). If the host name is not resolvable some programs refuse to work correctly.

> **Note:** If you want to use the same `hosts` file on a relatively large number of machines, it might we worth looking at the *dnsmasq* package. Dnsmasq can provide a `hosts` file on one host to a complete network through its DNS server.

Chapter 14. Printer configuration

Introduction

GNU/Linux supports a large share of the available USB, parallel and network printers. Libranet GNU/Linux provides the CUPS printing system. CUPS can be configured via adminmenu, as well as the CUPS web interface.

Adminmenu configuration

Adminmenu provides a special “Printer” section. This section contains three buttons for configuring CUPS. The “Detect printers” button will install CUPS, and tries to detect the printer. “Configure CUPS” provides a X interface for configuring CUPS, the second “Configure CUPS” launches a browser that will load the CUPS web interface.

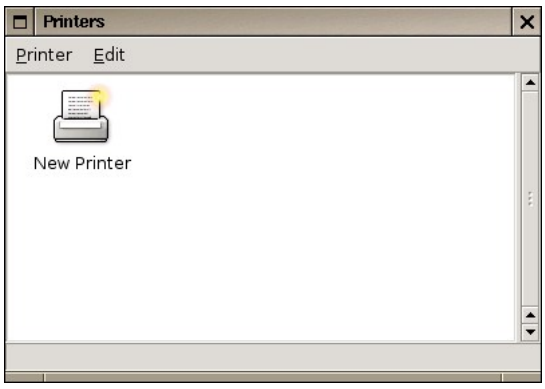
Detect printers

The “Detect printers” button launches a tool that tries to detect the printer, and configures CUPS automatically when a printer is found.

Configure CUPS

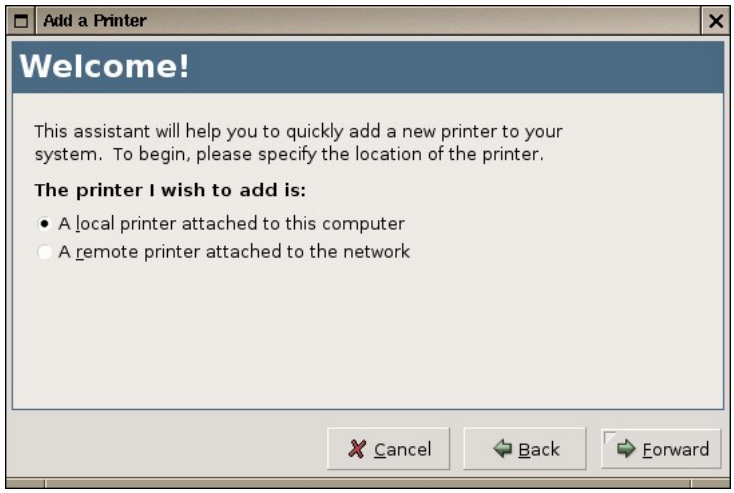
The “Configure CUPS” item in Xadminmenu provides a comfortable tool for configuring printers using CUPS. After launching this item a Window showing the configured printers pops up (Figure 14-1).

Figure 14-1. Configured printers



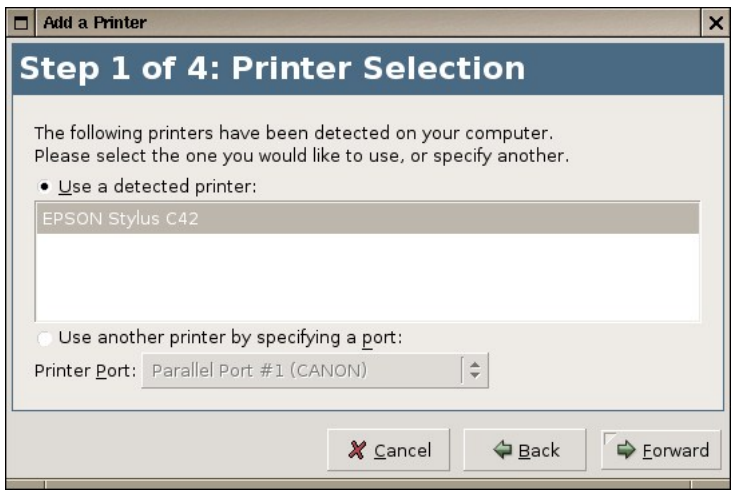
A new printer can be added by double-clicking the “New Printer” icon. This launches the printer configuration dialog (Figure 14-2, which allows you to select whether you would like to configure a local printer or a network printer. In this section we are going to look at how you can configure a local printer attached to the computer.

Figure 14-2. Selecting the kind of printer you would like to add



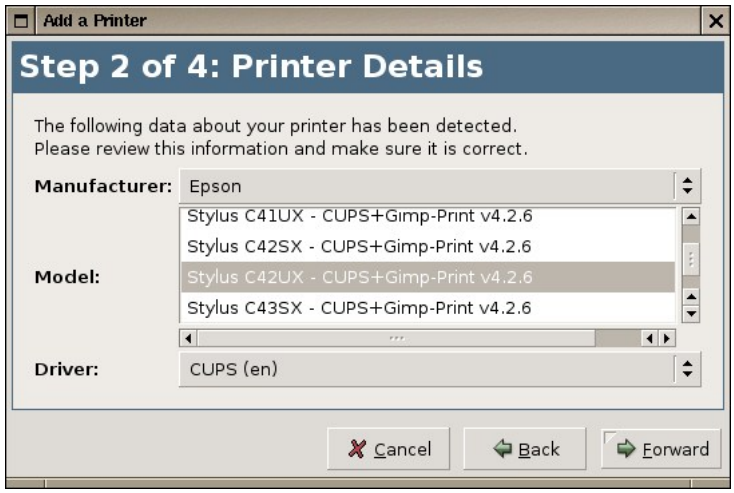
After specifying that you would like to setup a local printer, the dialog asks you to select which printer you would like to configure (Figure 14-3). If you have an USB printer that is turned on you can select it from the detected printers list. If the printer is not shown you can manually select the printer port manually. If you do not have an USB printer you should select “parallel port #1” most of the times.

Figure 14-3. Selecting the printer you would like to configure



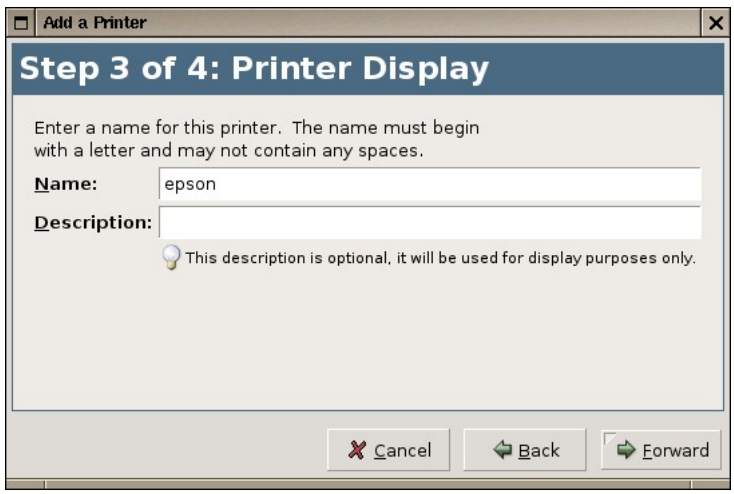
The next step is to specify the manufacturer and model of the printer (Figure 14-4). If your printer is not yet supported, it is a good idea to upgrade your Libranet installation using the Libranet safe archive. There is a good chance that newer drivers for CUPS support your printer.

Figure 14-4. Configuring the printer manufacturer and model



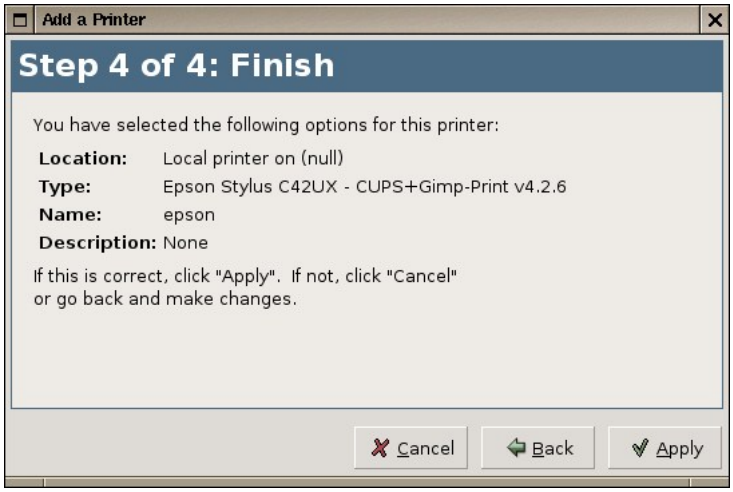
The next thing the printer needs is a name(Figure 14-5). The name and a description for the printer can be configured during the next step. The name of the printer is shown in many programs in the printer selection dialog. A description for the printer is only optional.

Figure 14-5. Setting the printer name



The final step will ask you for a confirmation (Figure 14-6). After confirming the settings the printer should show up in printers window.

Figure 14-6. Finishing the printer configuration



Web interface configuration

CUPS can be configured via a web interface. The configuration interface can be accessed with a web browser at the following URL: http://localhost:631/. Some parts of the web interface require that you authenticate yourself. If an authentication window pops up you can enter “root” as the user name, and fill in the root account password.

A printer can be added to the CUPS configuration by clicking on “Administrate”, and clicking on the “Add Printer” button after that. The web interface will ask for three options:

- *Name* - the name of the printer. Use a simple name, for example “epson”.
- *Location* - the physical location of the printer. This setting is not crucial, but handy for larger organizations.
- *Description* - a description of the printer, for example “Epson Stylus Color C42UX”.

You can proceed by clicking the “Continue” button. On the next page you can configure how the printer is connected. If you have an USB printer which is turned on, the web interface will show the name of the printer next to the USB port that is used. After configuring the printer port you can select the printer brand and model. After that the printer configuration is finished, and the printer will be added to the CUPS configuration.

An overview of the configured printers can be found on the “Printers” page. On this page you can also do some printer operations. For example, “Print Test Page” can be used to check the printer configuration by printing a test page.

Chapter 15. XFree86

X Configuration

The XFree86 configuration is stored in `/etc/X11/XF86Config-4`. XFree86 can be configured by hand by editing the `XF86Config-4` file. Fortunately Libranet provides some useful tools to configure XFree86.

Libranet Adminmenu

Adminmenu provides an excellent X configuration tool. This tool can be started by executing **adminmenu** in a shell. Select "Configure X-windows", and "Configure X-windows" once more in the next screen. This will launch the X configuration tool. The first thing that will be asked is whether you want to set up XFree86 automatically or manually. Detecting hardware automatically works perfectly most of the time.

Automatic detection

After selecting "auto" the X configuration tool will try to detect the display hardware, and will show the results afterward. If this is not correct, select "No", this will start the manual configuration.

Manual configuration

Select “manual” to start the manual configuration of X. The configuration tool will start by asking which video driver should be used. Right of the name of the driver the vendor and/or chipset series are displayed. Select the driver that is appropriate. The next screen will ask you how much video memory the card has. Normally you can leave this blank and just press <Enter>. This will bring you to the mouse type screen, select the mouse type you have, press <Enter>, and select the mouse protocol that should be used. The next screen asks you what kind of monitor you have. It is a good idea to have a look at the manual of your monitor to find out what specifications your screen has. Selecting a refresh rate that is too high can damage the monitor. Then select the resolution you want to use. The resolutions you can use are dependent on your monitor and video card. After that you can set the number of dots per inch (DPI) for your monitor. Font sizes will be adjusted to the DPI value to make fonts readable. The easiest way to set the DPI is to select “Set monitor DPI based on monitor size”, which allows you to set the DPI by the size of your monitor. Finally you can set the color depth that should be used.

Automatical configuration

The XFree86 server provides an option to automatically generate a configuration file. XFree86 will load all available driver modules, and will try to detect the hardware, and generate a configuration file. Execute the following command to generate a XFree86 configuration file:

```
$ XFree86 -configure
```

If X does not output any errors, the generated configuration can be copied to the `/etc/X11` directory. And X can be started to test the configuration:

```
$ cp /root/XF86Config /etc/X11/
$ startx
```

Interactive configuration

XFree86 provides two tools for configuring X interactively, **xf86cfg** and **xf86config**. **xf86cfg** tries to detect the video card automatically, and starts an tool which can be used to tune the configuration. Sometimes **xf86cfg** switches to a video mode which is not supported by the monitor. In that case **xf86cfg** can also be used in text-mode, by starting it with **xf86cfg -textmode**.

xf86config differs from the tools described above, it does not detect hardware and will ask detailed questions about your hardware. If you only have little experience configuring XFree86 it is a good idea to avoid **xf86config**.

Window manager

The "look and feel" of XFree86 is managed by a so-called window manager. Libranet GNU/Linux provides many window managers. The following window managers are widely used:

- WindowMaker: A relatively light window manager, which is part of the GNUStep project.
- BlackBlox: Light window manager, BlackBox has no dependencies except the X11 libraries.
- KDE: A complete desktop environment, including browser, e-mail program and an office suite (KOffice).
- GNOME: Like KDE a complete desktop environment.

If you are used to a desktop environment, using KDE or GNOME is a logical choice. But it is a good idea to try some of the lighter window managers. They are faster, and consumer less memory, besides that most KDE and GNOME applications are perfectly usable under other window managers.

Chapter 16. Security

Introduction

With the increasing usage of the Internet and wireless networks security is getting more important every day. It is impossible to cover this subject in a single chapter of an introduction to GNU/Linux. This chapter covers some basic security techniques that provide a good start for desktop and server security.

Before we go on to specific subjects, it is a good idea to make some remarks about passwords. Computer authorization largely relies on passwords. Be sure to use good passwords in all situations. Avoid using words, names, birth dates and short passwords. These passwords can easily be cracked with dictionary attacks or brute force attacks against hosts or password hashes. Use long passwords, ideally eight characters or longer, consisting of random letters (including capitals) and numbers.

E-Mail security

Introduction

GNU/Linux supports two major of securing e-mails, but before we will look into these specifically we are going to look what e-mail security provides. There are two ways in which you can secure e-mails: signing e-mail and encrypting e-mail. Signing e-mail means that a special digital signature is added to the e-mail by the mail user agent. The recipient can use the signature to check whether an e-mail is really from the sender it claims to be from or not, and that the message is not in any way changed during the transmission. E-Mail encryption codes the e-mail in a way that only the intended recipient can decode it.

This system relies on two keys: the private and the public key. Public keys are used to encrypt messages, and messages can only be decrypted with the private key. This means that one can sent his public key out to people. People can use this key to send encrypted e-mails, that only the person with the private key can decode. Of course, this means that the security of this system depends on how well the private is kept secret.

One might wonder why he or she should use one of these techniques. While most people do not feel the need to encrypt most of their e-mails, it generally is a good idea to sign your e-mails. There are, for example, a lot of viruses these days that use other people's e-mail addresses in the *From:* field of viruses. If the people who you are communicating with know that you sign your e-mails, they will not open fake e-mail from viruses. Besides that it looks much more professional if people can check your identity, especially in business transactions. For example, who would you rather trust, *vampire_boy93853@hotmail.com*, or someone using a professional e-mail address with digitally signed e-mails?

GnuPG or S/MIME

At the moment there two major standards that are supported by GNU/Linux, the OpenPGP standard, by the GnuPG tool, and S/MIME. Both are about equally good concerning the strength of the used encryption algorithms. The big difference is that S/MIME relies on a certificate authority (CA), GnuPG does not. A certificate authority is an organization that is authorized to hand out keys. This

means that the certificate authority validates that the e-mail address and/or name in the certificate is authentic. This means that either:

- The certificate (aka private and public key) belongs to the owner of the e-mail address. E-Mail verification is used by the certificate authority to check this.
- The certificate belongs to the owner of the e-mail address, and to the name in the certificate. This involves an extra step than e-mail verification. To have a certificate with a name, your identity has to be checked by the certificate authority.

As you can see this system is quite bullet-proof, and you can reasonably expect that a certificate is authentically representing the e-mail address and/or person. GnuPG does not use a certificate authority. This means that anybody can make a key with your name and e-mail address. Trust is established by signing keys. For example, person A verifies that person B's key really belongs to person B, by meeting him in real life, and he signs his key. The authenticity of a key is more certain if it is signed by people who you trust. As you can guess this system is not completely bullet proof, and it can take some time to get your key trusted by other people. Besides, if you use another e-mail address, ant want to use a new key you will have to start all over again. The advantage of this approach is that your private key is not known to a certificate authority.

GnuPG used to be supported a lot better on GNU/Linux, but more and more e-mail clients support S/MIME these days. On Windows S/MIME is better supported, because Outlook Express and Mozilla Mail (two commonly used e-mail clients) both have S/MIME support. If you are communicating with many people that use Windows, it is a good choice to go for S/MIME, because it is worthless to use signed e-mails if the recipient can not verify them by default.

Configuring and using GnuPG

This section gives a short introduction on how to configure and use GnuPG. Make sure you have GnuPG installed. If you have not, you can install it with APT:

```
# apt-get install gnupg
```

Generating your private and public keys

Generating public and private keys is a bit complicated, because GnuPG uses DSA keys by default. DSA is an encryption algorithm, the problem is that the maximum key length of DSA is 1024 bits, this is considered too short for the longer term. That is why it is a good idea to use 2048 bit RSA keys. This section describers how this can be done.

> **Note:** 1024-bit keys were believed to be secure for a long time. But D.J. Bernstein's paper "Circuits for Integer Factorization: a Proposal" contests this, the bottom line is that it is quite feasible for national security agencies to produce hardware that can break keys in a relatively short amount of time. Besides that it has be shown that 512-bit RSA keys can be broken in a relatively short time using common hardware. More information about these issues can by found in this e-mail to the cypherpunks list:
> http://lists.saigon.com/vault/security/encryption/rsa1024.html

We can generate a key by executing:

```
$ gpg --gen-key
```

The first question is what kind of key you would like to make. We will choose *(4) RSA (sign only)*:

```
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (4) RSA (sign only)
Your selection? 4
```

You will then be asked what the size of the key you want to generate has to be. Type in *2048* to generate a 2048 bit key, and press enter to continue.

```
What keysize do you want? (1024) 2048
```

The next question is simple to answer, just choose what you like. Generally speaking it is not a bad idea to let the key be valid infinitely. You can always deactivate the key with a special revocation certificate.

```
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
```

GnuPG will then ask for confirmation. After confirming your name and e-mail address will be requested. GnuPG will also ask for a comment, you can leave this blank, or you could fill in something like “Work” or “Private”, to indicate what the key is used for. For example:

```
Real name: John Doe
Email address: john@doe.com
Comment: Work
You selected this USER-ID:
    "John Doe (Work) <john@doe.com>"
```

GnuPG will the ask you to confirm your user ID. After confirming it GnuPG will ask you to enter a password. Be sure to use a good password:

```
You need a Passphrase to protect your secret key.

Enter passphrase:
```

After entering the password twice GnuPG will generate the keys. But we are not done yet. GnuPG has only generated a key for signing information, not for encryption of information. To continue, have a look at the output, and look for the key ID. In the information about the key you will see *pub 2048R/*. The key ID is printed after this fragment. In this example:

```
public and secret key created and signed.
key marked as ultimately trusted.

pub  2048R/8D080768 2004-07-16 John Doe (Work) <john@doe.com>
     Key fingerprint = 625A 269A 16B9 C652 B953  8B64 389A E0C9 8D08 0768
```

the key ID is *8D080768*. If you lost the output of the key generation you can still find the key ID in the output of the **gpg --list-keys** command. Use the key ID to tell GnuPG that you want to edit your key:

```
$ gpg --edit-key <Key ID>
```

With the example key above the command would be:

```
$ gpg --edit-key 8D080768
```

GnuPG will now display a command prompt. Execute the **addkey** command on this command prompt:

```
Command> addkey
```

GnuPG will now ask the password you used for your key:

```
Key is protected.

You need a passphrase to unlock the secret key for
user: "John Doe (Work) <john@doe.com>"
2048-bit RSA key, ID 8D080768, created 2004-07-16

Enter passphrase:
```

After entering the password GnuPG will ask you what kind of key you would like to create. Choose *RSA (encrypt only)*, and fill in the information like you did earlier (be sure to use a 2048 bit key). For example:

```
Please select what kind of key you want:
   (2) DSA (sign only)
   (3) ElGamal (encrypt only)
   (4) RSA (sign only)
   (5) RSA (encrypt only)
Your selection? 5
What keysize do you want? (1024) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
```

And confirm that the information is correct. After the key is generated you can leave the GnuPG command prompt, and save the new key with the **save** command:

```
Command> save
```

Congratulations! You have now generated the necessary keys to encrypt and decrypt e-mails and files. You can now configure your e-mail client to use GnuPG. It is a good idea to store the contents of the `.gnupg` directory on some reliable medium, and store that in a safe place! If your private key is lost you can't decrypt files and messages that were encrypted with your public key. If the private key, and your password are stolen, the security of this system is completely compromised.

Exporting your public key

To make GnuPG useful, you have to give your public key to people who send you files or e-mails. They can use your public key to encrypt files, or use it to verify whether a file has a correct signature

or not. The key can be exported using the *--export* parameter. It is also a good idea to specify the *--output* parameter, this will save the key in a file. The following command would save the public key of *John Doe*, used in earlier examples, to the file `key.gpg`:

```
$ gpg --output key.gpg --export john@doe.com
```

This saves the key in binary format. Often it is more convenient to use the so-called “ASCII armored output”, which fits better for adding the key to e-mails, or websites. You export an ASCII armored version of the key by adding the *--armor* parameter:

```
$ gpg --armor --output key.gpg --export john@doe.com
```

If you look at the `key.gpg` file you will notice that the ASCII armored key is a much more comfortable format.

Signatures

With GPG you can make a signature for a file. This signature is unique, because your signature can only be made with your private key. This means that other people can check whether the file was really sent by you, and whether it was in any way altered or not. Files can be signed with the *--detach-sign* parameter. Let us look at an example. This command will make a signature for the `memo.txt` file. The signature will be stored in `memo.txt.sig`.

```
$ gpg --output memo.txt.sig --detach-sign memo.txt

You need a passphrase to unlock the secret key for
user: "John Doe (Work) <john@doe.com>"
2048-bit RSA key, ID 8D080768, created 2004-07-16

Enter passphrase:
```

As you can see, GnuPG will ask you to enter the password for your private key. After you have entered the right key the signature file (`memo.txt.sig`) will be created.

You can verify a file with its signature using the *--verify* parameter. Specify the signature file as a parameter to the *--verify parameter*. The file that needs to be verified can be specified as the final parameter:

```
$  gpg --verify memo.txt.sig memo.txt
gpg: Signature made Tue Jul 20 23:47:45 2004 CEST using RSA key ID 8D080768
gpg: Good signature from "John Doe (Work) <john@doe.com>"
```

This will confirm that the file was indeed signed by *John Doe (Work) <john@doe.com>*, with the key *8D080768*, and that the file is unchanged. Suppose the file was changed, GnuPG would have complained about it loudly:

```
$ gpg --verify memo.txt.sig memo.txt
gpg: Signature made Tue Jul 20 23:47:45 2004 CEST using RSA key ID 8D080768
gpg: BAD signature from "John Doe (Work) <john@doe.com>"
```

Closing services

Introduction

Many GNU/Linux run some services that are open to a local network or the Internet. Other hosts can connect to these services by connecting to specific ports. For example, port 80 is used for WWW traffic. The `/etc/services` file contains a table with all commonly used services, and the port numbers that are used for these services.

A secure system should only run the services that are necessary. So, suppose that a host is acting as a web server, it should not have ports open (thus servicing) FTP or SMTP. With more open ports security risks increase very fast, because there is a bigger chance that the software servicing a port has a vulnerability, or is badly configured. The following few sections will help you tracking down which ports are open, and closing them.

Finding open ports

Open ports can be found using a port scanner. Probably the most famous port scanner for GNU/Linux is **nmap**. **nmap** is available through the Debian package repositories. You can install **nmap** through you favorite package manager, or using APT:

```
# apt-get install nmap
```

The basic **nmap** syntax is: **nmap host**. The *host* parameter can either be a hostname or IP address. Suppose that we would like to scan the host that **nmap** is installed on. In this case we could specify the *localhost* IP address, *127.0.0.1*:

```
$ nmap 127.0.0.1

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on localhost (127.0.0.1):
(The 1596 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     open        telnet
80/tcp     open        http
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

In this example you can see that the host has five open ports that are being serviced; ftp, ssh, telnet, http and X11.

inetd

There are two ways to offer TCP/IP services: by running server applications stand-alone as a daemon or by using the internet super server, **inetd**(8). inetd is a daemon which monitors a range of ports. If a client attempts to connect to a port inetd handles the connection and forwards the connection to the server software which handles that kind of connection. The advantage of this approach is that it adds an extra layer of security and it makes it easier to log incoming connections. The disadvantage is that

it is somewhat slower than using a stand-alone daemon. It is thus a good idea to run a stand-alone daemon on, for example, a heavily loaded FTP server.

You can check whether **inetd** is running on a host or not with **ps**, for example:

```
$ ps ax | grep inetd
 2845 ?        S      0:00 /usr/sbin/inetd
```

In this example inetd is running with PID (process ID) 2845. **inetd** can be configured using the `/etc/inetd.conf` file. Let's have a look at an example line from inetd.conf:

```
# File Transfer Protocol (FTP) server:
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  proftpd
```

This line specifies that inetd should accept FTP connections and pass them to **tcpd**. This may seem a bit odd, because **proftpd** normally handles FTP connections. You can also specify to use proftpd directly in `inetd.conf`, but it is a good idea to give the connection to **tcpd**. This program passes the connection to **proftpd** in turn, as specified. **tcpd** is used to monitor services and to provide host based access control.

Services can be disabled by adding the comment character (#) at the beginning of the line. It is a good idea to disable all services and enable services you need one at a time. After changing `/etc/inetd.conf` **inetd** needs to be restarted to activate the changes. This can be done by sending the HUP signal to the inetd process:

```
# ps ax | grep 'inetd'
   2845 ?        S      0:00 /usr/sbin/inetd
# kill -HUP 2845
```

If you do not need **inetd** at all, it is a good idea to remove it. If you want to keep it installed, but do not want Libranet to load it at the booting process, execute the following command as root:

```
# update-rc.d -f inetd remove
```

V. Appendices

Appendix A. Running Windows programs on Libranet

Introduction

For a few years there are a few solutions for running Windows on Linux or running Linux programs under Linux. This article article describes some of these solutions, and how well they run with Libranet GNU/Linux.

CrossOver Office

Introduction

CrossOver Office is a product based on Wine, which allows one to run some widely used Windows applications. It started as a product for running Microsoft Office under Linux, but nowadays it supports other applications as well. At the moment of writing supported applications include most Microsoft Office programs, Adobe Photoshop, Quicken and Lotus Notes. The CodeWeavers site has an application database with support ratings for many applications (gold, silver, etc.).

Facts

- **Reviewed version:** 3.0
- **Website:** http://www.codeweavers.com/

VMWare

Introduction

VMWare is a virtual machine, this means it emulates a PC on which different operating systems can be installed. Hardware access to the virtual machine is translated to instructions for the real machine. The disadvantage of this approach is that it is slower than normal hardware access, especially disk access is much slower. And be aware that the hardware is emulated, so you are bound by the limitations of the emulated hardware. Besides that VMWare is probably to expensive for most home users.

The big advantage of this approach is that one can run multiple operating systems, even simultaneously. Officially supported guest operating systems include Windows 95/98/ME/2000/XP, Novel Netware, FreeBSD and Linux.

Facts

- **Reviewed version:** 4.0.5
- **Website:** http://www.vmware.com/

Serenity Virtual Station

Introduction

Serenity Virtual Station (or SVista in short) is a virtual machine, much like VMWare, that is currently being beta-tested. It targets a wider selection of host operating systems, at the moment Linux, FreeBSD, OS/2 and Windows. Besides that it is less expensive than VMWare (the guestimated retail price for one platform is $99, and $199 for all platforms). The good support for OS/2 guests makes it a good migration tool for OS/2 users.

Facts

- **Reviewed version:** open beta 3
- **Website:** http://www.serenityvirtual.com/

Win4Lin

Introduction

Win4Lin is a special piece of software that allows you to run Windows 95/98/ME on Linux. Unlike VMWare it is not a virtual machine. It provides a special DOS environment in which Windows runs. At first Windows is loaded from CD-ROM to the hard disk, after that Windows can be installed per user. Win4Lin provides special hardware to Windows, which communicates with the Linux kernel. The Windows files are installed in the Linux filesystem (in contrast to most virtual machines, which use a hard disk image). This results in a very fast way to run Windows on GNU/Linux. On modern systems Windows boots in about five seconds. Windows can be used in a Window on in X or full-screen (with the **fwin** command). The fwin launches X with a Windows as a full screen program.

The biggest disadvantage is that Win4Lin requires kernel patches. These can be applied to the vanilla Libranet Linux kernel and requires a kernel recompile (which shouldn’t be a problem for most users, thanks to AdminMenu). People who do a lot of gaming might be disappointed by the lack of Direct3D and OpenGL support (DirectDraw is partly supported). But overall it is a very good and fast solution to be able to run a large share of Windows programs.

Facts

- **Reviewed version:** 5.1
- **Website:** http://www.netraverse.com/

Wine

Introduction

Wine is free software that will run Windows 95/98/ME programs on Linux. Wine's name is a recursive acronym: "Wine Is Not an Emulator": Wine does not emulate the Intel x86 processor. The goal of Wine is a full re-implementation of the Windows API which will make Windows unnecessary. Wine does not require Microsoft Windows, as it is a completely alternative implementation consisting of 100% Microsoft-free code, but it can optionally use native system DLLs if they are available. Well written windows programs will run as fast or faster in wine than they will in their native OS.

There are many wine packages: libwine (library) libwine-alsa (ALSA Sound Module) libwine-arts (aRts Sound Module) libwine-capi(ISDN Module) libwine-dev (Development files) libwinejack (JACK Sound Module) libwine-nas (NAS Sound Module) libwine-print (Printing Module) libwine-twain (Scanner Module) wine (Binary Emulator) wine-doc (Documentation) wine-utils (Utilities) winesetuptk (Configuration and Setup Tool). At a minimum, you will need wine and libwine. Winesetuptk is recomended for easier configuration.

The biggest disadvantages to using wine is having to do more work installing seperate configuration packages and it may not have as many recent APIs as some of the commercial programs. The big advantage is cost and custom configuration.

Facts

- **Reviewed version:** 0.0.20040505-1
- **Website:** http://winehq.org/

Cedega

Introduction

Cedega is a commercial version of WINE from Transgaming that implement parts of the DirectX API. This makes it possible to run a selection of Windows(tm) games on GNU/Linux. A slightly impaired version of Cedega is available through the Transgaming CVS tree, this version misses some crucial code to run some Windows games. Cedega is offered a subscription, during the subscription period a subscriber can download the latest version of Cedega, and subscribers can vote which games have the highest priority to be supported in future versions.

Facts

- **Reviewed version:** 3.1
- **Website:** http://www.transgaming.com/

Appendix B. GNU Free Documentation License

Version 1.2, November 2002

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as

Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant

Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the

original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.