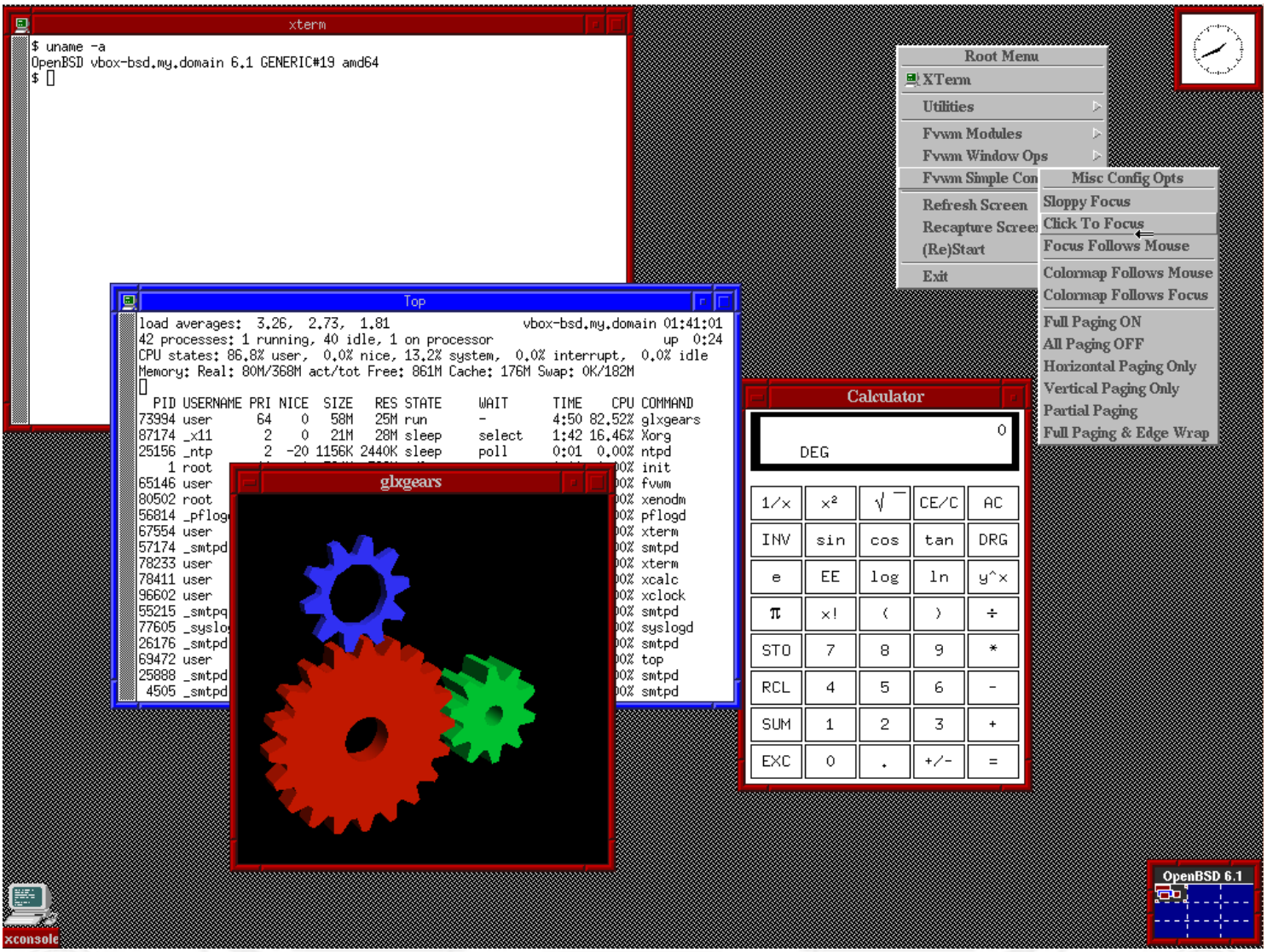
The Media Streaming Journal

September 2018



Covering Audio and Video Internet Broadcasting

Brought To You By

RADIOSOLUTION

www.radiosolution.info

publicdomainvectors.org/en/free-clipart/Vintage-microphone-vector-graphics/6111.html

The Media Streaming Journal Staff

Derek Bullard
Publication Director
info@radiosolution.info

David Childers
Editor In Chief
editor@radiosolution.info

Advertising
advertising@radiosolution.info

www.radiosolution.info

Welcome to The Media Streaming Journal

Greetings,

Open Source Software has a long history of dedicated developers that have provided world-class applications and a series of operating systems. The very core of the Internet is run on both Open Standard and Open Source Software. There is a minority group of people that have the notion that Open Source Software is somehow inferior in quality to closed source / proprietary software. This notion is far from the truth.

The Linux computer operating system provides a rock solid architecture for both server and workstation use. There are hundreds of thousands of pre-packaged applications free for the choosing that can be used in a production ready environment, as well as the ability to compile and use thousands of more applications.

System compatibility and architecture security is a heavy focus for Linux. Another salient point is that users are not required to continually upgrade hardware for each new kernel release or core system enhancement that is pushed out.

I personally use a Dell Optiplex 745 that was designed for Windows XP as my default workstation with Debian 9.0 installed. It does absolutely everything I need and so much more. I use it for Audio, Video, Graphics, Desktop Publishing, and Computer Code creation.

Broadcasting is more than just pushing a few buttons. Broadcasting requires the ability to understand the processes and methodology in use and the ability to control those processes.

Challenge yourself and learn something new, you may like it!

Namaste

David Childers

Editor In Chief

David Childers

The Grand Master of Digital Disaster

Current Member: International Association Of Internet Broadcasters

Former Member: Society of Motion Picture and Television Engineers

Published Author

Introduction To Internet Broadcasting
Amazon Publishing

Numerous Creative Commons Computer, Technical and Internet Broadcasting Guides
http://www.ScenicRadio.com/Library/BroadGuide/index.html

Newspaper Interviews

New York Times

Internet TV: Don't Touch That Mouse!
Tim Gnatek
July 1, 2004

Lagniappe - "Something Extra for Mobile"

Mobile Gets Hoaxed
Rob Holbert
Mar 16, 2016

Cited By

Five Essays on Copyright In the Digital Era
Ville Oksanen
2009

Turre Publishing
Helsinki Finland

Open Source Developer

Developed software architecture to continuously source multimedia content to Youtube Live servers.
Scenic Television – The sights and sounds of nature on the Internet.
http://www.ScenicRadio.com

Projects

Researched and developed documentation for Peercast P2P multimedia streaming project.
http://en.wikipedia.org/wiki/PeerCast

Researched and developed technical documentation for NSV / Winamp Television.
https://web.archive.org/web/20080601000000*/http://www.scvi.net

MidSummer Eve Webfest

A virtual International festival focusing on Digital art and Free Software that was coordinated by OrganicaDTM Design Studio.

Presentation and discussion regarding Internet multimedia content distribution.
https://web.archive.org/web/20061104230522/http://www.organicadtm.com/index.php?module=articles&func=display&catid=37&aid=61

LinkedIn Contact Information

http://www.linkedin.com/pub/david-childers/4/736/72a

The Media Streaming Journal

What is in this edition of the Media Streaming Journal

Linux Fundamentals
Nah Soo Hoe and Colin Charles

User Guide to Using the Linux Desktop
Paul Cobbaut

Command-line Bootcamp
Keith Bradnam

Thanks for reading and supporting The Media Streaming Journal!

Join our technical discussion on Facebook

http://www.facebook.com/groups/internetradiosupport/

Magazine cover: Database server:
https://commons.wikimedia.org/wiki/File:Openbsd61_desktop.png

RADIOSOLUTION

www.radiosolution.info

Our Mission

Let our friendly, knowledgeable staff assist you to build your project, such as an online radio station using our high end reliable video and audio streaming technologies. We want to become your partner for all your hosting needs, as well as your one stop shop for radio products such as custom DJ drops and radio ID's.

Start An Internet Radio Station

Whatever you need to start Internet radio station, we will deliver! We provide high quality Internet Radio services to make your music radio project a success. We can provide Wowza, Icecast, SHOUTcast hosting and internet radio services to hobbyists, deejays, amateurs and established professionals. No radio station client is too big or too small for Radiosolution.

Choose between complete hassle-free service packages or new features to add to start internet radio station. Benefit from customized services and the latest in internet radio technology. You will receive professional, personalized and better Internet Radio Station services than you have received up till now. If you already have an Icecast or SHOUTcast hosting provider, we can still help you transfer your radio server over to us with no hassle and at no charge.

Internet Radio Station Services

Launch your internet, digital, satellite or AM/FM radio station anywhere in the world with all of the right tools. A broadcasting specialist is on standby to help you get started with an SHOUTcast or Icecast hosting package. We have servers ready for reliable streaming in North America and Europe. Our hosting packages have all the features you need to make your radio station project a success.

If you stream live or with an Auto DJ, we can provide you with the latest in web-based Cloud technology. You will love the simple to use control panel. Discover how easy it is to manage live deejays, upload fresh music and create custom scheduled programming. You will be able to track your listeners by getting real time statistics.

Starting your own Internet radio has never been easier. Get in touch with us anytime to start your Internet radio station.

Radiosolution is a SHOUTcast hosting provider located in Quebec Canada. We also offer Icecast, Wowza and Web Hosting services. Contact us to discuss the best option available as you start internet radio station. Radiosolution can provide personalized service in English, Dutch, and French. Starting an internet radio station can be intimidating, many people want to start one, but have no idea where to start. Radiosolution will be there for you every step of the way. Everyday people are searching the internet for free SHOUTcast servers. With Radiosolution SHOUTcast hosting we will allow you to try our services for FREE. By trying our services, you can be confident that you have chosen the best radio server hosting provider. You have nothing to loose because we offer a 30 day satisfaction guarantee. What are you waiting for? Contact us now! Radiosolution offers everything you need to start internet radio station. You will not need to go anywhere else. We can create your website, market your station and help you submit your station to online directories. We also feature the voice of Derek Bullard aka Dibblebee He can create affordable commercials, DJ intros, sweepers, jingles, ids and so much more.

Relax With The Sights And Sounds Of Nature

Scenic Television

Your Window To The World

Scenic Television is an Internet television station that presents the sights and sounds of nature 24 hours a day. Let us soothe and relax you wherever you are. Savor the tropical beaches of Puerto Rico or relax at a rain forest in Costa Rica. Meditate at the Danube River in Germany, or relish the view of Lake Zurich in Switzerland. We have scenic videos from locations all over the world.

Scenic Television originates from the Gulf coast of South Alabama and broadcasts to a global audience. The television broadcast is accessible on any device with an Internet connection. Such electronic devices include desktop computers, laptops, tablets, smartphones, game platforms, and Internet-connected televisions.

http://www.scenicradio.com

all-free-download.com/free-vector/download/magnifying_glass_clip_art_23181.html

We Are Your Information Resource

Are you looking for specialized data?

Are you swamped with information overload?

Do you need help finding the right information?

We Can Help You
Find The Information
That You Need

Our experienced data research analysts can wade through the vast information wasteland and find the information that you need.

We can save you both time and money.

We can streamline data requirement planning.

We can provide business critical information acquisition.

Contact us today

info@radiosolution.info

- Linux Fundamentals

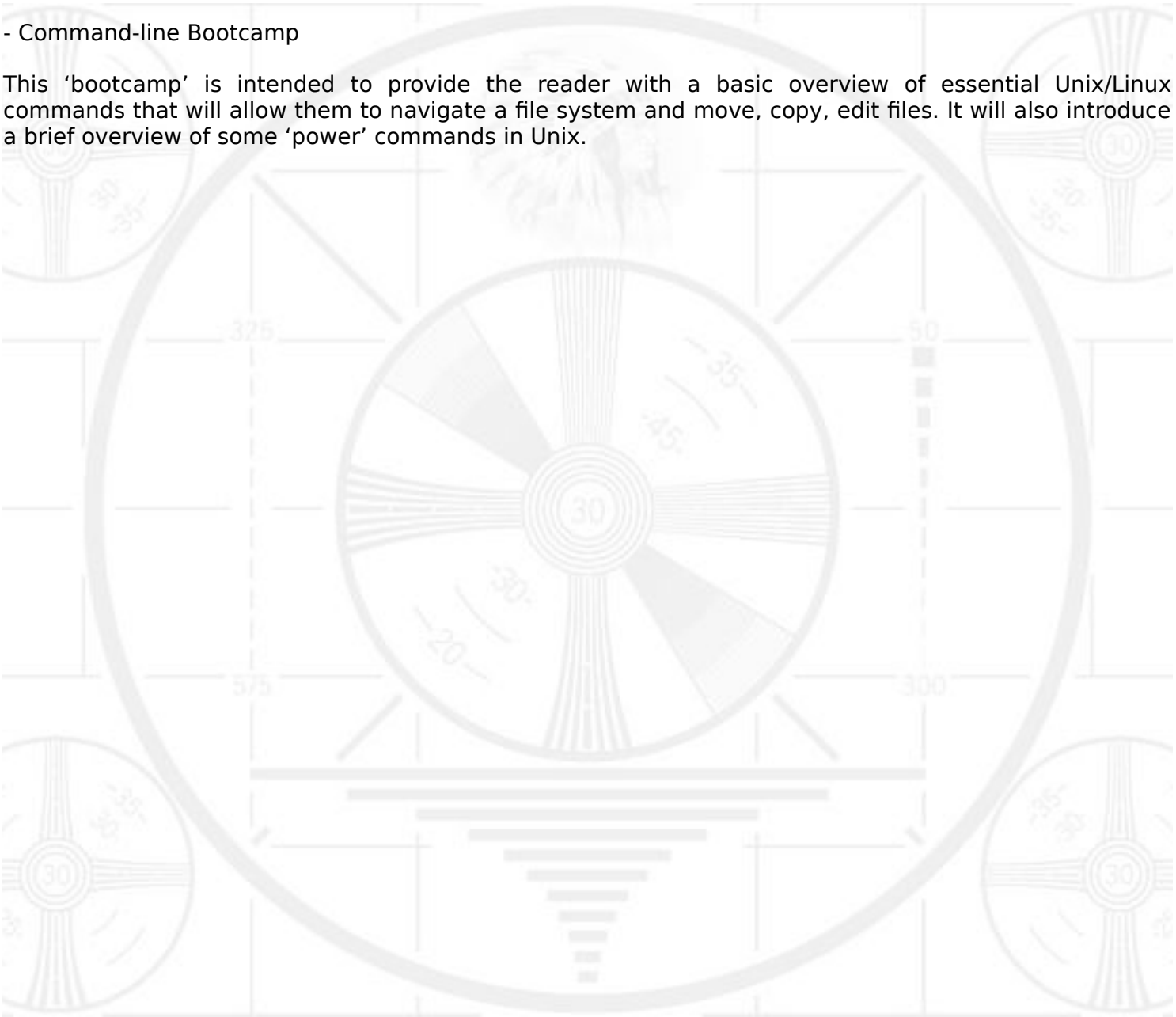
This book is aimed at novice Linux system administrators (and might be interesting and useful for home users that want to know a bit more about their Linux system).

- User Guide to Using the Linux Desktop

This user guide is meant as an introductory guide for a user to use a modern personal computer (PC) running the Linux operating system. The main aim is to provide a self-learning guide on how to use a modern Linux desktop system.

- Command-line Bootcamp

This ‘bootcamp’ is intended to provide the reader with a basic overview of essential Unix/Linux commands that will allow them to navigate a file system and move, copy, edit files. It will also introduce a brief overview of some ‘power’ commands in Unix.



Linux Fundamentals

Paul Cobbaut

Linux Fundamentals

Paul Cobbaut

Publication date 2015-05-24 CEST

Abstract

This book is meant to be used in an instructor-led training. For self-study, the intent is to read this book next to a working Linux computer so you can immediately do every subject, practicing each command.

This book is aimed at novice Linux system administrators (and might be interesting and useful for home users that want to know a bit more about their Linux system). However, this book is not meant as an introduction to Linux desktop applications like text editors, browsers, mail clients, multimedia or office applications.

More information and free .pdf available at **http://linux-training.be** .

Feel free to contact the author:

- Paul Cobbaut: paul.cobbaut@gmail.com, http://www.linkedin.com/in/cobbaut

Contributors to the Linux Training project are:

- Serge van Ginderachter: serge@ginsys.eu, build scripts and infrastructure setup
- Ywein Van den Brande: ywein@crealaw.eu, license and legal sections
- Hendrik De Vloed: hendrik.devloed@ugent.be, buildheader.pl script

We'd also like to thank our reviewers:

- Wouter Verhelst: wo@uter.be, http://grep.be
- Geert Goossens: mail.goossens.geert@gmail.com, http://www.linkedin.com/in/geertgoossens
- Elie De Brauwer: elie@de-brauwer.be, http://www.de-brauwer.be
- Christophe Vandeplas: christophe@vandeplas.com, http://christophe.vandeplas.com
- Bert Desmet: bert@devnox.be, http://blog.bdesmet.be
- Rich Yonts: richyonts@gmail.com,

Table of Contents

List of Tables

Part I. introduction to Linux

Table of Contents

Chapter 1. Linux history

This chapter briefly tells the history of Unix and where Linux fits in.

If you are eager to start working with Linux without this blah, blah, blah over history, distributions, and licensing then jump straight to **Part II - Chapter 8. Working with Directories** page 73.

1.1. 1969

All modern operating systems have their roots in 1969 when **Dennis Ritchie** and **Ken Thompson** developed the C language and the **Unix** operating system at AT&T Bell Labs. They shared their source code (yes, there was open source back in the Seventies) with the rest of the world, including the hippies in Berkeley California. By 1975, when AT&T started selling Unix commercially, about half of the source code was written by others. The hippies were not happy that a commercial company sold software that they had written; the resulting (legal) battle ended in there being two versions of **Unix**: the official AT&T Unix, and the free **BSD** Unix.

Development of BSD descendants like FreeBSD, OpenBSD, NetBSD, DragonFly BSD and PC-BSD is still active today.

```
https://en.wikipedia.org/wiki/Dennis_Ritchie
https://en.wikipedia.org/wiki/Ken_Thompson
https://en.wikipedia.org/wiki/BSD
https://en.wikipedia.org/wiki/Comparison_of_BSD_operating_systems
```

1.2. 1980s

In the Eighties many companies started developing their own Unix: IBM created AIX, Sun SunOS (later Solaris), HP HP-UX and about a dozen other companies did the same. The result was a mess of Unix dialects and a dozen different ways to do the same thing. And here is the first real root of **Linux**, when **Richard Stallman** aimed to end this era of Unix separation and everybody re-inventing the wheel by starting the **GNU** project (GNU is Not Unix). His goal was to make an operating system that was freely available to everyone, and where everyone could work together (like in the Seventies). Many of the command line tools that you use today on **Linux** are GNU tools.

```
https://en.wikipedia.org/wiki/Richard_Stallman
https://en.wikipedia.org/wiki/IBM_AIX
https://en.wikipedia.org/wiki/HP-UX
```

1.3. 1990s

The Nineties started with **Linus Torvalds**, a Swedish speaking Finnish student, buying a 386 computer and writing a brand new POSIX compliant kernel. He put the source code online, thinking it would never support anything but 386 hardware. Many people embraced the combination of this kernel with the GNU tools, and the rest, as they say, is history.

```
http://en.wikipedia.org/wiki/Linus_Torvalds
https://en.wikipedia.org/wiki/History_of_Linux
https://en.wikipedia.org/wiki/Linux
https://lwn.net
http://www.levenez.com/unix/   (a huge Unix history poster)
```

1.4. 2015

Today more than 97 percent of the world's supercomputers (including the complete top 10), more than 80 percent of all smartphones, many millions of desktop computers, around 70 percent of all web servers, a large chunk of tablet computers, and several appliances (dvd-players, washing machines, dsl modems, routers, self-driving cars, space station laptops...) run **Linux**. Linux is by far the most commonly used operating system in the world.

Linux kernel version 4.0 was released in April 2015. Its source code grew by several hundred thousand lines (compared to version 3.19 from February 2015) thanks to contributions of thousands of developers paid by hundreds of commercial companies including Red Hat, Intel, Samsung, Broadcom, Texas Instruments, IBM, Novell, Qualcomm, Nokia, Oracle, Google, AMD and even Microsoft (and many more).

```
http://kernelnewbies.org/DevelopmentStatistics
http://kernel.org
http://www.top500.org
```

Chapter 2. distributions

This chapter gives a short overview of current Linux distributions.

A Linux **distribution** is a collection of (usually open source) software on top of a Linux kernel. A distribution (or short, distro) can bundle server software, system management tools, documentation and many desktop applications in a **central secure software repository**. A distro aims to provide a common look and feel, secure and easy software management and often a specific operational purpose.

Let's take a look at some popular distributions.

2.1. Red Hat

Red Hat is a billion dollar commercial Linux company that puts a lot of effort in developing Linux. They have hundreds of Linux specialists and are known for their excellent support. They give their products (Red Hat Enterprise Linux and Fedora) away for free. While **Red Hat Enterprise Linux** (RHEL) is well tested before release and supported for up to seven years after release, **Fedora** is a distro with faster updates but without support.

2.2. Ubuntu

Canonical started sending out free compact discs with **Ubuntu** Linux in 2004 and quickly became popular for home users (many switching from Microsoft Windows). Canonical wants Ubuntu to be an easy to use graphical Linux desktop without need to ever see a command line. Of course they also want to make a profit by selling support for Ubuntu.

2.3. Debian

There is no company behind **Debian**. Instead there are thousands of well organised developers that elect a **Debian Project Leader** every two years. Debian is seen as one of the most stable Linux distributions. It is also the basis of every release of Ubuntu. Debian comes in three versions: stable, testing and unstable. Every Debian release is named after a character in the movie Toy Story.

2.4. Other

Distributions like CentOS, Oracle Enterprise Linux and Scientific Linux are based on Red Hat Enterprise Linux and share many of the same principles, directories and system administration techniques. **Linux Mint**, Edubuntu and many other *buntu named distributions are based on Ubuntu and thus share a lot with Debian. There are hundreds of other Linux distributions.

2.5. Which to choose ?

Below are some very personal opinions on some of the most popular Linux Distributions. Keep in mind that any of the below Linux distributions can be a stable server and a nice graphical desktop client.

Table 2.1. choosing a Linux distro

distribution name	reason(s) for using
Red Hat Enterprise (RHEL)	You are a manager and you want a good support contract.
CentOS	You want Red Hat without the support contract from Red Hat.
Fedora	You want Red Hat on your laptop/desktop.
Linux Mint	You want a personal graphical desktop to play movies, music and games.
Debian	My personal favorite for servers, laptops, and any other device.
Ubuntu	Very popular, based on Debian, not my favorite.
Kali	You want a pointy-clicky hacking interface.
others	Advanced users may prefer Arch, Gentoo, OpenSUSE, Scientific, ...

When you are new to Linux in 2015, go for the latest Mint or Fedora. If you only want to practice the Linux command line then install one Debian server and/or one CentOS server (without graphical interface).

Here are some links to help you choose:

```
distrowatch.com
redhat.com
centos.org
debian.org
www.linuxmint.com
ubuntu.com
```

Chapter 3. licensing

This chapter briefly explains the different licenses used for distributing operating systems software.

Many thanks go to **Ywein Van den Brande** for writing most of this chapter.

Ywein is an attorney at law, co-author of **The International FOSS Law Book** and author of **Praktijkboek Informaticarecht** (in Dutch).


```
http://ifosslawbook.org
http://www.crealaw.eu
```

3.1. about software licenses

There are two predominant software paradigms: **Free and Open Source Software** (FOSS) and **proprietary software**. The criteria for differentiation between these two approaches is based on control over the software. With **proprietary software**, control tends to lie more with the vendor, while with **Free and Open Source Software** it tends to be more weighted towards the end user. But even though the paradigms differ, they use the same **copyright laws** to reach and enforce their goals. From a legal perspective, **Free and Open Source Software** can be considered as software to which users generally receive more rights via their license agreement than they would have with a **proprietary software license**, yet the underlying license mechanisms are the same.

Legal theory states that the author of FOSS, contrary to the author of **public domain** software, has in no way whatsoever given up his rights on his work. FOSS supports on the rights of the author (the **copyright**) to impose FOSS license conditions. The FOSS license conditions need to be respected by the user in the same way as proprietary license conditions. Always check your license carefully before you use third party software.

Examples of proprietary software are **AIX** from IBM, **HP-UX** from HP and **Oracle Database 11g**. You are not authorised to install or use this software without paying a licensing fee. You are not authorised to distribute copies and you are not authorised to modify the closed source code.

3.2. public domain software and freeware

Software that is original in the sense that it is an intellectual creation of the author benefits **copyright** protection. Non-original software does not come into consideration for **copyright** protection and can, in principle, be used freely.

Public domain software is considered as software to which the author has given up all rights and on which nobody is able to enforce any rights. This software can be used, reproduced or executed freely, without permission or the payment of a fee. Public domain software can in certain cases even be presented by third parties as own work, and by modifying the original work, third parties can take certain versions of the public domain software out of the public domain again.

Freeware is not public domain software or FOSS. It is proprietary software that you can use without paying a license cost. However, the often strict license terms need to be respected.

Examples of freeware are **Adobe Reader**, **Skype** and **Command and Conquer: Tiberian Sun** (this game was sold as proprietary in 1999 and is since 2011 available as freeware).

3.3. Free Software or Open Source Software

Both the **Free Software** (translates to **vrije software** in Dutch and to **Logiciel Libre** in French) and the **Open Source Software** movement largely pursue similar goals and endorse similar software licenses. But historically, there has been some perception of differentiation due to different emphases. Where the **Free Software** movement focuses on the rights (the

four freedoms) which Free Software provides to its users, the **Open Source Software** movement points to its Open Source Definition and the advantages of peer-to-peer software development.

Recently, the term free and open source software or FOSS has arisen as a neutral alternative. A lesser-used variant is free/libre/open source software (FLOSS), which uses **libre** to clarify the meaning of free as in **freedom** rather than as in **at no charge**.

Examples of **free software** are **gcc**, **MySQL** and **gimp**.

Detailed information about the **four freedoms** can be found here:

```
http://www.gnu.org/philosophy/free-sw.html
```

The **open source definition** can be found at:

```
http://www.opensource.org/docs/osd
```

The above definition is based on the **Debian Free Software Guidelines** available here:

```
http://www.debian.org/social_contract#guidelines
```

3.4. GNU General Public License

More and more software is being released under the **GNU GPL** (in 2006 Java was released under the GPL). This license (v2 and v3) is the main license endorsed by the Free Software Foundation. It's main characteristic is the **copyleft** principle. This means that everyone in the chain of consecutive users, in return for the right of use that is assigned, needs to distribute the improvements he makes to the software and his derivative works under the same conditions to other users, if he chooses to distribute such improvements or derivative works. In other words, software which incorporates GNU GPL software, needs to be distributed in turn as GNU GPL software (or compatible, see below). It is not possible to incorporate copyright protected parts of GNU GPL software in a proprietary licensed work. The GPL has been upheld in court.

3.5. using GPLv3 software

You can use **GPLv3 software** almost without any conditions. If you solely run the software you even don't have to accept the terms of the GPLv3. However, any other use - such as modifying or distributing the software - implies acceptance.

In case you use the software internally (including over a network), you may modify the software without being obliged to distribute your modification. You may hire third parties to work on the software exclusively for you and under your direction and control. But if you modify the software and use it otherwise than merely internally, this will be considered as distribution. You must distribute your modifications under GPLv3 (the copyleft principle). Several more obligations apply if you distribute GPLv3 software. Check the GPLv3 license carefully.

You create output with GPLv3 software: The GPLv3 does not automatically apply to the output.

3.6. BSD license

There are several versions of the original Berkeley Distribution License. The most common one is the 3-clause license ("New BSD License" or "Modified BSD License").

This is a permissive free software license. The license places minimal restrictions on how the software can be redistributed. This is in contrast to copyleft licenses such as the GPLv. 3 discussed above, which have a copyleft mechanism.

This difference is of less importance when you merely use the software, but kicks in when you start redistributing verbatim copies of the software or your own modified versions.

3.7. other licenses

FOSS or not, there are many kind of licenses on software. You should read and understand them before using any software.

3.8. combination of software licenses

When you use several sources or wishes to redistribute your software under a different license, you need to verify whether all licenses are compatible. Some FOSS licenses (such as BSD) are compatible with proprietary licenses, but most are not. If you detect a license incompatibility, you must contact the author to negotiate different license conditions or refrain from using the incompatible software.

Part II. installing Linux

Table of Contents

Chapter 4. installing Debian 8

This module is a step by step demonstration of an actual installation of **Debian 8** (also known as **Jessie**).

We start by downloading an image from the internet and install **Debian 8** as a virtual machine in **Virtualbox**. We will also do some basic configuration of this new machine like setting an **ip address** and fixing a **hostname**.

This procedure should be very similar for other versions of **Debian**, and also for distributions like **Linux Mint**, **xubuntu/ubuntu/kubuntu** or **Mepis**. This procedure can also be helpful if you are using another virtualization solution.

Go to the next chapter if you want to install **CentOS, Fedora, Red Hat Enterprise Linux, ...**.

4.1. Debian

Debian is one of the oldest Linux distributions. I use Debian myself on almost every computer that I own (including **raspbian** on the **Raspberry Pi**).

Debian comes in **releases** named after characters in the movie **Toy Story**. The **Jessie** release contains about 36000 packages.

Table 4.1. Debian releases

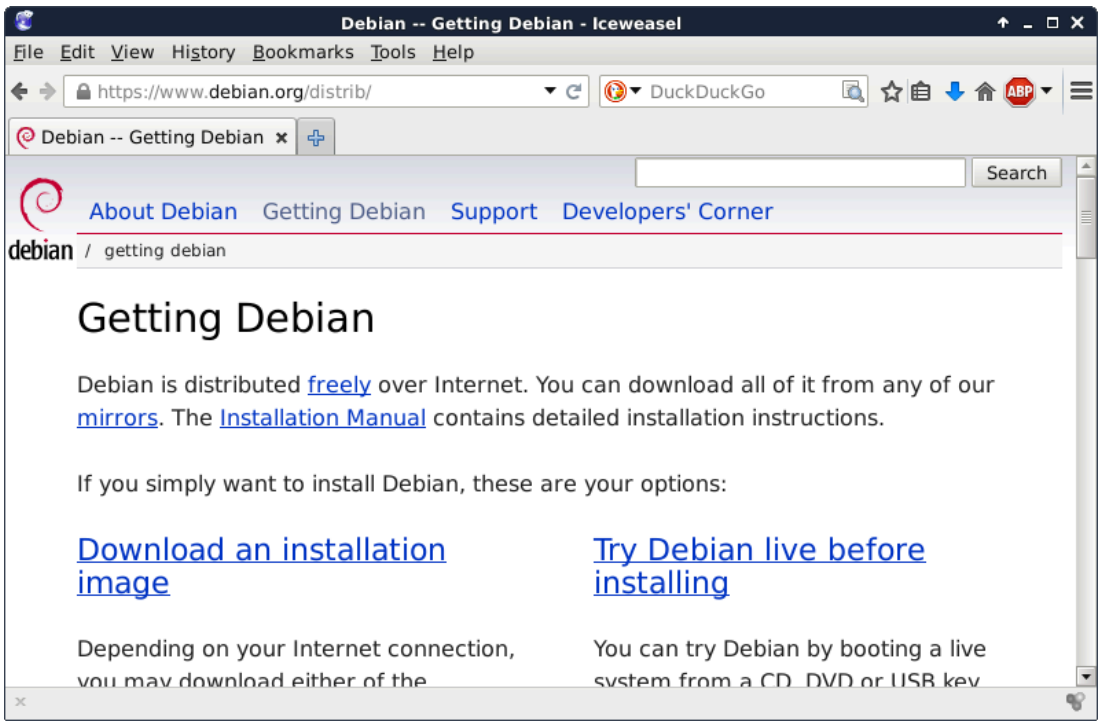
name	number	year
Woody	3.0	2002
Sarge	3.1	2005
Etch	4.0	2007
Lenny	5.0	2009
Squeeze	6.0	2011
Wheezy	7	2013
Jessie	8	2015

There is never a fixed date for the next **Debian** release. The next version is released when it is ready.

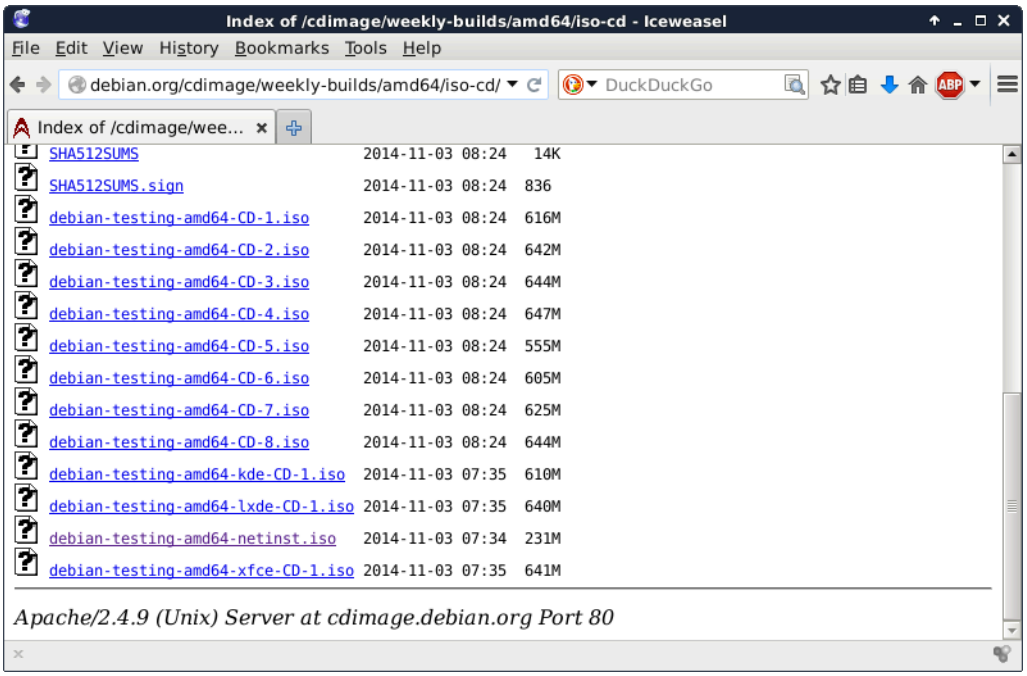
4.2. Downloading

All these screenshots were made in November 2014, which means **Debian 8** was still in 'testing' (but in 'freeze', so there will be no major changes when it is released).

Download Debian here:

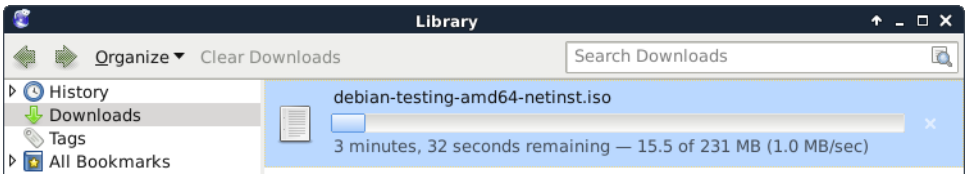


After a couple of clicks on that website, I ended up downloading **Debian 8** (testing) here. It should be only one click once **Debian 8** is released (somewhere in 2015).



You have many other options to download and install **Debian**. We will discuss them much later.

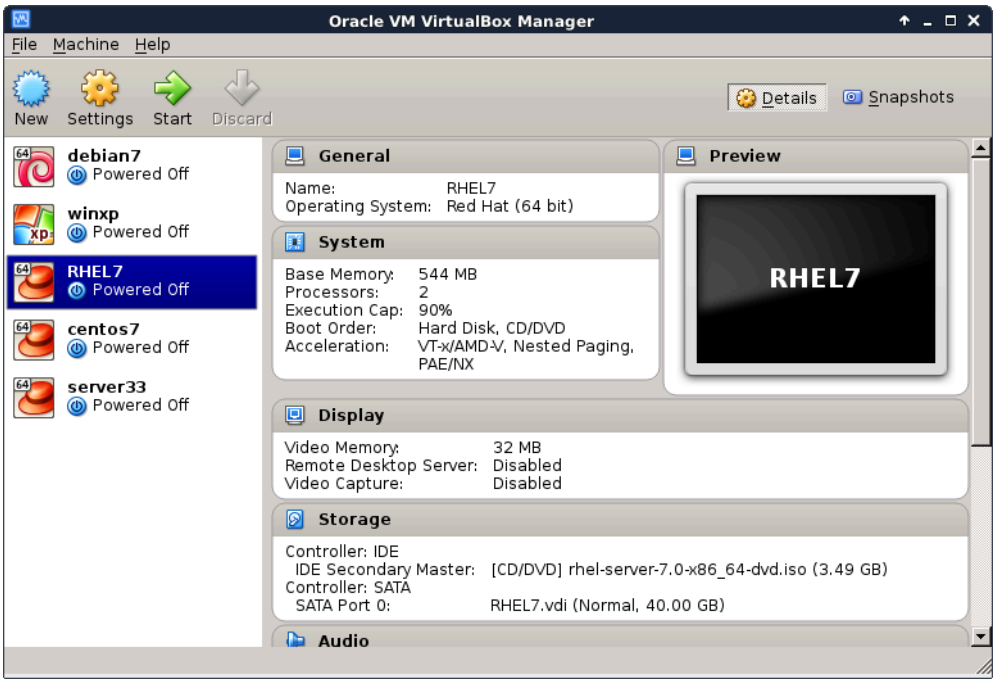
This small screenshot shows the downloading of a **netinst** .iso file. Most of the software will be downloaded during the installation. This also means that you will have the most recent version of all packages when the install is finished.



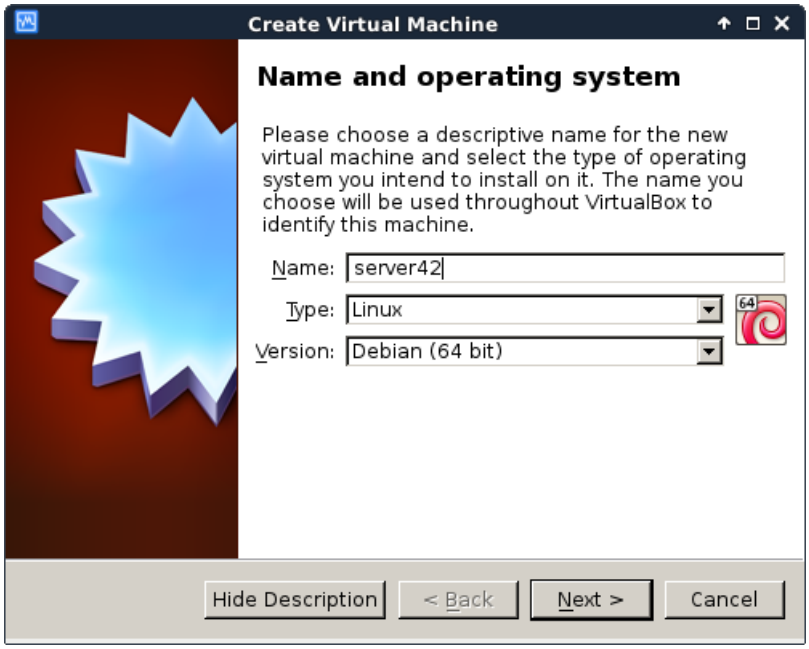
I already have Debian 8 installed on my laptop (hence the **paul@debian8** prompt). Anyway, this is the downloaded file just before starting the installation.

```
paul@debian8:~$ ls -hl debian-testing-amd64-netinst.iso
-rw-r--r-- 1 paul paul 231M Nov 10 17:59 debian-testing-amd64-netinst.iso
```

Create a new virtualbox machine (I already have five, you might have zero for now). Click the **New** button to start a wizard that will help you create a virtual machine.

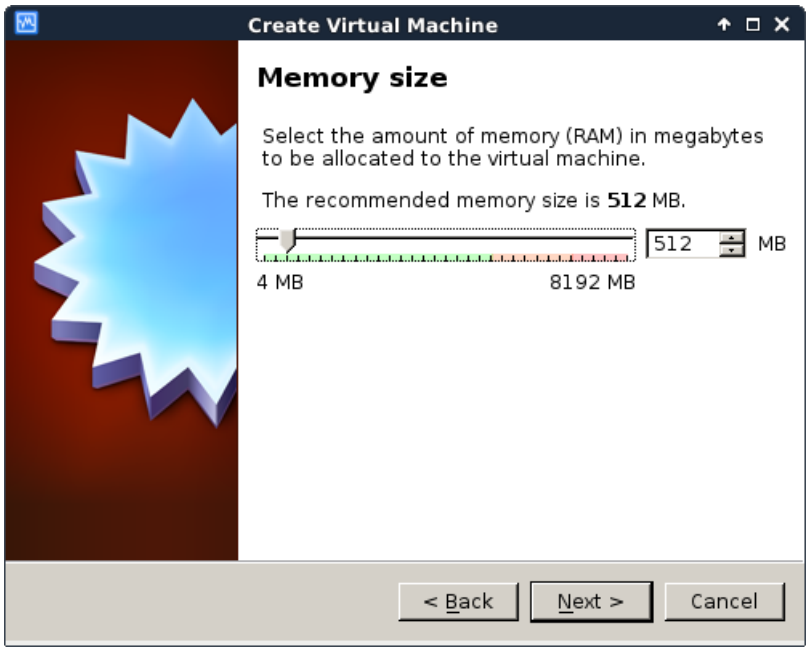


The machine needs a name, this screenshot shows that I named it **server42**.

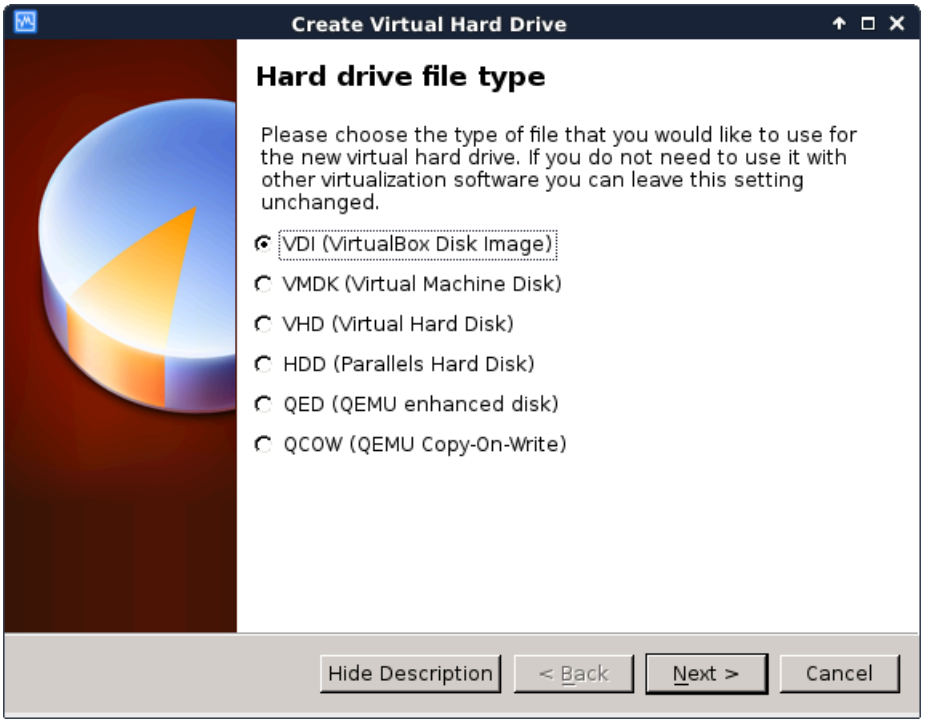


Most of the defaults in Virtualbox are ok.

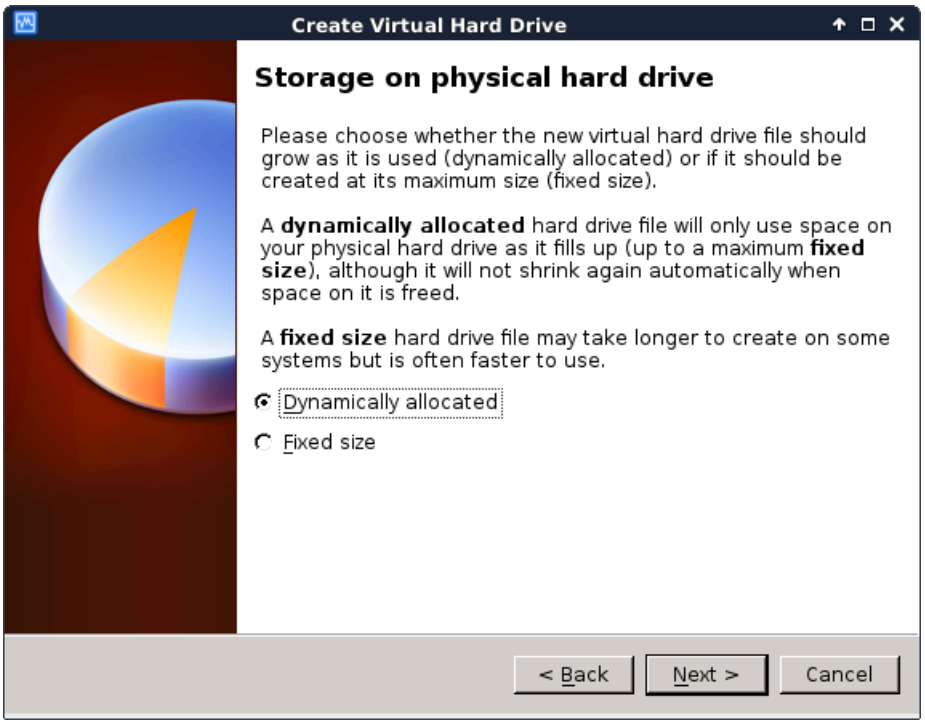
512MB of RAM is enough to practice all the topics in this book.



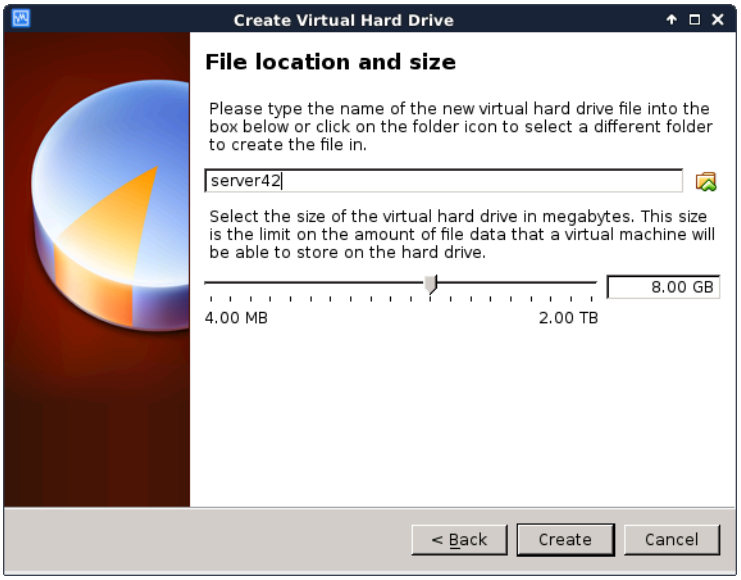
We do not care about the virtual disk format.



Choosing **dynamically allocated** will save you some disk space (for a small performance hit).

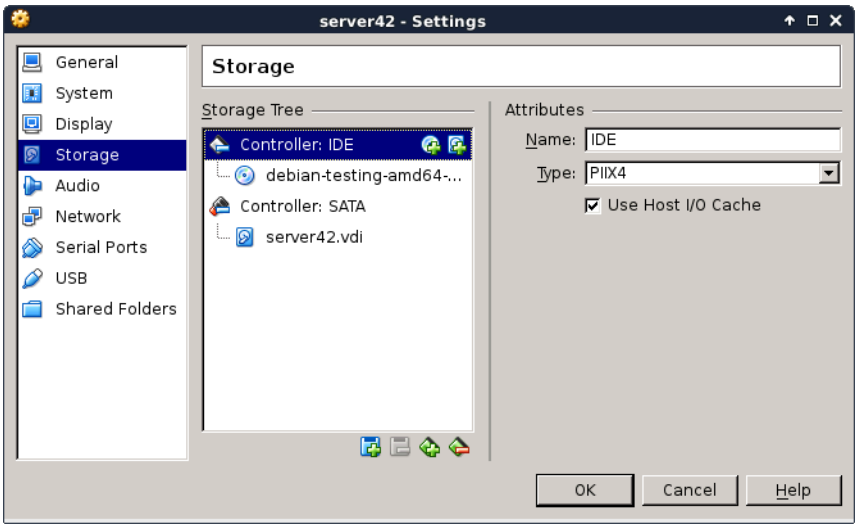


8GB should be plenty for learning about Linux servers.



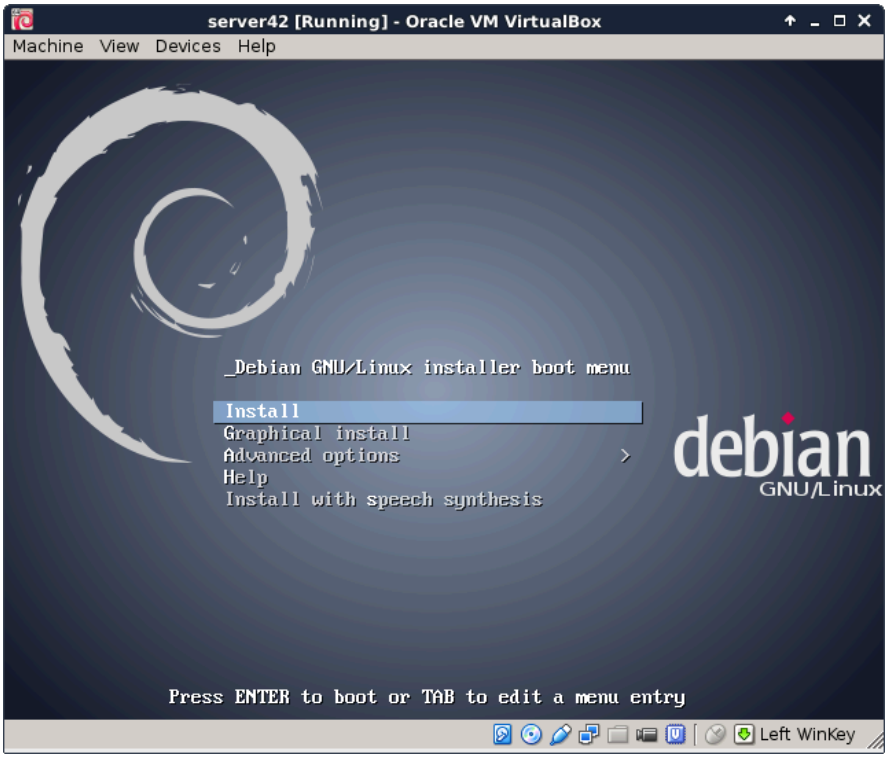
This finishes the wizard. You virtual machine is almost ready to begin the installation.

First, make sure that you attach the downloaded .iso image to the virtual CD drive. (by opening **Settings**, **Storage** followed by a mouse click on the round CD icon)

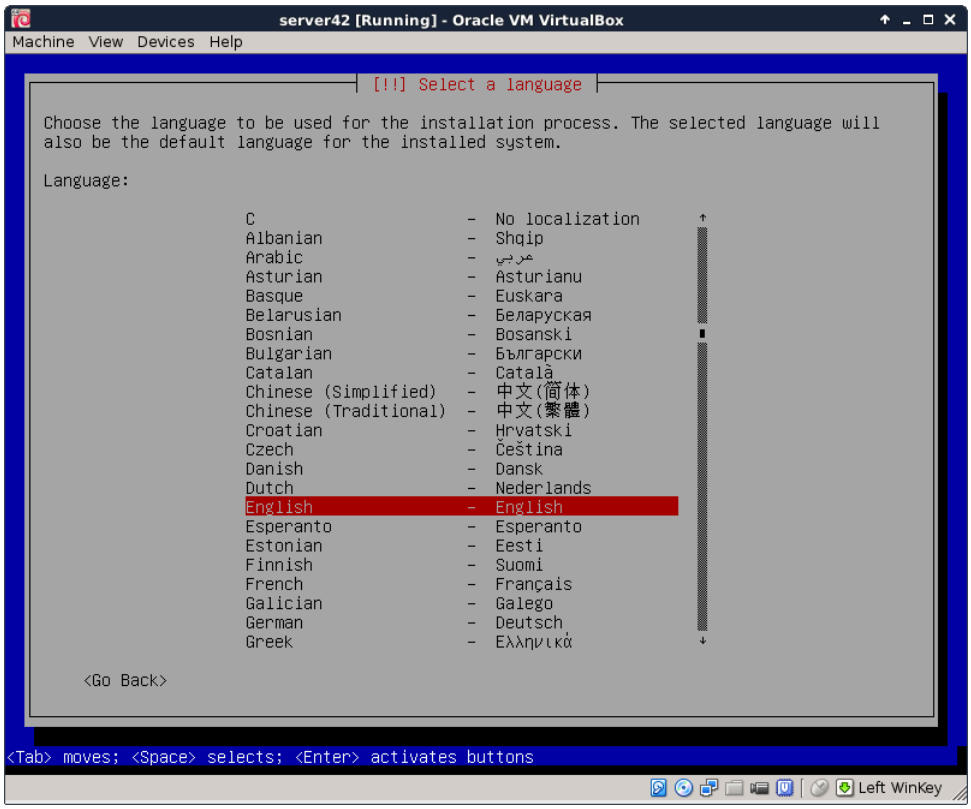


Personally I also disable sound and usb, because I never use these features. I also remove the floppy disk and use a PS/2 mouse pointer. This is probably not very important, but I like the idea that it saves some resources.

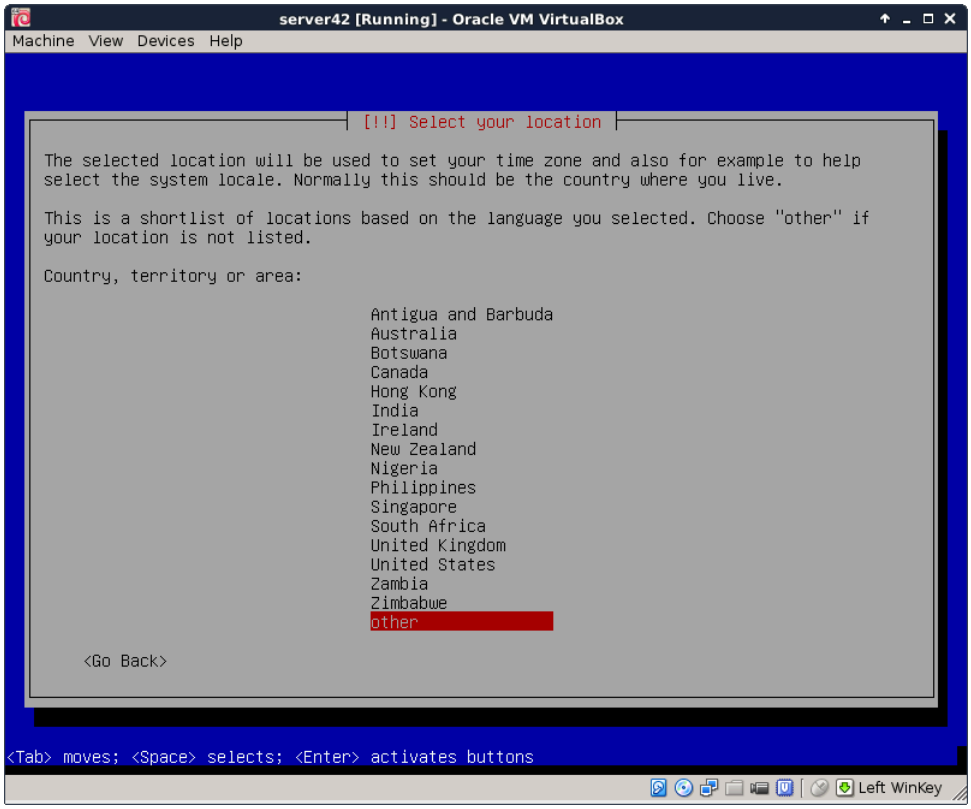
Now boot the virtual machine and begin the actual installation. After a couple of seconds you should see a screen similar to this. Choose **Install** to begin the installation of Debian.



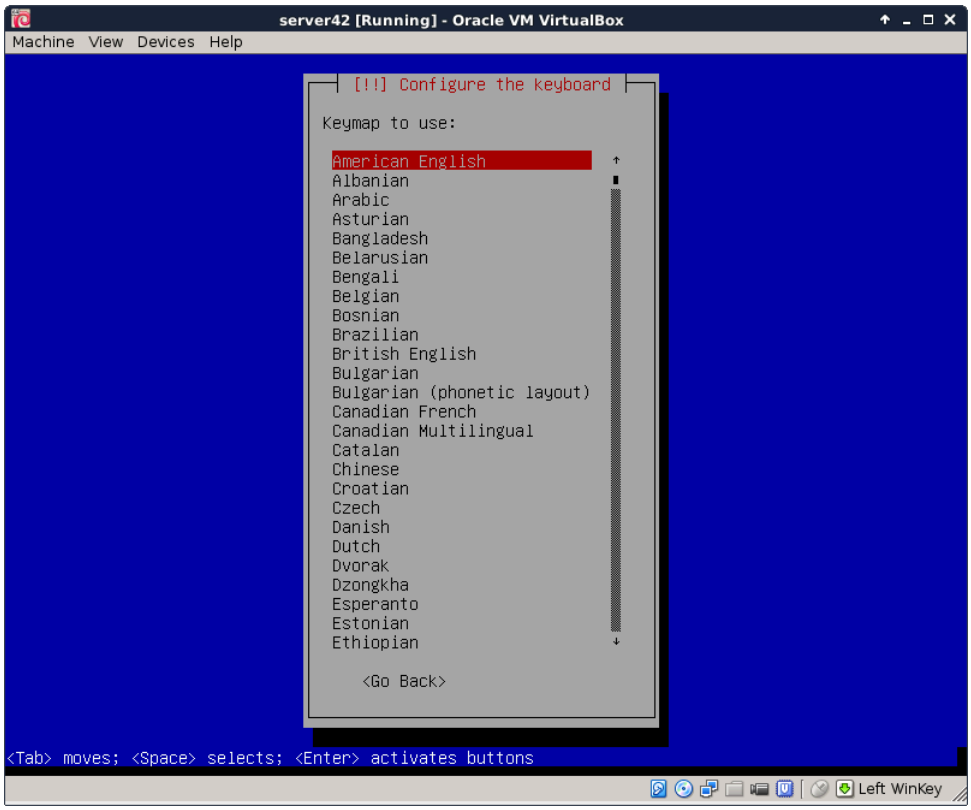
First select the language you want to use.



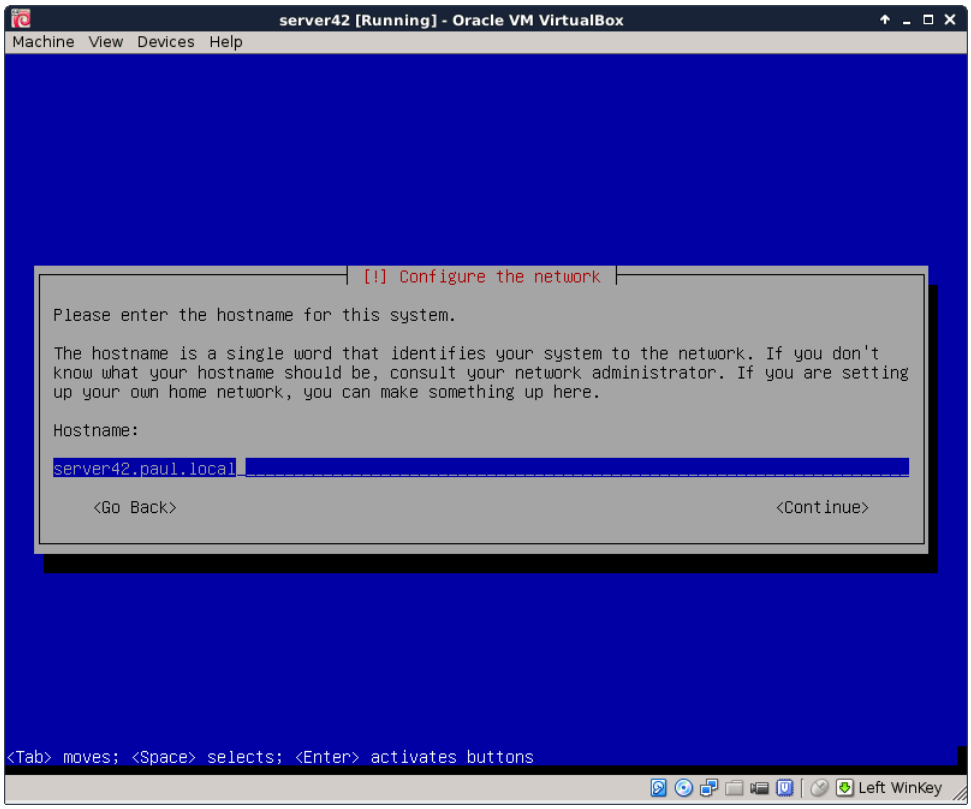
Choose your country. This information will be used to suggest a download mirror.



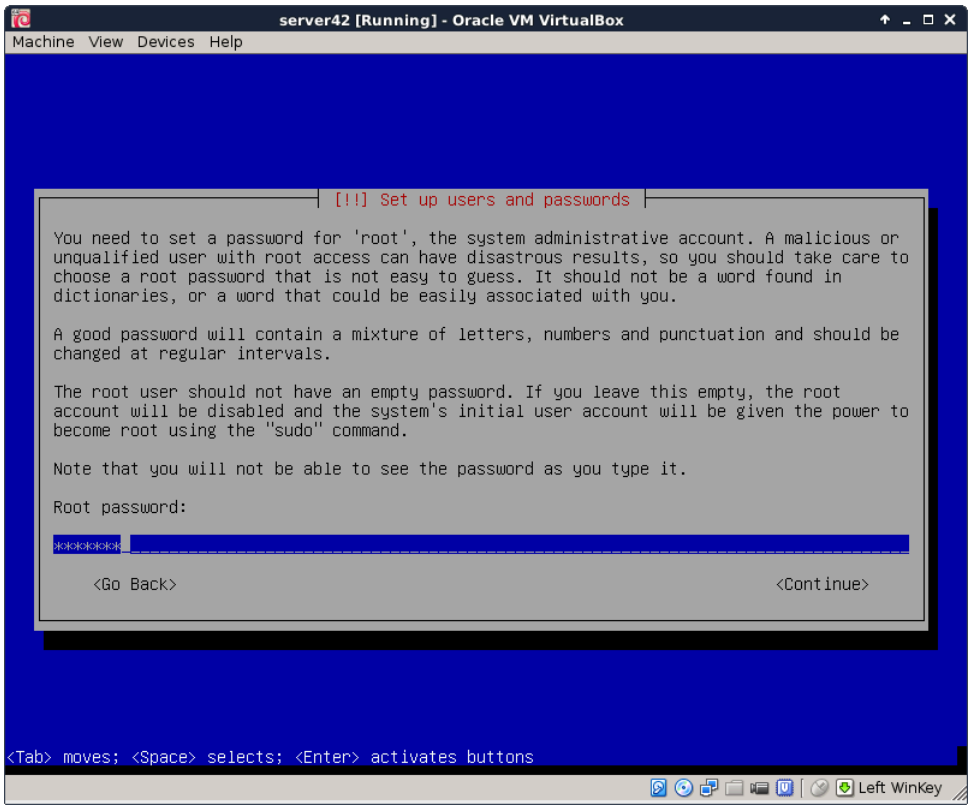
Choose the correct keyboard. On servers this is of no importance since most servers are remotely managed via **ssh**.



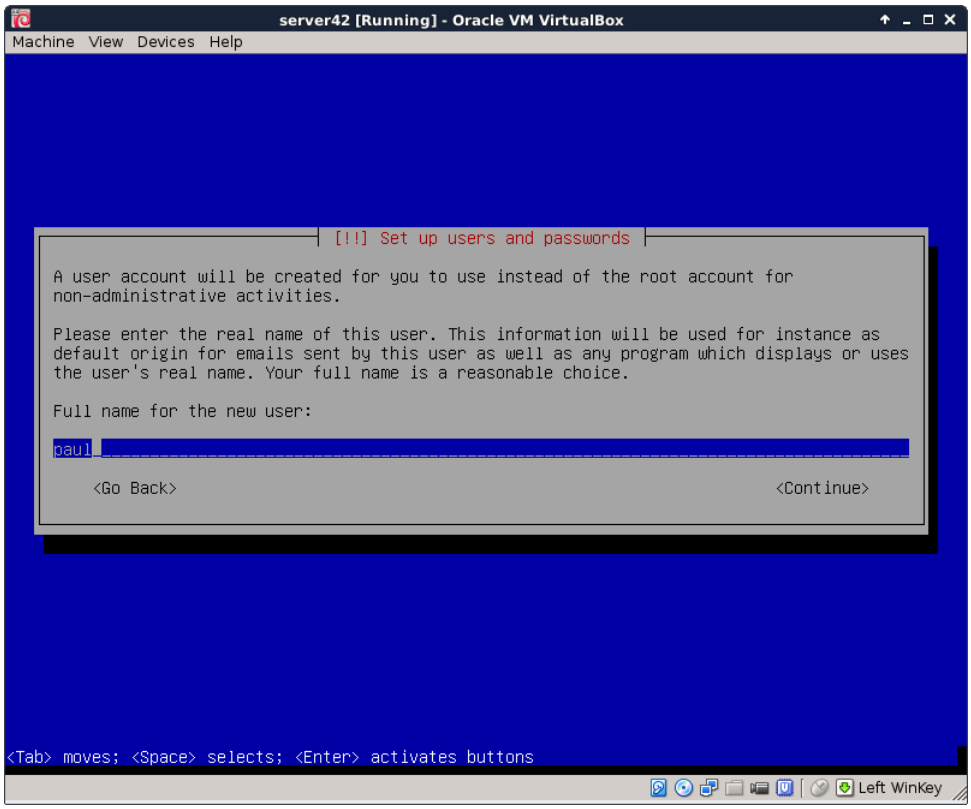
Enter a **hostname** (with **fqdn** to set a **dnsdomainname**).



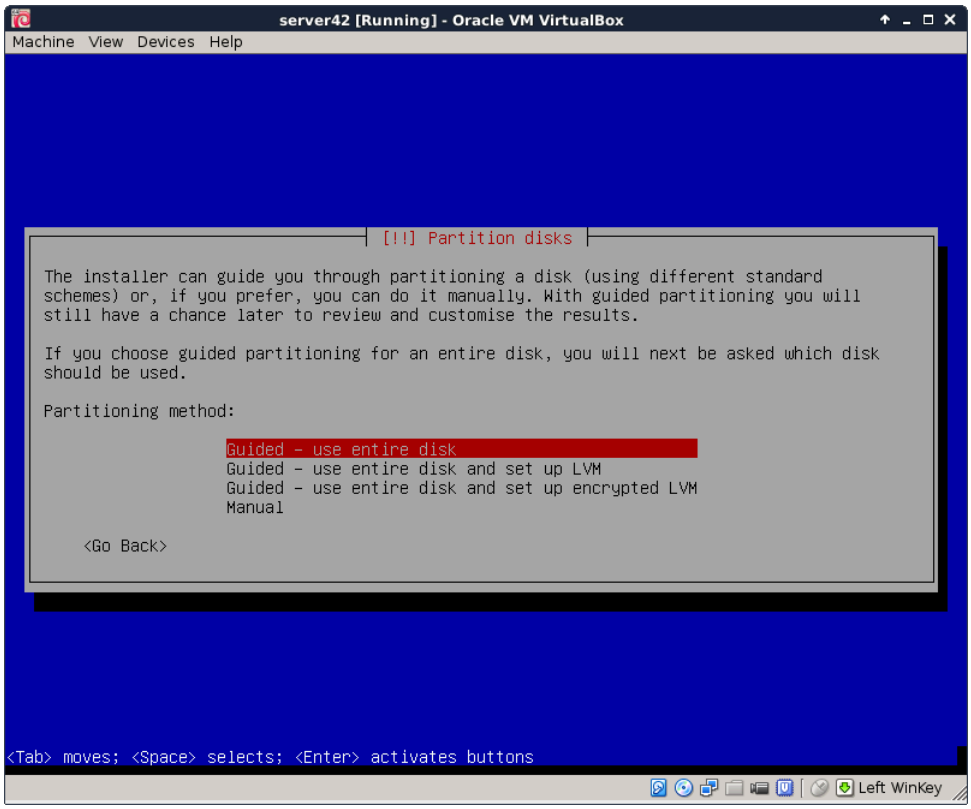
Give the **root** user a password. Remember this password (or use **hunter2**).



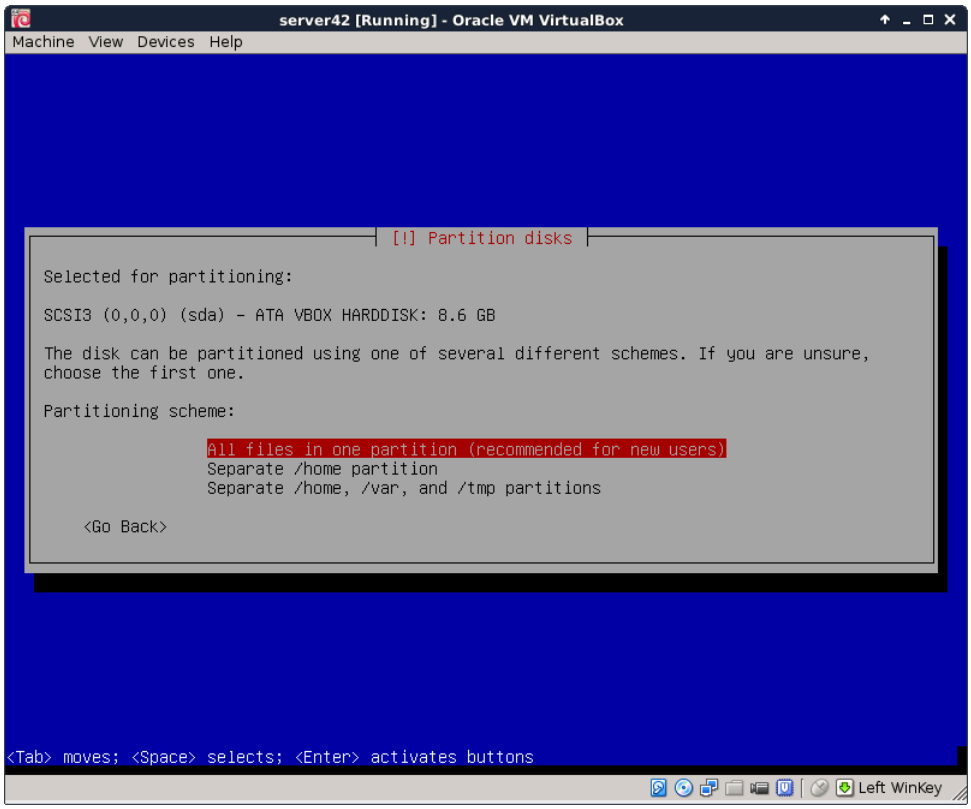
It is adviced to also create a normal user account. I don't give my full name, Debian 8 accepts an identical username and full name **paul**.



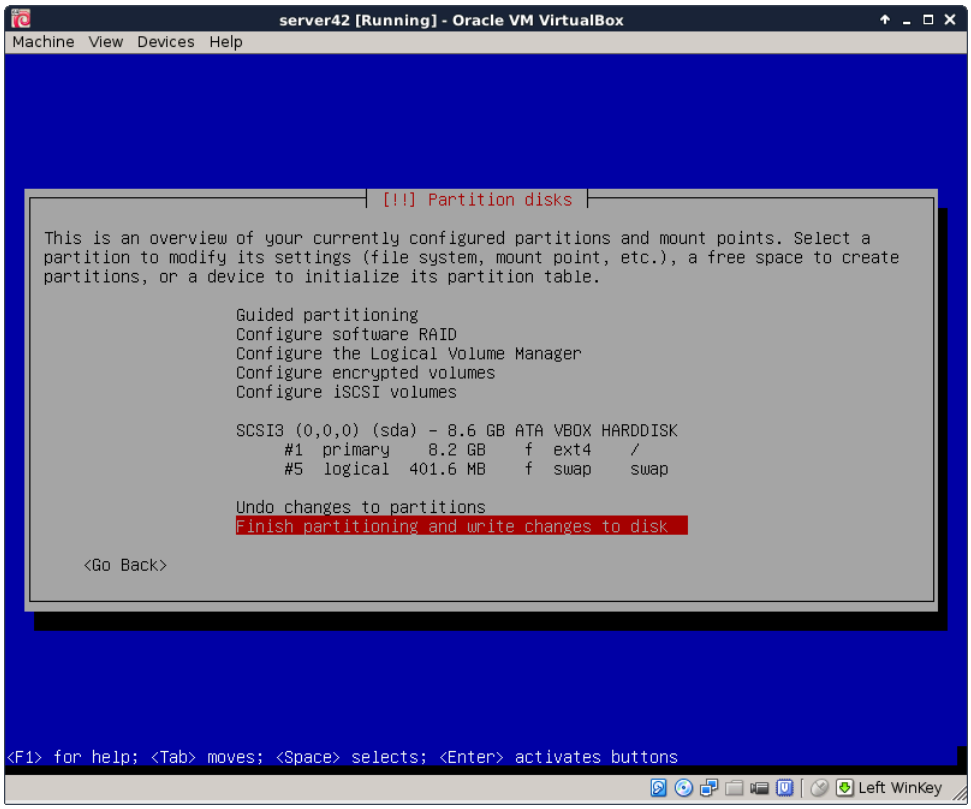
The **use entire disk** refers to the **virtual disk** that you created before in **Virtualbox**..



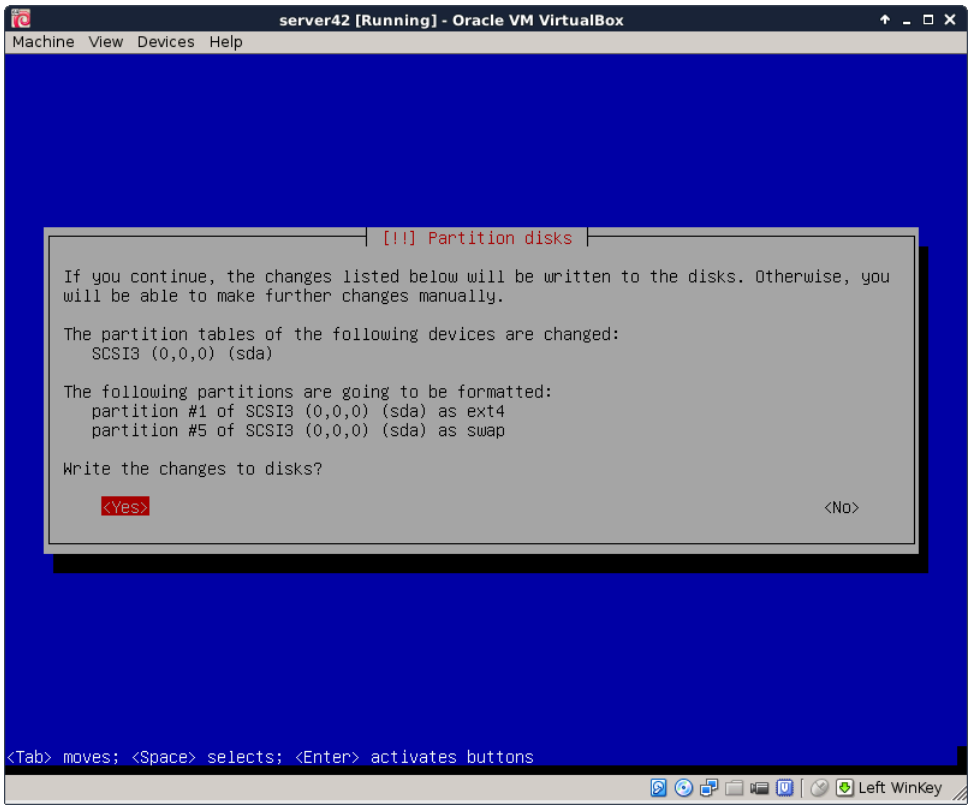
Again the default is probably what you want. Only change partitioning if you really know what you are doing.



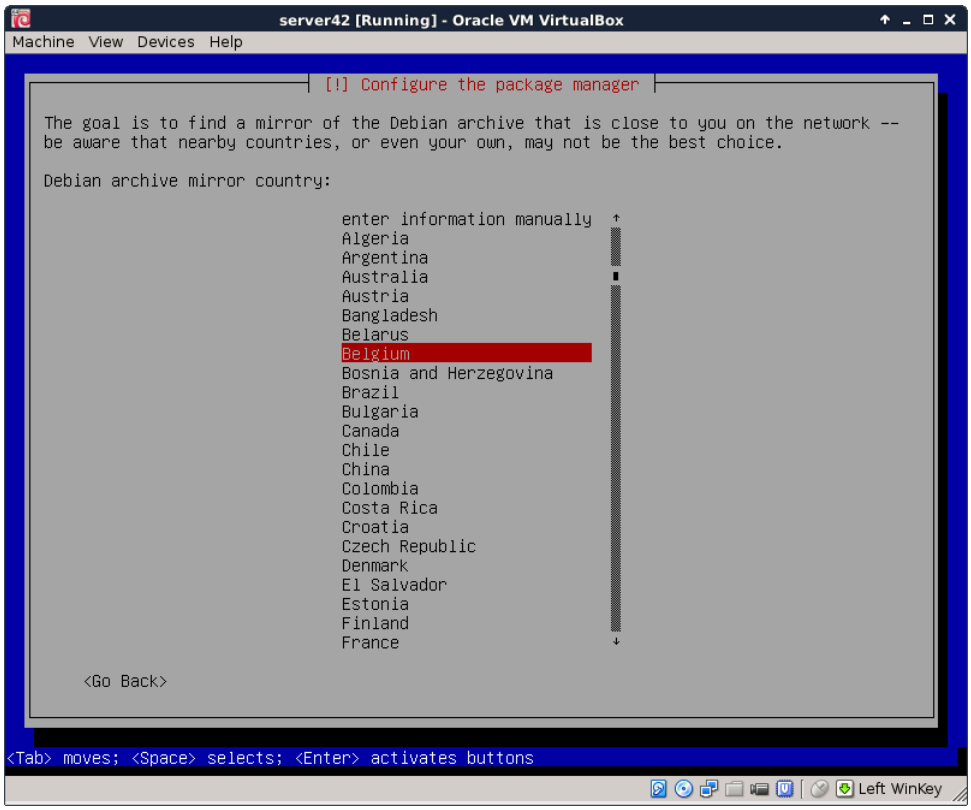
Accept the partition layout (again only change if you really know what you are doing).



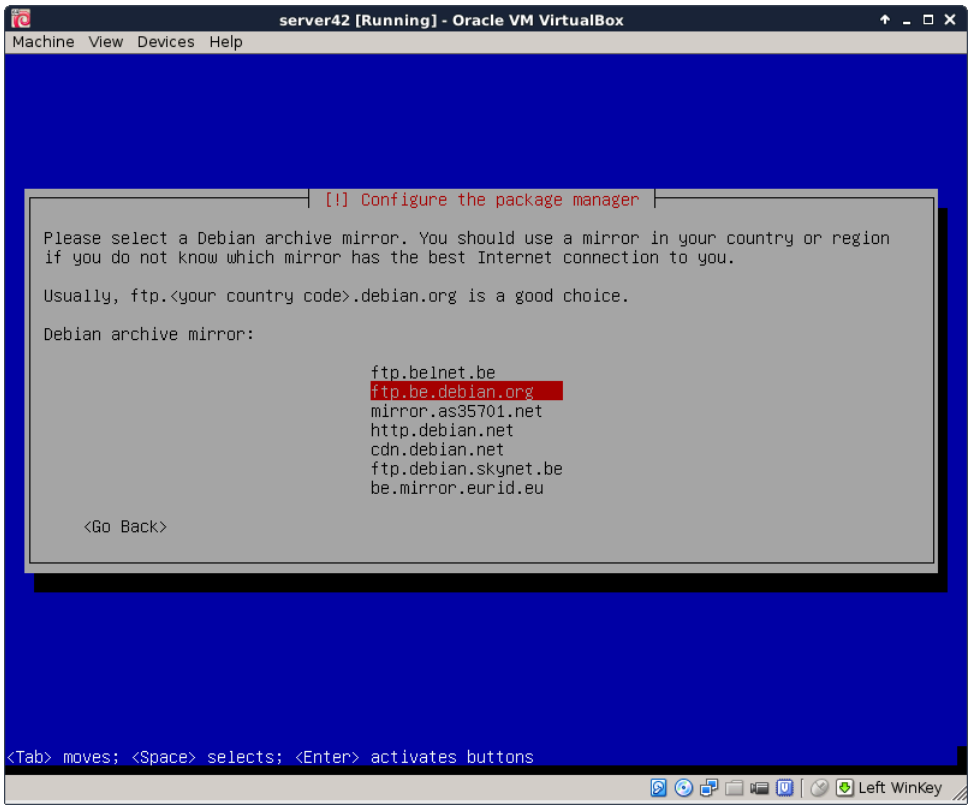
This is the point of no return, the magical moment where pressing **yes** will forever erase data on the (virtual) computer.



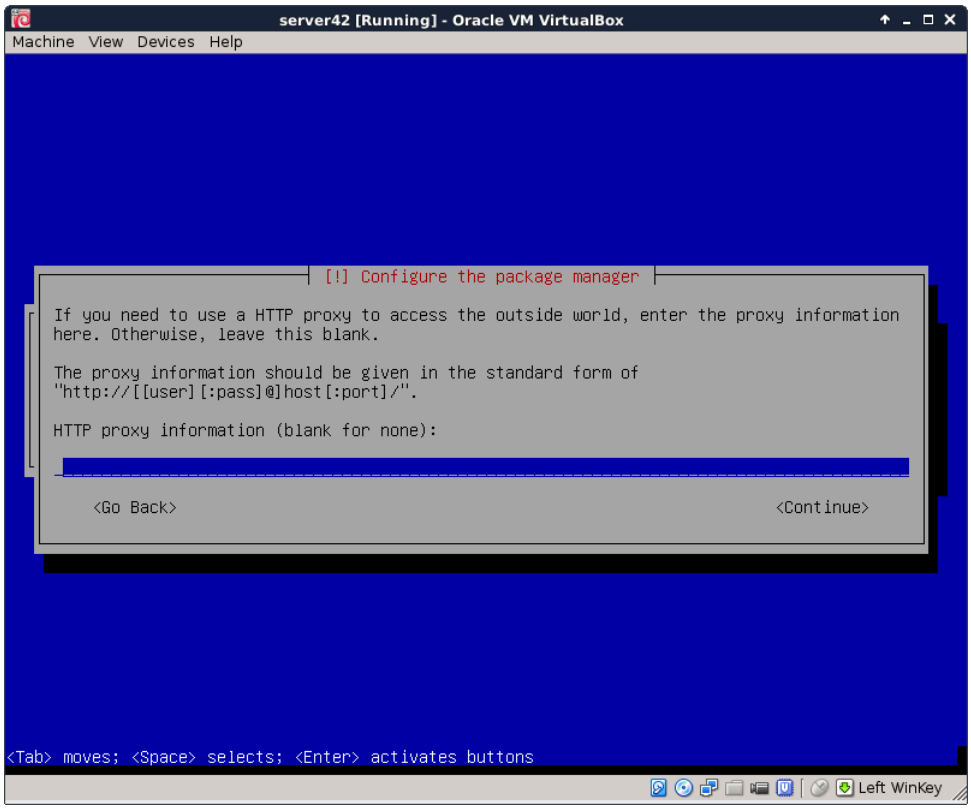
Software is downloaded from a mirror repository, preferably choose one that is close by (as in the same country).



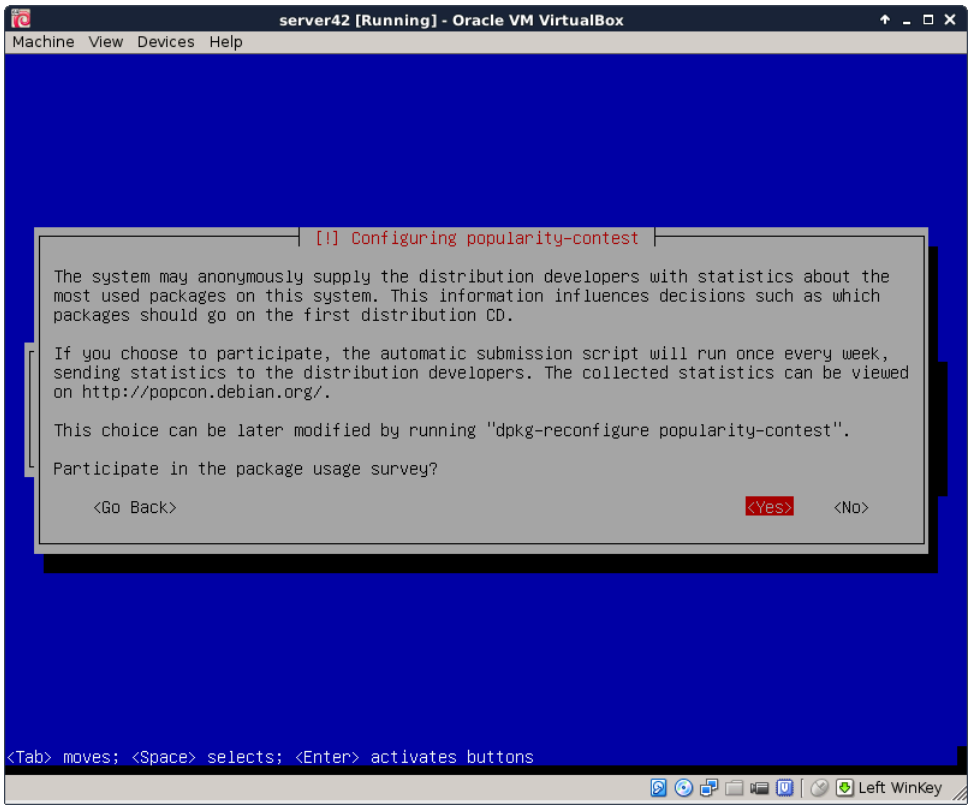
This setup was done in Belgium.



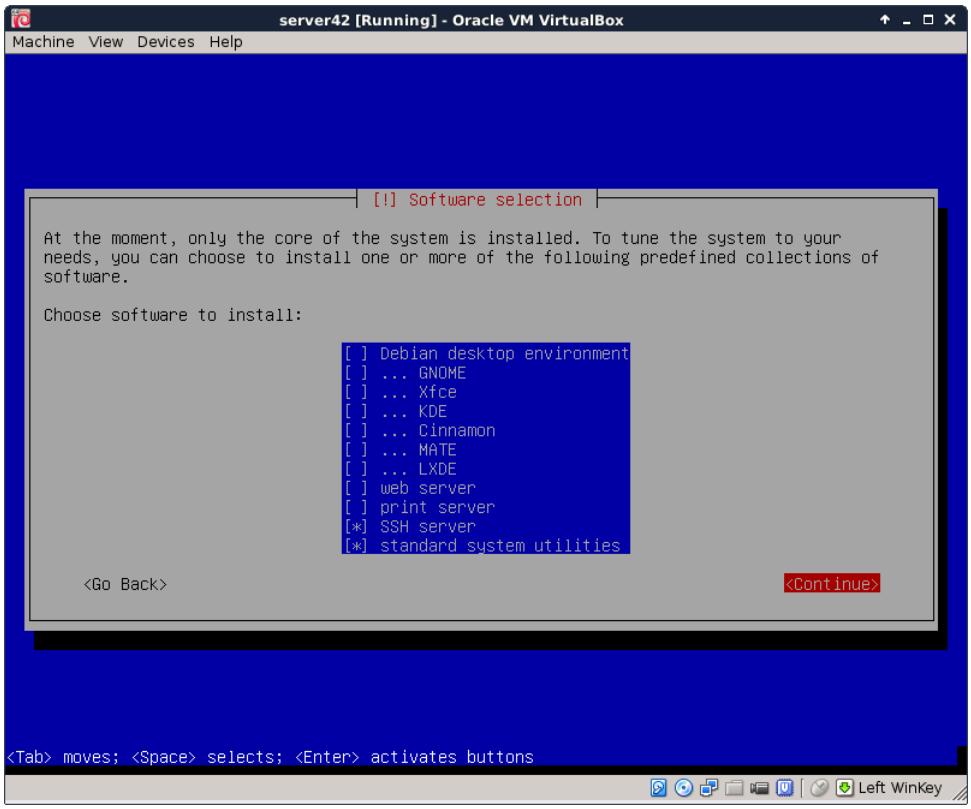
Leave the proxy field empty (unless you are sure that you are behind a proxy server).



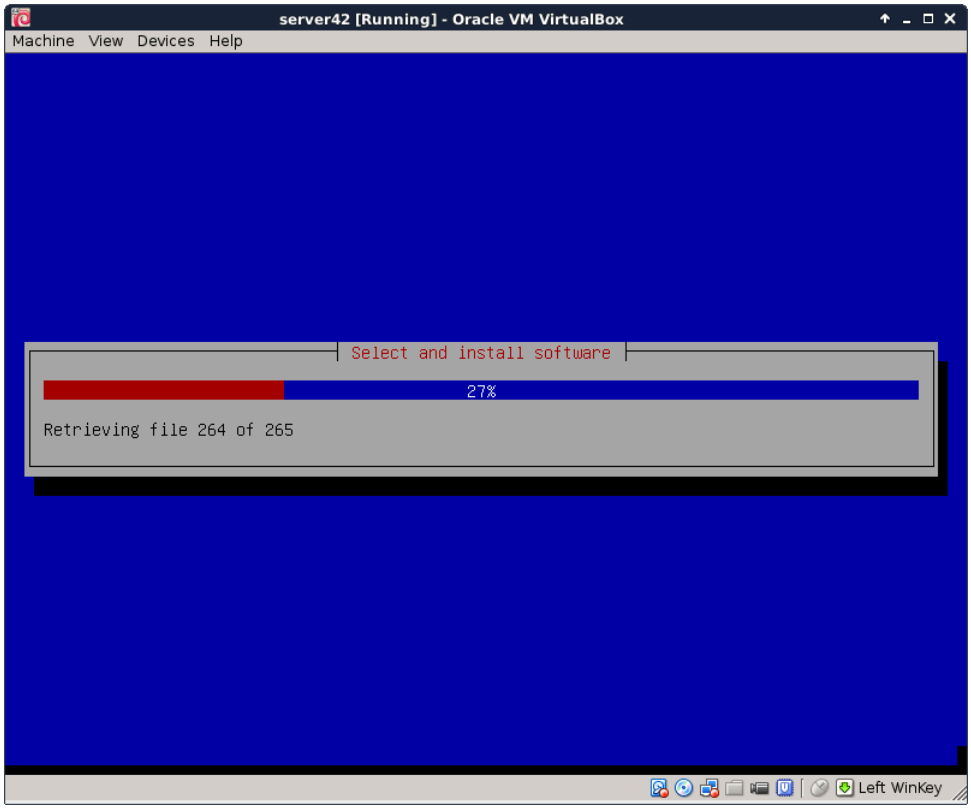
Choose whether you want to send anonymous statistics to the Debian project (it gathers data about installed packages). You can view the statistics here **http://popcon.debian.org/**.



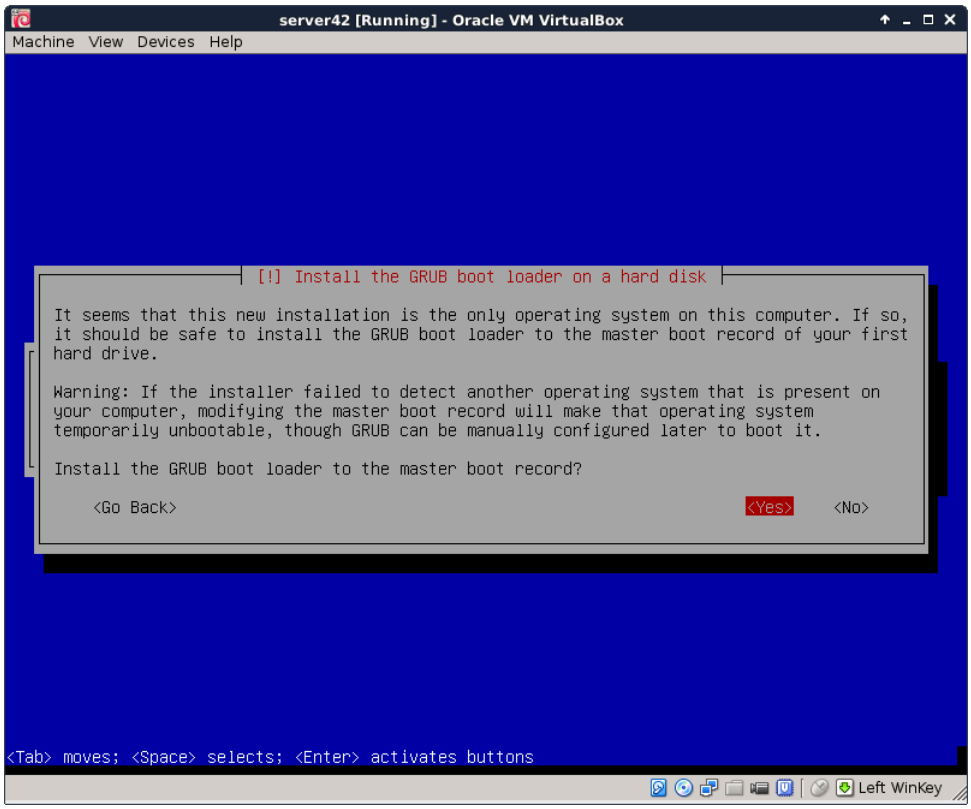
Choose what software to install, we do not need any graphical stuff for this training.



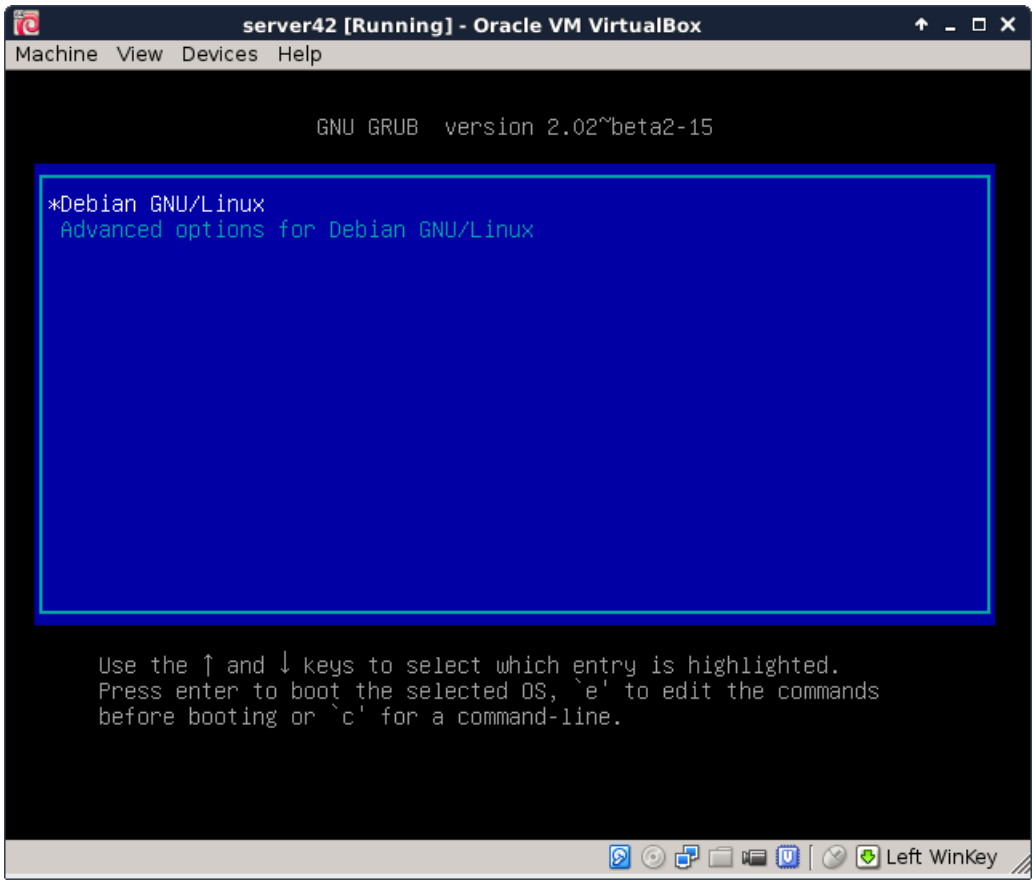
The latest versions are being downloaded.



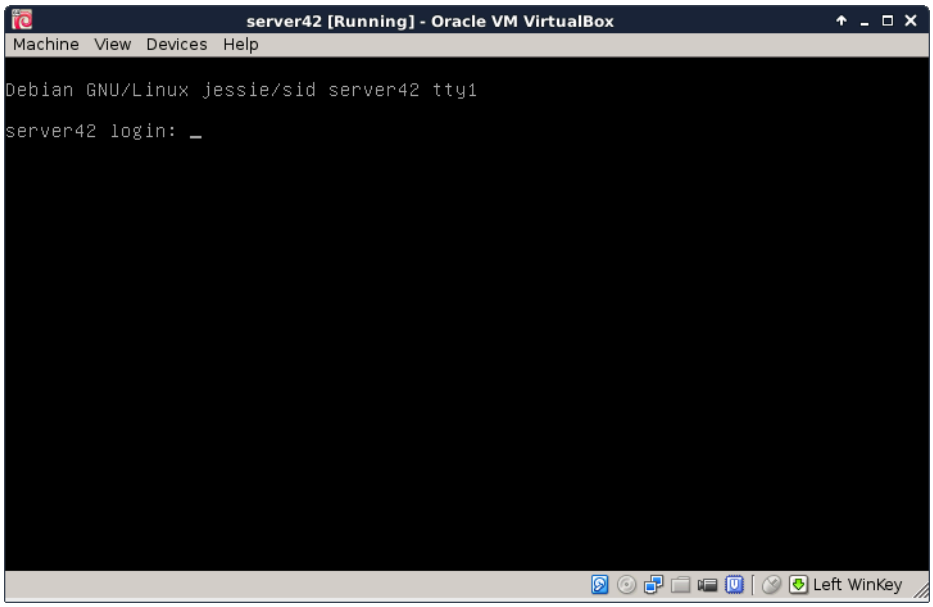
Say yes to install the bootloader on the virtual machine.



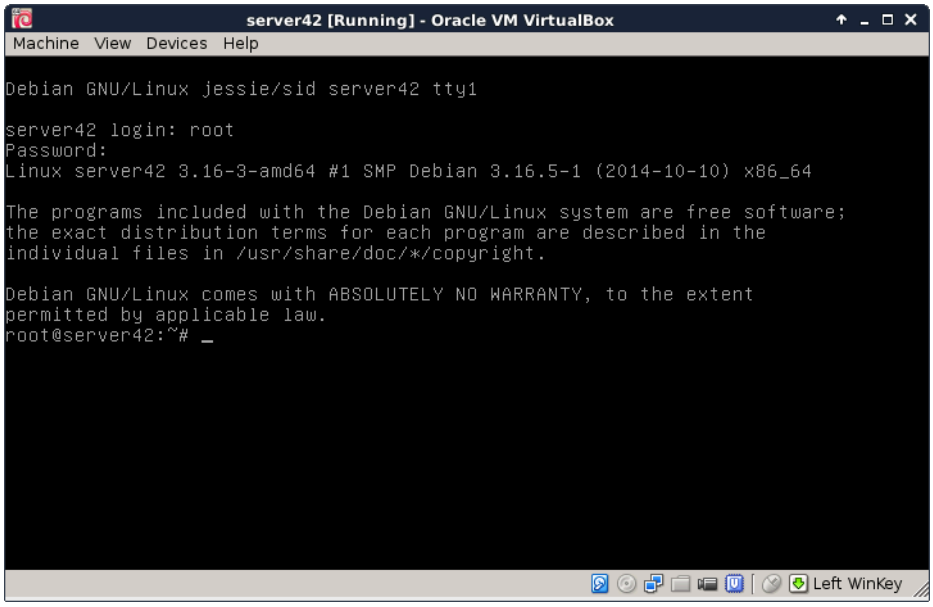
Booting for the first time shows the grub screen



A couple seconds later you should see a lot of text scrolling of the screen (**dmesg**). After which you are presented with this **getty** and are allowed your first logon.



You should now be able to log on to your virtual machine with the **root** account. Do you remember the password ? Was it **hunter2** ?



The screenshots in this book will look like this from now on. You can just type those commands in the terminal (after you logged on).

```
root@server42:~# who am i
root     tty1         2014-11-10 18:21
root@server42:~# hostname
server42
root@server42:~# date
Mon Nov 10 18:21:56 CET 2014
```

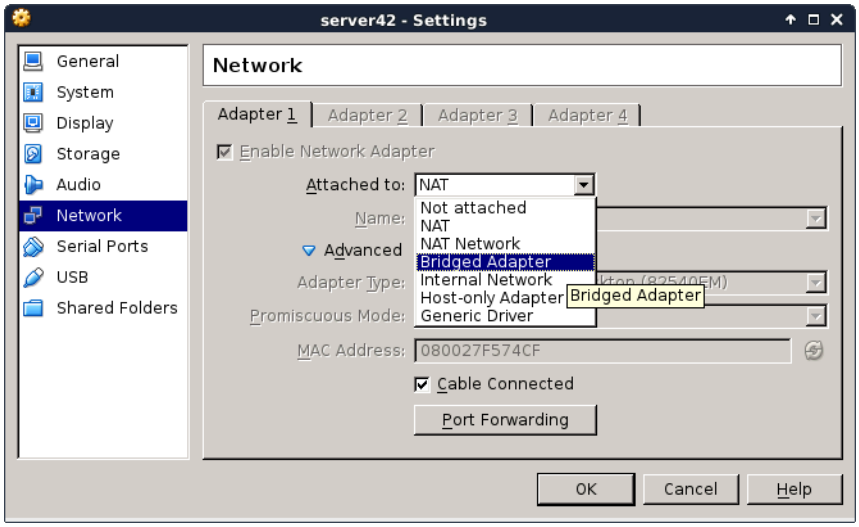
4.3. virtualbox networking

You can also log on from remote (or from your Windows/Mac/Linux host computer) using **ssh** or **putty**. Change the **network** settings in the virtual machine to **bridge**. This will enable your virtual machine to receive an ip address from your local dhcp server.

The default virtualbox networking is to attach virtual network cards to **nat**. This screenshiot shows the ip address **10.0.2.15** when on **nat**:

```
root@server42:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:f5:74:cf
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef5:74cf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2352 (2.2 KiB)  TX bytes:1988 (1.9 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```



By shutting down the network interface and enabling it again, we force Debian to renew an ip address from the bridged network.

```
root@server42:~# # do not run ifdown while connected over ssh!
root@server42:~# ifdown eth0
Killed old client process
Internet Systems Consortium DHCP Client 4.3.1
Copyright 2004-2014 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/08:00:27:f5:74:cf
Sending on   LPF/eth0/08:00:27:f5:74:cf
```

```
Sending on   Socket/fallback
DHCPRELEASE on eth0 to 10.0.2.2 port 67
root@server42:~# # now enable bridge in virtualbox settings
root@server42:~# ifup eth0
Internet Systems Consortium DHCP Client 4.3.1
Copyright 2004-2014 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/08:00:27:f5:74:cf
Sending on   LPF/eth0/08:00:27:f5:74:cf
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPOFFER from 192.168.1.42
DHCPACK from 192.168.1.42
bound to 192.168.1.111 -- renewal in 2938 seconds.
root@server42:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 08:00:27:f5:74:cf
          inet addr:192.168.1.111  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef5:74cf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3156 (3.0 KiB)  TX bytes:3722 (3.6 KiB)
root@server42:~#
```

Here is an example of **ssh** to this freshly installed computer. Note that **Debian 8** has disabled remote root access, so i need to use the normal user account.

```
paul@debian8:~$ ssh paul@192.168.1.111
paul@192.168.1.111's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
paul@server42:~$
paul@server42:~$ su -
Password:
root@server42:~#
```

TODO: putty screenshot here...

4.4. setting the hostname

The hostname of the server is asked during installation, so there is no need to configure this manually.

```
root@server42:~# hostname
server42
root@server42:~# cat /etc/hostname
server42
root@server42:~# dnsdomainname
paul.local
root@server42:~# grep server42 /etc/hosts
127.0.1.1       server42.paul.local     server42
root@server42:~#
```

4.5. adding a static ip address

This example shows how to add a static ip address to your server.

You can use **ifconfig** to set a static address that is active until the next **reboot** (or until the next **ifdown**).
a

```
root@server42:~# ifconfig eth0:0 10.104.33.39
```

Adding a couple of lines to the **/etc/network/interfaces** file to enable an extra ip address forever.

```
root@server42:~# vi /etc/network/interfaces
root@server42:~# tail -4 /etc/network/interfaces
auto eth0:0
iface eth0:0 inet static
address 10.104.33.39
netmask 255.255.0.0
root@server42:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:f5:74:cf
          inet addr:192.168.1.111  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef5:74cf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:528 errors:0 dropped:0 overruns:0 frame:0
          TX packets:333 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:45429 (44.3 KiB)  TX bytes:48763 (47.6 KiB)

eth0:0    Link encap:Ethernet  HWaddr 08:00:27:f5:74:cf
          inet addr:10.104.33.39  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@server42:~#
```

4.6. Debian package management

To get all information about the newest packages form the online repository:

```
root@server42:~# aptitude update
Get: 1 http://ftp.be.debian.org jessie InRelease [191 kB]
Get: 2 http://security.debian.org jessie/updates InRelease [84.1 kB]
Get: 3 http://ftp.be.debian.org jessie-updates InRelease [117 kB]
Get: 4 http://ftp.be.debian.org jessie-backports InRelease [118 kB]
Get: 5 http://security.debian.org jessie/updates/main Sources [14 B]
Get: 6 http://ftp.be.debian.org jessie/main Sources/DiffIndex [7,876 B]
... (output truncated)
```

To download and apply all updates for all installed packages:

```
root@server42:~# aptitude upgrade
Resolving dependencies...
The following NEW packages will be installed:
  firmware-linux-free{a} irqbalance{a} libnuma1{a} linux-image-3.16.0-4-amd64{a}
The following packages will be upgraded:
  busybox file libc-bin libc6 libexpat1 libmagic1 libpaper-utils libpaper1 libsqlite3-0
  linux-image-amd64 locales multiarch-support
12 packages upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 44.9 MB of archives. After unpacking 161 MB will be used.
Do you want to continue? [Y/n/?]
... (output truncated)
```

To install new software (**vim** and **tmux** in this example):

```
root@server42:~# aptitude install vim tmux
The following NEW packages will be installed:
  tmux vim vim-runtime{a}
0 packages upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 6,243 kB of archives. After unpacking 29.0 MB will be used.
Do you want to continue? [Y/n/?]
Get: 1 http://ftp.be.debian.org/debian/ jessie/main tmux amd64 1.9-6 [245 kB]
Get: 2 http://ftp.be.debian.org/debian/ jessie/main vim-runtime all 2:7.4.488-1 [5,046 kB]
Get: 3 http://ftp.be.debian.org/debian/ jessie/main vim amd64 2:7.4.488-1 [952 kB]
```

Refer to the **package management** chapter in LinuxAdm.pdf for more information.

Chapter 5. installing CentOS 7

This module is a step by step demonstration of an actual installation of **CentOS 7**.

We start by downloading an image from the internet and install **CentOS 7** as a virtual machine in **Virtualbox**. We will also do some basic configuration of this new machine like setting an **ip address** and fixing a **hostname**.

This procedure should be very similar for other versions of **CentOS**, and also for distributions like **RHEL** (Red Hat Enterprise Linux) or **Fedora**. This procedure can also be helpful if you are using another virtualization solution.

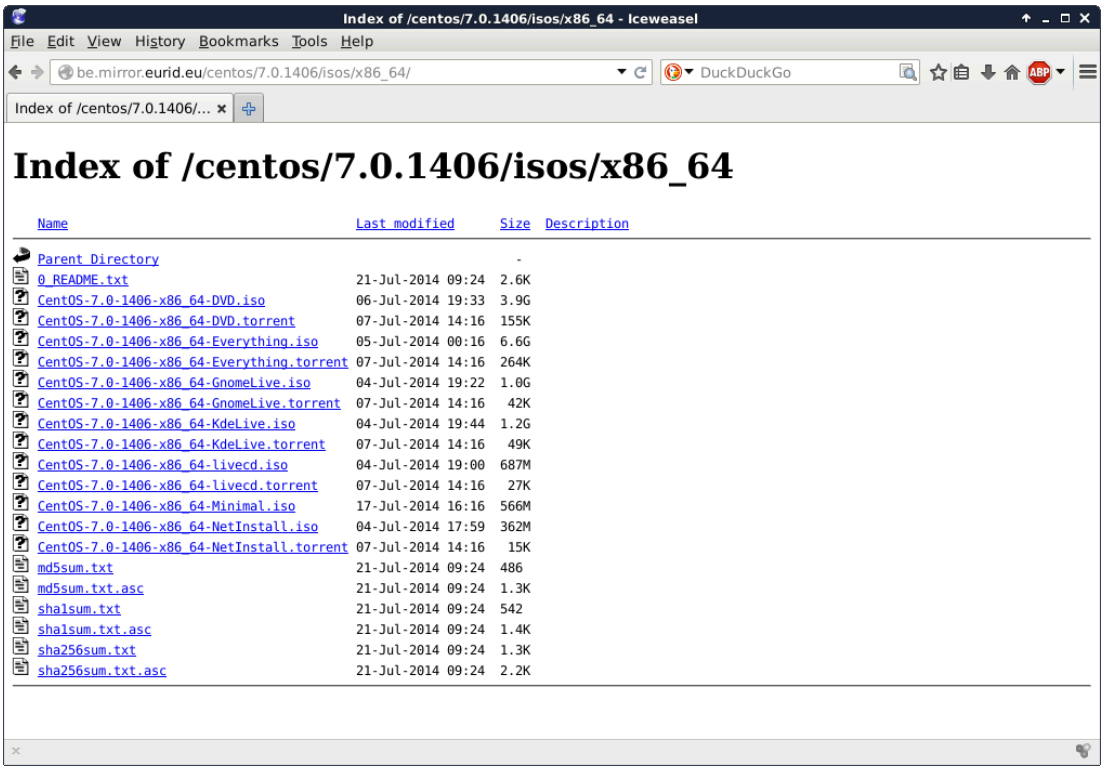
5.1. download a CentOS 7 image

This demonstration uses a laptop computer with **Virtualbox** to install **CentOS 7** as a virtual machine. The first task is to download an **.iso** image of **CentOS 7**.

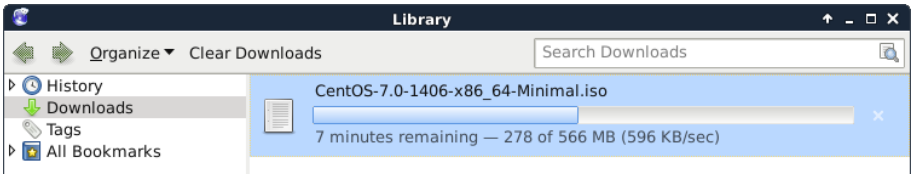
The **CentOS 7** website looks like this today (November 2014). They change the look regularly, so it may look different when you visit it.

You can download a full DVD, which allows for an off line installation of a graphical **CentOS 7** desktop. You can select this because it should be easy and complete, and should get you started with a working **CentOS 7** virtual machine.

But I clicked instead on 'alternative downloads', selected **CentOS 7** and **x86_64** and ended up on a **mirror list**. Each mirror is a server that contains copies of **CentOS 7** media. I selected a Belgian mirror because I currently am in Belgium.



There is again the option for full DVD's and more. This demonstration will use the **minimal** .iso file, because it is much smaller in size. The download takes a couple of minutes.



Verify the size of the file after download to make sure it is complete. Probably a right click on the file and selecting 'properties' (if you use Windows or Mac OSX).

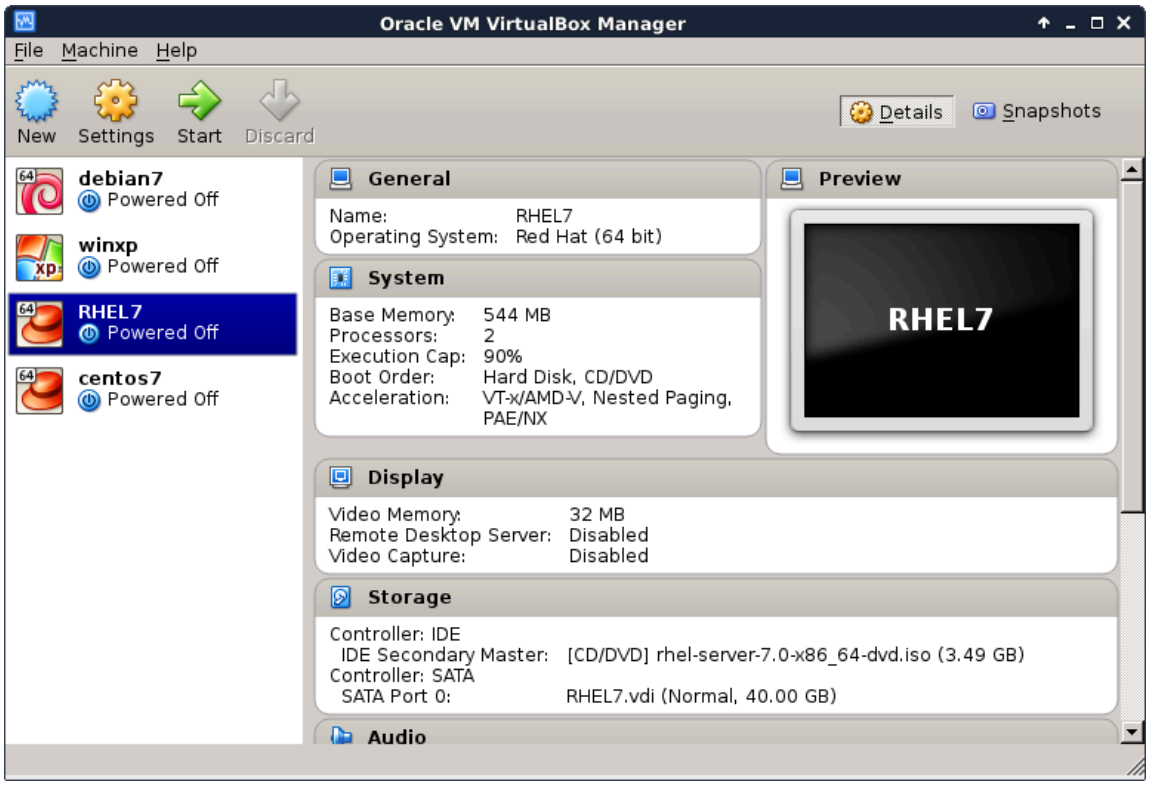
I use Linux on the laptop already:

```
paul@debian8:~$ ls -lh CentOS-7.0-1406-x86_64-Minimal.iso
-rw-r--r-- 1 paul paul 566M Nov  1 14:45 CentOS-7.0-1406-x86_64-Minimal.iso
```

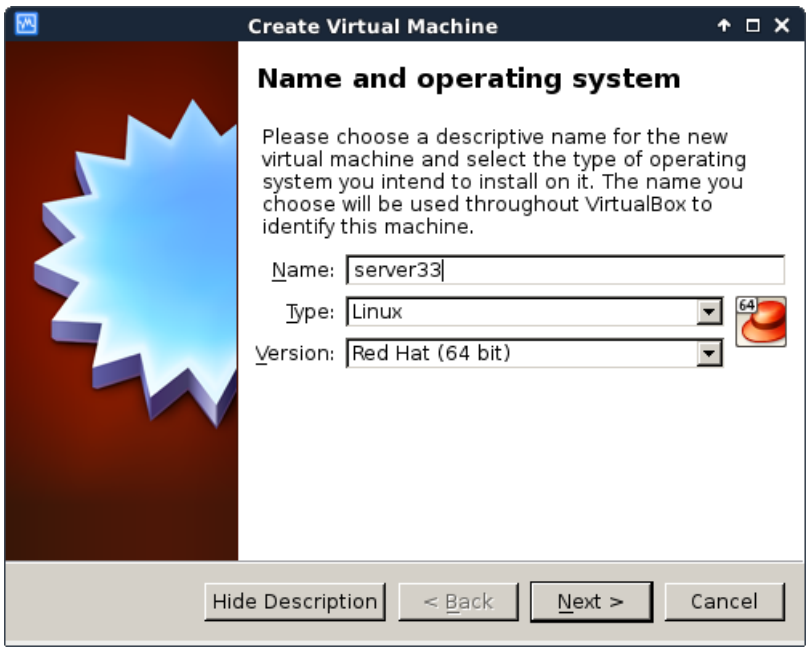
Do not worry if you do no understand the above command. Just try to make sure that the size of this file is the same as the size that is mentioned on the **CentOS 7** website.

5.2. Virtualbox

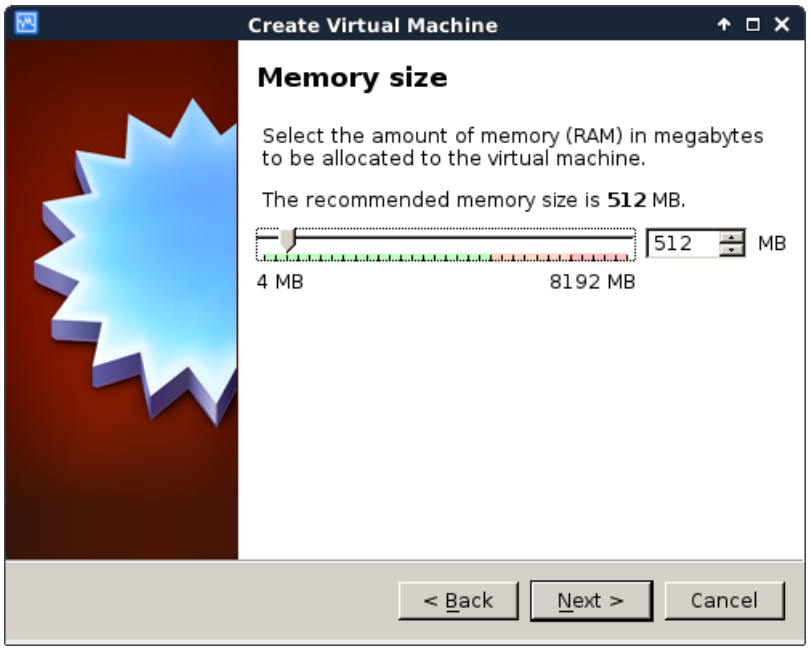
This screenshot shows up when I start Virtualbox. I already have four virtual machines, you might have none.



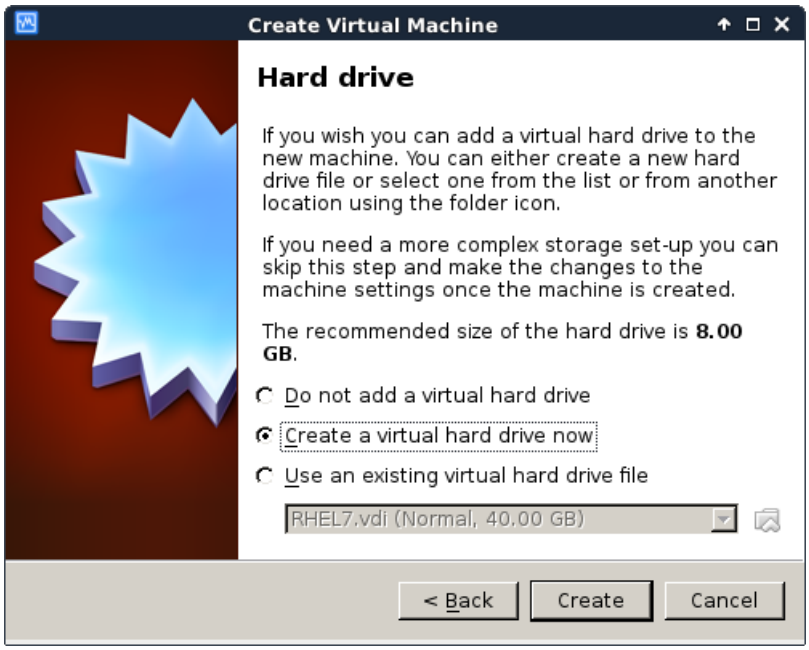
Below are the steps for creating a new virtual machine. Start by clicking **New** and give your machine a name (I chose **server33**). Click **Next**.



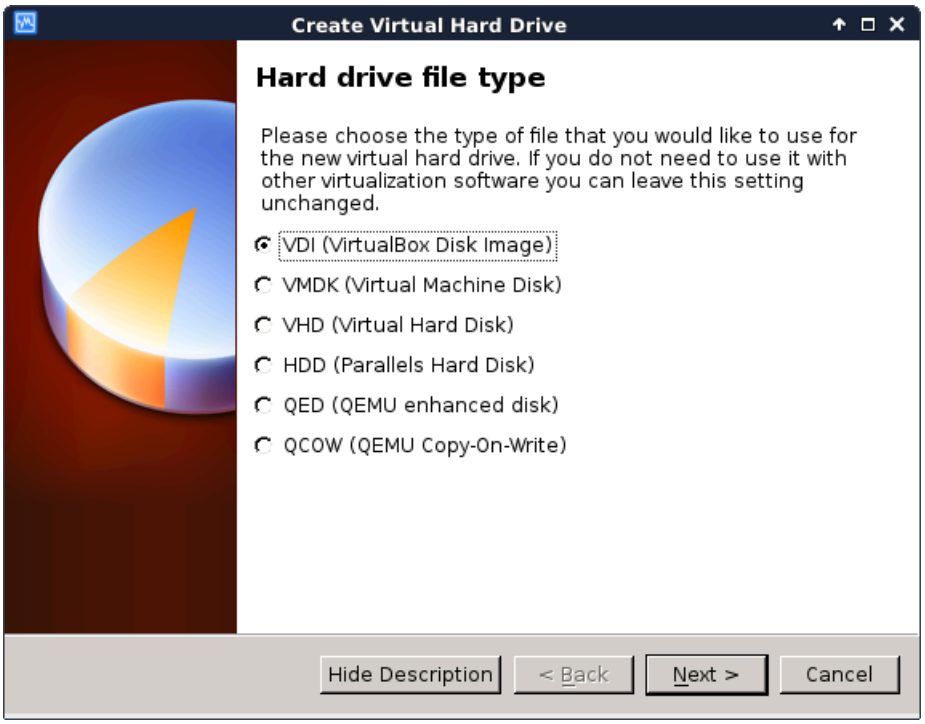
A Linux computer without graphical interface will run fine on **half a gigabyte** of RAM.



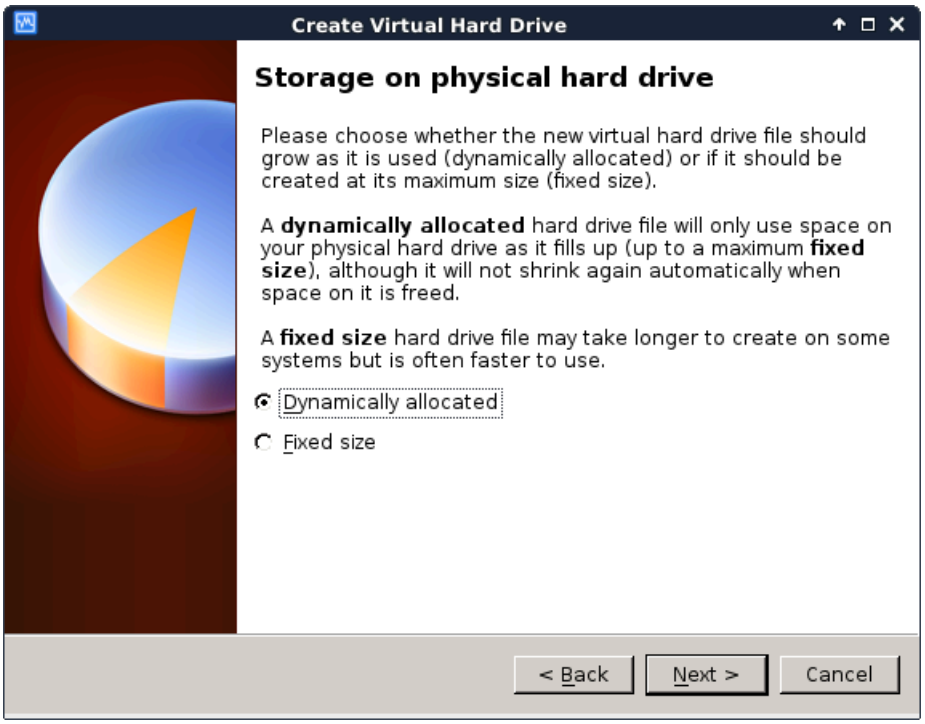
A Linux virtual machine will need a **virtual hard drive**.



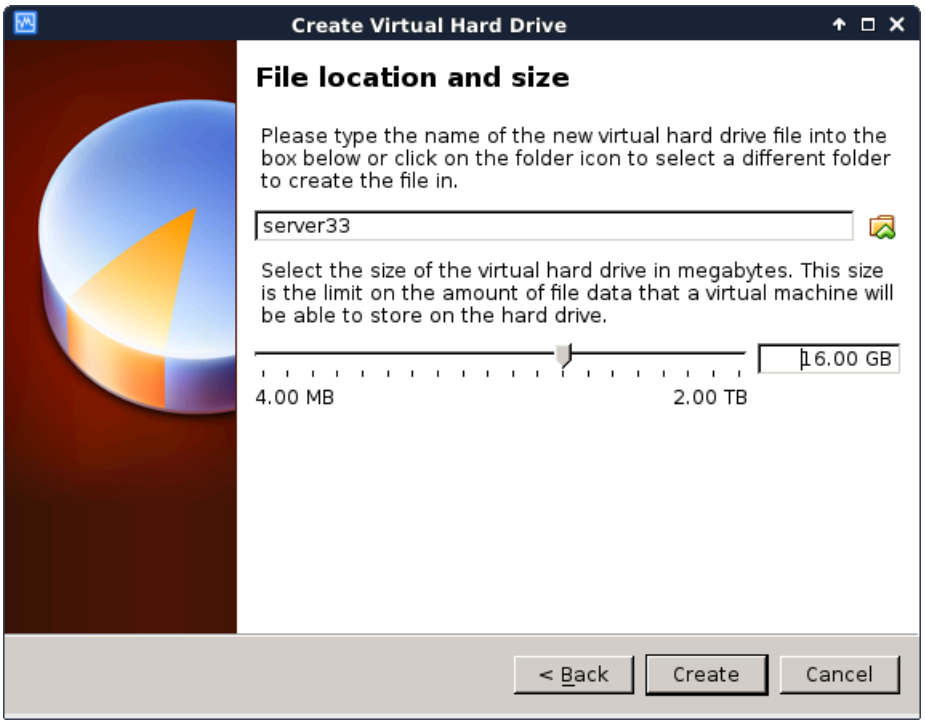
Any format will do for our purpose, so I left the default **vdi**.



The default **dynamically allocated** type will save disk space (until we fill the virtual disk up to 100 percent). It makes the virtual machine a bit slower than **fixed size**, but the **fixed size** speed improvement is not worth it for our purpose.

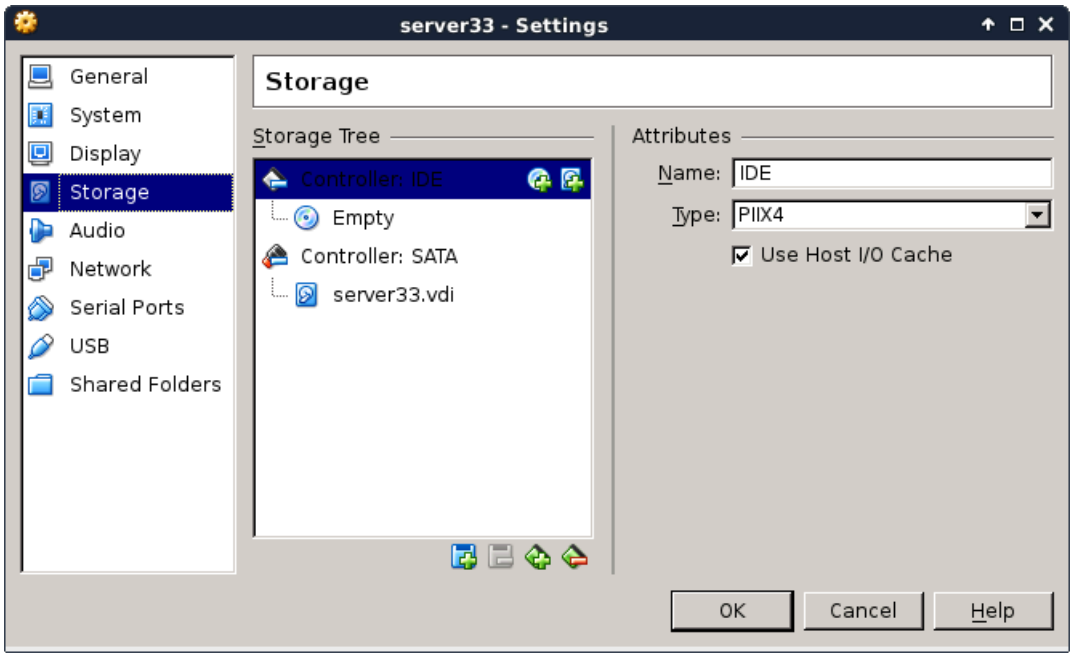


The name of the virtual disk file on the host computer will be **server33.vdi** in my case (I left it default and it uses the vm name). Also 16 GB should be enough to practice Linux. The file will stay much smaller than 16GB, unless you copy a lot of files to the virtual machine.

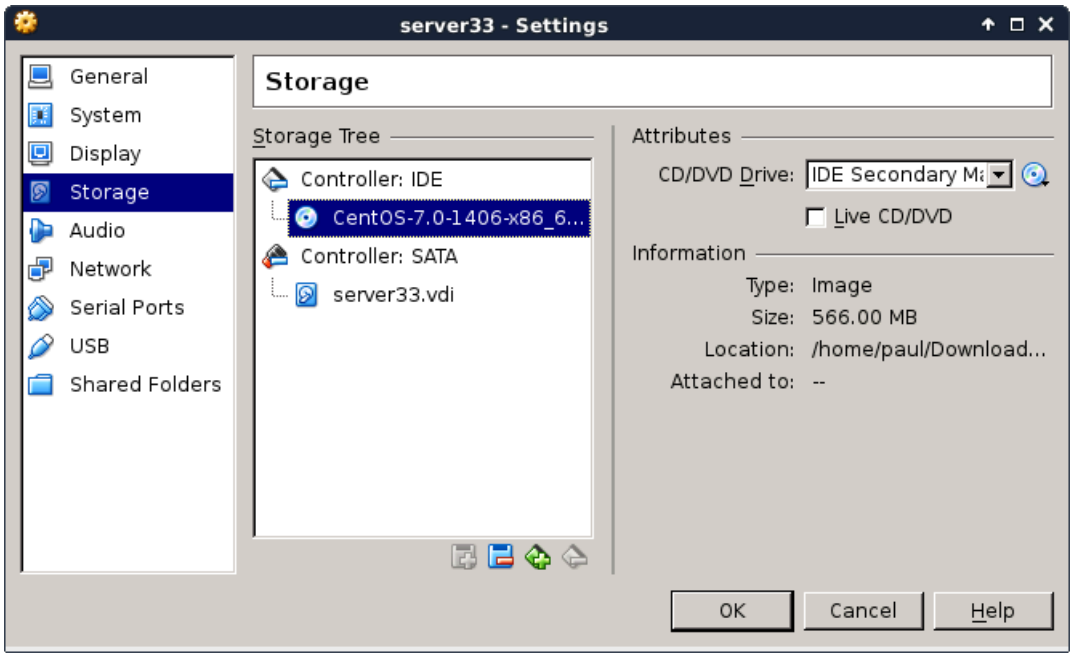


You should now be back to the start screen of **Virtualbox**. If all went well, then you should see the machine you just created in the list.

After finishing the setup, we go into the **Settings** of our virtual machine and attach the **.iso** file we downloaded before. Below is the default screenshot.



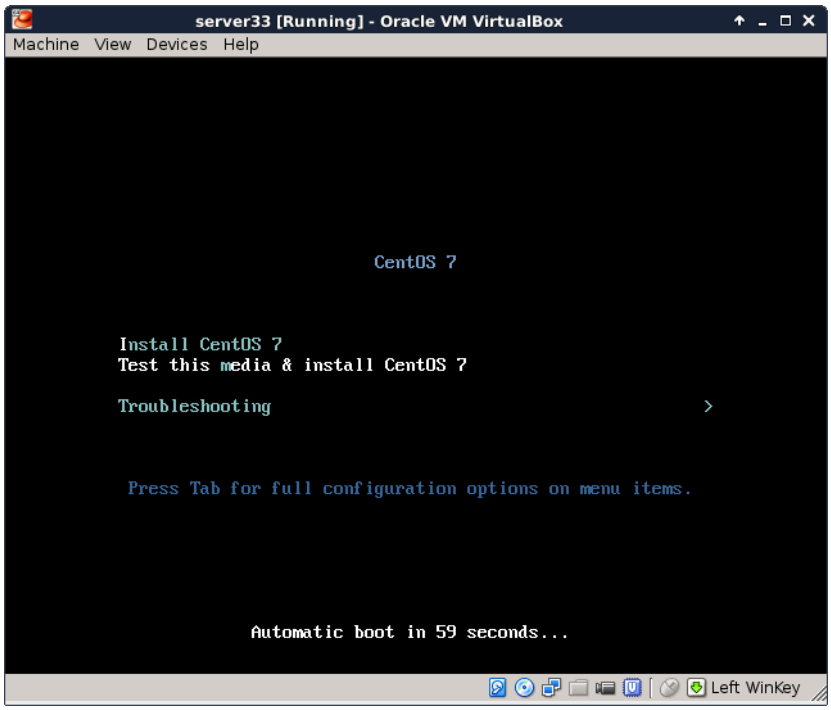
This is a screenshot with the **.iso** file properly attached.



5.3. CentOS 7 installing

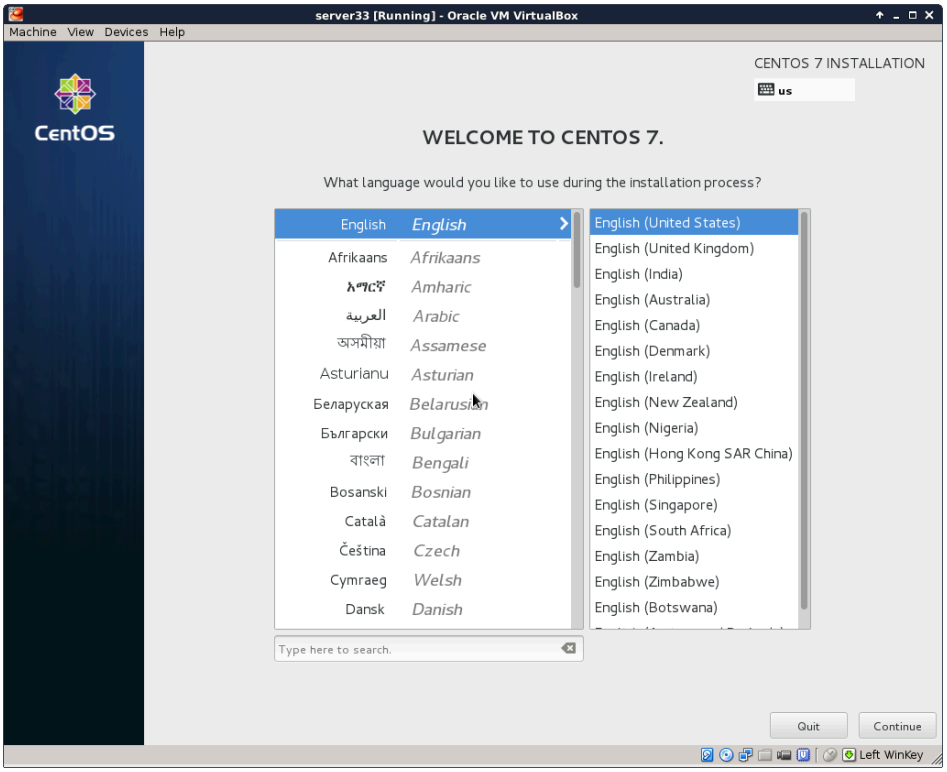
The screenshots below will show every step from starting the virtual machine for the first time (with the .iso file attached) until the first logon.

You should see this when booting, otherwise verify the attachment of the .iso file form the previous steps. Select **Test this media and install CentOS 7**.

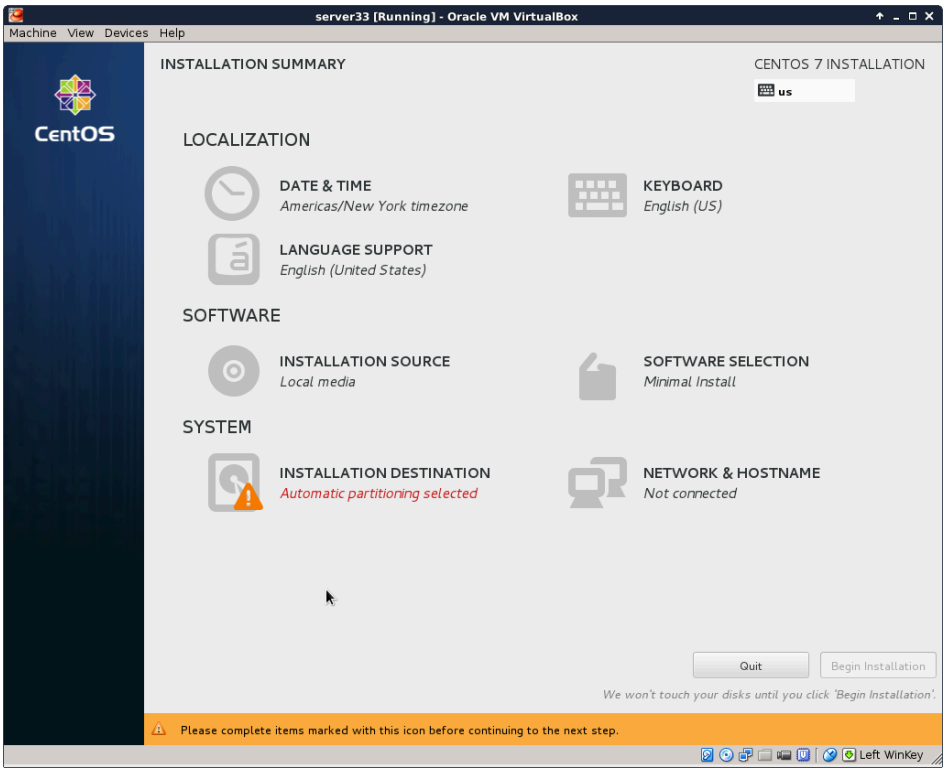


Carefully select the language in which you want your **CentOS**. I always install operating systems in English, even though my native language is not English.

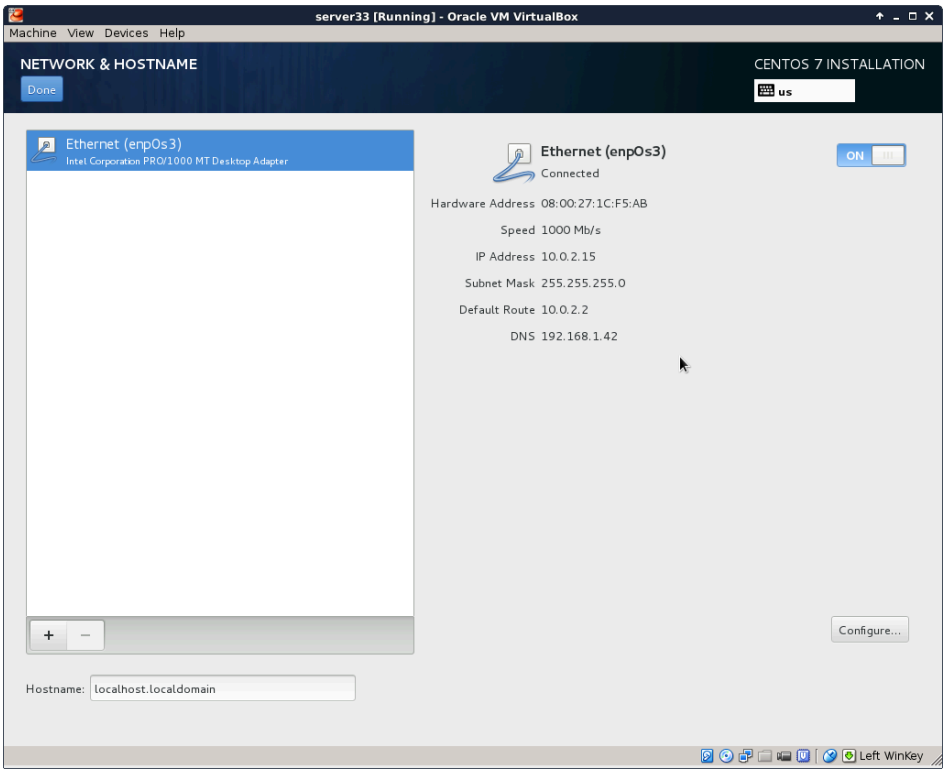
Also select the right keyboard, mine is a US qwerty, but yours may be different.



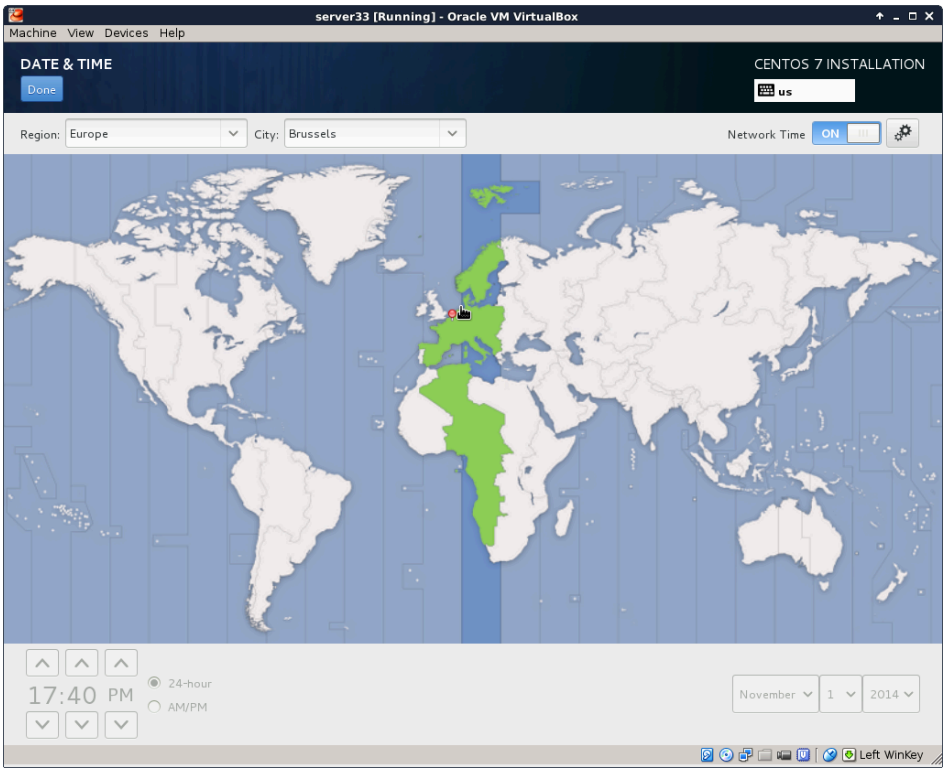
You should arrive at a summary page (with one or more warnings).



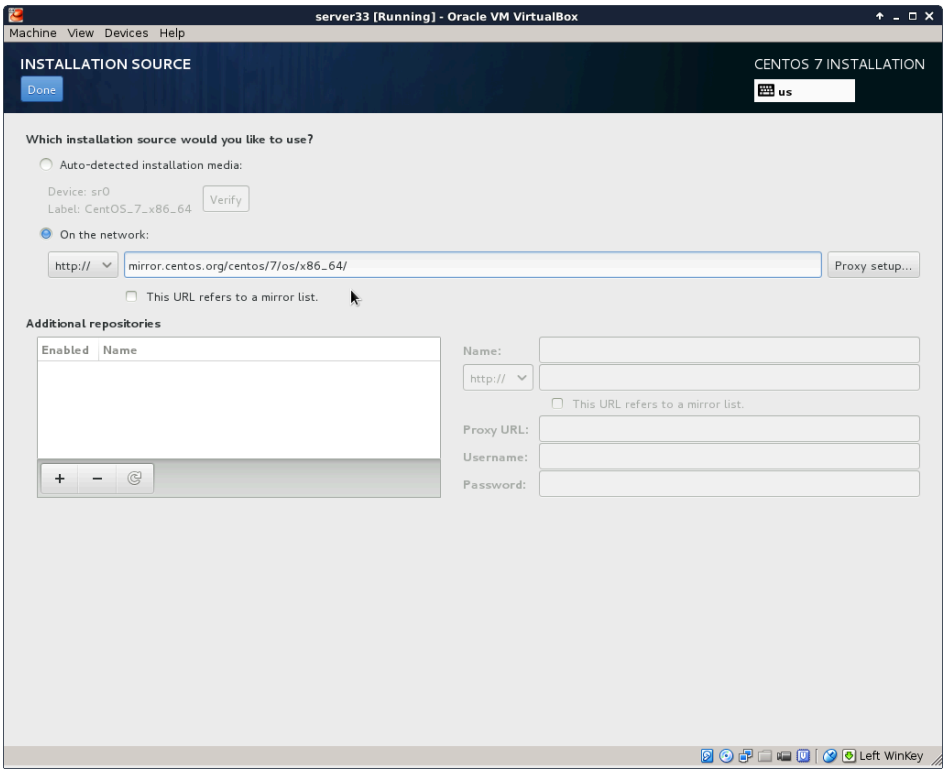
Start by configuring the network. During this demonstration I had a DHCP server running at 192.168.1.42, yours is probably different. Ask someone (a network administator ?) for help if this step fails.



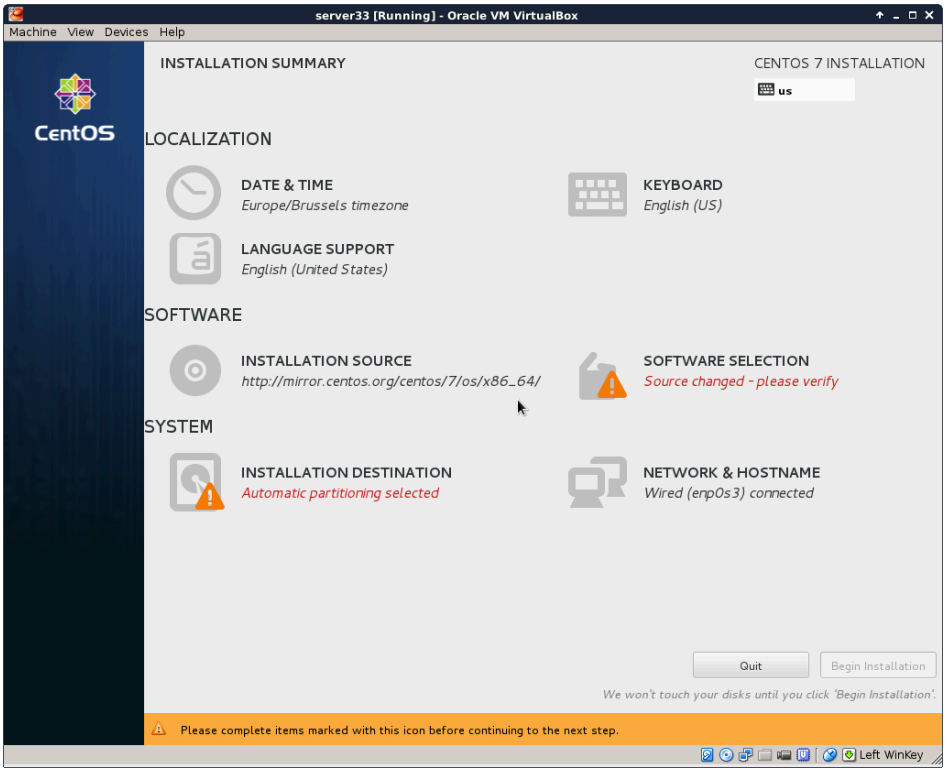
Select your time zone, and activate **ntp**.



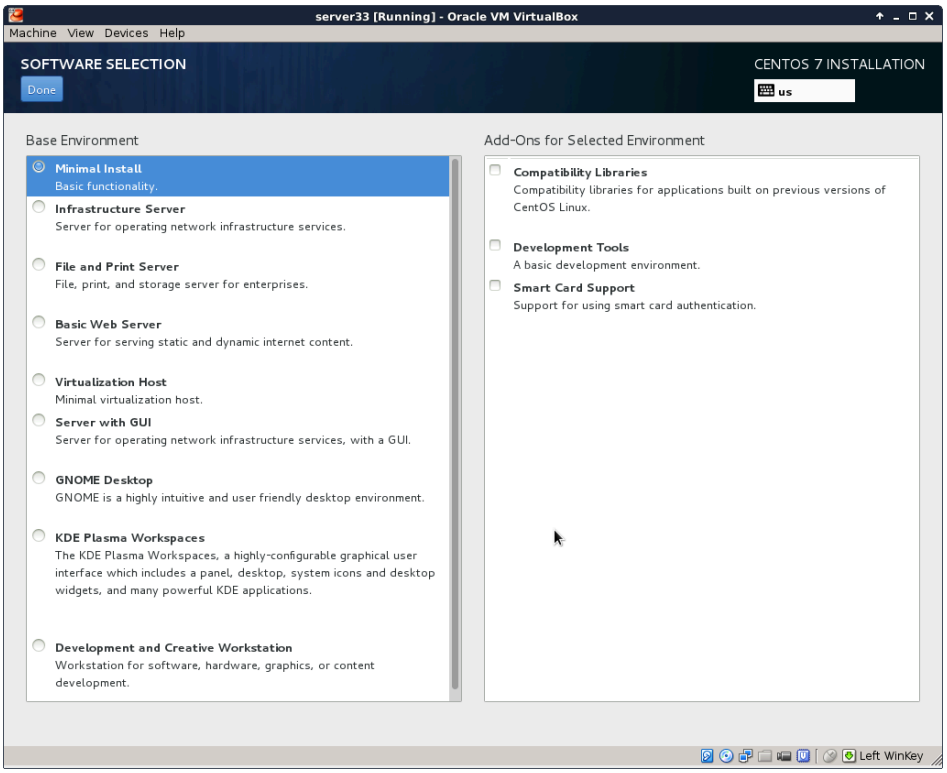
Choose a mirror that is close to you. If you can't find a local mirror, then you can copy the one from this screenshot (it is a general **CentOS** mirror).



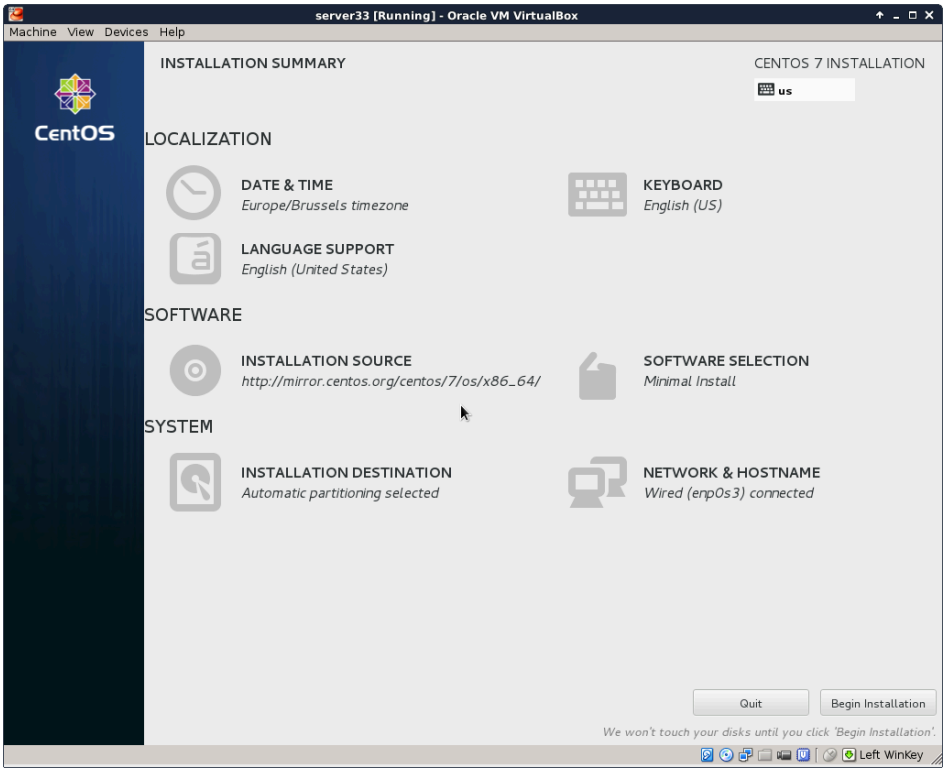
It can take a couple of seconds before the mirror is verified.



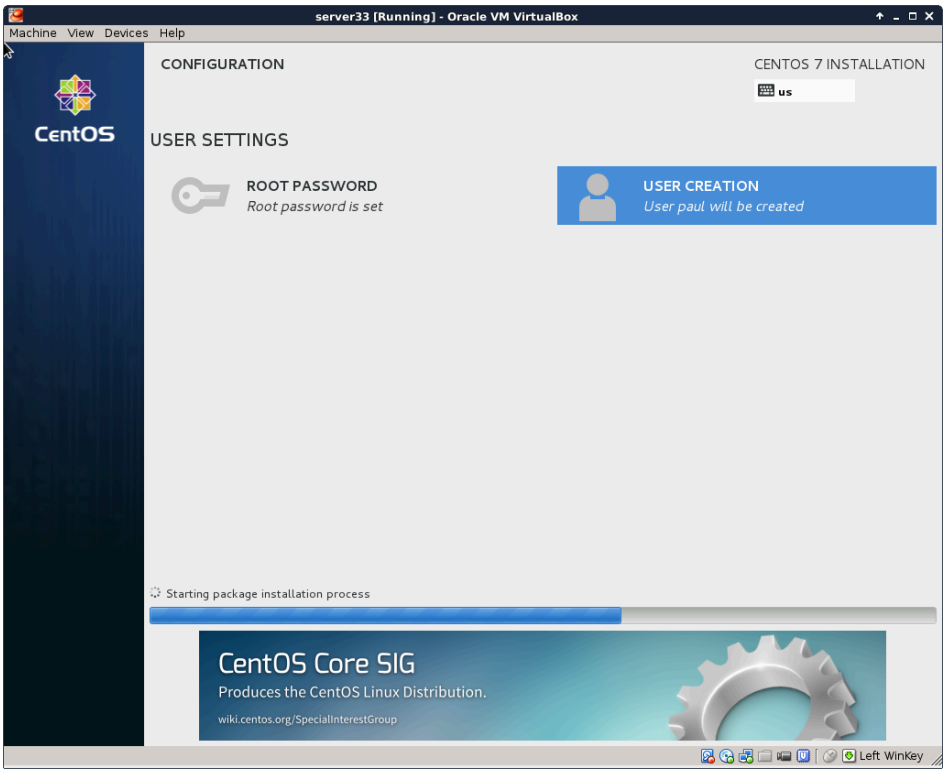
I did not select any software here (because I want to show it all in this training).



After configuring network, location, software and all, you should be back on this page. Make sure there are no warnings anymore (and that you made the correct choice everywhere).

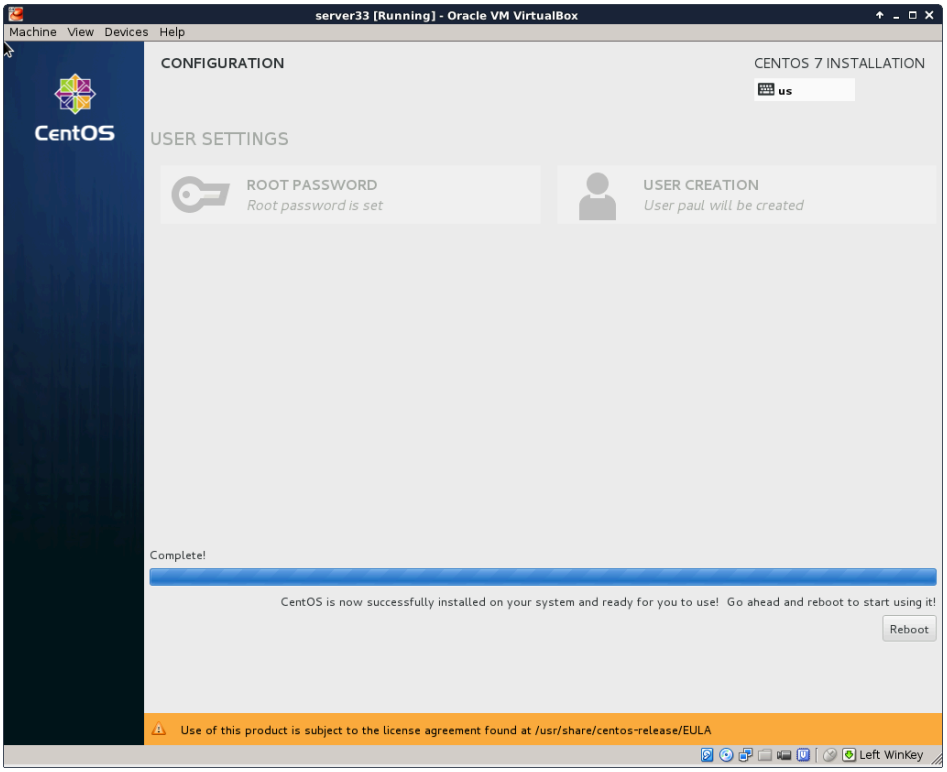


You can enter a **root password** and create a **user account** while the installation is downloading from the internet. This is the longest step, it can take several minutes (or up to an hour if you have a slow internet connection).

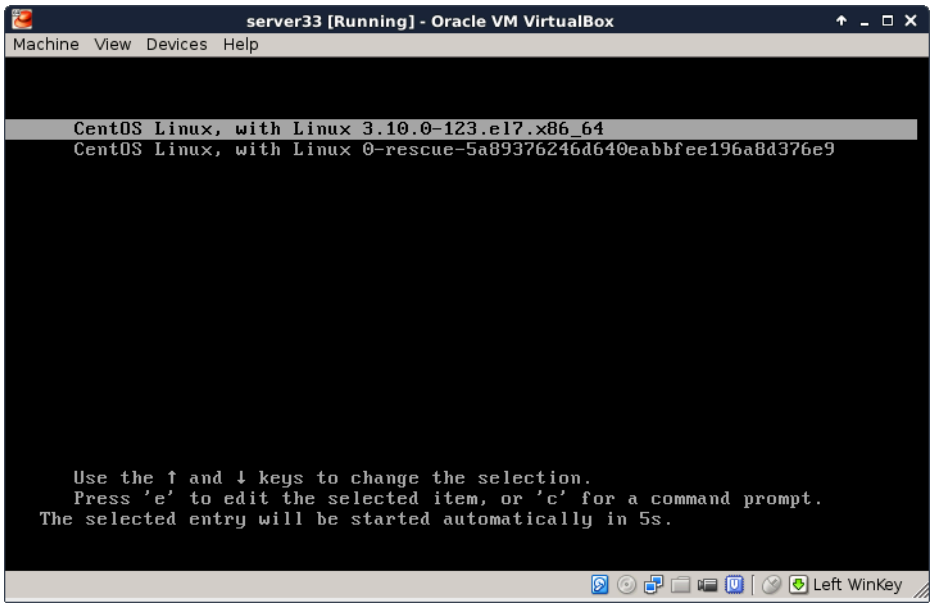


If you see this, then the installation was successful.

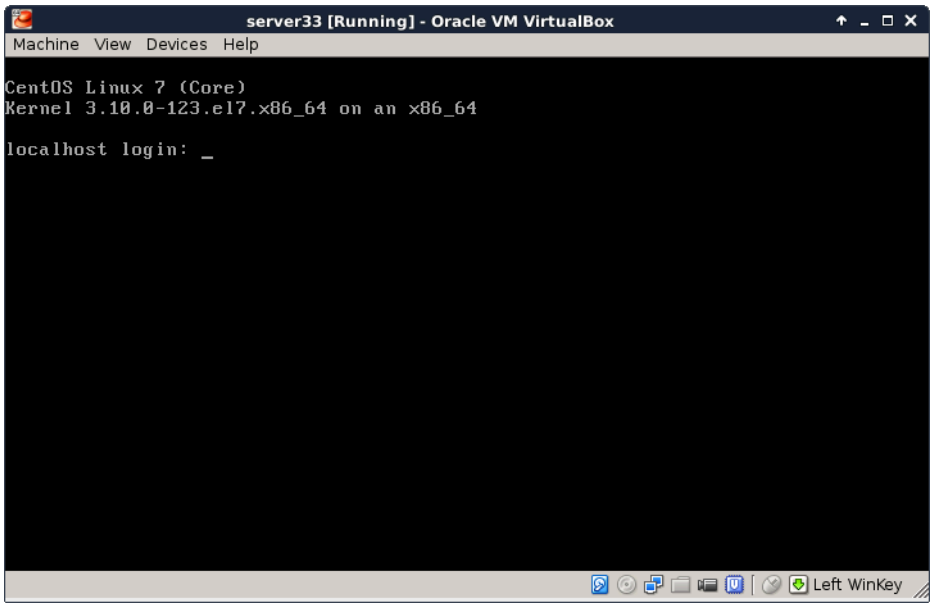
Time to reboot the computer and start **CentOS 7** for the first time.



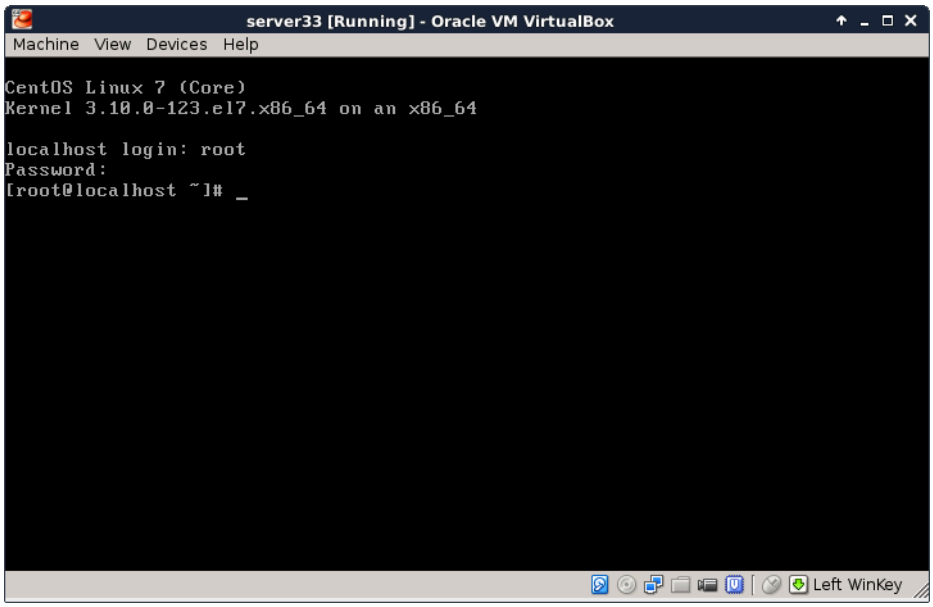
This screen will appear briefly when the virtual machines starts. You don't have to do anything.



After a couple of seconds, you should see a logon screen. This is called a **tty** or a **getty**. Here you can type **root** as username. The **login process** will then ask your password (nothing will appear on screen when you type your password).



And this is what it looks like after logon. You are logged on to your own Linux machine, very good.



All subsequent screenshots will be text only, no images anymore.

For example this screenshot shows three commands being typed on my new CentOS 7 install.

```
[root@localhost ~]# who am i
root     pts/0        2014-11-01 22:14
[root@localhost ~]# hostname
localhost.localdomain
[root@localhost ~]# date
Sat Nov  1 22:14:37 CET 2014
```

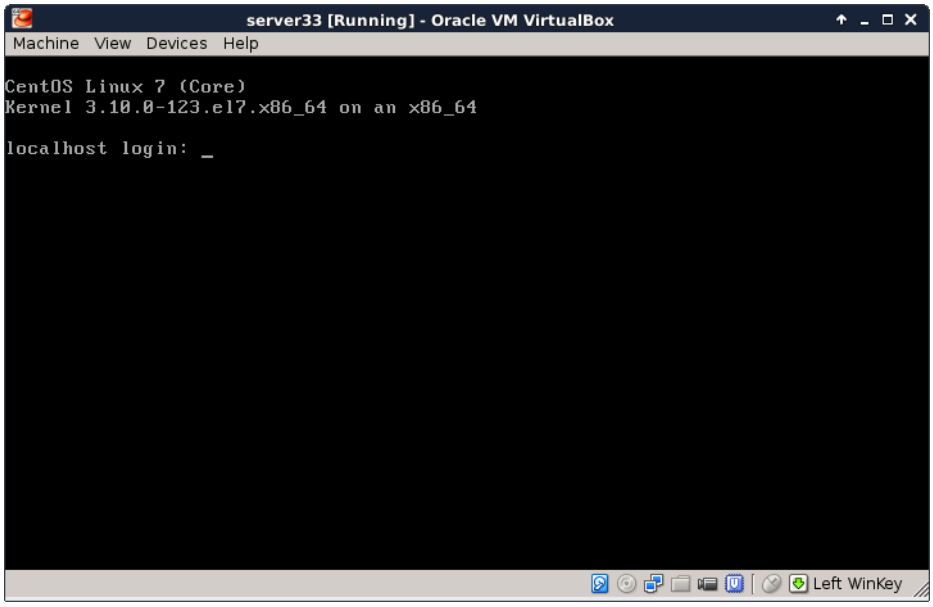
When using **ssh** the same commands will give this screenshot:

```
[root@localhost ~]# who am i
root     pts/0        2014-11-01 21:00 (192.168.1.35)
[root@localhost ~]# hostname
localhost.localdomain
[root@localhost ~]# date
Sat Nov  1 22:10:04 CET 2014
[root@localhost ~]#
```

If the last part is a bit too fast, take a look at the next topic **CentOS 7 first logon**.

5.4. CentOS 7 first logon

All you have to log on, after finishing the installation, is this screen in Virtualbox.



This is workable to learn Linux, and you will be able to practice a lot. But there are more ways to access your virtual machine, the next chapters discuss some of these and will also introduce some basic system configuration.

5.4.1. setting the hostname

Setting the hostname is a simple as changing the **/etc/hostname** file. As you can see here, it is set to **localhost.localdomain** by default.

```
[root@localhost ~]# cat /etc/hostname
localhost.localdomain
```

You could do **echo server33.netsec.local > /etc/hostname** followed by a **reboot**. But there is also the new **CentOS 7** way of setting a new hostname.

```
[root@localhost ~]# nmtui
```

The above command will give you a menu to choose from with a **set system hostname** option. Using this **nmtui** option will edit the **/etc/hostname** file for you.

```
[root@localhost ~]# cat /etc/hostname
server33.netsec.local
[root@localhost ~]# hostname
server33.netsec.local
[root@localhost ~]# dnsdomainname
netsec.local
```

For some reason the documentation on the **centos.org** and **docs.redhat.com** websites tell you to also execute this command:

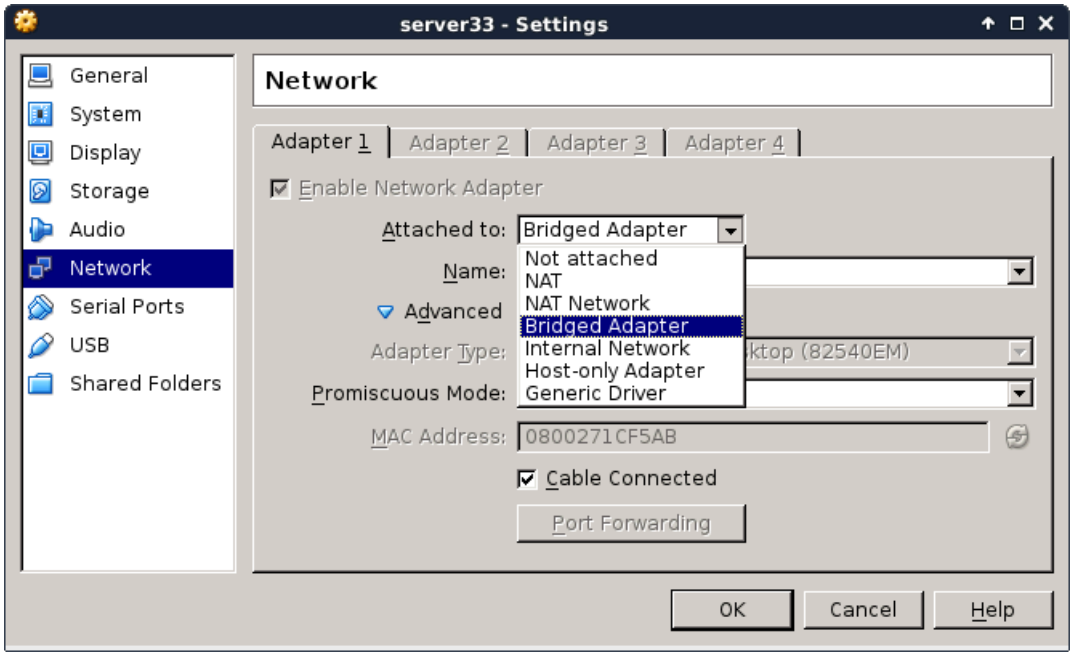
```
[root@localhost ~]# systemctl restart systemd-hostnamed
```

5.5. Virtualbox network interface

By default **Virtualbox** will connect your virtual machine over a **nat** interface. This will show up as a 10.0.2.15 (or similar).

```
[root@server33 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast s\
tate UP qlen 1000
    link/ether 08:00:27:1c:f5:ab brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 86399sec preferred_lft 86399sec
    inet6 fe80::a00:27ff:fe1c:f5ab/64 scope link
       valid_lft forever preferred_lft forever
```

You can change this to **bridge** (over your wi-fi or over the ethernet cable) and thus make it appear as if your virtual machine is directly on your local network (receiving an ip address from your real dhcp server).



You can make this change while the vm is running, provided that you execute this command:

```
[root@server33 ~]# systemctl restart network
[root@server33 ~]# ip a s dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast s\
tate UP qlen 1000
    link/ether 08:00:27:1c:f5:ab brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.110/24 brd 192.168.1.255 scope global dynamic enp0s3
       valid_lft 7199sec preferred_lft 7199sec
    inet6 fe80::a00:27ff:fe1c:f5ab/64 scope link
       valid_lft forever preferred_lft forever
[root@server33 ~]#
```

5.6. configuring the network

The new way of changing network configuration is through the **nmtui** tool. If you want to manually play with the files in **/etc/sysconfig/network-scripts** then you will first need to verify (and disable) **NetworkManager** on that interface.

Verify whether an interface is controlled by **NetworkManager** using the **nmcli** command (connected means managed bu NM).

```
[root@server33 ~]# nmcli dev status
DEVICE  TYPE      STATE      CONNECTION
enp0s3  ethernet  connected  enp0s3
lo      loopback  unmanaged  --
```

Disable **NetworkManager** on an interface (enp0s3 in this case):

```
echo 'NM_CONTROLLED=no' >> /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

You can restart the network without a reboot like this:

```
[root@server33 ~]# systemctl restart network
```

Also, forget **ifconfig** and instead use **ip a**.

```
[root@server33 ~]# ip a s dev enp0s3 | grep inet
    inet 192.168.1.110/24 brd 192.168.1.255 scope global dynamic enp0s3
    inet6 fe80::a00:27ff:fe1c:f5ab/64 scope link
[root@server33 ~]#
```

5.7. adding one static ip address

This example shows how to add one static ip address to your computer.

```
[root@server33 ~]# nmtui edit enp0s3
```

In this interface leave the IPv4 configuration to automatic, and add an ip address just below.

```
          IPv4 CONFIGURATION <Automatic>                          <Hide>
          Addresses 10.104.33.32/16__________ <Remove>
```

Execute this command after exiting **nmtui**.

```
[root@server33 ~]# systemctl restart network
```

And verify with **ip** (not with **ifconfig**):

```
[root@server33 ~]# ip a s dev enp0s3 | grep inet
    inet 192.168.1.110/24 brd 192.168.1.255 scope global dynamic enp0s3
    inet 10.104.33.32/16 brd 10.104.255.255 scope global enp0s3
    inet6 fe80::a00:27ff:fe1c:f5ab/64 scope link
[root@server33 ~]#
```

5.8. package management

Even with a network install, **CentOS 7** did not install the latest version of some packages. Luckily there is only one command to run (as root). This can take a while.

```
[root@server33 ~]# yum update
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.weepeetelecom.be
 * extras: centos.weepeetelecom.be
 * updates: centos.weepeetelecom.be
Resolving Dependencies
--> Running transaction check
---> Package NetworkManager.x86_64 1:0.9.9.1-13.git20140326.4dba720.el7 \
will be updated
... (output truncated)
```

You can also use **yum** to install one or more packages. Do not forget to run **yum update** from time to time.

```
[root@server33 ~]# yum update -y && yum install vim -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.weepeetelecom.be
... (output truncated)
```

Refer to the package management chapter for more information on installing and removing packages.

5.9. logon from Linux and MacOSX

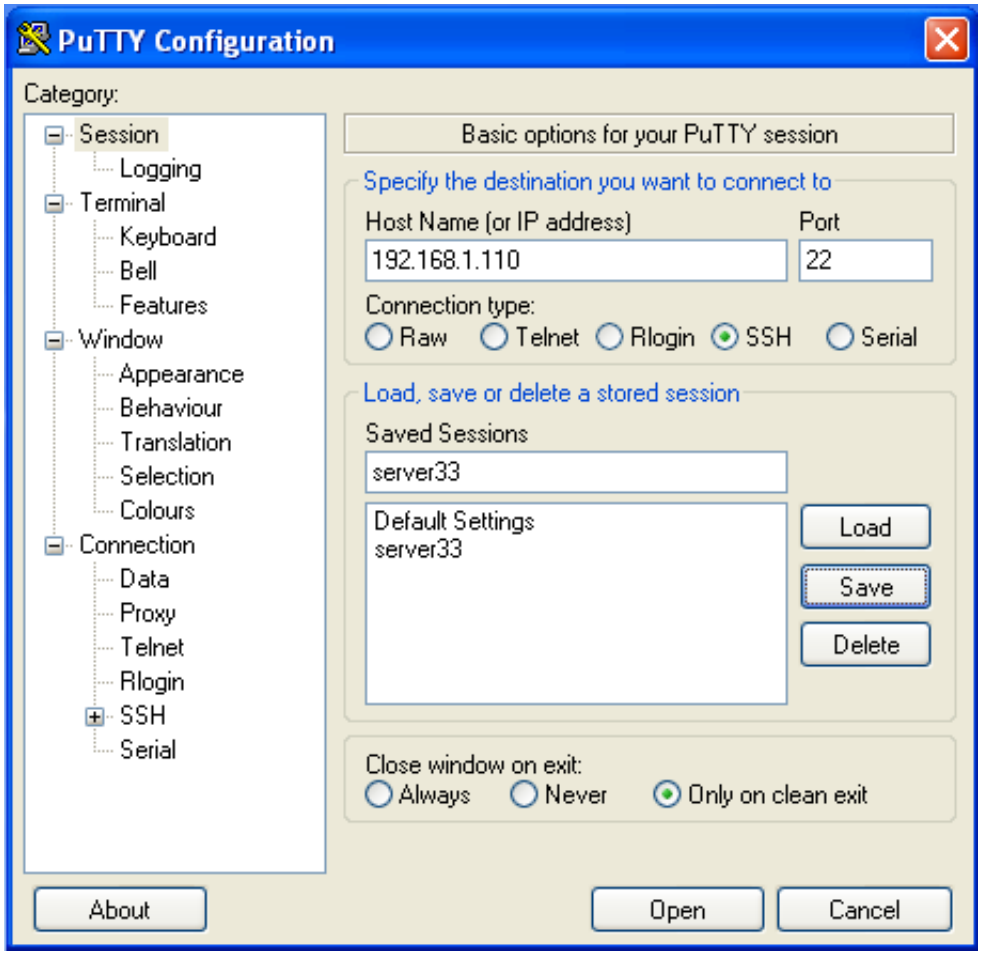
You can now open a terminal on Linux or MacOSX and use **ssh** to log on to your virtual machine.

```
paul@debian8:~$ ssh root@192.168.1.110
root@192.168.1.110's password:
Last login: Sun Nov  2 11:53:57 2014
[root@server33 ~]# hostname
server33.netsec.local
[root@server33 ~]#
```

5.10. logon from MS Windows

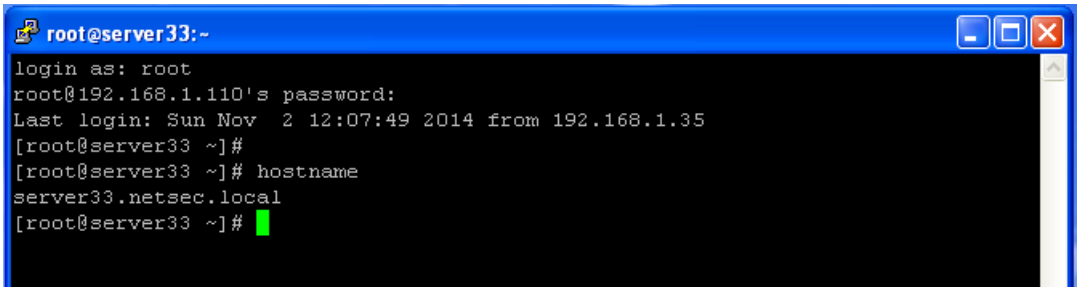
There is no **ssh** installed on MS Windows, but you can download **putty.exe** from **http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html** (just Google it).

Use **putty.exe** as shown in this screenshot (I saved the ip address by giving it a name 'server33' and presing the 'save' button).



The first time you will get a message about keys, accept this (this is explained in the ssh chapter).

Enter your userid (or root) and the correct password (nothing will appear on the screen when typing a password).



Chapter 6. getting Linux at home

This chapter shows a Ubuntu install in Virtualbox. Consider it legacy and use CentOS7 or Debian8 instead (each have their own chapter now).

This book assumes you have access to a working Linux computer. Most companies have one or more Linux servers, if you have already logged on to it, then you 're all set (skip this chapter and go to the next).

Another option is to insert a Ubuntu Linux CD in a computer with (or without) Microsoft Windows and follow the installation. Ubuntu will resize (or create) partitions and setup a menu at boot time to choose Windows or Linux.

If you do not have access to a Linux computer at the moment, and if you are unable or unsure about installing Linux on your computer, then this chapter proposes a third option: installing Linux in a virtual machine.

Installation in a virtual machine (provided by **Virtualbox**) is easy and safe. Even when you make mistakes and crash everything on the virtual Linux machine, then nothing on the real computer is touched.

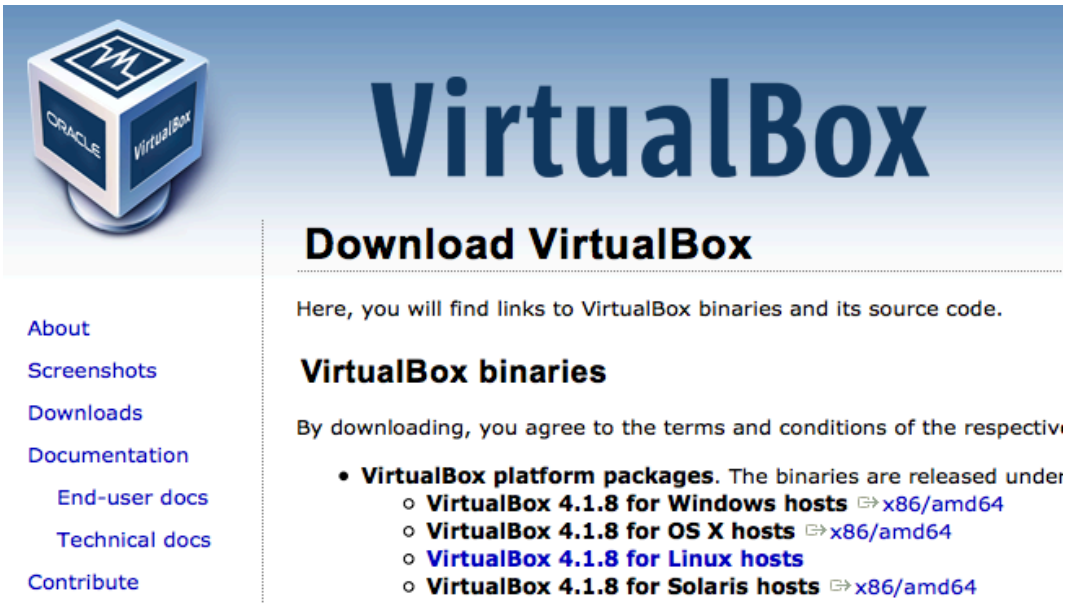
This chapter gives easy steps and screenshots to get a working Ubuntu server in a Virtualbox virtual machine. The steps are very similar to installing Fedora or CentOS or even Debian, and if you like you can also use VMWare instead of Virtualbox.

6.1. download a Linux CD image

Start by downloading a Linux CD image (an .ISO file) from the distribution of your choice from the Internet. Take care selecting the correct cpu architecture of your computer; choose **i386** if unsure. Choosing the wrong cpu type (like x86_64 when you have an old Pentium) will almost immediately fail to boot the CD.

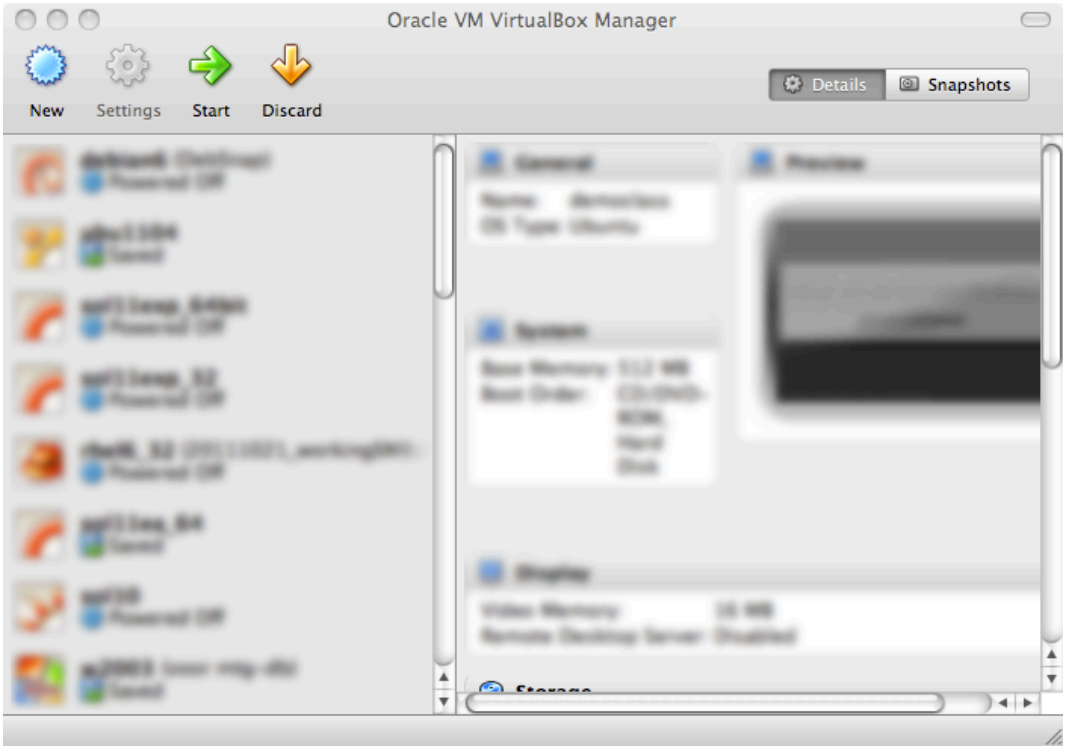
6.2. download Virtualbox

Step two (when the .ISO file has finished downloading) is to download Virtualbox. If you are currently running Microsoft Windows, then download and install Virtualbox for Windows!



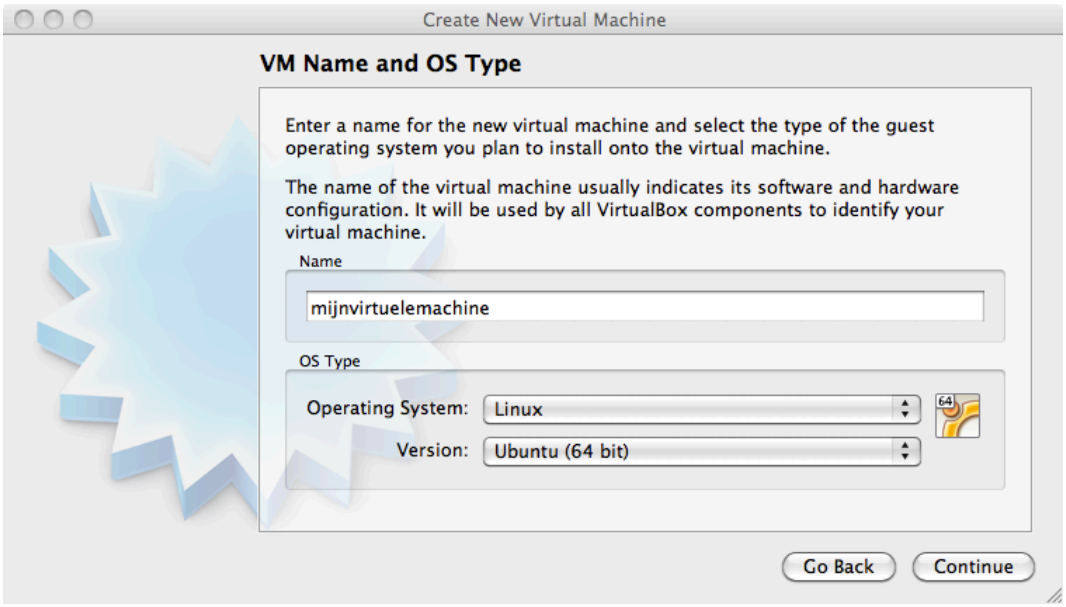
6.3. create a virtual machine

Now start Virtualbox. Contrary to the screenshot below, your left pane should be empty.

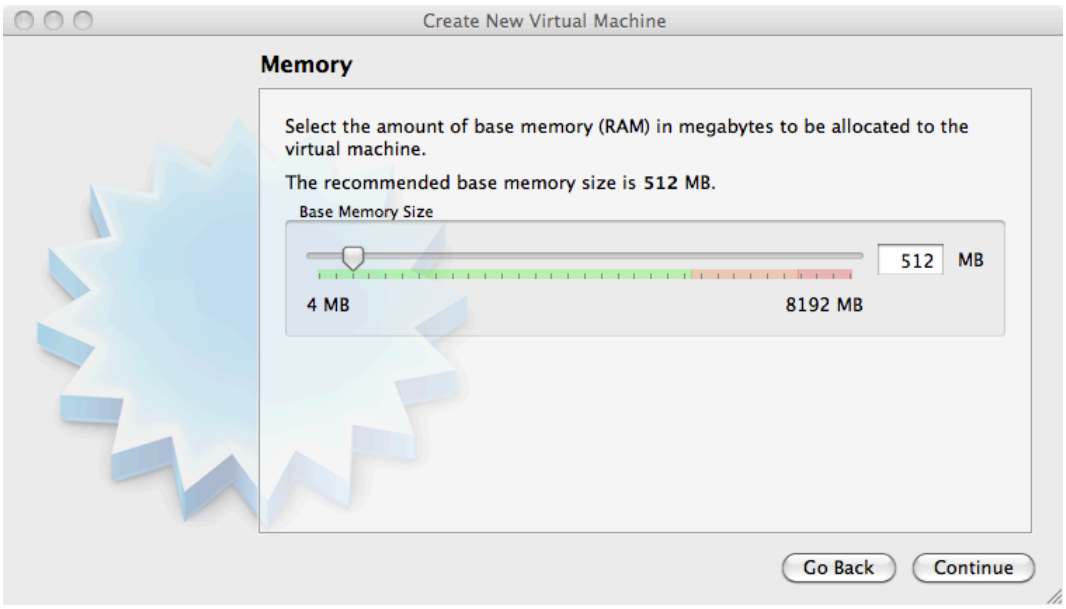


Click **New** to create a new virtual machine. We will walk together through the wizard. The screenshots below are taken on Mac OSX; they will be slightly different if you are running Microsoft Windows.

Name your virtual machine (and maybe select 32-bit or 64-bit).



Give the virtual machine some memory (512MB if you have 2GB or more, otherwise select 256MB).

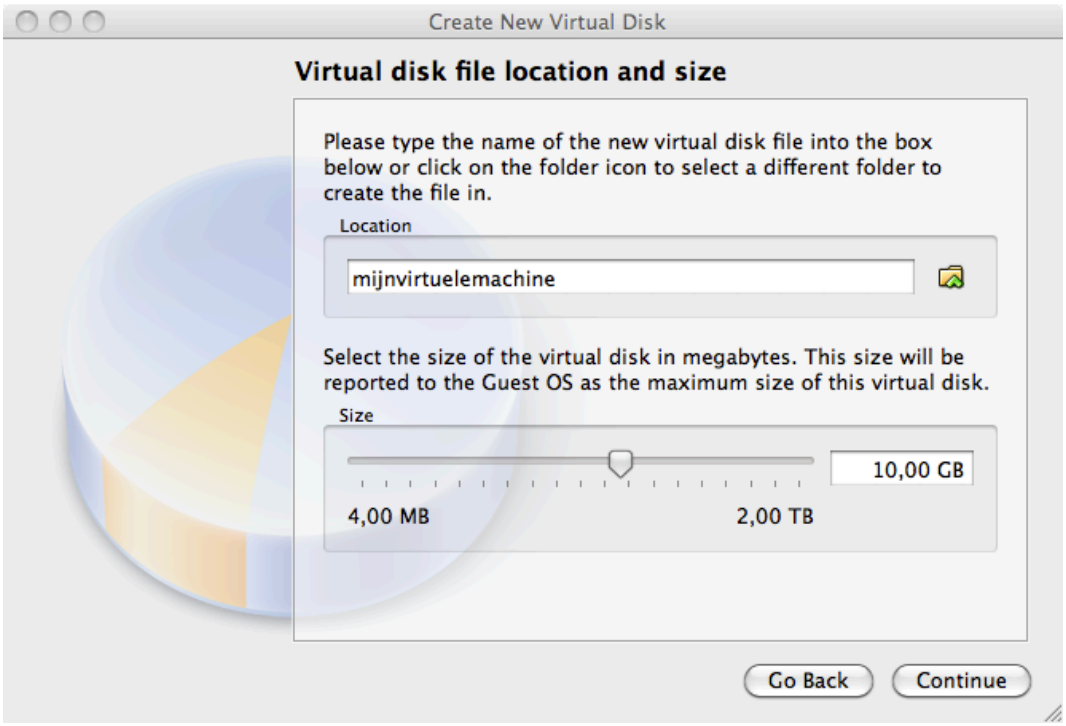


Select to create a new disk (remember, this will be a virtual disk).

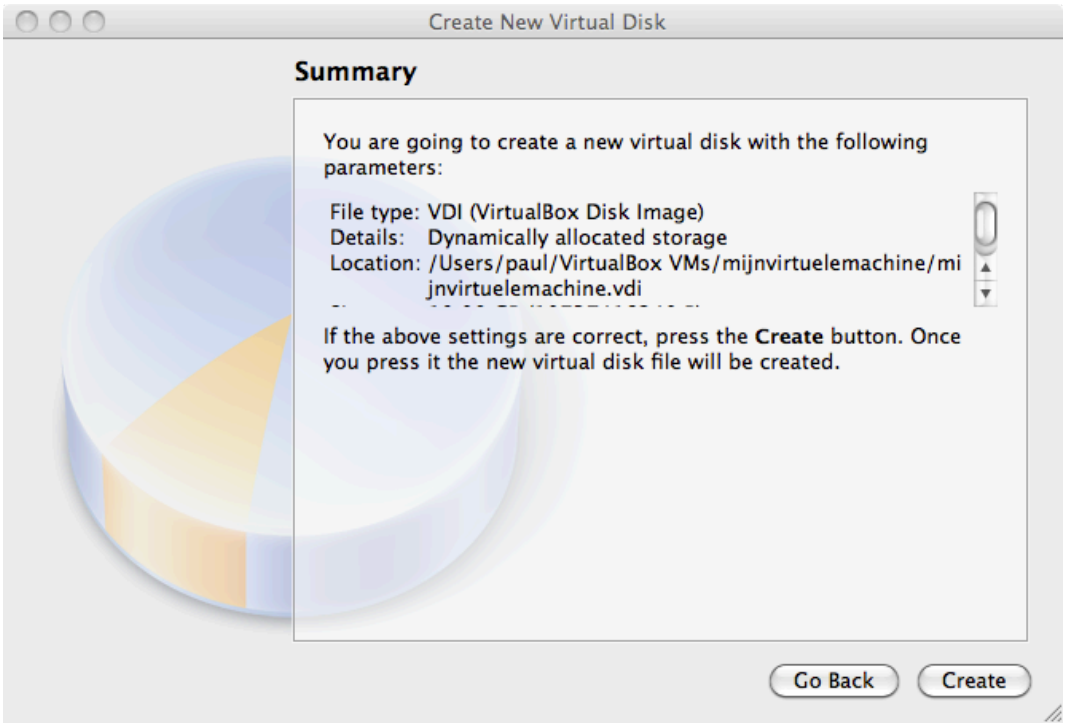
If you get the question below, choose vdi.

Choose **dynamically allocated** (fixed size is only useful in production or on really old, slow hardware).

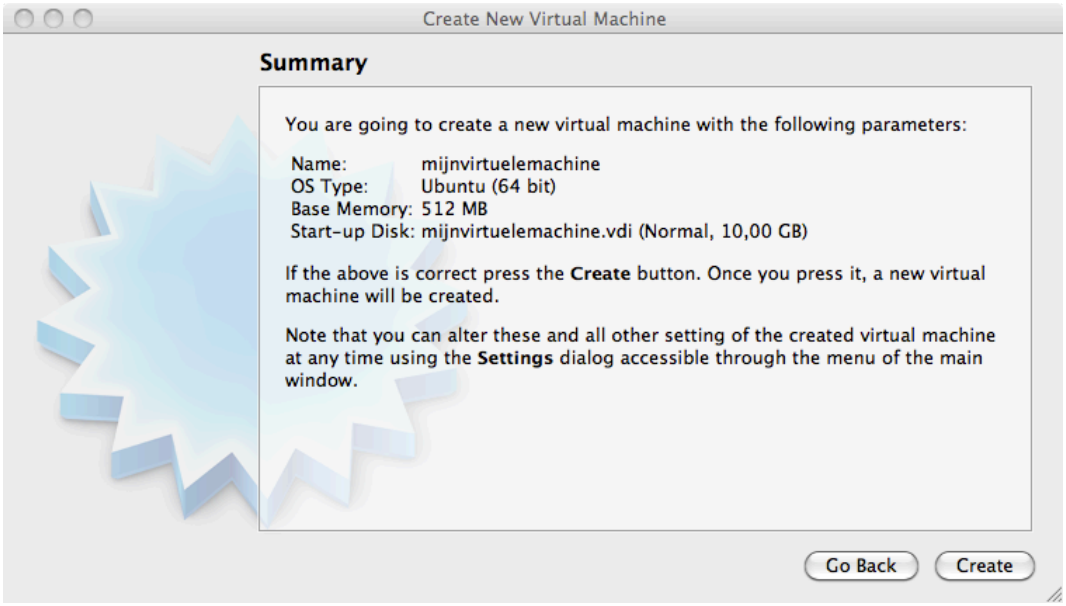
Choose between 10GB and 16GB as the disk size.



Click **create** to create the virtual disk.

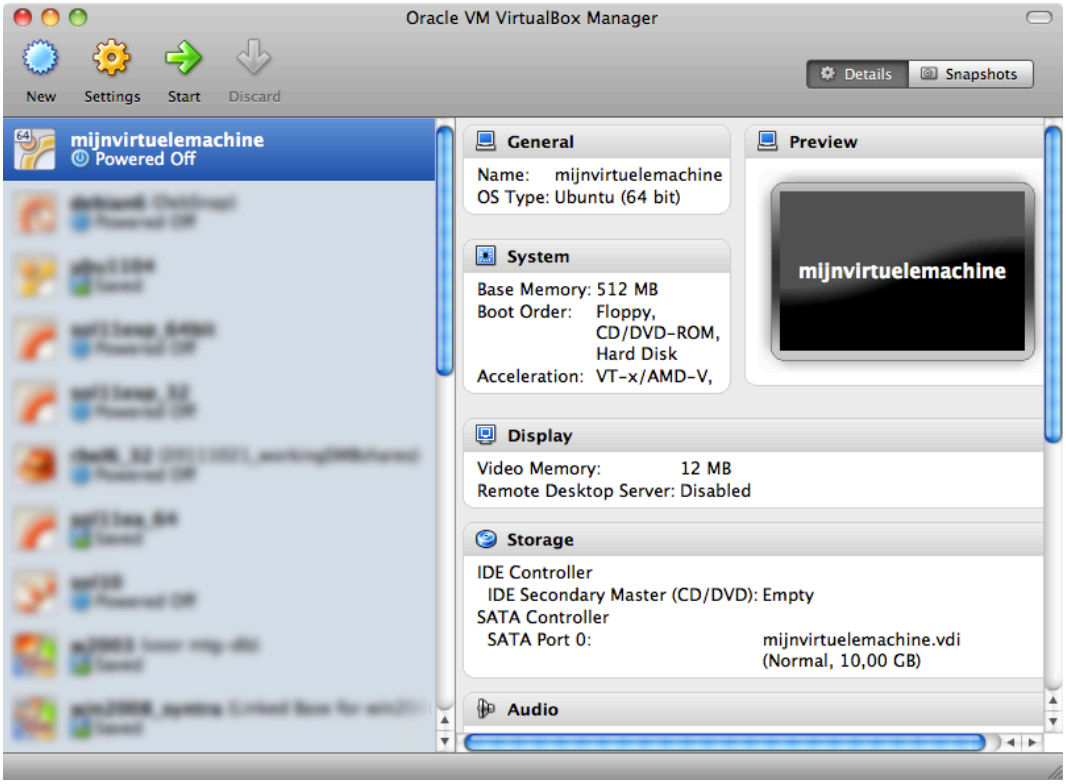


Click **create** to create the virtual machine.

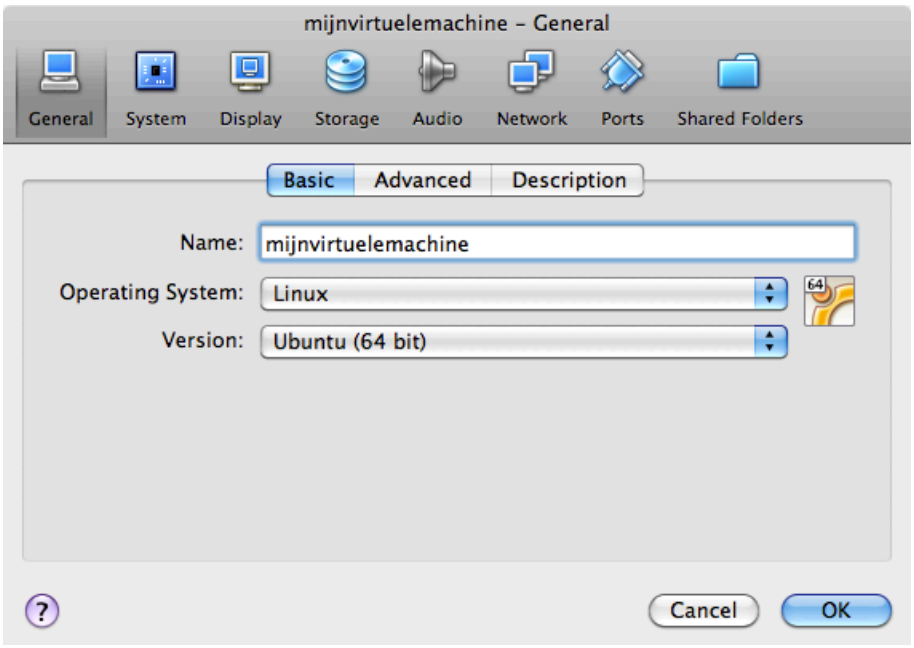


6.4. attach the CD image

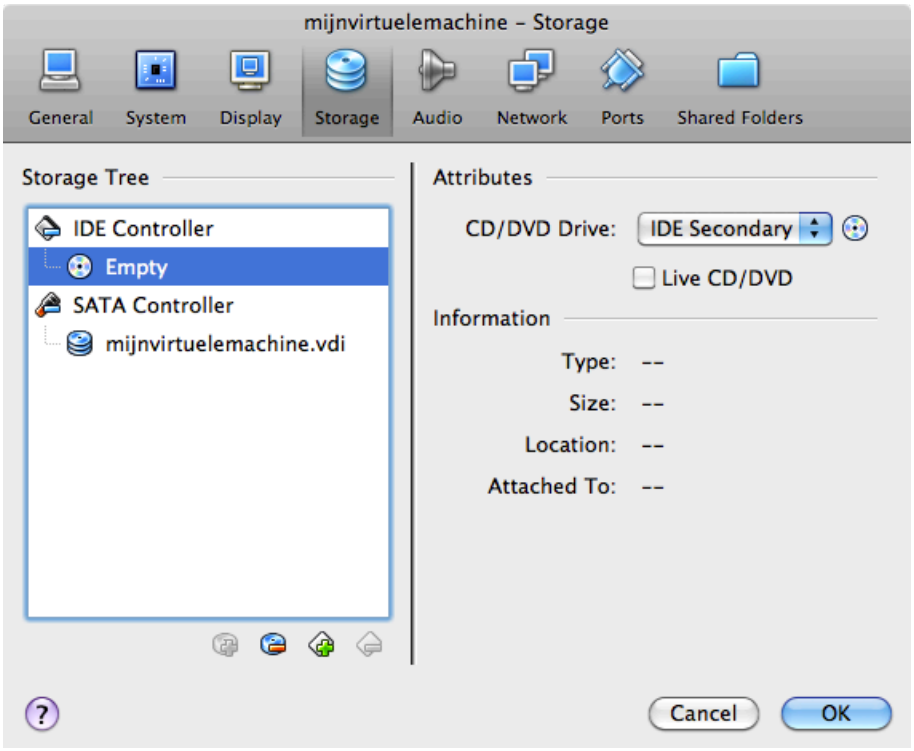
Before we start the virtual computer, let us take a look at some settings (click **Settings**).



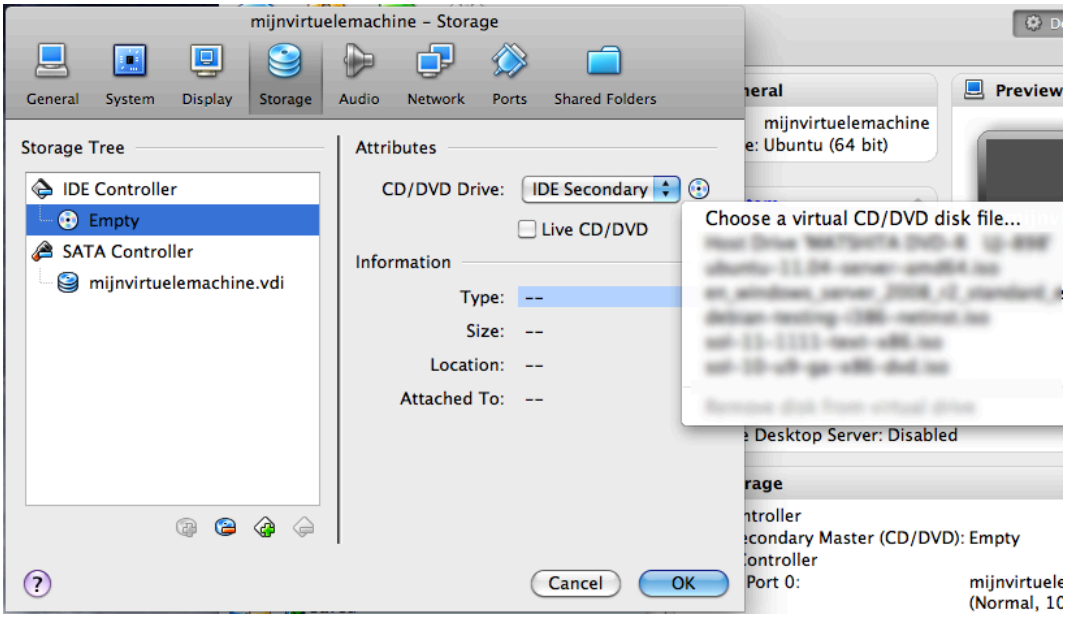
Do not worry if your screen looks different, just find the button named **storage**.



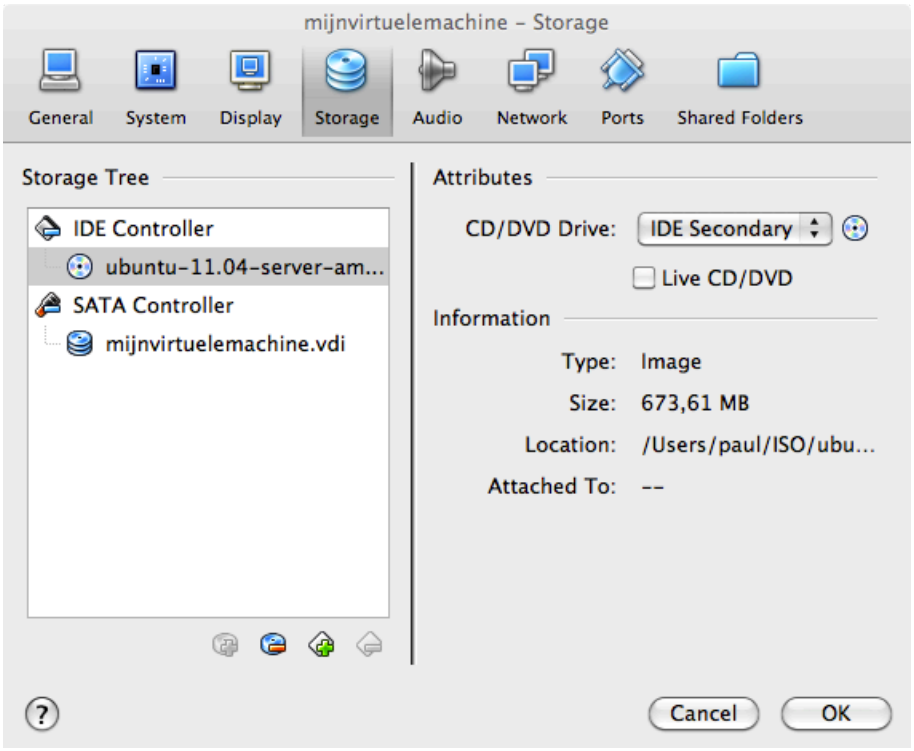
Remember the .ISO file you downloaded? Connect this .ISO file to this virtual machine by clicking on the CD icon next to **Empty**.



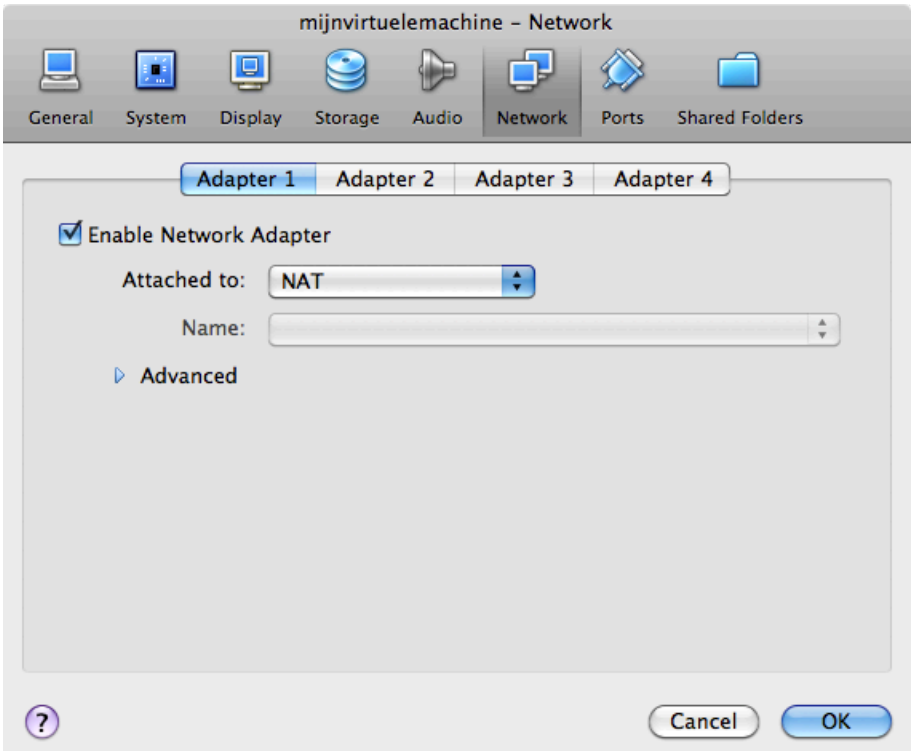
Now click on the other CD icon and attach your ISO file to this virtual CD drive.



Verify that your download is accepted. If Virtualbox complains at this point, then you probably did not finish the download of the CD (try downloading it again).



It could be useful to set the network adapter to bridge instead of NAT. Bridged usually will connect your virtual computer to the Internet.



6.5. install Linux

The virtual machine is now ready to start. When given a choice at boot, select **install** and follow the instructions on the screen. When the installation is finished, you can log on to the machine and start practising Linux!

Part III. first steps on the command line

Table of Contents

Chapter 7. man pages

This chapter will explain the use of **man** pages (also called **manual pages**) on your Unix or Linux computer.

You will learn the **man** command together with related commands like **whereis**, **whatis** and **mandb**.

Most Unix files and commands have pretty good man pages to explain their use. Man pages also come in handy when you are using multiple flavours of Unix or several Linux distributions since options and parameters sometimes vary.

7.1. man $command

Type **man** followed by a command (for which you want help) and start reading. Press **q** to quit the manpage. Some man pages contain examples (near the end).

```
paul@laika:~$ man whois
Reformatting whois(1), please wait...
```

7.2. man $configfile

Most **configuration files** have their own manual.

```
paul@laika:~$ man syslog.conf
Reformatting syslog.conf(5), please wait...
```

7.3. man $daemon

This is also true for most **daemons** (background programs) on your system..

```
paul@laika:~$ man syslogd
Reformatting syslogd(8), please wait...
```

7.4. man -k (apropos)

man -k (or **apropos**) shows a list of man pages containing a string.

```
paul@laika:~$ man -k syslog
lm-syslog-setup (8)  - configure laptop mode to switch syslog.conf ...
logger (1)           - a shell command interface to the syslog(3) ...
syslog-facility (8)  - Setup and remove LOCALx facility for sysklogd
syslog.conf (5)      - syslogd(8) configuration file
syslogd (8)          - Linux system logging utilities.
syslogd-listfiles (8) - list system logfiles
```

7.5. whatis

To see just the description of a manual page, use **whatis** followed by a string.

```
paul@u810:~$ whatis route
route (8)            - show / manipulate the IP routing table
```

7.6. whereis

The location of a manpage can be revealed with **whereis**.

```
paul@laika:~$ whereis -m whois
whois: /usr/share/man/man1/whois.1.gz
```

This file is directly readable by **man**.

```
paul@laika:~$ man /usr/share/man/man1/whois.1.gz
```

7.7. man sections

By now you will have noticed the numbers between the round brackets. **man man** will explain to you that these are section numbers. Executable programs and shell commands reside in section one.

```
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 conventions), e.g. man(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]
```

7.8. man $section $file

Therefor, when referring to the man page of the passwd command, you will see it written as **passwd(1)**; when referring to the **passwd file**, you will see it written as **passwd(5)**. The screenshot explains how to open the man page in the correct section.

```
[paul@RHEL52 ~]$ man passwd      # opens the first manual found
[paul@RHEL52 ~]$ man 5 passwd    # opens a page from section 5
```

7.9. man man

If you want to know more about **man**, then Read The Fantastic Manual (RTFM).

Unfortunately, manual pages do not have the answer to everything...

```
paul@laika:~$ man woman
No manual entry for woman
```

7.10. mandb

Should you be convinced that a man page exists, but you can't access it, then try running **mandb** on Debian/Mint.

```
root@laika:~# mandb
0 man subdirectories contained newer manual pages.
0 manual pages were added.
0 stray cats were added.
0 old database entries were purged.
```

Or run **makewhatis** on CentOS/Redhat.

```
[root@centos65 ~]# apropos scsi
scsi: nothing appropriate
[root@centos65 ~]# makewhatis
[root@centos65 ~]# apropos scsi
hpsa                 (4)  - HP Smart Array SCSI driver
lsscsi               (8)  - list SCSI devices (or hosts) and their attributes
sd                   (4)  - Driver for SCSI Disk Drives
st                   (4)  - SCSI tape device
```

Chapter 8. working with directories

This module is a brief overview of the most common commands to work with directories: **pwd**, **cd**, **ls**, **mkdir** and **rmdir**. These commands are available on any Linux (or Unix) system.

This module also discusses **absolute** and **relative paths** and **path completion** in the **bash** shell.

8.1. pwd

The **you are here** sign can be displayed with the **pwd** command (Print Working Directory). Go ahead, try it: Open a command line interface (also called a terminal, console or xterm) and type **pwd**. The tool displays your **current directory**.

```
paul@debian8:~$ pwd
/home/paul
```

8.2. cd

You can change your current directory with the **cd** command (Change Directory).

```
paul@debian8$ cd /etc
paul@debian8$ pwd
/etc
paul@debian8$ cd /bin
paul@debian8$ pwd
/bin
paul@debian8$ cd /home/paul/
paul@debian8$ pwd
/home/paul
```

8.2.1. cd ~

The **cd** is also a shortcut to get back into your home directory. Just typing **cd** without a target directory, will put you in your home directory. Typing **cd ~** has the same effect.

```
paul@debian8$ cd /etc
paul@debian8$ pwd
/etc
paul@debian8$ cd
paul@debian8$ pwd
/home/paul
paul@debian8$ cd ~
paul@debian8$ pwd
/home/paul
```

8.2.2. cd ..

To go to the **parent directory** (the one just above your current directory in the directory tree), type **cd ..** .

```
paul@debian8$ pwd
/usr/share/games
paul@debian8$ cd ..
paul@debian8$ pwd
/usr/share
```

*To stay in the current directory, type **cd .** ;-)* We will see useful use of the **.** character representing the current directory later.

8.2.3. cd -

Another useful shortcut with **cd** is to just type **cd -** to go to the previous directory.

```
paul@debian8$ pwd
/home/paul
paul@debian8$ cd /etc
paul@debian8$ pwd
/etc
paul@debian8$ cd -
/home/paul
paul@debian8$ cd -
/etc
```

8.3. absolute and relative paths

You should be aware of **absolute and relative paths** in the file tree. When you type a path starting with a **slash (/)**, then the **root** of the file tree is assumed. If you don't start your path with a slash, then the current directory is the assumed starting point.

The screenshot below first shows the current directory **/home/paul**. From within this directory, you have to type **cd /home** instead of **cd home** to go to the **/home** directory.

```
paul@debian8$ pwd
/home/paul
paul@debian8$ cd home
bash: cd: home: No such file or directory
paul@debian8$ cd /home
paul@debian8$ pwd
/home
```

When inside **/home**, you have to type **cd paul** instead of **cd /paul** to enter the subdirectory **paul** of the current directory **/home**.

```
paul@debian8$ pwd
/home
paul@debian8$ cd /paul
bash: cd: /paul: No such file or directory
paul@debian8$ cd paul
paul@debian8$ pwd
/home/paul
```

In case your current directory is the **root directory /**, then both **cd /home** and **cd home** will get you in the **/home** directory.

```
paul@debian8$ pwd
/
paul@debian8$ cd home
paul@debian8$ pwd
/home
paul@debian8$ cd /
paul@debian8$ cd /home
paul@debian8$ pwd
/home
```

This was the last screenshot with **pwd** statements. From now on, the current directory will often be displayed in the prompt. Later in this book we will explain how the shell variable **$PS1** can be configured to show this.

8.4. path completion

The **tab key** can help you in typing a path without errors. Typing **cd /et** followed by the **tab key** will expand the command line to **cd /etc/**. When typing **cd /Et** followed by the **tab key**, nothing will happen because you typed the wrong **path** (upper case E).

You will need fewer key strokes when using the **tab key**, and you will be sure your typed **path** is correct!

8.5. ls

You can list the contents of a directory with **ls**.

```
paul@debian8:~$ ls
allfiles.txt  dmesg.txt  services   stuff  summer.txt
paul@debian8:~$
```

8.5.1. ls -a

A frequently used option with ls is **-a** to show all files. Showing all files means including the **hidden files**. When a file name on a Linux file system starts with a dot, it is considered a **hidden file** and it doesn't show up in regular file listings.

```
paul@debian8:~$ ls
allfiles.txt  dmesg.txt  services  stuff  summer.txt
paul@debian8:~$ ls -a
.   allfiles.txt   .bash_profile  dmesg.txt   .lesshst  stuff
..  .bash_history  .bashrc        services    .ssh      summer.txt
paul@debian8:~$
```

8.5.2. ls -l

Many times you will be using options with **ls** to display the contents of the directory in different formats or to display different parts of the directory. Typing just **ls** gives you a list of files in the directory. Typing **ls -l** (that is a letter L, not the number 1) gives you a long listing.

```
paul@debian8:~$ ls -l
total 17296
-rw-r--r-- 1 paul paul 17584442 Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul    96650 Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul    19558 Sep 17 00:04 services
drwxr-xr-x 2 paul paul     4096 Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul        0 Sep 17 00:04 summer.txt
```

8.5.3. ls -lh

Another frequently used ls option is **-h**. It shows the numbers (file sizes) in a more human readable format. Also shown below is some variation in the way you can give the options to **ls**. We will explain the details of the output later in this book.

Note that we use the letter L as an option in this screenshot, not the number 1.

```
paul@debian8:~$ ls -l -h
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$ ls -lh
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$ ls -hl
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$ ls -h -l
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$
```

8.6. mkdir

Walking around the Unix file tree is fun, but it is even more fun to create your own directories with **mkdir**. You have to give at least one parameter to **mkdir**, the name of the new directory to be created. Think before you type a leading / .

```
paul@debian8:~$ mkdir mydir
paul@debian8:~$ cd mydir
paul@debian8:~/mydir$ ls -al
total 8
drwxr-xr-x  2 paul paul 4096 Sep 17 00:07 .
drwxr-xr-x 48 paul paul 4096 Sep 17 00:07 ..
paul@debian8:~/mydir$ mkdir stuff
paul@debian8:~/mydir$ mkdir otherstuff
paul@debian8:~/mydir$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 otherstuff
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 stuff
paul@debian8:~/mydir$
```

8.6.1. mkdir -p

The following command will fail, because the **parent directory** of **threedirsdeep** does not exist.

```
paul@debian8:~$ mkdir mydir2/mysubdir2/threedirsdeep
mkdir: cannot create directory ‘mydir2/mysubdir2/threedirsdeep’: No such fi\
le or directory
```

When given the option **-p**, then **mkdir** will create **parent directories** as needed.

```
paul@debian8:~$ mkdir -p mydir2/mysubdir2/threedirsdeep
paul@debian8:~$ cd mydir2
paul@debian8:~/mydir2$ ls -l
total 4
drwxr-xr-x 3 paul paul 4096 Sep 17 00:11 mysubdir2
paul@debian8:~/mydir2$ cd mysubdir2
paul@debian8:~/mydir2/mysubdir2$ ls -l
total 4
drwxr-xr-x 2 paul paul 4096 Sep 17 00:11 threedirsdeep
paul@debian8:~/mydir2/mysubdir2$ cd threedirsdeep/
paul@debian8:~/mydir2/mysubdir2/threedirsdeep$ pwd
/home/paul/mydir2/mysubdir2/threedirsdeep
```

8.7. rmdir

When a directory is empty, you can use **rmdir** to remove the directory.

```
paul@debian8:~/mydir$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 otherstuff
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 stuff
paul@debian8:~/mydir$ rmdir otherstuff
paul@debian8:~/mydir$ cd ..
paul@debian8:~$ rmdir mydir
rmdir: failed to remove ‘mydir’: Directory not empty
paul@debian8:~$ rmdir mydir/stuff
paul@debian8:~$ rmdir mydir
paul@debian8:~$
```

8.7.1. rmdir -p

And similar to the **mkdir -p** option, you can also use **rmdir** to recursively remove directories.

```
paul@debian8:~$ mkdir -p test42/subdir
paul@debian8:~$ rmdir -p test42/subdir
paul@debian8:~$
```

8.8. practice: working with directories

1. Display your current directory.

2. Change to the /etc directory.

3. Now change to your home directory using only three key presses.

4. Change to the /boot/grub directory using only eleven key presses.

5. Go to the parent directory of the current directory.

6. Go to the root directory.

7. List the contents of the root directory.

8. List a long listing of the root directory.

9. Stay where you are, and list the contents of /etc.

10. Stay where you are, and list the contents of /bin and /sbin.

11. Stay where you are, and list the contents of ~.

12. List all the files (including hidden files) in your home directory.

13. List the files in /boot in a human readable format.

14. Create a directory testdir in your home directory.

15. Change to the /etc directory, stay here and create a directory newdir in your home directory.

16. Create in one command the directories ~/dir1/dir2/dir3 (dir3 is a subdirectory from dir2, and dir2 is a subdirectory from dir1).

17. Remove the directory testdir.

18. If time permits (or if you are waiting for other students to finish this practice), use and understand **pushd** and **popd**. Use the man page of **bash** to find information about these commands.

8.9. solution: working with directories

1. Display your current directory.

```
pwd
```

2. Change to the /etc directory.

```
cd /etc
```

3. Now change to your home directory using only three key presses.

```
cd (and the enter key)
```

4. Change to the /boot/grub directory using only eleven key presses.

```
cd /boot/grub (use the tab key)
```

5. Go to the parent directory of the current directory.

```
cd .. (with space between cd and ..)
```

6. Go to the root directory.

```
cd /
```

7. List the contents of the root directory.

```
ls
```

8. List a long listing of the root directory.

```
ls -l
```

9. Stay where you are, and list the contents of /etc.

```
ls /etc
```

10. Stay where you are, and list the contents of /bin and /sbin.

```
ls /bin /sbin
```

11. Stay where you are, and list the contents of ~.

```
ls ~
```

12. List all the files (including hidden files) in your home directory.

```
ls -al ~
```

13. List the files in /boot in a human readable format.

```
ls -lh /boot
```

14. Create a directory testdir in your home directory.

```
mkdir ~/testdir
```

15. Change to the /etc directory, stay here and create a directory newdir in your home directory.

```
cd /etc ; mkdir ~/newdir
```

16. Create in one command the directories ~/dir1/dir2/dir3 (dir3 is a subdirectory from dir2, and dir2 is a subdirectory from dir1).

```
mkdir -p ~/dir1/dir2/dir3
```

17. Remove the directory testdir.

```
rmdir testdir
```

18. If time permits (or if you are waiting for other students to finish this practice), use and understand **pushd** and **popd**. Use the man page of **bash** to find information about these commands.

```
man bash            # opens the manual
/pushd              # searches for pushd
n                   # next (do this two/three times)
```

The Bash shell has two built-in commands called **pushd** and **popd**. Both commands work with a common stack of previous directories. Pushd adds a directory to the stack and changes to a new current directory, popd removes a directory from the stack and sets the current directory.

```
paul@debian7:/etc$ cd /bin
paul@debian7:/bin$ pushd /lib
/lib /bin
paul@debian7:/lib$ pushd /proc
/proc /lib /bin
paul@debian7:/proc$ popd
/lib /bin
paul@debian7:/lib$ popd
/bin
```

Chapter 9. working with files

In this chapter we learn how to recognise, create, remove, copy and move files using commands like **file, touch, rm, cp, mv** and **rename**.

9.1. all files are case sensitive

Files on Linux (or any Unix) are **case sensitive**. This means that **FILE1** is different from **file1**, and **/etc/hosts** is different from **/etc/Hosts** (the latter one does not exist on a typical Linux computer).

This screenshot shows the difference between two files, one with upper case **W**, the other with lower case **w**.

```
paul@laika:~/Linux$ ls
winter.txt  Winter.txt
paul@laika:~/Linux$ cat winter.txt
It is cold.
paul@laika:~/Linux$ cat Winter.txt
It is very cold!
```

9.2. everything is a file

A **directory** is a special kind of **file**, but it is still a (case sensitive!) **file**. Each terminal window (for example **/dev/pts/4**), any hard disk or partition (for example **/dev/sdb1**) and any process are all represented somewhere in the **file system** as a **file**. It will become clear throughout this course that everything on Linux is a **file**.

9.3. file

The **file** utility determines the file type. Linux does not use extensions to determine the file type. The command line does not care whether a file ends in .txt or .pdf. As a system administrator, you should use the **file** command to determine the file type. Here are some examples on a typical Linux system.

```
paul@laika:~$ file pic33.png
pic33.png: PNG image data, 3840 x 1200, 8-bit/color RGBA, non-interlaced
paul@laika:~$ file /etc/passwd
/etc/passwd: ASCII text
paul@laika:~$ file HelloWorld.c
HelloWorld.c: ASCII C program text
```

The file command uses a magic file that contains patterns to recognise file types. The magic file is located in **/usr/share/file/magic**. Type **man 5 magic** for more information.

It is interesting to point out **file -s** for special files like those in **/dev** and **/proc**.

```
root@debian6~# file /dev/sda
/dev/sda: block special
root@debian6~# file -s /dev/sda
/dev/sda: x86 boot sector; partition 1: ID=0x83, active, starthead...
root@debian6~# file /proc/cpuinfo
/proc/cpuinfo: empty
root@debian6~# file -s /proc/cpuinfo
/proc/cpuinfo: ASCII C++ program text
```

9.4. touch

9.4.1. create an empty file

One easy way to create an empty file is with **touch**. (We will see many other ways for creating files later in this book.)

This screenshot starts with an empty directory, creates two files with **touch** and the lists those files.

```
paul@debian7:~$ ls -l
total 0
paul@debian7:~$ touch file42
paul@debian7:~$ touch file33
paul@debian7:~$ ls -l
total 0
-rw-r--r-- 1 paul paul 0 Oct 15 08:57 file33
-rw-r--r-- 1 paul paul 0 Oct 15 08:56 file42
paul@debian7:~$
```

9.4.2. touch -t

The **touch** command can set some properties while creating empty files. Can you determine what is set by looking at the next screenshot? If not, check the manual for **touch**.

```
paul@debian7:~$ touch -t 200505050000 SinkoDeMayo
paul@debian7:~$ touch -t 130207111630 BigBattle.txt
paul@debian7:~$ ls -l
total 0
-rw-r--r-- 1 paul paul 0 Jul 11  1302 BigBattle.txt
-rw-r--r-- 1 paul paul 0 Oct 15 08:57 file33
-rw-r--r-- 1 paul paul 0 Oct 15 08:56 file42
-rw-r--r-- 1 paul paul 0 May  5  2005 SinkoDeMayo
paul@debian7:~$
```

9.5. rm

9.5.1. remove forever

When you no longer need a file, use **rm** to remove it. Unlike some graphical user interfaces, the command line in general does not have a **waste bin** or **trash can** to recover files. When you use **rm** to remove a file, the file is gone. Therefore, be careful when removing files!

```
paul@debian7:~$ ls
BigBattle.txt  file33  file42  SinkoDeMayo
paul@debian7:~$ rm BigBattle.txt
paul@debian7:~$ ls
file33  file42  SinkoDeMayo
paul@debian7:~$
```

9.5.2. rm -i

To prevent yourself from accidentally removing a file, you can type **rm -i**.

```
paul@debian7:~$ ls
file33  file42  SinkoDeMayo
paul@debian7:~$ rm -i file33
rm: remove regular empty file `file33'? yes
paul@debian7:~$ rm -i SinkoDeMayo
rm: remove regular empty file `SinkoDeMayo'? n
paul@debian7:~$ ls
file42  SinkoDeMayo
paul@debian7:~$
```

9.5.3. rm -rf

By default, **rm -r** will not remove non-empty directories. However **rm** accepts several options that will allow you to remove any directory. The **rm -rf** statement is famous because it will erase anything (providing that you have the permissions to do so). When you are logged on as root, be very careful with **rm -rf** (the **f** means **force** and the **r** means **recursive**) since being root implies that permissions don't apply to you. You can literally erase your entire file system by accident.

```
paul@debian7:~$ mkdir test
paul@debian7:~$ rm test
rm: cannot remove `test': Is a directory
paul@debian7:~$ rm -rf test
paul@debian7:~$ ls test
ls: cannot access test: No such file or directory
paul@debian7:~$
```

9.6. cp

9.6.1. copy one file

To copy a file, use **cp** with a source and a target argument.

```
paul@debian7:~$ ls
file42  SinkoDeMayo
paul@debian7:~$ cp file42 file42.copy
paul@debian7:~$ ls
file42  file42.copy  SinkoDeMayo
```

9.6.2. copy to another directory

If the target is a directory, then the source files are copied to that target directory.

```
paul@debian7:~$ mkdir dir42
paul@debian7:~$ cp SinkoDeMayo dir42
paul@debian7:~$ ls dir42/
SinkoDeMayo
```

9.6.3. cp -r

To copy complete directories, use **cp -r** (the **-r** option forces **recursive** copying of all files in all subdirectories).

```
paul@debian7:~$ ls
dir42  file42  file42.copy  SinkoDeMayo
paul@debian7:~$ cp -r dir42/ dir33
paul@debian7:~$ ls
dir33  dir42  file42  file42.copy  SinkoDeMayo
paul@debian7:~$ ls dir33/
SinkoDeMayo
```

9.6.4. copy multiple files to directory

You can also use cp to copy multiple files into a directory. In this case, the last argument (a.k.a. the target) must be a directory.

```
paul@debian7:~$ cp file42 file42.copy SinkoDeMayo dir42/
paul@debian7:~$ ls dir42/
file42  file42.copy  SinkoDeMayo
```

9.6.5. cp -i

To prevent **cp** from overwriting existing files, use the **-i** (for interactive) option.

```
paul@debian7:~$ cp SinkoDeMayo file42
paul@debian7:~$ cp SinkoDeMayo file42
paul@debian7:~$ cp -i SinkoDeMayo file42
cp: overwrite `file42'? n
paul@debian7:~$
```

9.7. mv

9.7.1. rename files with mv

Use **mv** to rename a file or to move the file to another directory.

```
paul@debian7:~$ ls
dir33  dir42  file42  file42.copy  SinkoDeMayo
paul@debian7:~$ mv file42 file33
paul@debian7:~$ ls
dir33  dir42  file33  file42.copy  SinkoDeMayo
paul@debian7:~$
```

When you need to rename only one file then **mv** is the preferred command to use.

9.7.2. rename directories with mv

The same **mv** command can be used to rename directories.

```
paul@debian7:~$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir33
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir42
-rw-r--r-- 1 paul paul    0 Oct 15 09:38 file33
-rw-r--r-- 1 paul paul    0 Oct 15 09:16 file42.copy
-rw-r--r-- 1 paul paul    0 May  5  2005 SinkoDeMayo
paul@debian7:~$ mv dir33 backup
paul@debian7:~$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 backup
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir42
-rw-r--r-- 1 paul paul    0 Oct 15 09:38 file33
-rw-r--r-- 1 paul paul    0 Oct 15 09:16 file42.copy
-rw-r--r-- 1 paul paul    0 May  5  2005 SinkoDeMayo
paul@debian7:~$
```

9.7.3. mv -i

The **mv** also has a **-i** switch similar to **cp** and **rm**.

this screenshot shows that **mv -i** will ask permission to overwrite an existing file.

```
paul@debian7:~$ mv -i file33 SinkoDeMayo
mv: overwrite `SinkoDeMayo'? no
paul@debian7:~$
```

9.8. rename

9.8.1. about rename

The **rename** command is one of the rare occasions where the Linux Fundamentals book has to make a distinction between Linux distributions. Almost every command in the **Fundamentals** part of this book works on almost every Linux computer. But **rename** is different.

Try to use **mv** whenever you need to rename only a couple of files.

9.8.2. rename on Debian/Ubuntu

The **rename** command on Debian uses regular expressions (regular expression or shor regex are explained in a later chapter) to rename many files at once.

Below a **rename** example that switches all occurrences of txt to png for all file names ending in .txt.

```
paul@debian7:~/test42$ ls
abc.txt  file33.txt  file42.txt
paul@debian7:~/test42$ rename 's/\.txt/\.png/' *.txt
paul@debian7:~/test42$ ls
abc.png  file33.png  file42.png
```

This second example switches all (first) occurrences of **file** into **document** for all file names ending in .png.

```
paul@debian7:~/test42$ ls
abc.png  file33.png  file42.png
paul@debian7:~/test42$ rename 's/file/document/' *.png
paul@debian7:~/test42$ ls
abc.png  document33.png  document42.png
paul@debian7:~/test42$
```

9.8.3. rename on CentOS/RHEL/Fedora

On Red Hat Enterprise Linux, the syntax of **rename** is a bit different. The first example below renames all *.conf files replacing any occurrence of .conf with .backup.

```
[paul@centos7 ~]$ touch one.conf two.conf three.conf
[paul@centos7 ~]$ rename .conf .backup *.conf
[paul@centos7 ~]$ ls
one.backup  three.backup  two.backup
[paul@centos7 ~]$
```

The second example renames all (*) files replacing one with ONE.

```
[paul@centos7 ~]$ ls
one.backup  three.backup  two.backup
[paul@centos7 ~]$ rename one ONE *
[paul@centos7 ~]$ ls
ONE.backup  three.backup  two.backup
[paul@centos7 ~]$
```

9.9. practice: working with files

1. List the files in the /bin directory

2. Display the type of file of /bin/cat, /etc/passwd and /usr/bin/passwd.

3a. Download wolf.jpg and LinuxFun.pdf from http://linux-training.be (wget http://linux-training.be/files/studentfiles/wolf.jpg and wget http://linux-training.be/files/books/LinuxFun.pdf)

```
wget http://linux-training.be/files/studentfiles/wolf.jpg
wget http://linux-training.be/files/studentfiles/wolf.png
wget http://linux-training.be/files/books/LinuxFun.pdf
```

3b. Display the type of file of wolf.jpg and LinuxFun.pdf

3c. Rename wolf.jpg to wolf.pdf (use mv).

3d. Display the type of file of wolf.pdf and LinuxFun.pdf.

4. Create a directory ~/touched and enter it.

5. Create the files today.txt and yesterday.txt in touched.

6. Change the date on yesterday.txt to match yesterday's date.

7. Copy yesterday.txt to copy.yesterday.txt

8. Rename copy.yesterday.txt to kim

9. Create a directory called ~/testbackup and copy all files from ~/touched into it.

10. Use one command to remove the directory ~/testbackup and all files into it.

11. Create a directory ~/etcbackup and copy all *.conf files from /etc into it. Did you include all subdirectories of /etc ?

12. Use rename to rename all *.conf files to *.backup . (if you have more than one distro available, try it on all!)

9.10. solution: working with files

1. List the files in the /bin directory

```
ls /bin
```

2. Display the type of file of /bin/cat, /etc/passwd and /usr/bin/passwd.

```
file /bin/cat /etc/passwd /usr/bin/passwd
```

3a. Download wolf.jpg and LinuxFun.pdf from http://linux-training.be (wget http://linux-training.be/files/studentfiles/wolf.jpg and wget http://linux-training.be/files/books/LinuxFun.pdf)

```
wget http://linux-training.be/files/studentfiles/wolf.jpg
wget http://linux-training.be/files/studentfiles/wolf.png
wget http://linux-training.be/files/books/LinuxFun.pdf
```

3b. Display the type of file of wolf.jpg and LinuxFun.pdf

```
file wolf.jpg LinuxFun.pdf
```

3c. Rename wolf.jpg to wolf.pdf (use mv).

```
mv wolf.jpg wolf.pdf
```

3d. Display the type of file of wolf.pdf and LinuxFun.pdf.

```
file wolf.pdf LinuxFun.pdf
```

4. Create a directory ~/touched and enter it.

```
mkdir ~/touched ; cd ~/touched
```

5. Create the files today.txt and yesterday.txt in touched.

```
touch today.txt yesterday.txt
```

6. Change the date on yesterday.txt to match yesterday's date.

```
touch -t 200810251405 yesterday.txt (substitute 20081025 with yesterday)
```

7. Copy yesterday.txt to copy.yesterday.txt

```
cp yesterday.txt copy.yesterday.txt
```

8. Rename copy.yesterday.txt to kim

```
mv copy.yesterday.txt kim
```

9. Create a directory called ~/testbackup and copy all files from ~/touched into it.

```
mkdir ~/testbackup ; cp -r ~/touched ~/testbackup/
```

10. Use one command to remove the directory ~/testbackup and all files into it.

```
rm -rf ~/testbackup
```

11. Create a directory ~/etcbackup and copy all *.conf files from /etc into it. Did you include all subdirectories of /etc ?

```
cp -r /etc/*.conf ~/etcbackup
```

```
Only *.conf files that are directly in /etc/ are copied.
```

12. Use rename to rename all *.conf files to *.backup . (if you have more than one distro available, try it on all!)

```
On RHEL: touch 1.conf 2.conf ; rename conf backup *.conf
```

```
On Debian: touch 1.conf 2.conf ; rename 's/conf/backup/' *.conf
```

Chapter 10. working with file contents

In this chapter we will look at the contents of **text files** with **head, tail, cat, tac, more, less** and **strings**.

We will also get a glimpse of the possibilities of tools like **cat** on the command line.

10.1. head

You can use **head** to display the first ten lines of a file.

```
paul@debian7~$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
root@debian7~#
```

The **head** command can also display the first **n** lines of a file.

```
paul@debian7~$ head -4 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
paul@debian7~$
```

And **head** can also display the first **n bytes**.

```
paul@debian7~$ head -c14 /etc/passwd
root:x:0:0:roopaul@debian7~$
```

10.2. tail

Similar to **head**, the **tail** command will display the last ten lines of a file.

```
paul@debian7~$ tail /etc/services
vboxd           20012/udp
binkp           24554/tcp                       # binkp fidonet protocol
asp             27374/tcp                       # Address Search Protocol
asp             27374/udp
csync2          30865/tcp                       # cluster synchronization tool
dircproxy       57000/tcp                       # Detachable IRC Proxy
tfido           60177/tcp                       # fidonet EMSI over telnet
fido            60179/tcp                       # fidonet EMSI over TCP

# Local services
paul@debian7~$
```

You can give **tail** the number of lines you want to see.

```
paul@debian7~$ tail -3 /etc/services
fido            60179/tcp                       # fidonet EMSI over TCP

# Local services
paul@debian7~$
```

The **tail** command has other useful options, some of which we will use during this course.

10.3. cat

The **cat** command is one of the most universal tools, yet all it does is copy **standard input** to **standard output**. In combination with the shell this can be very powerful and diverse. Some examples will give a glimpse into the possibilities. The first example is simple, you can use cat to display a file on the screen. If the file is longer than the screen, it will scroll to the end.

```
paul@debian8:~$ cat /etc/resolv.conf
domain linux-training.be
search linux-training.be
nameserver 192.168.1.42
```

10.3.1. concatenate

cat is short for **concatenate**. One of the basic uses of **cat** is to concatenate files into a bigger (or complete) file.

```
paul@debian8:~$ echo one >part1
paul@debian8:~$ echo two >part2
paul@debian8:~$ echo three >part3
paul@debian8:~$ cat part1
one
paul@debian8:~$ cat part2
two
paul@debian8:~$ cat part3
three
paul@debian8:~$ cat part1 part2 part3
one
two
three
paul@debian8:~$ cat part1 part2 part3 >all
paul@debian8:~$ cat all
one
two
three
paul@debian8:~$
```

10.3.2. create files

You can use **cat** to create flat text files. Type the **cat > winter.txt** command as shown in the screenshot below. Then type one or more lines, finishing each line with the enter key. After the last line, type and hold the Control (Ctrl) key and press d.

```
paul@debian8:~$ cat > winter.txt
It is very cold today!
paul@debian8:~$ cat winter.txt
It is very cold today!
paul@debian8:~$
```

The **Ctrl d** key combination will send an **EOF** (End of File) to the running process ending the **cat** command.

10.3.3. custom end marker

You can choose an end marker for **cat** with << as is shown in this screenshot. This construction is called a **here directive** and will end the **cat** command.

```
paul@debian8:~$ cat > hot.txt <<stop
> It is hot today!
> Yes it is summer.
> stop
paul@debian8:~$ cat hot.txt
It is hot today!
Yes it is summer.
paul@debian8:~$
```

10.3.4. copy files

In the third example you will see that cat can be used to copy files. We will explain in detail what happens here in the bash shell chapter.

```
paul@debian8:~$ cat winter.txt
It is very cold today!
paul@debian8:~$ cat winter.txt > cold.txt
paul@debian8:~$ cat cold.txt
It is very cold today!
paul@debian8:~$
```

10.4. tac

Just one example will show you the purpose of **tac** (cat backwards).

```
paul@debian8:~$ cat count
one
two
three
four
paul@debian8:~$ tac count
four
three
two
one
```

10.5. more and less

The **more** command is useful for displaying files that take up more than one screen. More will allow you to see the contents of the file page by page. Use the space bar to see the next page, or **q** to quit. Some people prefer the **less** command to **more**.

10.6. strings

With the **strings** command you can display readable ascii strings found in (binary) files. This example locates the **ls** binary then displays readable strings in the binary file (output is truncated).

```
paul@laika:~$ which ls
/bin/ls
paul@laika:~$ strings /bin/ls
/lib/ld-linux.so.2
librt.so.1
__gmon_start__
_Jv_RegisterClasses
clock_gettime
libacl.so.1
...
```

10.7. practice: file contents

1. Display the first 12 lines of **/etc/services**.

2. Display the last line of **/etc/passwd**.

3. Use cat to create a file named **count.txt** that looks like this:

```
One
Two
Three
Four
Five
```

4. Use **cp** to make a backup of this file to **cnt.txt**.

5. Use **cat** to make a backup of this file to **catcnt.txt**.

6. Display **catcnt.txt**, but with all lines in reverse order (the last line first).

7. Use more to display **/etc/services**.

8. Display the readable character strings from the **/usr/bin/passwd** command.

9. Use **ls** to find the biggest file in **/etc**.

10. Open two terminal windows (or tabs) and make sure you are in the same directory in both. Type **echo this is the first line > tailing.txt** in the first terminal, then issue **tail -f tailing.txt** in the second terminal. Now go back to the first terminal and type **echo This is another line >> tailing.txt** (note the double >>), verify that the **tail -f** in the second terminal shows both lines. Stop the **tail -f** with **Ctrl-C**.

11. Use **cat** to create a file named **tailing.txt** that contains the contents of **tailing.txt** followed by the contents of **/etc/passwd**.

12. Use **cat** to create a file named **tailing.txt** that contains the contents of **tailing.txt** preceded by the contents of **/etc/passwd**.

10.8. solution: file contents

1. Display the first 12 lines of **/etc/services**.

```
head -12 /etc/services
```

2. Display the last line of **/etc/passwd**.

```
tail -1 /etc/passwd
```

3. Use cat to create a file named **count.txt** that looks like this:

```
cat > count.txt
One
Two
Three
Four
Five (followed by Ctrl-d)
```

4. Use **cp** to make a backup of this file to **cnt.txt**.

```
cp count.txt cnt.txt
```

5. Use **cat** to make a backup of this file to **catcnt.txt**.

```
cat count.txt > catcnt.txt
```

6. Display **catcnt.txt**, but with all lines in reverse order (the last line first).

```
tac catcnt.txt
```

7. Use more to display **/etc/services**.

```
more /etc/services
```

8. Display the readable character strings from the **/usr/bin/passwd** command.

```
strings /usr/bin/passwd
```

9. Use **ls** to find the biggest file in **/etc**.

```
ls -lrS /etc
```

10. Open two terminal windows (or tabs) and make sure you are in the same directory in both. Type **echo this is the first line > tailing.txt** in the first terminal, then issue **tail -f tailing.txt** in the second terminal. Now go back to the first terminal and type **echo This is another line >> tailing.txt** (note the double >>), verify that the **tail -f** in the second terminal shows both lines. Stop the **tail -f** with **Ctrl-C**.

11. Use **cat** to create a file named **tailing.txt** that contains the contents of **tailing.txt** followed by the contents of **/etc/passwd**.

```
cat /etc/passwd >> tailing.txt
```

12. Use **cat** to create a file named **tailing.txt** that contains the contents of **tailing.txt** preceded by the contents of **/etc/passwd**.

```
mv tailing.txt tmp.txt ; cat /etc/passwd tmp.txt > tailing.txt
```

Chapter 11. the Linux file tree

This chapter takes a look at the most common directories in the **Linux file tree**. It also shows that on Unix everything is a file.

11.1. filesystem hierarchy standard

Many Linux distributions partially follow the **Filesystem Hierarchy Standard**. The **FHS** may help make more Unix/Linux file system trees conform better in the future. The **FHS** is available online at **http://www.pathname.com/fhs/** where we read: "The filesystem hierarchy standard has been designed to be used by Unix distribution developers, package developers, and system implementers. However, it is primarily intended to be a reference and is not a tutorial on how to manage a Unix filesystem or directory hierarchy."

11.2. man hier

There are some differences in the filesystems between **Linux distributions**. For help about your machine, enter **man hier** to find information about the file system hierarchy. This manual will explain the directory structure on your computer.

11.3. the root directory /

All Linux systems have a directory structure that starts at the **root directory**. The root directory is represented by a **forward slash**, like this: /. Everything that exists on your Linux system can be found below this root directory. Let's take a brief look at the contents of the root directory.

```
[paul@RHELv4u3 ~]$ ls /
bin   dev  home  media  mnt  proc  sbin     srv  tftpboot  usr
boot  etc  lib   misc   opt  root  selinux  sys  tmp       var
```

11.4. binary directories

Binaries are files that contain compiled source code (or machine code). Binaries can be **executed** on the computer. Sometimes binaries are called **executables**.

11.4.1. /bin

The **/bin** directory contains **binaries** for use by all users. According to the FHS the **/bin** directory should contain **/bin/cat** and **/bin/date** (among others).

In the screenshot below you see common Unix/Linux commands like cat, cp, cpio, date, dd, echo, grep, and so on. Many of these will be covered in this book.

```
paul@laika:~$ ls /bin
archdetect       egrep             mt               setupcon
autopartition    false             mt-gnu           sh
bash             fgconsole         mv               sh.distrib
bunzip2          fgrep             nano             sleep
bzcat            fuser             nc               stralign
bzcmp            fusermount        nc.traditional   stty
bzdiff           get_mountoptions  netcat           su
bzegrep          grep              netstat          sync
bzexe            gunzip            ntfs-3g          sysfs
bzfgrep          gzexe             ntfs-3g.probe    tailf
bzgrep           gzip              parted_devices   tar
bzip2            hostname          parted_server    tempfile
bzip2recover     hw-detect         partman          touch
bzless           ip                partman-commit   true
bzmore           kbd_mode          perform_recipe   ulockmgr
cat              kill              pidof            umount
...
```

11.4.2. other /bin directories

You can find a **/bin subdirectory** in many other directories. A user named **serena** could put her own programs in **/home/serena/bin**.

Some applications, often when installed directly from source will put themselves in **/opt**. A **samba server** installation can use **/opt/samba/bin** to store its binaries.

11.4.3. /sbin

/sbin contains binaries to configure the operating system. Many of the **system binaries** require **root** privilege to perform certain tasks.

Below a screenshot containing **system binaries** to change the ip address, partition a disk and create an ext4 file system.

```
paul@ubu1010:~$ ls -l /sbin/ifconfig /sbin/fdisk /sbin/mkfs.ext4
-rwxr-xr-x 1 root root 97172 2011-02-02 09:56 /sbin/fdisk
-rwxr-xr-x 1 root root 65708 2010-07-02 09:27 /sbin/ifconfig
-rwxr-xr-x 5 root root 55140 2010-08-18 18:01 /sbin/mkfs.ext4
```

11.4.4. /lib

Binaries found in **/bin** and **/sbin** often use **shared libraries** located in **/lib**. Below is a screenshot of the partial contents of **/lib**.

```
paul@laika:~$ ls /lib/libc*
/lib/libc-2.5.so     /lib/libcfont.so.0.0.0  /lib/libcom_err.so.2.1
/lib/libcap.so.1     /lib/libcidn-2.5.so     /lib/libconsole.so.0
/lib/libcap.so.1.10  /lib/libcidn.so.1       /lib/libconsole.so.0.0.0
/lib/libcfont.so.0   /lib/libcom_err.so.2    /lib/libcrypt-2.5.so
```

/lib/modules

Typically, the **Linux kernel** loads kernel modules from **/lib/modules/$kernel-version/**. This directory is discussed in detail in the Linux kernel chapter.

/lib32 and /lib64

We currently are in a transition between **32-bit** and **64-bit** systems. Therefore, you may encounter directories named **/lib32** and **/lib64** which clarify the register size used during compilation time of the libraries. A 64-bit computer may have some 32-bit binaries and libraries for compatibility with legacy applications. This screenshot uses the **file** utility to demonstrate the difference.

```
paul@laika:~$ file /lib32/libc-2.5.so
/lib32/libc-2.5.so: ELF 32-bit LSB shared object, Intel 80386, \
version 1 (SYSV), for GNU/Linux 2.6.0, stripped
paul@laika:~$ file /lib64/libcap.so.1.10
/lib64/libcap.so.1.10: ELF 64-bit LSB shared object, AMD x86-64, \
version 1 (SYSV), stripped
```

The ELF **(Executable and Linkable Format)** is used in almost every Unix-like operating system since **System V**.

11.4.5. /opt

The purpose of **/opt** is to store **optional** software. In many cases this is software from outside the distribution repository. You may find an empty **/opt** directory on many systems.

A large package can install all its files in **/bin**, **/lib**, **/etc** subdirectories within **/opt/$packagename/**. If for example the package is called wp, then it installs in **/opt/wp**, putting binaries in **/opt/wp/bin** and manpages in **/opt/wp/man**.

11.5. configuration directories

11.5.1. /boot

The **/boot** directory contains all files needed to boot the computer. These files don't change very often. On Linux systems you typically find the **/boot/grub** directory here. **/boot/grub** contains **/boot/grub/grub.cfg** (older systems may still have **/boot/grub/grub.conf**) which defines the boot menu that is displayed before the kernel starts.

11.5.2. /etc

All of the machine-specific **configuration files** should be located in **/etc**. Historically **/etc** stood for **etcetera**, today people often use the **Editable Text Configuration** backronym.

Many times the name of a configuration files is the same as the application, daemon, or protocol with **.conf** added as the extension.

```
paul@laika:~$ ls /etc/*.conf
/etc/adduser.conf        /etc/ld.so.conf       /etc/scrollkeeper.conf
/etc/brltty.conf         /etc/lftp.conf        /etc/sysctl.conf
/etc/ccertificates.conf  /etc/libao.conf       /etc/syslog.conf
/etc/cvs-cron.conf       /etc/logrotate.conf   /etc/ucf.conf
/etc/ddclient.conf       /etc/ltrace.conf      /etc/uniconf.conf
/etc/debconf.conf        /etc/mke2fs.conf      /etc/updatedb.conf
/etc/deluser.conf        /etc/netscsid.conf    /etc/usplash.conf
/etc/fdmount.conf        /etc/nsswitch.conf    /etc/uswsusp.conf
/etc/hdparm.conf         /etc/pam.conf         /etc/vnc.conf
/etc/host.conf           /etc/pnm2ppa.conf     /etc/wodim.conf
/etc/inetd.conf          /etc/povray.conf      /etc/wvdial.conf
/etc/kernel-img.conf     /etc/resolv.conf
paul@laika:~$
```

There is much more to be found in **/etc**.

/etc/init.d/

A lot of Unix/Linux distributions have an **/etc/init.d** directory that contains scripts to start and stop **daemons**. This directory could disappear as Linux migrates to systems that replace the old **init** way of starting all **daemons**.

/etc/X11/

The graphical display (aka **X Window System** or just **X**) is driven by software from the X.org foundation. The configuration file for your graphical display is **/etc/X11/xorg.conf**.

/etc/skel/

The **skeleton** directory **/etc/skel** is copied to the home directory of a newly created user. It usually contains hidden files like a **.bashrc** script.

/etc/sysconfig/

This directory, which is not mentioned in the FHS, contains a lot of **Red Hat Enterprise Linux** configuration files. We will discuss some of them in greater detail. The screenshot below is the **/etc/sysconfig** directory from RHELv4u4 with everything installed.

```
paul@RHELv4u4:~$ ls /etc/sysconfig/
apmd          firstboot     irda              network      saslauthd
apm-scripts   grub          irqbalance        networking   selinux
authconfig    hidd          keyboard          ntpd         spamassassin
autofs        httpd         kudzu             openib.conf  squid
bluetooth     hwconf        lm_sensors        pand         syslog
clock         i18n          mouse             pcmcia       sys-config-sec
console       init          mouse.B           pgsql        sys-config-users
crond         installinfo   named             prelink      sys-logviewer
desktop       ipmi          netdump           rawdevices   tux
diskdump      iptables      netdump_id_dsa    rhn          vncservers
dund          iptables-cfg  netdump_id_dsa.p  samba        xinetd
paul@RHELv4u4:~$
```

The file **/etc/sysconfig/firstboot** tells the Red Hat Setup Agent not to run at boot time. If you want to run the Red Hat Setup Agent at the next reboot, then simply remove this file, and run **chkconfig --level 5 firstboot on**. The Red Hat Setup Agent allows you to install the latest updates, create a user account, join the Red Hat Network and more. It will then create the /etc/sysconfig/firstboot file again.

```
paul@RHELv4u4:~$ cat /etc/sysconfig/firstboot
RUN_FIRSTBOOT=NO
```

The **/etc/sysconfig/harddisks** file contains some parameters to tune the hard disks. The file explains itself.

You can see hardware detected by **kudzu** in **/etc/sysconfig/hwconf**. Kudzu is software from Red Hat for automatic discovery and configuration of hardware.

The keyboard type and keymap table are set in the **/etc/sysconfig/keyboard** file. For more console keyboard information, check the manual pages of **keymaps(5)**, **dumpkeys(1)**, **loadkeys(1)** and the directory **/lib/kbd/keymaps/**.

```
root@RHELv4u4:/etc/sysconfig# cat keyboard
KEYBOARDTYPE="pc"
KEYTABLE="us"
```

We will discuss networking files in this directory in the networking chapter.

11.6. data directories

11.6.1. /home

Users can store personal or project data under **/home**. It is common (but not mandatory by the fhs) practice to name the users home directory after the user name in the format **/home/$USERNAME**. For example:

```
paul@ubu606:~$ ls /home
geert  annik  sandra  paul  tom
```

Besides giving every user (or every project or group) a location to store personal files, the home directory of a user also serves as a location to store the user profile. A typical Unix user profile contains many hidden files (files whose file name starts with a dot). The hidden files of the Unix user profiles contain settings specific for that user.

```
paul@ubu606:~$ ls -d /home/paul/.*
/home/paul/.              /home/paul/.bash_profile  /home/paul/.ssh
/home/paul/..             /home/paul/.bashrc        /home/paul/.viminfo
/home/paul/.bash_history  /home/paul/.lesshst
```

11.6.2. /root

On many systems **/root** is the default location for personal data and profile of the **root user**. If it does not exist by default, then some administrators create it.

11.6.3. /srv

You may use **/srv** for data that is **served by your system**. The FHS allows locating cvs, rsync, ftp and www data in this location. The FHS also approves administrative naming in /srv, like /srv/project55/ftp and /srv/sales/www.

On Sun Solaris (or Oracle Solaris) **/export** is used for this purpose.

11.6.4. /media

The **/media** directory serves as a mount point for **removable media devices** such as CD-ROM's, digital cameras, and various usb-attached devices. Since **/media** is rather new in the Unix world, you could very well encounter systems running without this directory. Solaris 9 does not have it, Solaris 10 does. Most Linux distributions today mount all removable media in **/media**.

```
paul@debian5:~$ ls /media/
cdrom  cdrom0  usbdisk
```

11.6.5. /mnt

The **/mnt** directory should be empty and should only be used for temporary mount points (according to the FHS).

Unix and Linux administrators used to create many directories here, like /mnt/something/. You likely will encounter many systems with more than one directory created and/or mounted inside **/mnt** to be used for various local and remote filesystems.

11.6.6. /tmp

Applications and users should use **/tmp** to store temporary data when needed. Data stored in **/tmp** may use either disk space or RAM. Both of which are managed by the operating system. Never use **/tmp** to store data that is important or which you wish to archive.

11.7. in memory directories

11.7.1. /dev

Device files in **/dev** appear to be ordinary files, but are not actually located on the hard disk. The **/dev** directory is populated with files as the kernel is recognising hardware.

common physical devices

Common hardware such as hard disk devices are represented by device files in **/dev**. Below a screenshot of SATA device files on a laptop and then IDE attached drives on a desktop. (The detailed meaning of these devices will be discussed later.)

```
#
# SATA or SCSI or USB
#
paul@laika:~$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sdb  /dev/sdb1  /dev/sdb2

#
# IDE or ATAPI
#
paul@barry:~$ ls /dev/hd*
/dev/hda  /dev/hda1  /dev/hda2  /dev/hdb  /dev/hdb1  /dev/hdb2  /dev/hdc
```

Besides representing physical hardware, some device files are special. These special devices can be very useful.

/dev/tty and /dev/pts

For example, **/dev/tty1** represents a terminal or console attached to the system. (Don't break your head on the exact terminology of 'terminal' or 'console', what we mean here is a command line interface.) When typing commands in a terminal that is part of a graphical interface like Gnome or KDE, then your terminal will be represented as **/dev/pts/1** (1 can be another number).

/dev/null

On Linux you will find other special devices such as **/dev/null** which can be considered a black hole; it has unlimited storage, but nothing can be retrieved from it. Technically speaking, anything written to /dev/null will be discarded. /dev/null can be useful to discard unwanted output from commands. */dev/null is not a good location to store your backups ;-).*

11.7.2. /proc conversation with the kernel

/proc is another special directory, appearing to be ordinary files, but not taking up disk space. It is actually a view of the kernel, or better, what the kernel manages, and is a means to interact with it directly. **/proc** is a proc filesystem.

```
paul@RHELv4u4:~$ mount -t proc
```

```
none on /proc type proc (rw)
```

When listing the /proc directory you will see many numbers (on any Unix) and some interesting files (on Linux)

```
mul@laika:~$ ls /proc
1      2339   4724  5418  6587  7201       cmdline      mounts
10175  2523   4729  5421  6596  7204       cpuinfo      mtrr
10211  2783   4741  5658  6599  7206       crypto       net
10239  2975   4873  5661  6638  7214       devices      pagetypeinfo
141    29775  4874  5665  6652  7216       diskstats    partitions
15045  29792  4878  5927  6719  7218       dma          sched_debug
1519   2997   4879  6     6736  7223       driver       scsi
1548   3      4881  6032  6737  7224       execdomains  self
1551   30228  4882  6033  6755  7227       fb           slabinfo
1554   3069   5     6145  6762  7260       filesystems  stat
1557   31422  5073  6298  6774  7267       fs           swaps
1606   3149   5147  6414  6816  7275       ide          sys
180    31507  5203  6418  6991  7282       interrupts   sysrq-trigger
181    3189   5206  6419  6993  7298       iomem        sysvipc
182    3193   5228  6420  6996  7319       ioports      timer_list
18898  3246   5272  6421  7157  7330       irq          timer_stats
19799  3248   5291  6422  7163  7345       kallsyms     tty
19803  3253   5294  6423  7164  7513       kcore        uptime
19804  3372   5356  6424  7171  7525       key-users    version
1987   4      5370  6425  7175  7529       kmsg         version_signature
1989   42     5379  6426  7188  9964       loadavg      vmcore
2      45     5380  6430  7189  acpi       locks        vmnet
20845  4542   5412  6450  7191  asound     meminfo      vmstat
221    46     5414  6551  7192  buddyinfo  misc         zoneinfo
2338   4704   5416  6568  7199  bus        modules
```

Let's investigate the file properties inside **/proc**. Looking at the date and time will display the current date and time showing the files are constantly updated (a view on the kernel).

```
paul@RHELv4u4:~$ date
Mon Jan 29 18:06:32 EST 2007
paul@RHELv4u4:~$ ls -al /proc/cpuinfo
-r--r--r--  1 root root 0 Jan 29 18:06 /proc/cpuinfo
paul@RHELv4u4:~$
paul@RHELv4u4:~$  ...time passes...
paul@RHELv4u4:~$
paul@RHELv4u4:~$ date
Mon Jan 29 18:10:00 EST 2007
paul@RHELv4u4:~$ ls -al /proc/cpuinfo
-r--r--r--  1 root root 0 Jan 29 18:10 /proc/cpuinfo
```

Most files in /proc are 0 bytes, yet they contain data--sometimes a lot of data. You can see this by executing cat on files like **/proc/cpuinfo**, which contains information about the CPU.

```
paul@RHELv4u4:~$ file /proc/cpuinfo
/proc/cpuinfo: empty
paul@RHELv4u4:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 43
```

```
model name      : AMD Athlon(tm) 64 X2 Dual Core Processor 4600+
stepping        : 1
cpu MHz         : 2398.628
cache size      : 512 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge...
bogomips        : 4803.54
```

Just for fun, here is /proc/cpuinfo on a Sun Sunblade 1000...

```
paul@pasha:~$ cat /proc/cpuinfo
cpu : TI UltraSparc III (Cheetah)
fpu : UltraSparc III integrated FPU
promlib : Version 3 Revision 2
prom : 4.2.2
type : sun4u
ncpus probed : 2
ncpus active : 2
Cpu0Bogo : 498.68
Cpu0ClkTck : 000000002cb41780
Cpu1Bogo : 498.68
Cpu1ClkTck : 000000002cb41780
MMU Type : Cheetah
State:
CPU0: online
CPU1: online
```

Most of the files in /proc are read only, some require root privileges, some files are writable, and many files in **/proc/sys** are writable. Let's discuss some of the files in /proc.

/proc/interrupts

On the x86 architecture, **/proc/interrupts** displays the interrupts.

```
paul@RHELv4u4:~$ cat /proc/interrupts
           CPU0
  0:   13876877    IO-APIC-edge  timer
  1:         15    IO-APIC-edge  i8042
  8:          1    IO-APIC-edge  rtc
  9:          0   IO-APIC-level  acpi
 12:         67    IO-APIC-edge  i8042
 14:        128    IO-APIC-edge  ide0
 15:     124320    IO-APIC-edge  ide1
169:     111993   IO-APIC-level  ioc0
177:       2428   IO-APIC-level  eth0
NMI:          0
LOC:   13878037
ERR:          0
MIS:          0
```

On a machine with two CPU's, the file looks like this.

```
paul@laika:~$ cat /proc/interrupts
           CPU0       CPU1
  0:    860013          0  IO-APIC-edge     timer
  1:      4533          0  IO-APIC-edge     i8042
  7:         0          0  IO-APIC-edge     parport0
  8:   6588227          0  IO-APIC-edge     rtc
 10:      2314          0  IO-APIC-fasteoi  acpi
 12:       133          0  IO-APIC-edge     i8042
 14:         0          0  IO-APIC-edge     libata
 15:     72269          0  IO-APIC-edge     libata
 18:         1          0  IO-APIC-fasteoi  yenta
 19:    115036          0  IO-APIC-fasteoi  eth0
 20:    126871          0  IO-APIC-fasteoi  libata, ohci1394
 21:     30204          0  IO-APIC-fasteoi  ehci_hcd:usb1, uhci_hcd:usb2
 22:      1334          0  IO-APIC-fasteoi  saa7133[0], saa7133[0]
 24:    234739          0  IO-APIC-fasteoi  nvidia
NMI:        72         42
LOC:    860000     859994
ERR:         0
```

/proc/kcore

The physical memory is represented in **/proc/kcore**. Do not try to cat this file, instead use a debugger. The size of /proc/kcore is the same as your physical memory, plus four bytes.

```
paul@laika:~$ ls -lh /proc/kcore
-r-------- 1 root root 2.0G 2007-01-30 08:57 /proc/kcore
paul@laika:~$
```

11.7.3. /sys Linux 2.6 hot plugging

The **/sys** directory was created for the Linux 2.6 kernel. Since 2.6, Linux uses **sysfs** to support **usb** and **IEEE 1394** (**FireWire**) hot plug devices. See the manual pages of udev(8) (the successor of **devfs**) and hotplug(8) for more info (or visit http://linux-hotplug.sourceforge.net/).

Basically the **/sys** directory contains kernel information about hardware.

11.8. /usr Unix System Resources

Although **/usr** is pronounced like user, remember that it stands for **Unix System Resources**. The **/usr** hierarchy should contain **shareable, read only** data. Some people choose to mount **/usr** as read only. This can be done from its own partition or from a read only NFS share (NFS is discussed later).

11.8.1. /usr/bin

The **/usr/bin** directory contains a lot of commands.

```
paul@deb508:~$ ls /usr/bin | wc -l
1395
```

(On Solaris the **/bin** directory is a symbolic link to **/usr/bin**.)

11.8.2. /usr/include

The **/usr/include** directory contains general use include files for C.

```
paul@ubu1010:~$ ls /usr/include/
aalib.h         expat_config.h      math.h           search.h
af_vfs.h        expat_external.h    mcheck.h         semaphore.h
aio.h           expat.h             memory.h         setjmp.h
AL              fcntl.h             menu.h           sgtty.h
aliases.h       features.h          mntent.h         shadow.h
...
```

11.8.3. /usr/lib

The **/usr/lib** directory contains libraries that are not directly executed by users or scripts.

```
paul@deb508:~$ ls /usr/lib | head -7
4Suite
ao
apt
arj
aspell
avahi
bonobo
```

11.8.4. /usr/local

The **/usr/local** directory can be used by an administrator to install software locally.

```
paul@deb508:~$ ls /usr/local/
bin  etc  games  include  lib  man  sbin  share  src
paul@deb508:~$ du -sh /usr/local/
128K /usr/local/
```

11.8.5. /usr/share

The **/usr/share** directory contains architecture independent data. As you can see, this is a fairly large directory.

```
paul@deb508:~$ ls /usr/share/ | wc -l
```

```
263
paul@deb508:~$ du -sh /usr/share/
1.3G /usr/share/
```

This directory typically contains **/usr/share/man** for manual pages.

```
paul@deb508:~$ ls /usr/share/man
cs  fr    hu  it.UTF-8  man2 man6  pl.ISO8859-2  sv
de  fr.ISO8859-1  id  ja   man3 man7  pl.UTF-8     tr
es  fr.UTF-8   it  ko   man4 man8  pt_BR     zh_CN
fi  gl    it.ISO8859-1 man1   man5 pl    ru     zh_TW
```

And it contains **/usr/share/games** for all static game data (so no high-scores or play logs).

```
paul@ubu1010:~$ ls /usr/share/games/
openttd  wesnoth
```

11.8.6. /usr/src

The **/usr/src** directory is the recommended location for kernel source files.

```
paul@deb508:~$ ls -l /usr/src/
total 12
drwxr-xr-x  4 root root 4096 2011-02-01 14:43 linux-headers-2.6.26-2-686
drwxr-xr-x 18 root root 4096 2011-02-01 14:43 linux-headers-2.6.26-2-common
drwxr-xr-x  3 root root 4096 2009-10-28 16:01 linux-kbuild-2.6.26
```

11.9. /var variable data

Files that are unpredictable in size, such as log, cache and spool files, should be located in **/var**.

11.9.1. /var/log

The **/var/log** directory serves as a central point to contain all log files.

```
[paul@RHEL4b ~]$ ls /var/log
acpid           cron.2    maillog.2   quagga           secure.4
amanda          cron.3    maillog.3   radius           spooler
anaconda.log    cron.4    maillog.4   rpmpkgs          spooler.1
anaconda.syslog cups      mailman     rpmpkgs.1        spooler.2
anaconda.xlog   dmesg     messages    rpmpkgs.2        spooler.3
audit           exim      messages.1  rpmpkgs.3        spooler.4
boot.log        gdm       messages.2  rpmpkgs.4        squid
boot.log.1      httpd     messages.3  sa               uucp
boot.log.2      iiim      messages.4  samba            vbox
boot.log.3      iptraf    mysqld.log  scrollkeeper.log vmware-tools-guestd
boot.log.4      lastlog   news        secure           wtmp
canna           mail      pgsql       secure.1         wtmp.1
cron            maillog   ppp         secure.2         Xorg.0.log
cron.1          maillog.1 prelink.log secure.3         Xorg.0.log.old
```

11.9.2. /var/log/messages

A typical first file to check when troubleshooting on Red Hat (and derivatives) is the **/var/log/messages** file. By default this file will contain information on what just happened to the system. The file is called **/var/log/syslog** on Debian and Ubuntu.

```
[root@RHEL4b ~]# tail /var/log/messages
Jul 30 05:13:56 anacron: anacron startup succeeded
Jul 30 05:13:56 atd: atd startup succeeded
Jul 30 05:13:57 messagebus: messagebus startup succeeded
Jul 30 05:13:57 cups-config-daemon: cups-config-daemon startup succeeded
Jul 30 05:13:58 haldaemon: haldaemon startup succeeded
Jul 30 05:14:00 fstab-sync[3560]: removed all generated mount points
Jul 30 05:14:01 fstab-sync[3628]: added mount point /media/cdrom for...
Jul 30 05:14:01 fstab-sync[3646]: added mount point /media/floppy for...
Jul 30 05:16:46 sshd(pam_unix)[3662]: session opened for user paul by...
Jul 30 06:06:37 su(pam_unix)[3904]: session opened for user root by paul
```

11.9.3. /var/cache

The **/var/cache** directory can contain **cache data** for several applications.

```
paul@ubu1010:~$ ls /var/cache/
apt      dictionaries-common    gdm       man        software-center
binfmts  flashplugin-installer  hald      pm-utils
cups     fontconfig             jockey    pppconfig
debconf  fonts                  ldconfig  samba
```

11.9.4. /var/spool

The **/var/spool** directory typically contains spool directories for **mail** and **cron**, but also serves as a parent directory for other spool files (for example print spool files).

11.9.5. /var/lib

The **/var/lib** directory contains application state information.

Red Hat Enterprise Linux for example keeps files pertaining to **rpm** in **/var/lib/rpm/**.

11.9.6. /var/...

/var also contains Process ID files in **/var/run** (soon to be replaced with **/run**) and temporary files that survive a reboot in **/var/tmp** and information about file locks in **/var/lock**. There will be more examples of **/var** usage further in this book.

11.10. practice: file system tree

1. Does the file **/bin/cat** exist ? What about **/bin/dd** and **/bin/echo**. What is the type of these files ?

2. What is the size of the Linux kernel file(s) (vmlinu*) in **/boot** ?

3. Create a directory ~/test. Then issue the following commands:

```
cd ~/test
```

```
dd if=/dev/zero of=zeroes.txt count=1 bs=100
```

```
od zeroes.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file **/dev/zero** to ~/test/zeroes.txt. Can you describe the functionality of **/dev/zero** ?

4. Now issue the following command:

```
dd if=/dev/random of=random.txt count=1 bs=100 ; od random.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file **/dev/random** to ~/test/random.txt. Can you describe the functionality of **/dev/random** ?

5. Issue the following two commands, and look at the first character of each output line.

```
ls -l /dev/sd* /dev/hd*
```

```
ls -l /dev/tty* /dev/input/mou*
```

The first ls will show block(b) devices, the second ls shows character(c) devices. Can you tell the difference between block and character devices ?

6. Use cat to display **/etc/hosts** and **/etc/resolv.conf**. What is your idea about the purpose of these files ?

7. Are there any files in **/etc/skel/** ? Check also for hidden files.

8. Display **/proc/cpuinfo**. On what architecture is your Linux running ?

9. Display **/proc/interrupts**. What is the size of this file ? Where is this file stored ?

10. Can you enter the **/root** directory ? Are there (hidden) files ?

11. Are ifconfig, fdisk, parted, shutdown and grub-install present in **/sbin** ? Why are these binaries in **/sbin** and not in **/bin** ?

12. Is **/var/log** a file or a directory ? What about **/var/spool** ?

13. Open two command prompts (Ctrl-Shift-T in gnome-terminal) or terminals (Ctrl-Alt-F1, Ctrl-Alt-F2, ...) and issue the **who am i** in both. Then try to echo a word from one terminal to the other.

14. Read the man page of **random** and explain the difference between **/dev/random** and **/dev/urandom**.

11.11. solution: file system tree

1. Does the file **/bin/cat** exist ? What about **/bin/dd** and **/bin/echo**. What is the type of these files ?

```
ls /bin/cat ; file /bin/cat
```

```
ls /bin/dd ; file /bin/dd
```

```
ls /bin/echo ; file /bin/echo
```

2. What is the size of the Linux kernel file(s) (vmlinu*) in **/boot** ?

```
ls -lh /boot/vm*
```

3. Create a directory ~/test. Then issue the following commands:

```
cd ~/test
```

```
dd if=/dev/zero of=zeroes.txt count=1 bs=100
```

```
od zeroes.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file **/dev/zero** to ~/test/zeroes.txt. Can you describe the functionality of **/dev/zero** ?

/dev/zero is a Linux special device. It can be considered a source of zeroes. You cannot send something to **/dev/zero**, but you can read zeroes from it.

4. Now issue the following command:

```
dd if=/dev/random of=random.txt count=1 bs=100 ; od random.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file **/dev/random** to ~/test/random.txt. Can you describe the functionality of **/dev/random** ?

/dev/random acts as a **random number generator** on your Linux machine.

5. Issue the following two commands, and look at the first character of each output line.

```
ls -l /dev/sd* /dev/hd*
```

```
ls -l /dev/tty* /dev/input/mou*
```

The first ls will show block(b) devices, the second ls shows character(c) devices. Can you tell the difference between block and character devices ?

Block devices are always written to (or read from) in blocks. For hard disks, blocks of 512 bytes are common. Character devices act as a stream of characters (or bytes). Mouse and keyboard are typical character devices.

6. Use cat to display **/etc/hosts** and **/etc/resolv.conf**. What is your idea about the purpose of these files ?

/etc/hosts contains hostnames with their ip address

/etc/resolv.conf should contain the ip address of a DNS name server.

7. Are there any files in **/etc/skel/** ? Check also for hidden files.

```
Issue "ls -al /etc/skel/". Yes, there should be hidden files there.
```

8. Display **/proc/cpuinfo**. On what architecture is your Linux running ?

```
The file should contain at least one line with Intel or other cpu.
```

9. Display **/proc/interrupts**. What is the size of this file ? Where is this file stored ?

The size is zero, yet the file contains data. It is not stored anywhere because /proc is a virtual file system that allows you to talk with the kernel. (If you answered "stored in RAM-memory, that is also correct...).

10. Can you enter the **/root** directory ? Are there (hidden) files ?

```
Try "cd /root". The /root directory is not accessible for normal users on most modern Linux sy
```

11. Are ifconfig, fdisk, parted, shutdown and grub-install present in **/sbin** ? Why are these binaries in **/sbin** and not in /bin ?

```
Because those files are only meant for system administrators.
```

12. Is **/var/log** a file or a directory ? What about **/var/spool** ?

```
Both are directories.
```

13. Open two command prompts (Ctrl-Shift-T in gnome-terminal) or terminals (Ctrl-Alt-F1, Ctrl-Alt-F2, ...) and issue the **who am i** in both. Then try to echo a word from one terminal to the other.

```
tty-terminal: echo Hello > /dev/tty1
```

```
pts-terminal: echo Hello > /dev/pts/1
```

14. Read the man page of **random** and explain the difference between **/dev/random** and **/dev/urandom**.

```
man 4 random
```

Part IV. shell expansion

Table of Contents

Chapter 12. commands and arguments

This chapter introduces you to **shell expansion** by taking a close look at **commands** and **arguments**. Knowing **shell expansion** is important because many **commands** on your Linux system are processed and most likely changed by the **shell** before they are executed.

The command line interface or **shell** used on most Linux systems is called **bash**, which stands for **Bourne again shell**. The **bash** shell incorporates features from **sh** (the original Bourne shell), **csh** (the C shell), and **ksh** (the Korn shell).

This chapter frequently uses the **echo** command to demonstrate shell features. The **echo** command is very simple: it echoes the input that it receives.

```
paul@laika:~$ echo Burtonville
Burtonville
paul@laika:~$ echo Smurfs are blue
Smurfs are blue
```

12.1. arguments

One of the primary features of a shell is to perform a **command line scan**. When you enter a command at the shell's command prompt and press the enter key, then the shell will start scanning that line, cutting it up in **arguments**. While scanning the line, the shell may make many changes to the **arguments** you typed.

This process is called **shell expansion**. When the shell has finished scanning and modifying that line, then it will be executed.

12.2. white space removal

Parts that are separated by one or more consecutive **white spaces** (or tabs) are considered separate **arguments**, any white space is removed. The first **argument** is the command to be executed, the other **arguments** are given to the command. The shell effectively cuts your command into one or more arguments.

This explains why the following four different command lines are the same after **shell expansion**.

```
[paul@RHELv4u3 ~]$ echo Hello World
Hello World
[paul@RHELv4u3 ~]$ echo Hello   World
Hello World
[paul@RHELv4u3 ~]$ echo   Hello   World
Hello World
[paul@RHELv4u3 ~]$    echo      Hello      World
Hello World
```

The **echo** command will display each argument it receives from the shell. The **echo** command will also add a new white space between the arguments it received.

12.3. single quotes

You can prevent the removal of white spaces by quoting the spaces. The contents of the quoted string are considered as one argument. In the screenshot below the **echo** receives only one **argument**.

```
[paul@RHEL4b ~]$ echo 'A line with      single    quotes'
A line with      single    quotes
[paul@RHEL4b ~]$
```

12.4. double quotes

You can also prevent the removal of white spaces by double quoting the spaces. Same as above, **echo** only receives one **argument**.

```
[paul@RHEL4b ~]$ echo "A line with      double    quotes"
A line with      double    quotes
[paul@RHEL4b ~]$
```

Later in this book, when discussing **variables** we will see important differences between single and double quotes.

12.5. echo and quotes

Quoted lines can include special escaped characters recognised by the **echo** command (when using **echo -e**). The screenshot below shows how to use **\n** for a newline and **\t** for a tab (usually eight white spaces).

```
[paul@RHEL4b ~]$ echo -e "A line with \na newline"
A line with
a newline
[paul@RHEL4b ~]$ echo -e 'A line with \na newline'
A line with
a newline
[paul@RHEL4b ~]$ echo -e "A line with \ta tab"
A line with     a tab
[paul@RHEL4b ~]$ echo -e 'A line with \ta tab'
A line with     a tab
[paul@RHEL4b ~]$
```

The echo command can generate more than white spaces, tabs and newlines. Look in the man page for a list of options.

12.6. commands

12.6.1. external or builtin commands ?

Not all commands are external to the shell, some are **builtin**. **External commands** are programs that have their own binary and reside somewhere in the file system. Many external commands are located in **/bin** or **/sbin**. **Builtin commands** are an integral part of the shell program itself.

12.6.2. type

To find out whether a command given to the shell will be executed as an **external command** or as a **builtin command**, use the **type** command.

```
paul@laika:~$ type cd
cd is a shell builtin
paul@laika:~$ type cat
cat is /bin/cat
```

As you can see, the **cd** command is **builtin** and the **cat** command is **external**.

You can also use this command to show you whether the command is **aliased** or not.

```
paul@laika:~$ type ls
ls is aliased to `ls --color=auto'
```

12.6.3. running external commands

Some commands have both builtin and external versions. When one of these commands is executed, the builtin version takes priority. To run the external version, you must enter the full path to the command.

```
paul@laika:~$ type -a echo
echo is a shell builtin
echo is /bin/echo
paul@laika:~$ /bin/echo Running the external echo command...
Running the external echo command...
```

12.6.4. which

The **which** command will search for binaries in the **$PATH** environment variable (variables will be explained later). In the screenshot below, it is determined that **cd** is **builtin**, and **ls, cp, rm, mv, mkdir, pwd,** and **which** are external commands.

```
[root@RHEL4b ~]# which cp ls cd mkdir pwd
/bin/cp
/bin/ls
/usr/bin/which: no cd in (/usr/kerberos/sbin:/usr/kerberos/bin:...
/bin/mkdir
/bin/pwd
```

12.7. aliases

12.7.1. create an alias

The shell allows you to create **aliases**. Aliases are often used to create an easier to remember name for an existing command or to easily supply parameters.

```
[paul@RHELv4u3 ~]$ cat count.txt
one
two
three
[paul@RHELv4u3 ~]$ alias dog=tac
[paul@RHELv4u3 ~]$ dog count.txt
three
two
one
```

12.7.2. abbreviate commands

An **alias** can also be useful to abbreviate an existing command.

```
paul@laika:~$ alias ll='ls -lh --color=auto'
paul@laika:~$ alias c='clear'
paul@laika:~$
```

12.7.3. default options

Aliases can be used to supply commands with default options. The example below shows how to set the **-i** option default when typing **rm**.

```
[paul@RHELv4u3 ~]$ rm -i winter.txt
rm: remove regular file `winter.txt'? no
[paul@RHELv4u3 ~]$ rm winter.txt
[paul@RHELv4u3 ~]$ ls winter.txt
ls: winter.txt: No such file or directory
[paul@RHELv4u3 ~]$ touch winter.txt
[paul@RHELv4u3 ~]$ alias rm='rm -i'
[paul@RHELv4u3 ~]$ rm winter.txt
rm: remove regular empty file `winter.txt'? no
[paul@RHELv4u3 ~]$
```

Some distributions enable default aliases to protect users from accidentally erasing files ('rm -i', 'mv -i', 'cp -i')

12.7.4. viewing aliases

You can provide one or more aliases as arguments to the **alias** command to get their definitions. Providing no arguments gives a complete list of current aliases.

```
paul@laika:~$ alias c ll
alias c='clear'
alias ll='ls -lh --color=auto'
```

12.7.5. unalias

You can undo an alias with the **unalias** command.

```
[paul@RHEL4b ~]$ which rm
/bin/rm
[paul@RHEL4b ~]$ alias rm='rm -i'
[paul@RHEL4b ~]$ which rm
alias rm='rm -i'
        /bin/rm
[paul@RHEL4b ~]$ unalias rm
[paul@RHEL4b ~]$ which rm
/bin/rm
[paul@RHEL4b ~]$
```

12.8. displaying shell expansion

You can display shell expansion with **set -x**, and stop displaying it with **set +x**. You might want to use this further on in this course, or when in doubt about exactly what the shell is doing with your command.

```
[paul@RHELv4u3 ~]$ set -x
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ echo $USER
+ echo paul
paul
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ echo \$USER
+ echo '$USER'
$USER
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ set +x
+ set +x
[paul@RHELv4u3 ~]$ echo $USER
paul
```

12.9. practice: commands and arguments

1. How many **arguments** are in this line (not counting the command itself).

```
touch '/etc/cron/cron.allow' 'file 42.txt' "file 33.txt"
```

2. Is **tac** a shell builtin command ?

3. Is there an existing alias for **rm** ?

4. Read the man page of **rm**, make sure you understand the **-i** option of rm. Create and remove a file to test the **-i** option.

5. Execute: **alias rm='rm -i'** . Test your alias with a test file. Does this work as expected ?

6. List all current aliases.

7a. Create an alias called 'city' that echoes your hometown.

7b. Use your alias to test that it works.

8. Execute **set -x** to display shell expansion for every command.

9. Test the functionality of **set -x** by executing your **city** and **rm** aliases.

10 Execute **set +x** to stop displaying shell expansion.

11. Remove your city alias.

12. What is the location of the **cat** and the **passwd** commands ?

13. Explain the difference between the following commands:

```
echo
```

```
/bin/echo
```

14. Explain the difference between the following commands:

```
echo Hello
```

```
echo -n Hello
```

15. Display **A B C** with two spaces between B and C.

(optional)16. Complete the following command (do not use spaces) to display exactly the following output:

```
4+4     =8
10+14   =24
```

17. Use **echo** to display the following exactly:

```
??\\
```

Find two solutions with single quotes, two with double quotes and one without quotes (and say thank you to René and Darioush from Google for this extra).

18. Use one **echo** command to display three words on three lines.

12.10. solution: commands and arguments

1. How many **arguments** are in this line (not counting the command itself).

```
touch '/etc/cron/cron.allow' 'file 42.txt' "file 33.txt"
```

```
answer: three
```

2. Is **tac** a shell builtin command ?

```
type tac
```

3. Is there an existing alias for **rm** ?

```
alias rm
```

4. Read the man page of **rm**, make sure you understand the **-i** option of rm. Create and remove a file to test the **-i** option.

```
man rm
```

```
touch testfile
```

```
rm -i testfile
```

5. Execute: **alias rm='rm -i'** . Test your alias with a test file. Does this work as expected ?

```
touch testfile
```

```
rm testfile (should ask for confirmation)
```

6. List all current aliases.

```
alias
```

7a. Create an alias called 'city' that echoes your hometown.

```
alias city='echo Antwerp'
```

7b. Use your alias to test that it works.

```
city (it should display Antwerp)
```

8. Execute **set -x** to display shell expansion for every command.

```
set -x
```

9. Test the functionality of **set -x** by executing your **city** and **rm** aliases.

```
shell should display the resolved aliases and then execute the command:
paul@deb503:~$ set -x
paul@deb503:~$ city
+ echo antwerp
antwerp
```

10 Execute **set +x** to stop displaying shell expansion.

```
set +x
```

11. Remove your city alias.

```
unalias city
```

12. What is the location of the **cat** and the **passwd** commands ?

```
which cat (probably /bin/cat)
```

```
which passwd (probably /usr/bin/passwd)
```

13. Explain the difference between the following commands:

```
echo
```

```
/bin/echo
```

The **echo** command will be interpreted by the shell as the **built-in echo** command. The **/bin/echo** command will make the shell execute the **echo binary** located in the **/bin** directory.

14. Explain the difference between the following commands:

```
echo Hello
```

```
echo -n Hello
```

The -n option of the **echo** command will prevent echo from echoing a trailing newline. **echo Hello** will echo six characters in total, **echo -n hello** only echoes five characters.

(The -n option might not work in the Korn shell.)

15. Display **A B C** with two spaces between B and C.

```
echo "A B  C"
```

16. Complete the following command (do not use spaces) to display exactly the following output:

```
4+4     =8
10+14   =24
```

The solution is to use tabs with \t.

```
echo -e "4+4\t=8" ; echo -e "10+14\t=24"
```

17. Use **echo** to display the following exactly:

```
??\\
echo '??\\'
echo -e '??\\\\'
echo "??\\\\"
echo -e "??\\\\\\"
echo ??\\\\
```

Find two solutions with single quotes, two with double quotes and one without quotes (and say thank you to René and Darioush from Google for this extra).

18. Use one **echo** command to display three words on three lines.

```
echo -e "one \ntwo \nthree"
```

Chapter 13. control operators

In this chapter we put more than one command on the command line using **control operators**. We also briefly discuss related parameters ($?) and similar special characters(&).

13.1. ; semicolon

You can put two or more commands on the same line separated by a semicolon **;** . The shell will scan the line until it reaches the semicolon. All the arguments before this semicolon will be considered a separate command from all the arguments after the semicolon. Both series will be executed sequentially with the shell waiting for each command to finish before starting the next one.

```
[paul@RHELv4u3 ~]$ echo Hello
Hello
[paul@RHELv4u3 ~]$ echo World
World
[paul@RHELv4u3 ~]$ echo Hello ; echo World
Hello
World
[paul@RHELv4u3 ~]$
```

13.2. & ampersand

When a line ends with an ampersand **&**, the shell will not wait for the command to finish. You will get your shell prompt back, and the command is executed in background. You will get a message when this command has finished executing in background.

```
[paul@RHELv4u3 ~]$ sleep 20 &
[1] 7925
[paul@RHELv4u3 ~]$
...wait 20 seconds...
[paul@RHELv4u3 ~]$
[1]+  Done                    sleep 20
```

The technical explanation of what happens in this case is explained in the chapter about **processes**.

13.3. $? dollar question mark

The exit code of the previous command is stored in the shell variable **$?**. Actually **$?** is a shell parameter and not a variable, since you cannot assign a value to **$?**.

```
paul@debian5:~/test$ touch file1
paul@debian5:~/test$ echo $?
0
paul@debian5:~/test$ rm file1
paul@debian5:~/test$ echo $?
0
paul@debian5:~/test$ rm file1
rm: cannot remove `file1': No such file or directory
paul@debian5:~/test$ echo $?
1
paul@debian5:~/test$
```

13.4. && double ampersand

The shell will interpret **&&** as a **logical AND**. When using **&&** the second command is executed only if the first one succeeds (returns a zero exit status).

```
paul@barry:~$ echo first && echo second
first
second
paul@barry:~$ zecho first && echo second
-bash: zecho: command not found
```

Another example of the same **logical AND** principle. This example starts with a working **cd** followed by **ls**, then a non-working **cd** which is **not** followed by **ls**.

```
[paul@RHELv4u3 ~]$ cd gen && ls
file1  file3  File55  fileab  FileAB   fileabc
file2  File4  FileA   Fileab  fileab2
[paul@RHELv4u3 gen]$ cd gen && ls
-bash: cd: gen: No such file or directory
```

13.5. || double vertical bar

The || represents a **logical OR**. The second command is executed only when the first command fails (returns a non-zero exit status).

```
paul@barry:~$ echo first || echo second ; echo third
first
third
paul@barry:~$ zecho first || echo second ; echo third
-bash: zecho: command not found
second
third
paul@barry:~$
```

Another example of the same **logical OR** principle.

```
[paul@RHELv4u3 ~]$ cd gen || ls
[paul@RHELv4u3 gen]$ cd gen || ls
-bash: cd: gen: No such file or directory
file1  file3  File55  fileab  FileAB   fileabc
file2  File4  FileA   Fileab  fileab2
```

13.6. combining && and ||

You can use this logical AND and logical OR to write an **if-then-else** structure on the command line. This example uses **echo** to display whether the **rm** command was successful.

```
paul@laika:~/test$ rm file1 && echo It worked! || echo It failed!
It worked!
paul@laika:~/test$ rm file1 && echo It worked! || echo It failed!
rm: cannot remove `file1': No such file or directory
It failed!
paul@laika:~/test$
```

13.7. # pound sign

Everything written after a **pound sign** (#) is ignored by the shell. This is useful to write a **shell comment**, but has no influence on the command execution or shell expansion.

```
paul@debian4:~$ mkdir test    # we create a directory
paul@debian4:~$ cd test       #### we enter the directory
paul@debian4:~/test$ ls       # is it empty ?
paul@debian4:~/test$
```

13.8. \ escaping special characters

The backslash \ character enables the use of control characters, but without the shell interpreting it, this is called **escaping** characters.

```
[paul@RHELv4u3 ~]$ echo hello \; world
hello ; world
[paul@RHELv4u3 ~]$ echo hello\ \ \ world
hello   world
[paul@RHELv4u3 ~]$ echo escaping \\\ \#\ \&\ \"\ \'
escaping \ # & " '
[paul@RHELv4u3 ~]$ echo escaping \\\?\*\"\'
escaping \?*"'
```

13.8.1. end of line backslash

Lines ending in a backslash are continued on the next line. The shell does not interpret the newline character and will wait on shell expansion and execution of the command line until a newline without backslash is encountered.

```
[paul@RHEL4b ~]$ echo This command line \
> is split in three \
> parts
This command line is split in three parts
[paul@RHEL4b ~]$
```

13.9. practice: control operators

0. Each question can be answered by one command line!

1. When you type **passwd**, which file is executed ?

2. What kind of file is that ?

3. Execute the **pwd** command twice. (remember 0.)

4. Execute **ls** after **cd /etc**, but only if **cd /etc** did not error.

5. Execute **cd /etc** after **cd etc**, but only if **cd etc** fails.

6. Echo **it worked** when **touch test42** works, and echo **it failed** when the **touch** failed. All on one command line as a normal user (not root). Test this line in your home directory and in **/bin/** .

7. Execute **sleep 6**, what is this command doing ?

8. Execute **sleep 200** in background (do not wait for it to finish).

9. Write a command line that executes **rm file55**. Your command line should print 'success' if file55 is removed, and print 'failed' if there was a problem.

(optional)10. Use echo to display "Hello World with strange' characters \ * [} ~ \ \ ." (including all quotes)

13.10. solution: control operators

0. Each question can be answered by one command line!

1. When you type **passwd**, which file is executed ?

```
which passwd
```

2. What kind of file is that ?

```
file /usr/bin/passwd
```

3. Execute the **pwd** command twice. (remember 0.)

```
pwd ; pwd
```

4. Execute **ls** after **cd /etc**, but only if **cd /etc** did not error.

```
cd /etc && ls
```

5. Execute **cd /etc** after **cd etc**, but only if **cd etc** fails.

```
cd etc || cd /etc
```

6. Echo **it worked** when **touch test42** works, and echo **it failed** when the **touch** failed. All on one command line as a normal user (not root). Test this line in your home directory and in **/bin/** .

```
paul@deb503:~$ cd ; touch test42 && echo it worked || echo it failed
it worked
paul@deb503:~$ cd /bin; touch test42 && echo it worked || echo it failed
touch: cannot touch `test42': Permission denied
it failed
```

7. Execute **sleep 6**, what is this command doing ?

```
pausing for six seconds
```

8. Execute **sleep 200** in background (do not wait for it to finish).

```
sleep 200 &
```

9. Write a command line that executes **rm file55**. Your command line should print 'success' if file55 is removed, and print 'failed' if there was a problem.

```
rm file55 && echo success || echo failed
```

(optional)10. Use echo to display "Hello World with strange' characters \ * [} ~ \ \ ." (including all quotes)

```
echo \"Hello World with strange\' characters \\ \* \[ \} \~ \\\\ \. \"
```

```
or
```

```
echo \""Hello World with strange' characters \ * [ } ~ \\ . "\"
```

Chapter 14. shell variables

In this chapter we learn to manage environment **variables** in the shell. These **variables** are often needed by applications.

14.1. $ dollar sign

Another important character interpreted by the shell is the dollar sign **$**. The shell will look for an **environment variable** named like the string following the **dollar sign** and replace it with the value of the variable (or with nothing if the variable does not exist).

These are some examples using $HOSTNAME, $USER, $UID, $SHELL, and $HOME.

```
[paul@RHELv4u3 ~]$ echo This is the $SHELL shell
This is the /bin/bash shell
[paul@RHELv4u3 ~]$ echo This is $SHELL on computer $HOSTNAME
This is /bin/bash on computer RHELv4u3.localdomain
[paul@RHELv4u3 ~]$ echo The userid of $USER is $UID
The userid of paul is 500
[paul@RHELv4u3 ~]$ echo My homedir is $HOME
My homedir is /home/paul
```

14.2. case sensitive

This example shows that shell variables are case sensitive!

```
[paul@RHELv4u3 ~]$ echo Hello $USER
Hello paul
[paul@RHELv4u3 ~]$ echo Hello $user
Hello
```

14.3. creating variables

This example creates the variable **$MyVar** and sets its value. It then uses **echo** to verify the value.

```
[paul@RHELv4u3 gen]$ MyVar=555
[paul@RHELv4u3 gen]$ echo $MyVar
555
[paul@RHELv4u3 gen]$
```

14.4. quotes

Notice that double quotes still allow the parsing of variables, whereas single quotes prevent this.

```
[paul@RHELv4u3 ~]$ MyVar=555
[paul@RHELv4u3 ~]$ echo $MyVar
555
[paul@RHELv4u3 ~]$ echo "$MyVar"
555
[paul@RHELv4u3 ~]$ echo '$MyVar'
$MyVar
```

The bash shell will replace variables with their value in double quoted lines, but not in single quoted lines.

```
paul@laika:~$ city=Burtonville
paul@laika:~$ echo "We are in $city today."
We are in Burtonville today.
paul@laika:~$ echo 'We are in $city today.'
We are in $city today.
```

14.5. set

You can use the **set** command to display a list of environment variables. On Ubuntu and Debian systems, the **set** command will also list shell functions after the shell variables. Use **set | more** to see the variables then.

14.6. unset

Use the **unset** command to remove a variable from your shell environment.

```
[paul@RHEL4b ~]$ MyVar=8472
[paul@RHEL4b ~]$ echo $MyVar
8472
[paul@RHEL4b ~]$ unset MyVar
[paul@RHEL4b ~]$ echo $MyVar

[paul@RHEL4b ~]$
```

14.7. $PS1

The **$PS1** variable determines your shell prompt. You can use backslash escaped special characters like **\u** for the username or **\w** for the working directory. The **bash** manual has a complete reference.

In this example we change the value of **$PS1** a couple of times.

```
paul@deb503:~$ PS1=prompt
prompt
promptPS1='prompt '
prompt
prompt PS1='> '
>
> PS1='\u@\h$ '
paul@deb503$
paul@deb503$ PS1='\u@\h:\W$'
paul@deb503:~$
```

To avoid unrecoverable mistakes, you can set normal user prompts to green and the root prompt to red. Add the following to your **.bashrc** for a green user prompt:

```
# color prompt by paul
RED='\[\033[01;31m\]'
WHITE='\[\033[01;00m\]'
GREEN='\[\033[01;32m\]'
BLUE='\[\033[01;34m\]'
export PS1="${debian_chroot:+($debian_chroot)}$GREEN\u$WHITE@$BLUE\h$WHITE\w\$ "
```

14.8. $PATH

The **$PATH** variable is determines where the shell is looking for commands to execute (unless the command is builtin or aliased). This variable contains a list of directories, separated by colons.

```
[[paul@RHEL4b ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
```

The shell will not look in the current directory for commands to execute! (Looking for executables in the current directory provided an easy way to hack PC-DOS computers). If you want the shell to look in the current directory, then add a . at the end of your $PATH.

```
[paul@RHEL4b ~]$ PATH=$PATH:.
[paul@RHEL4b ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:.
[paul@RHEL4b ~]$
```

Your path might be different when using su instead of **su -** because the latter will take on the environment of the target user. The root user typically has **/sbin** directories added to the $PATH variable.

```
[paul@RHEL3 ~]$ su
Password:
[root@RHEL3 paul]# echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
[root@RHEL3 paul]# exit
[paul@RHEL3 ~]$ su -
Password:
[root@RHEL3 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:
[root@RHEL3 ~]#
```

14.9. env

The **env** command without options will display a list of **exported variables**. The difference with **set** with options is that **set** lists all variables, including those not exported to child shells.

But **env** can also be used to start a clean shell (a shell without any inherited environment). The **env -i** command clears the environment for the subshell.

Notice in this screenshot that **bash** will set the **$SHELL** variable on startup.

```
[paul@RHEL4b ~]$ bash -c 'echo $SHELL $HOME $USER'
/bin/bash /home/paul paul
[paul@RHEL4b ~]$ env -i bash -c 'echo $SHELL $HOME $USER'
/bin/bash
[paul@RHEL4b ~]$
```

You can use the **env** command to set the **$LANG**, or any other, variable for just one instance of **bash** with one command. The example below uses this to show the influence of the **$LANG** variable on file globbing (see the chapter on file globbing).

```
[paul@RHEL4b test]$ env LANG=C bash -c 'ls File[a-z]'
Filea  Fileb
[paul@RHEL4b test]$ env LANG=en_US.UTF-8 bash -c 'ls File[a-z]'
Filea  FileA  Fileb  FileB
[paul@RHEL4b test]$
```

14.10. export

You can export shell variables to other shells with the **export** command. This will export the variable to child shells.

```
[paul@RHEL4b ~]$ var3=three
[paul@RHEL4b ~]$ var4=four
[paul@RHEL4b ~]$ export var4
[paul@RHEL4b ~]$ echo $var3 $var4
three four
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $var3 $var4
four
```

But it will not export to the parent shell (previous screenshot continued).

```
[paul@RHEL4b ~]$ export var5=five
[paul@RHEL4b ~]$ echo $var3 $var4 $var5
four five
[paul@RHEL4b ~]$ exit
exit
[paul@RHEL4b ~]$ echo $var3 $var4 $var5
three four
[paul@RHEL4b ~]$
```

14.11. delineate variables

Until now, we have seen that bash interprets a variable starting from a dollar sign, continuing until the first occurrence of a non-alphanumeric character that is not an underscore. In some situations, this can be a problem. This issue can be resolved with curly braces like in this example.

```
[paul@RHEL4b ~]$ prefix=Super
[paul@RHEL4b ~]$ echo Hello $prefixman and $prefixgirl
Hello  and
[paul@RHEL4b ~]$ echo Hello ${prefix}man and ${prefix}girl
Hello Superman and Supergirl
[paul@RHEL4b ~]$
```

14.12. unbound variables

The example below tries to display the value of the **$MyVar** variable, but it fails because the variable does not exist. By default the shell will display nothing when a variable is unbound (does not exist).

```
[paul@RHELv4u3 gen]$ echo $MyVar

[paul@RHELv4u3 gen]$
```

There is, however, the **nounset** shell option that you can use to generate an error when a variable does not exist.

```
paul@laika:~$ set -u
paul@laika:~$ echo $Myvar
bash: Myvar: unbound variable
paul@laika:~$ set +u
paul@laika:~$ echo $Myvar

paul@laika:~$
```

In the bash shell **set -u** is identical to **set -o nounset** and likewise **set +u** is identical to **set +o nounset**.

14.13. practice: shell variables

1. Use echo to display Hello followed by your username. (use a bash variable!)

2. Create a variable **answer** with a value of **42**.

3. Copy the value of $LANG to $MyLANG.

4. List all current shell variables.

5. List all exported shell variables.

6. Do the **env** and **set** commands display your variable ?

6. Destroy your **answer** variable.

7. Create two variables, and **export** one of them.

8. Display the exported variable in an interactive child shell.

9. Create a variable, give it the value 'Dumb', create another variable with value 'do'. Use **echo** and the two variables to echo Dumbledore.

10. Find the list of backslash escaped characters in the manual of bash. Add the time to your **PS1** prompt.

14.14. solution: shell variables

1. Use echo to display Hello followed by your username. (use a bash variable!)

```
echo Hello $USER
```

2. Create a variable **answer** with a value of **42**.

```
answer=42
```

3. Copy the value of $LANG to $MyLANG.

```
MyLANG=$LANG
```

4. List all current shell variables.

```
set
```

```
set|more on Ubuntu/Debian
```

5. List all exported shell variables.

```
env
export
declare -x
```

6. Do the **env** and **set** commands display your variable ?

```
env | more
set | more
```

6. Destroy your **answer** variable.

```
unset answer
```

7. Create two variables, and **export** one of them.

```
var1=1; export var2=2
```

8. Display the exported variable in an interactive child shell.

```
bash
echo $var2
```

9. Create a variable, give it the value 'Dumb', create another variable with value 'do'. Use **echo** and the two variables to echo Dumbledore.

```
varx=Dumb; vary=do
```

```
echo ${varx}le${vary}re
solution by Yves from Dexia : echo $varx'le'$vary're'
solution by Erwin from Telenet : echo "$varx"le"$vary"re
```

10. Find the list of backslash escaped characters in the manual of bash. Add the time to your **PS1** prompt.

```
PS1='\t \u@\h \W$ '
```

Chapter 15. shell embedding and options

This chapter takes a brief look at **child shells**, **embedded shells** and **shell options**.

15.1. shell embedding

Shells can be **embedded** on the command line, or in other words, the command line scan can spawn new processes containing a fork of the current shell. You can use variables to prove that new shells are created. In the screenshot below, the variable $var1 only exists in the (temporary) sub shell.

```
[paul@RHELv4u3 gen]$ echo $var1

[paul@RHELv4u3 gen]$ echo $(var1=5;echo $var1)
5
[paul@RHELv4u3 gen]$ echo $var1

[paul@RHELv4u3 gen]$
```

You can embed a shell in an **embedded shell**, this is called **nested embedding** of shells.

This screenshot shows an embedded shell inside an embedded shell.

```
paul@deb503:~$ A=shell
paul@deb503:~$ echo $C$B$A $(B=sub;echo $C$B$A; echo $(C=sub;echo $C$B$A))
shell subshell subsubshell
```

15.1.1. backticks

Single embedding can be useful to avoid changing your current directory. The screenshot below uses **backticks** instead of dollar-bracket to embed.

```
[paul@RHELv4u3 ~]$ echo `cd /etc; ls -d * | grep pass`
passwd passwd- passwd.OLD
[paul@RHELv4u3 ~]$
```

You can only use the **$()** notation to nest embedded shells, **backticks** cannot do this.

15.1.2. backticks or single quotes

Placing the embedding between **backticks** uses one character less than the dollar and parenthesis combo. Be careful however, backticks are often confused with single quotes. The technical difference between ' and ` is significant!

```
[paul@RHELv4u3 gen]$ echo `var1=5;echo $var1`
5
[paul@RHELv4u3 gen]$ echo 'var1=5;echo $var1'
var1=5;echo $var1
[paul@RHELv4u3 gen]$
```

15.2. shell options

Both **set** and **unset** are builtin shell commands. They can be used to set options of the bash shell itself. The next example will clarify this. By default, the shell will treat unset variables as a variable having no value. By setting the -u option, the shell will treat any reference to unset variables as an error. See the man page of bash for more information.

```
[paul@RHEL4b ~]$ echo $var123

[paul@RHEL4b ~]$ set -u
[paul@RHEL4b ~]$ echo $var123
-bash: var123: unbound variable
[paul@RHEL4b ~]$ set +u
[paul@RHEL4b ~]$ echo $var123

[paul@RHEL4b ~]$
```

To list all the set options for your shell, use **echo $-**. The **noclobber** (or **-C**) option will be explained later in this book (in the I/O redirection chapter).

```
[paul@RHEL4b ~]$ echo $-
himBH
[paul@RHEL4b ~]$ set -C ; set -u
[paul@RHEL4b ~]$ echo $-
himuBCH
[paul@RHEL4b ~]$ set +C ; set +u
[paul@RHEL4b ~]$ echo $-
himBH
[paul@RHEL4b ~]$
```

When typing **set** without options, you get a list of all variables without function when the shell is on **posix** mode. You can set bash in posix mode typing **set -o posix**.

15.3. practice: shell embedding

1. Find the list of shell options in the man page of **bash**. What is the difference between **set -u** and **set -o nounset**?

2. Activate **nounset** in your shell. Test that it shows an error message when using non-existing variables.

3. Deactivate nounset.

4. Execute **cd /var** and **ls** in an embedded shell.

The **echo** command is only needed to show the result of the **ls** command. Omitting will result in the shell trying to execute the first file as a command.

5. Create the variable embvar in an embedded shell and echo it. Does the variable exist in your current shell now ?

6. Explain what "set -x" does. Can this be useful ?

(optional)7. Given the following screenshot, add exactly four characters to that command line so that the total output is FirstMiddleLast.

```
[paul@RHEL4b ~]$ echo  First; echo  Middle; echo  Last
```

8. Display a **long listing** (ls -l) of the **passwd** command using the **which** command inside an embedded shell.

15.4. solution: shell embedding

1. Find the list of shell options in the man page of **bash**. What is the difference between **set -u** and **set -o nounset**?

read the manual of bash (man bash), search for nounset -- both mean the same thing.

2. Activate **nounset** in your shell. Test that it shows an error message when using non-existing variables.

```
set -u
OR
set -o nounset
```

Both these lines have the same effect.

3. Deactivate nounset.

```
set +u
OR
set +o nounset
```

4. Execute **cd /var** and **ls** in an embedded shell.

```
echo $(cd /var ; ls)
```

The **echo** command is only needed to show the result of the **ls** command. Omitting will result in the shell trying to execute the first file as a command.

5. Create the variable embvar in an embedded shell and echo it. Does the variable exist in your current shell now ?

```
echo $(embvar=emb;echo $embvar) ; echo $embvar #the last echo fails
```

```
$embvar does not exist in your current shell
```

6. Explain what "set -x" does. Can this be useful ?

```
It displays shell expansion for troubleshooting your command.
```

(optional)7. Given the following screenshot, add exactly four characters to that command line so that the total output is FirstMiddleLast.

```
[paul@RHEL4b ~]$ echo  First; echo  Middle; echo  Last
```

```
echo -n First; echo -n Middle; echo Last
```

8. Display a **long listing** (ls -l) of the **passwd** command using the **which** command inside an embedded shell.

```
ls -l $(which passwd)
```

Chapter 16. shell history

The shell makes it easy for us to repeat commands, this chapter explains how.

16.1. repeating the last command

To repeat the last command in bash, type **!!**. This is pronounced as **bang bang**.

```
paul@debian5:~/test42$ echo this will be repeated > file42.txt
paul@debian5:~/test42$ !!
echo this will be repeated > file42.txt
paul@debian5:~/test42$
```

16.2. repeating other commands

You can repeat other commands using one **bang** followed by one or more characters. The shell will repeat the last command that started with those characters.

```
paul@debian5:~/test42$ touch file42
paul@debian5:~/test42$ cat file42
paul@debian5:~/test42$ !to
touch file42
paul@debian5:~/test42$
```

16.3. history

To see older commands, use **history** to display the shell command history (or use **history n** to see the last n commands).

```
paul@debian5:~/test$ history 10
38  mkdir test
39  cd test
40  touch file1
41  echo hello > file2
42  echo It is very cold today > winter.txt
43  ls
44  ls -l
45  cp winter.txt summer.txt
46  ls -l
47  history 10
```

16.4. !n

When typing **!** followed by the number preceding the command you want repeated, then the shell will echo the command and execute it.

```
paul@debian5:~/test$ !43
ls
file1  file2  summer.txt  winter.txt
```

16.5. Ctrl-r

Another option is to use **ctrl-r** to search in the history. In the screenshot below i only typed **ctrl-r** followed by four characters **apti** and it finds the last command containing these four consecutive characters.

```
paul@debian5:~$
(reverse-i-search)`apti': sudo aptitude install screen
```

16.6. $HISTSIZE

The $HISTSIZE variable determines the number of commands that will be remembered in your current environment. Most distributions default this variable to 500 or 1000.

```
paul@debian5:~$ echo $HISTSIZE
500
```

You can change it to any value you like.

```
paul@debian5:~$ HISTSIZE=15000
paul@debian5:~$ echo $HISTSIZE
15000
```

16.7. $HISTFILE

The $HISTFILE variable points to the file that contains your history. The **bash** shell defaults this value to **~/.bash_history**.

```
paul@debian5:~$ echo $HISTFILE
/home/paul/.bash_history
```

A session history is saved to this file when you **exit** the session!

*Closing a gnome-terminal with the mouse, or typing **reboot** as root will NOT save your terminal's history.*

16.8. $HISTFILESIZE

The number of commands kept in your history file can be set using $HISTFILESIZE.

```
paul@debian5:~$ echo $HISTFILESIZE
15000
```

16.9. prevent recording a command

You can prevent a command from being recorded in **history** using a space prefix.

```
paul@debian8:~/github$ echo abc
abc
paul@debian8:~/github$  echo def
def
paul@debian8:~/github$ echo ghi
ghi
paul@debian8:~/github$ history 3
 9501  echo abc
 9502  echo ghi
 9503  history 3
```

16.10. (optional)regular expressions

It is possible to use **regular expressions** when using the **bang** to repeat commands. The screenshot below switches 1 into 2.

```
paul@debian5:~/test$ cat file1
paul@debian5:~/test$ !c:s/1/2
cat file2
hello
paul@debian5:~/test$
```

16.11. (optional) Korn shell history

Repeating a command in the **Korn shell** is very similar. The Korn shell also has the **history** command, but uses the letter **r** to recall lines from history.

This screenshot shows the history command. Note the different meaning of the parameter.

```
$ history 17
17  clear
18  echo hoi
19  history 12
20  echo world
21  history 17
```

Repeating with **r** can be combined with the line numbers given by the history command, or with the first few letters of the command.

```
$ r e
echo world
world
$ cd /etc
$ r
cd /etc
$
```

16.12. practice: shell history

1. Issue the command **echo The answer to the meaning of life, the universe and everything is 42**.

2. Repeat the previous command using only two characters (there are two solutions!)

3. Display the last 5 commands you typed.

4. Issue the long **echo** from question 1 again, using the line numbers you received from the command in question 3.

5. How many commands can be kept in memory for your current shell session ?

6. Where are these commands stored when exiting the shell ?

7. How many commands can be written to the **history file** when exiting your current shell session ?

8. Make sure your current bash shell remembers the next 5000 commands you type.

9. Open more than one console (by press Ctrl-shift-t in gnome-terminal, or by opening an extra putty.exe in MS Windows) with the same user account. When is command history written to the history file ?

16.13. solution: shell history

1. Issue the command **echo The answer to the meaning of life, the universe and everything is 42**.

```
echo The answer to the meaning of life, the universe and everything is 42
```

2. Repeat the previous command using only two characters (there are two solutions!)

```
!!
OR
!e
```

3. Display the last 5 commands you typed.

```
paul@ubu1010:~$ history 5
 52  ls -l
 53  ls
 54  df -h | grep sda
 55  echo The answer to the meaning of life, the universe and everything is 42
 56  history 5
```

You will receive different line numbers.

4. Issue the long **echo** from question 1 again, using the line numbers you received from the command in question 3.

```
paul@ubu1010:~$ !55
echo The answer to the meaning of life, the universe and everything is 42
The answer to the meaning of life, the universe and everything is 42
```

5. How many commands can be kept in memory for your current shell session ?

```
echo $HISTSIZE
```

6. Where are these commands stored when exiting the shell ?

```
echo $HISTFILE
```

7. How many commands can be written to the **history file** when exiting your current shell session ?

```
echo $HISTFILESIZE
```

8. Make sure your current bash shell remembers the next 5000 commands you type.

```
HISTSIZE=5000
```

9. Open more than one console (by press Ctrl-shift-t in gnome-terminal, or by opening an extra putty.exe in MS Windows) with the same user account. When is command history written to the history file ?

```
when you type exit
```

Chapter 17. file globbing

The shell is also responsible for **file globbing** (or dynamic filename generation). This chapter will explain **file globbing**.

17.1. * asterisk

The asterisk ***** is interpreted by the shell as a sign to generate filenames, matching the asterisk to any combination of characters (even none). When no path is given, the shell will use filenames in the current directory. See the man page of **glob(7)** for more information. (This is part of LPI topic 1.103.3.)

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls File*
File4  File55  FileA  Fileab  FileAB
[paul@RHELv4u3 gen]$ ls file*
file1  file2  file3  fileab  fileabc
[paul@RHELv4u3 gen]$ ls *ile55
File55
[paul@RHELv4u3 gen]$ ls F*ile55
File55
[paul@RHELv4u3 gen]$ ls F*55
File55
[paul@RHELv4u3 gen]$
```

17.2. ? question mark

Similar to the asterisk, the question mark **?** is interpreted by the shell as a sign to generate filenames, matching the question mark with exactly one character.

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls File?
File4  FileA
[paul@RHELv4u3 gen]$ ls Fil?4
File4
[paul@RHELv4u3 gen]$ ls Fil??
File4  FileA
[paul@RHELv4u3 gen]$ ls File??
File55  Fileab  FileAB
[paul@RHELv4u3 gen]$
```

17.3. [] square brackets

The square bracket **[** is interpreted by the shell as a sign to generate filenames, matching any of the characters between **[** and the first subsequent **]**. The order in this list between the brackets is not important. Each pair of brackets is replaced by exactly one character.

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls File[5A]
FileA
[paul@RHELv4u3 gen]$ ls File[A5]
FileA
[paul@RHELv4u3 gen]$ ls File[A5][5b]
File55
[paul@RHELv4u3 gen]$ ls File[a5][5b]
File55  Fileab
[paul@RHELv4u3 gen]$ ls File[a5][5b][abcdefghijklm]
ls: File[a5][5b][abcdefghijklm]: No such file or directory
[paul@RHELv4u3 gen]$ ls file[a5][5b][abcdefghijklm]
fileabc
[paul@RHELv4u3 gen]$
```

You can also exclude characters from a list between square brackets with the exclamation mark **!**. And you are allowed to make combinations of these **wild cards**.

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls file[a5][!Z]
fileab
[paul@RHELv4u3 gen]$ ls file[!5]*
file1  file2  file3  fileab  fileabc
[paul@RHELv4u3 gen]$ ls file[!5]?
fileab
[paul@RHELv4u3 gen]$
```

17.4. a-z and 0-9 ranges

The bash shell will also understand ranges of characters between brackets.

```
[paul@RHELv4u3 gen]$ ls
file1  file3  File55  fileab  FileAB   fileabc
file2  File4  FileA   Fileab  fileab2
[paul@RHELv4u3 gen]$ ls file[a-z]*
fileab  fileab2  fileabc
[paul@RHELv4u3 gen]$ ls file[0-9]
file1  file2  file3
[paul@RHELv4u3 gen]$ ls file[a-z][a-z][0-9]*
fileab2
[paul@RHELv4u3 gen]$
```

17.5. $LANG and square brackets

But, don't forget the influence of the **LANG** variable. Some languages include lower case letters in an upper case range (and vice versa).

```
paul@RHELv4u4:~/test$ ls [A-Z]ile?
file1  file2  file3  File4
paul@RHELv4u4:~/test$ ls [a-z]ile?
file1  file2  file3  File4
paul@RHELv4u4:~/test$ echo $LANG
en_US.UTF-8
paul@RHELv4u4:~/test$ LANG=C
paul@RHELv4u4:~/test$ echo $LANG
C
paul@RHELv4u4:~/test$ ls [a-z]ile?
file1  file2  file3
paul@RHELv4u4:~/test$ ls [A-Z]ile?
File4
paul@RHELv4u4:~/test$
```

If **$LC_ALL** is set, then this will also need to be reset to prevent file globbing.

17.6. preventing file globbing

The screenshot below should be no surprise. The **echo *** will echo a * when in an empty directory. And it will echo the names of all files when the directory is not empty.

```
paul@ubu1010:~$ mkdir test42
paul@ubu1010:~$ cd test42
paul@ubu1010:~/test42$ echo *
*
paul@ubu1010:~/test42$ touch file42 file33
paul@ubu1010:~/test42$ echo *
file33 file42
```

Globbing can be prevented using quotes or by escaping the special characters, as shown in this screenshot.

```
paul@ubu1010:~/test42$ echo *
file33 file42
paul@ubu1010:~/test42$ echo \*
*
paul@ubu1010:~/test42$ echo '*'
*
paul@ubu1010:~/test42$ echo "*"
*
```

17.7. practice: shell globbing

1. Create a test directory and enter it.

2. Create the following files :

```
file1
file10
file11
file2
File2
File3
file33
fileAB
filea
fileA
fileAAA
file(
file 2
```

(the last one has 6 characters including a space)

3. List (with ls) all files starting with file

4. List (with ls) all files starting with File

5. List (with ls) all files starting with file and ending in a number.

6. List (with ls) all files starting with file and ending with a letter

7. List (with ls) all files starting with File and having a digit as fifth character.

8. List (with ls) all files starting with File and having a digit as fifth character and nothing else.

9. List (with ls) all files starting with a letter and ending in a number.

10. List (with ls) all files that have exactly five characters.

11. List (with ls) all files that start with f or F and end with 3 or A.

12. List (with ls) all files that start with f have i or R as second character and end in a number.

13. List all files that do not start with the letter F.

14. Copy the value of $LANG to $MyLANG.

15. Show the influence of $LANG in listing A-Z or a-z ranges.

16. You receive information that one of your servers was cracked, the cracker probably replaced the **ls** command. You know that the **echo** command is safe to use. Can **echo** replace **ls** ? How can you list the files in the current directory with **echo** ?

17. Is there another command besides cd to change directories ?

17.8. solution: shell globbing

1. Create a test directory and enter it.

```
mkdir testdir; cd testdir
```

2. Create the following files :

```
file1
file10
file11
file2
File2
File3
file33
fileAB
filea
fileA
fileAAA
file(
file 2
```

(the last one has 6 characters including a space)

```
touch file1 file10 file11 file2 File2 File3
touch file33 fileAB filea fileA fileAAA
touch "file("
touch "file 2"
```

3. List (with ls) all files starting with file

```
ls file*
```

4. List (with ls) all files starting with File

```
ls File*
```

5. List (with ls) all files starting with file and ending in a number.

```
ls file*[0-9]
```

6. List (with ls) all files starting with file and ending with a letter

```
ls file*[a-z]
```

7. List (with ls) all files starting with File and having a digit as fifth character.

```
ls File[0-9]*
```

8. List (with ls) all files starting with File and having a digit as fifth character and nothing else.

```
ls File[0-9]
```

9. List (with ls) all files starting with a letter and ending in a number.

```
ls [a-z]*[0-9]
```

10. List (with ls) all files that have exactly five characters.

```
ls ?????
```

11. List (with ls) all files that start with f or F and end with 3 or A.

```
ls [fF]*[3A]
```

12. List (with ls) all files that start with f have i or R as second character and end in a number.

```
ls f[iR]*[0-9]
```

13. List all files that do not start with the letter F.

```
ls [!F]*
```

14. Copy the value of $LANG to $MyLANG.

```
MyLANG=$LANG
```

15. Show the influence of $LANG in listing A-Z or a-z ranges.

```
see example in book
```

16. You receive information that one of your servers was cracked, the cracker probably replaced the **ls** command. You know that the **echo** command is safe to use. Can **echo** replace **ls** ? How can you list the files in the current directory with **echo** ?

```
echo *
```

17. Is there another command besides cd to change directories ?

```
pushd popd
```

Part V. pipes and commands

Table of Contents

Chapter 18. I/O redirection

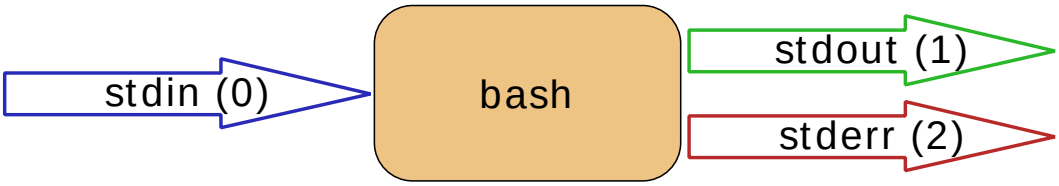
One of the powers of the Unix command line is the use of **input/output redirection** and **pipes**.

This chapter explains **redirection** of input, output and error streams.

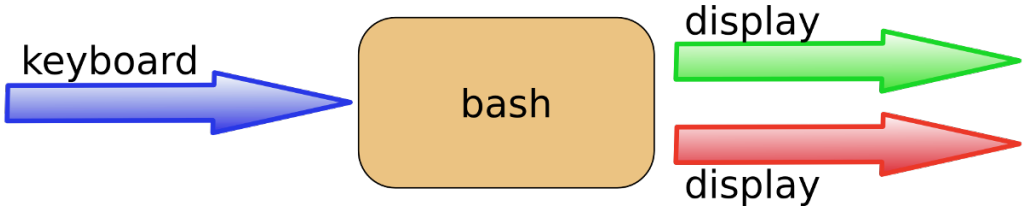
18.1. stdin, stdout, and stderr

The bash shell has three basic streams; it takes input from **stdin** (stream **0**), it sends output to **stdout** (stream **1**) and it sends error messages to **stderr** (stream **2**) .

The drawing below has a graphical interpretation of these three streams.



The keyboard often serves as **stdin**, whereas **stdout** and **stderr** both go to the display. This can be confusing to new Linux users because there is no obvious way to recognize **stdout** from **stderr**. Experienced users know that separating output from errors can be very useful.

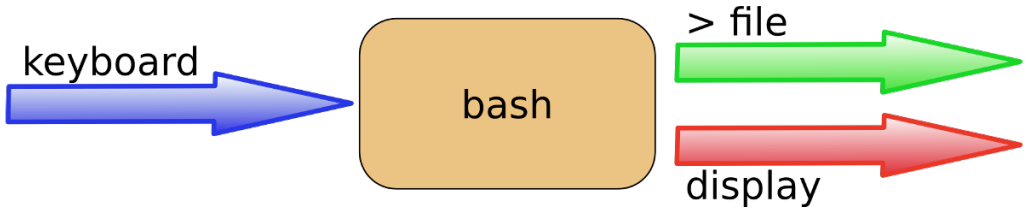


The next sections will explain how to redirect these streams.

18.2. output redirection

18.2.1. > stdout

stdout can be redirected with a **greater than** sign. While scanning the line, the shell will see the > sign and will clear the file.



The > notation is in fact the abbreviation of **1>** (**stdout** being referred to as stream **1**).

```
[paul@RHELv4u3 ~]$ echo It is cold today!
It is cold today!
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$
```

Note that the bash shell effectively **removes** the redirection from the command line before argument 0 is executed. This means that in the case of this command:

```
echo hello > greetings.txt
```

the shell only counts two arguments (echo = argument 0, hello = argument 1). The redirection is removed before the argument counting takes place.

18.2.2. output file is erased

While scanning the line, the shell will see the > sign and **will clear the file**! Since this happens before resolving **argument 0**, this means that even when the command fails, the file will have been cleared!

```
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$ zcho It is cold today! > winter.txt
-bash: zcho: command not found
[paul@RHELv4u3 ~]$ cat winter.txt
[paul@RHELv4u3 ~]$
```

18.2.3. noclobber

Erasing a file while using > can be prevented by setting the **noclobber** option.

```
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$ set -o noclobber
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
-bash: winter.txt: cannot overwrite existing file
[paul@RHELv4u3 ~]$ set +o noclobber
[paul@RHELv4u3 ~]$
```

18.2.4. overruling noclobber

The **noclobber** can be overruled with >|.

```
[paul@RHELv4u3 ~]$ set -o noclobber
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
-bash: winter.txt: cannot overwrite existing file
[paul@RHELv4u3 ~]$ echo It is very cold today! >| winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is very cold today!
[paul@RHELv4u3 ~]$
```

18.2.5. >> append

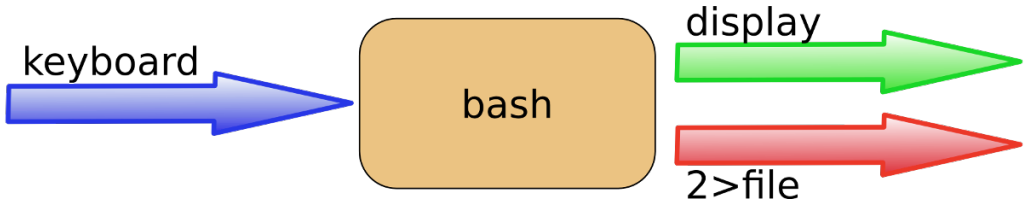
Use >> to **append** output to a file.

```
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$ echo Where is the summer ? >> winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
Where is the summer ?
[paul@RHELv4u3 ~]$
```

18.3. error redirection

18.3.1. 2> stderr

Redirecting **stderr** is done with **2>**. This can be very useful to prevent error messages from cluttering your screen.



The screenshot below shows redirection of **stdout** to a file, and **stderr** to **/dev/null**. Writing **1>** is the same as >.

```
[paul@RHELv4u3 ~]$ find / > allfiles.txt 2> /dev/null
[paul@RHELv4u3 ~]$
```

18.3.2. 2>&1

To redirect both **stdout** and **stderr** to the same file, use **2>&1**.

```
[paul@RHELv4u3 ~]$ find / > allfiles_and_errors.txt 2>&1
[paul@RHELv4u3 ~]$
```

Note that the order of redirections is significant. For example, the command

```
ls > dirlist 2>&1
```

directs both standard output (file descriptor 1) and standard error (file descriptor 2) to the file dirlist, while the command

```
ls 2>&1 > dirlist
```

directs only the standard output to file dirlist, because the standard error made a copy of the standard output before the standard output was redirected to dirlist.

18.4. output redirection and pipes

By default you cannot grep inside **stderr** when using pipes on the command line, because only **stdout** is passed.

```
paul@debian7:~$ rm file42 file33 file1201 | grep file42
rm: cannot remove ‘file42’: No such file or directory
rm: cannot remove ‘file33’: No such file or directory
rm: cannot remove ‘file1201’: No such file or directory
```

With **2>&1** you can force **stderr** to go to **stdout**. This enables the next command in the pipe to act on both streams.

```
paul@debian7:~$ rm file42 file33 file1201 2>&1 | grep file42
rm: cannot remove ‘file42’: No such file or directory
```

You cannot use both **1>&2** and **2>&1** to switch **stdout** and **stderr**.

```
paul@debian7:~$ rm file42 file33 file1201 2>&1 1>&2 | grep file42
rm: cannot remove ‘file42’: No such file or directory
paul@debian7:~$ echo file42 2>&1 1>&2 | sed 's/file42/FILE42/'
FILE42
```

You need a third stream to switch stdout and stderr after a pipe symbol.

```
paul@debian7:~$ echo file42 3>&1 1>&2 2>&3 | sed 's/file42/FILE42/'
file42
paul@debian7:~$ rm file42 3>&1 1>&2 2>&3 | sed 's/file42/FILE42/'
rm: cannot remove ‘FILE42’: No such file or directory
```

18.5. joining stdout and stderr

The **&>** construction will put both **stdout** and **stderr** in one stream (to a file).

```
paul@debian7:~$ rm file42 &> out_and_err
paul@debian7:~$ cat out_and_err
rm: cannot remove ‘file42’: No such file or directory
paul@debian7:~$ echo file42 &> out_and_err
paul@debian7:~$ cat out_and_err
file42
paul@debian7:~$
```

18.6. input redirection

18.6.1. < stdin

Redirecting **stdin** is done with < (short for 0<).

```
[paul@RHEL4b ~]$ cat < text.txt
one
two
[paul@RHEL4b ~]$ tr 'onetw' 'ONEZZ' < text.txt
ONE
ZZO
[paul@RHEL4b ~]$
```

18.6.2. << here document

The **here document** (sometimes called here-is-document) is a way to append input until a certain sequence (usually EOF) is encountered. The **EOF** marker can be typed literally or can be called with Ctrl-D.

```
[paul@RHEL4b ~]$ cat <<EOF > text.txt
> one
> two
> EOF
[paul@RHEL4b ~]$ cat text.txt
one
two
[paul@RHEL4b ~]$ cat <<brol > text.txt
> brel
> brol
[paul@RHEL4b ~]$ cat text.txt
brel
[paul@RHEL4b ~]$
```

18.6.3. <<< here string

The **here string** can be used to directly pass strings to a command. The result is the same as using **echo string | command** (but you have one less process running).

```
paul@ubu1110~$ base64 <<< linux-training.be
bGludXgtdHJhaW5pbmcuYmUK
paul@ubu1110~$ base64 -d <<< bGludXgtdHJhaW5pbmcuYmUK
linux-training.be
```

See rfc 3548 for more information about **base64**.

18.7. confusing redirection

The shell will scan the whole line before applying redirection. The following command line is very readable and is correct.

```
cat winter.txt > snow.txt 2> errors.txt
```

But this one is also correct, but less readable.

```
2> errors.txt cat winter.txt > snow.txt
```

Even this will be understood perfectly by the shell.

```
< winter.txt > snow.txt 2> errors.txt cat
```

18.8. quick file clear

So what is the quickest way to clear a file ?

```
>foo
```

And what is the quickest way to clear a file when the **noclobber** option is set ?

```
>|bar
```

18.9. practice: input/output redirection

1. Activate the **noclobber** shell option.

2. Verify that **noclobber** is active by repeating an **ls** on **/etc/** with redirected output to a file.

3. When listing all shell options, which character represents the **noclobber** option ?

4. Deactivate the **noclobber** option.

5. Make sure you have two shells open on the same computer. Create an empty **tailing.txt** file. Then type **tail -f tailing.txt**. Use the second shell to **append** a line of text to that file. Verify that the first shell displays this line.

6. Create a file that contains the names of five people. Use **cat** and output redirection to create the file and use a **here document** to end the input.

18.10. solution: input/output redirection

1. Activate the **noclobber** shell option.

```
set -o noclobber
set -C
```

2. Verify that **noclobber** is active by repeating an **ls** on **/etc/** with redirected output to a file.

```
ls /etc > etc.txt
ls /etc > etc.txt (should not work)
```

4. When listing all shell options, which character represents the **noclobber** option ?

```
echo $- (noclobber is visible as C)
```

5. Deactivate the **noclobber** option.

```
set +o noclobber
```

6. Make sure you have two shells open on the same computer. Create an empty **tailing.txt** file. Then type **tail -f tailing.txt**. Use the second shell to **append** a line of text to that file. Verify that the first shell displays this line.

```
paul@deb503:~$ > tailing.txt
paul@deb503:~$ tail -f tailing.txt
hello
world

in the other shell:
paul@deb503:~$ echo hello >> tailing.txt
paul@deb503:~$ echo world >> tailing.txt
```

7. Create a file that contains the names of five people. Use **cat** and output redirection to create the file and use a **here document** to end the input.

```
paul@deb503:~$ cat > tennis.txt << ace
> Justine Henin
> Venus Williams
> Serena Williams
> Martina Hingis
> Kim Clijsters
> ace
paul@deb503:~$ cat tennis.txt
Justine Henin
Venus Williams
Serena Williams
Martina Hingis
Kim Clijsters
paul@deb503:~$
```

Chapter 19. filters

Commands that are created to be used with a **pipe** are often called **filters**. These **filters** are very small programs that do one specific thing very efficiently. They can be used as **building blocks**.

This chapter will introduce you to the most common **filters**. The combination of simple commands and filters in a long **pipe** allows you to design elegant solutions.

19.1. cat

When between two **pipes**, the **cat** command does nothing (except putting **stdin** on **stdout**).

```
[paul@RHEL4b pipes]$ tac count.txt | cat | cat | cat | cat | cat
five
four
three
two
one
[paul@RHEL4b pipes]$
```

19.2. tee

Writing long **pipes** in Unix is fun, but sometimes you may want intermediate results. This is were **tee** comes in handy. The **tee** filter puts **stdin** on **stdout** and also into a file. So **tee** is almost the same as **cat**, except that it has two identical outputs.

```
[paul@RHEL4b pipes]$ tac count.txt | tee temp.txt | tac
one
two
three
four
five
[paul@RHEL4b pipes]$ cat temp.txt
five
four
three
two
one
[paul@RHEL4b pipes]$
```

19.3. grep

The **grep** filter is famous among Unix users. The most common use of **grep** is to filter lines of text containing (or not containing) a certain string.

```
[paul@RHEL4b pipes]$ cat tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$ cat tennis.txt | grep Williams
Serena Williams, usa
Venus Williams, USA
```

You can write this without the cat.

```
[paul@RHEL4b pipes]$ grep Williams tennis.txt
Serena Williams, usa
Venus Williams, USA
```

One of the most useful options of grep is **grep -i** which filters in a case insensitive way.

```
[paul@RHEL4b pipes]$ grep Bel tennis.txt
Justine Henin, Bel
[paul@RHEL4b pipes]$ grep -i Bel tennis.txt
```

```
Kim Clijsters, BEL
Justine Henin, Bel
[paul@RHEL4b pipes]$
```

Another very useful option is **grep -v** which outputs lines not matching the string.

```
[paul@RHEL4b pipes]$ grep -v Fra tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$
```

And of course, both options can be combined to filter all lines not containing a case insensitive string.

```
[paul@RHEL4b pipes]$ grep -vi usa tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
[paul@RHEL4b pipes]$
```

With **grep -A1** one line **after** the result is also displayed.

```
paul@debian5:~/pipes$ grep -A1 Henin tennis.txt
Justine Henin, Bel
Serena Williams, usa
```

With **grep -B1** one line **before** the result is also displayed.

```
paul@debian5:~/pipes$ grep -B1 Henin tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
```

With **grep -C1** (context) one line **before** and one **after** are also displayed. All three options (A,B, and C) can display any number of lines (using e.g. A2, B4 or C20).

```
paul@debian5:~/pipes$ grep -C1 Henin tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
```

19.4. cut

The **cut** filter can select columns from files, depending on a delimiter or a count of bytes. The screenshot below uses **cut** to filter for the username and userid in the **/etc/passwd** file. It uses the colon as a delimiter, and selects fields 1 and 3.

```
[[paul@RHEL4b pipes]$ cut -d: -f1,3 /etc/passwd | tail -4
Figo:510
Pfaff:511
Harry:516
Hermione:517
[paul@RHEL4b pipes]$
```

When using a space as the delimiter for **cut**, you have to quote the space.

```
[paul@RHEL4b pipes]$ cut -d" " -f1 tennis.txt
Amelie
Kim
Justine
Serena
Venus
[paul@RHEL4b pipes]$
```

This example uses **cut** to display the second to the seventh character of **/etc/passwd**.

```
[paul@RHEL4b pipes]$ cut -c2-7 /etc/passwd | tail -4
igo:x:
faff:x
arry:x
ermion
[paul@RHEL4b pipes]$
```

19.5. tr

You can translate characters with **tr**. The screenshot shows the translation of all occurrences of e to E.

```
[paul@RHEL4b pipes]$ cat tennis.txt | tr 'e' 'E'
AmEliE MaurEsmo, Fra
Kim ClijstErs, BEL
JustinE HEnin, BEl
SErEna Williams, usa
VEnus Williams, USA
```

Here we set all letters to uppercase by defining two ranges.

```
[paul@RHEL4b pipes]$ cat tennis.txt | tr 'a-z' 'A-Z'
AMELIE MAURESMO, FRA
KIM CLIJSTERS, BEL
JUSTINE HENIN, BEL
SERENA WILLIAMS, USA
VENUS WILLIAMS, USA
[paul@RHEL4b pipes]$
```

Here we translate all newlines to spaces.

```
[paul@RHEL4b pipes]$ cat count.txt
one
two
```

```
three
four
five
[paul@RHEL4b pipes]$ cat count.txt | tr '\n' ' '
one two three four five [paul@RHEL4b pipes]$
```

The **tr -s** filter can also be used to squeeze multiple occurrences of a character to one.

```
[paul@RHEL4b pipes]$ cat spaces.txt
one    two        three
     four   five  six
[paul@RHEL4b pipes]$ cat spaces.txt | tr -s ' '
one two three
 four five six
[paul@RHEL4b pipes]$
```

You can also use **tr** to 'encrypt' texts with **rot13**.

```
[paul@RHEL4b pipes]$ cat count.txt | tr 'a-z' 'nopqrstuvwxyzabcdefghijklm'
bar
gjb
guerr
sbhe
svir
[paul@RHEL4b pipes]$ cat count.txt | tr 'a-z' 'n-za-m'
bar
gjb
guerr
sbhe
svir
[paul@RHEL4b pipes]$
```

This last example uses **tr -d** to delete characters.

```
paul@debian5:~/pipes$ cat tennis.txt | tr -d e
Amli Maursmo, Fra
Kim Clijstrs, BEL
Justin Hnin, Bl
Srna Williams, usa
Vnus Williams, USA
```

19.6. wc

Counting words, lines and characters is easy with **wc**.

```
[paul@RHEL4b pipes]$ wc tennis.txt
  5  15 100 tennis.txt
[paul@RHEL4b pipes]$ wc -l tennis.txt
5 tennis.txt
[paul@RHEL4b pipes]$ wc -w tennis.txt
15 tennis.txt
[paul@RHEL4b pipes]$ wc -c tennis.txt
100 tennis.txt
[paul@RHEL4b pipes]$
```

19.7. sort

The **sort** filter will default to an alphabetical sort.

```
paul@debian5:~/pipes$ cat music.txt
Queen
Brel
Led Zeppelin
Abba
paul@debian5:~/pipes$ sort music.txt
Abba
Brel
Led Zeppelin
Queen
```

But the **sort** filter has many options to tweak its usage. This example shows sorting different columns (column 1 or column 2).

```
[paul@RHEL4b pipes]$ sort -k1 country.txt
Belgium, Brussels, 10
France, Paris, 60
Germany, Berlin, 100
Iran, Teheran, 70
Italy, Rome, 50
[paul@RHEL4b pipes]$ sort -k2 country.txt
Germany, Berlin, 100
Belgium, Brussels, 10
France, Paris, 60
Italy, Rome, 50
Iran, Teheran, 70
```

The screenshot below shows the difference between an alphabetical sort and a numerical sort (both on the third column).

```
[paul@RHEL4b pipes]$ sort -k3 country.txt
Belgium, Brussels, 10
Germany, Berlin, 100
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
[paul@RHEL4b pipes]$ sort -n -k3 country.txt
Belgium, Brussels, 10
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
Germany, Berlin, 100
```

19.8. uniq

With **uniq** you can remove duplicates from a **sorted list**.

```
paul@debian5:~/pipes$ cat music.txt
Queen
Brel
Queen
Abba
paul@debian5:~/pipes$ sort music.txt
Abba
Brel
Queen
Queen
paul@debian5:~/pipes$ sort music.txt |uniq
Abba
Brel
Queen
```

uniq can also count occurrences with the **-c** option.

```
paul@debian5:~/pipes$ sort music.txt |uniq -c
      1 Abba
      1 Brel
      2 Queen
```

19.9. comm

Comparing streams (or files) can be done with the **comm**. By default **comm** will output three columns. In this example, Abba, Cure and Queen are in both lists, Bowie and Sweet are only in the first file, Turner is only in the second.

```
paul@debian5:~/pipes$ cat > list1.txt
Abba
Bowie
Cure
Queen
Sweet
paul@debian5:~/pipes$ cat > list2.txt
Abba
Cure
Queen
Turner
paul@debian5:~/pipes$ comm list1.txt list2.txt
                Abba
Bowie
                Cure
                Queen
Sweet
        Turner
```

The output of **comm** can be easier to read when outputting only a single column. The digits point out which output columns should not be displayed.

```
paul@debian5:~/pipes$ comm -12 list1.txt list2.txt
Abba
Cure
Queen
paul@debian5:~/pipes$ comm -13 list1.txt list2.txt
Turner
paul@debian5:~/pipes$ comm -23 list1.txt list2.txt
Bowie
Sweet
```

19.10. od

European humans like to work with ascii characters, but computers store files in bytes. The example below creates a simple file, and then uses **od** to show the contents of the file in hexadecimal bytes

```
paul@laika:~/test$ cat > text.txt
abcdefg
1234567
paul@laika:~/test$ od -t x1 text.txt
0000000 61 62 63 64 65 66 67 0a 31 32 33 34 35 36 37 0a
0000020
```

The same file can also be displayed in octal bytes.

```
paul@laika:~/test$ od -b text.txt
0000000 141 142 143 144 145 146 147 012 061 062 063 064 065 066 067 012
0000020
```

And here is the file in ascii (or backslashed) characters.

```
paul@laika:~/test$ od -c text.txt
0000000   a   b   c   d   e   f   g  \n   1   2   3   4   5   6   7  \n
0000020
```

19.11. sed

The **s**tream **ed**itor **sed** can perform editing functions in the stream, using **regular expressions**.

```
paul@debian5:~/pipes$ echo level5 | sed 's/5/42/'
level42
paul@debian5:~/pipes$ echo level5 | sed 's/level/jump/'
jump5
```

Add **g** for global replacements (all occurrences of the string per line).

```
paul@debian5:~/pipes$ echo level5 level7 | sed 's/level/jump/'
jump5 level7
paul@debian5:~/pipes$ echo level5 level7 | sed 's/level/jump/g'
jump5 jump7
```

With **d** you can remove lines from a stream containing a character.

```
paul@debian5:~/test42$ cat tennis.txt
Venus Williams, USA
Martina Hingis, SUI
Justine Henin, BE
Serena williams, USA
Kim Clijsters, BE
Yanina Wickmayer, BE
paul@debian5:~/test42$ cat tennis.txt | sed '/BE/d'
Venus Williams, USA
Martina Hingis, SUI
Serena williams, USA
```

19.12. pipe examples

19.12.1. who | wc

How many users are logged on to this system ?

```
[paul@RHEL4b pipes]$ who
root     tty1         Jul 25 10:50
paul     pts/0        Jul 25 09:29 (laika)
Harry    pts/1        Jul 25 12:26 (barry)
paul     pts/2        Jul 25 12:26 (pasha)
[paul@RHEL4b pipes]$ who | wc -l
4
```

19.12.2. who | cut | sort

Display a sorted list of logged on users.

```
[paul@RHEL4b pipes]$ who | cut -d' ' -f1 | sort
Harry
paul
paul
root
```

Display a sorted list of logged on users, but every user only once .

```
[paul@RHEL4b pipes]$ who | cut -d' ' -f1 | sort | uniq
Harry
paul
root
```

19.12.3. grep | cut

Display a list of all bash **user accounts** on this computer. Users accounts are explained in detail later.

```
paul@debian5:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
paul:x:1000:1000:paul,,,:/home/paul:/bin/bash
serena:x:1001:1001::/home/serena:/bin/bash
paul@debian5:~$ grep bash /etc/passwd | cut -d: -f1
root
paul
serena
```

19.13. practice: filters

1. Put a sorted list of all bash users in bashusers.txt.

2. Put a sorted list of all logged on users in onlineusers.txt.

3. Make a list of all filenames in **/etc** that contain the string **conf** in their filename.

4. Make a sorted list of all files in **/etc** that contain the case insensitive string **conf** in their filename.

5. Look at the output of **/sbin/ifconfig**. Write a line that displays only ip address and the subnet mask.

6. Write a line that removes all non-letters from a stream.

7. Write a line that receives a text file, and outputs all words on a separate line.

8. Write a spell checker on the command line. (There may be a dictionary in **/usr/share/dict/** .)

19.14. solution: filters

1. Put a sorted list of all bash users in bashusers.txt.

```
grep bash /etc/passwd | cut -d: -f1 | sort > bashusers.txt
```

2. Put a sorted list of all logged on users in onlineusers.txt.

```
who | cut -d' ' -f1 | sort > onlineusers.txt
```

3. Make a list of all filenames in **/etc** that contain the string **conf** in their filename.

```
ls /etc | grep conf
```

4. Make a sorted list of all files in **/etc** that contain the case insensitive string **conf** in their filename.

```
ls /etc | grep -i conf | sort
```

5. Look at the output of **/sbin/ifconfig**. Write a line that displays only ip address and the subnet mask.

```
/sbin/ifconfig | head -2 | grep 'inet ' | tr -s ' ' | cut -d' ' -f3,5
```

6. Write a line that removes all non-letters from a stream.

```
paul@deb503:~$ cat text
This is, yes really! , a text with ?&* too many str$ange# characters ;-)
paul@deb503:~$ cat text | tr -d ',!$?.*&^%#@;()-'
This is yes really  a text with  too many strange characters
```

7. Write a line that receives a text file, and outputs all words on a separate line.

```
paul@deb503:~$ cat text2
it is very cold today without the sun

paul@deb503:~$ cat text2 | tr ' ' '\n'
it
is
very
cold
today
without
the
sun
```

8. Write a spell checker on the command line. (There may be a dictionary in **/usr/share/dict/** .)

```
paul@rhel ~$ echo "The zun is shining today" > text

paul@rhel ~$ cat > DICT
is
shining
sun
the
```

```
today

paul@rhel ~$ cat text | tr 'A-Z ' 'a-z\n' | sort | uniq | comm -23 - DICT
zun
```

You could also add the solution from question number 6 to remove non-letters, and **tr -s ' '** to remove redundant spaces.

Chapter 20. basic Unix tools

This chapter introduces commands to **find** or **locate** files and to **compress** files, together with other common tools that were not discussed before. While the tools discussed here are technically not considered **filters**, they can be used in **pipes**.

20.1. find

The **find** command can be very useful at the start of a pipe to search for files. Here are some examples. You might want to add **2>/dev/null** to the command lines to avoid cluttering your screen with error messages.

Find all files in **/etc** and put the list in etcfiles.txt

```
find /etc > etcfiles.txt
```

Find all files of the entire system and put the list in allfiles.txt

```
find / > allfiles.txt
```

Find files that end in .conf in the current directory (and all subdirs).

```
find . -name "*.conf"
```

Find files of type file (not directory, pipe or etc.) that end in .conf.

```
find . -type f -name "*.conf"
```

Find files of type directory that end in .bak .

```
find /data -type d -name "*.bak"
```

Find files that are newer than file42.txt

```
find . -newer file42.txt
```

Find can also execute another command on every file found. This example will look for *.odf files and copy them to /backup/.

```
find /data -name "*.odf" -exec cp {} /backup/ \;
```

Find can also execute, after your confirmation, another command on every file found. This example will remove *.odf files if you approve of it for every file found.

```
find /data -name "*.odf" -ok rm {} \;
```

20.2. locate

The **locate** tool is very different from **find** in that it uses an index to locate files. This is a lot faster than traversing all the directories, but it also means that it is always outdated. If the index does not exist yet, then you have to create it (as root on Red Hat Enterprise Linux) with the **updatedb** command.

```
[paul@RHEL4b ~]$ locate Samba
warning: locate: could not open database: /var/lib/slocate/slocate.db:...
warning: You need to run the 'updatedb' command (as root) to create th...
Please have a look at /etc/updatedb.conf to enable the daily cron job.
[paul@RHEL4b ~]$ updatedb
fatal error: updatedb: You are not authorized to create a default sloc...
[paul@RHEL4b ~]$ su -
Password:
[root@RHEL4b ~]# updatedb
[root@RHEL4b ~]#
```

Most Linux distributions will schedule the **updatedb** to run once every day.

20.3. date

The **date** command can display the date, time, time zone and more.

```
paul@rhel55 ~$ date
Sat Apr 17 12:44:30 CEST 2010
```

A date string can be customised to display the format of your choice. Check the man page for more options.

```
paul@rhel55 ~$ date +'%A %d-%m-%Y'
Saturday 17-04-2010
```

Time on any Unix is calculated in number of seconds since 1969 (the first second being the first second of the first of January 1970). Use **date +%s** to display Unix time in seconds.

```
paul@rhel55 ~$ date +%s
1271501080
```

When will this seconds counter reach two thousand million ?

```
paul@rhel55 ~$ date -d '1970-01-01 + 2000000000 seconds'
Wed May 18 04:33:20 CEST 2033
```

20.4. cal

The **cal** command displays the current month, with the current day highlighted.

```
paul@rhel55 ~$ cal
     April 2010
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

You can select any month in the past or the future.

```
paul@rhel55 ~$ cal 2 1970
   February 1970
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
```

20.5. sleep

The **sleep** command is sometimes used in scripts to wait a number of seconds. This example shows a five second **sleep**.

```
paul@rhel55 ~$ sleep 5
paul@rhel55 ~$
```

20.6. time

The **time** command can display how long it takes to execute a command. The **date** command takes only a little time.

```
paul@rhel55 ~$ time date
Sat Apr 17 13:08:27 CEST 2010

real    0m0.014s
user    0m0.008s
sys     0m0.006s
```

The **sleep 5** command takes five **real** seconds to execute, but consumes little **cpu time**.

```
paul@rhel55 ~$ time sleep 5

real    0m5.018s
user    0m0.005s
sys     0m0.011s
```

This **bzip2** command compresses a file and uses a lot of **cpu time**.

```
paul@rhel55 ~$ time bzip2 text.txt

real    0m2.368s
user    0m0.847s
sys     0m0.539s
```

20.7. gzip - gunzip

Users never have enough disk space, so compression comes in handy. The **gzip** command can make files take up less space.

```
paul@rhel55 ~$ ls -lh text.txt
-rw-rw-r-- 1 paul paul 6.4M Apr 17 13:11 text.txt
paul@rhel55 ~$ gzip text.txt
paul@rhel55 ~$ ls -lh text.txt.gz
-rw-rw-r-- 1 paul paul 760K Apr 17 13:11 text.txt.gz
```

You can get the original back with **gunzip**.

```
paul@rhel55 ~$ gunzip text.txt.gz
paul@rhel55 ~$ ls -lh text.txt
-rw-rw-r-- 1 paul paul 6.4M Apr 17 13:11 text.txt
```

20.8. zcat - zmore

Text files that are compressed with **gzip** can be viewed with **zcat** and **zmore**.

```
paul@rhel55 ~$ head -4 text.txt
/
/opt
/opt/VBoxGuestAdditions-3.1.6
/opt/VBoxGuestAdditions-3.1.6/routines.sh
paul@rhel55 ~$ gzip text.txt
paul@rhel55 ~$ zcat text.txt.gz | head -4
/
/opt
/opt/VBoxGuestAdditions-3.1.6
/opt/VBoxGuestAdditions-3.1.6/routines.sh
```

20.9. bzip2 - bunzip2

Files can also be compressed with **bzip2** which takes a little more time than **gzip**, but compresses better.

```
paul@rhel55 ~$ bzip2 text.txt
paul@rhel55 ~$ ls -lh text.txt.bz2
-rw-rw-r-- 1 paul paul 569K Apr 17 13:11 text.txt.bz2
```

Files can be uncompressed again with **bunzip2**.

```
paul@rhel55 ~$ bunzip2 text.txt.bz2
paul@rhel55 ~$ ls -lh text.txt
-rw-rw-r-- 1 paul paul 6.4M Apr 17 13:11 text.txt
```

20.10. bzcat - bzmore

And in the same way **bzcat** and **bzmore** can display files compressed with **bzip2**.

```
paul@rhel55 ~$ bzip2 text.txt
paul@rhel55 ~$ bzcat text.txt.bz2 | head -4
/
/opt
/opt/VBoxGuestAdditions-3.1.6
/opt/VBoxGuestAdditions-3.1.6/routines.sh
```

20.11. practice: basic Unix tools

1. Explain the difference between these two commands. This question is very important. If you don't know the answer, then look back at the **shell** chapter.

```
find /data -name "*.txt"
```

```
find /data -name *.txt
```

2. Explain the difference between these two statements. Will they both work when there are 200 **.odf** files in **/data** ? How about when there are 2 million .odf files ?

```
find /data -name "*.odf" > data_odf.txt
```

```
find /data/*.odf > data_odf.txt
```

3. Write a find command that finds all files created after January 30th 2010.

4. Write a find command that finds all *.odf files created in September 2009.

5. Count the number of *.conf files in /etc and all its subdirs.

6. Here are two commands that do the same thing: copy *.odf files to /backup/ . What would be a reason to replace the first command with the second ? Again, this is an important question.

```
cp -r /data/*.odf /backup/
```

```
find /data -name "*.odf" -exec cp {} /backup/ \;
```

7. Create a file called **loctest.txt**. Can you find this file with **locate** ? Why not ? How do you make locate find this file ?

8. Use find and -exec to rename all .htm files to .html.

9. Issue the **date** command. Now display the date in YYYY/MM/DD format.

10. Issue the **cal** command. Display a calendar of 1582 and 1752. Notice anything special ?

20.12. solution: basic Unix tools

1. Explain the difference between these two commands. This question is very important. If you don't know the answer, then look back at the **shell** chapter.

```
find /data -name "*.txt"
```

```
find /data -name *.txt
```

When ***.txt** is quoted then the shell will not touch it. The **find** tool will look in the **/data** for all files ending in **.txt**.

When ***.txt** is not quoted then the shell might expand this (when one or more files that ends in **.txt** exist in the current directory). The **find** might show a different result, or can result in a syntax error.

2. Explain the difference between these two statements. Will they both work when there are 200 **.odf** files in **/data** ? How about when there are 2 million .odf files ?

```
find /data -name "*.odf" > data_odf.txt
```

```
find /data/*.odf > data_odf.txt
```

The first **find** will output all **.odf** filenames in **/data** and all subdirectories. The shell will redirect this to a file.

The second find will output all files named **.odf** in **/data** and will also output all files that exist in directories named ***.odf** (in **/data**).

With two million files the command line would be expanded beyond the maximum that the shell can accept. The last part of the command line would be lost.

3. Write a find command that finds all files created after January 30th 2010.

```
touch -t 201001302359 marker_date
find . -type f -newer marker_date
```

```
There is another solution :
find . -type f -newerat "20100130 23:59:59"
```

4. Write a find command that finds all *.odf files created in September 2009.

```
touch -t 200908312359 marker_start
touch -t 200910010000 marker_end
find . -type f -name "*.odf" -newer marker_start ! -newer marker_end
```

The exclamation mark **! -newer** can be read as **not newer**.

5. Count the number of *.conf files in /etc and all its subdirs.

```
find /etc -type f -name '*.conf' | wc -l
```

6. Here are two commands that do the same thing: copy *.odf files to /backup/ . What would be a reason to replace the first command with the second ? Again, this is an important question.

```
cp -r /data/*.odf /backup/
```

```
find /data -name "*.odf" -exec cp {} /backup/ \;
```

The first might fail when there are too many files to fit on one command line.

7. Create a file called **loctest.txt**. Can you find this file with **locate** ? Why not ? How do you make locate find this file ?

You cannot locate this with **locate** because it is not yet in the index.

```
updatedb
```

8. Use find and -exec to rename all .htm files to .html.

```
paul@rhel55 ~$ find . -name '*.htm'
./one.htm
./two.htm
paul@rhel55 ~$ find . -name '*.htm' -exec mv {} {}l \;
paul@rhel55 ~$ find . -name '*.htm*'
./one.html
./two.html
```

9. Issue the **date** command. Now display the date in YYYY/MM/DD format.

```
date +%Y/%m/%d
```

10. Issue the **cal** command. Display a calendar of 1582 and 1752. Notice anything special ?

```
cal 1582
```

The calendars are different depending on the country. Check http://linux-training.be/files/studentfiles/dates.txt

Chapter 21. regular expressions

Regular expressions are a very powerful tool in Linux. They can be used with a variety of programs like bash, vi, rename, grep, sed, and more.

This chapter introduces you to the basics of **regular expressions**.

21.1. regex versions

There are three different versions of regular expression syntax:

```
BRE: Basic Regular Expressions
ERE: Extended Regular Expressions
PRCE: Perl Regular Expressions
```

Depending on the tool being used, one or more of these syntaxes can be used.

For example the **grep** tool has the **-E** option to force a string to be read as ERE while **-G** forces BRE and **-P** forces PRCE.

Note that **grep** also has **-F** to force the string to be read literally.

The **sed** tool also has options to choose a regex syntax.

Read the manual of the tools you use!

21.2. grep

21.2.1. print lines matching a pattern

grep is a popular Linux tool to search for lines that match a certain pattern. Below are some examples of the simplest **regular expressions**.

This is the contents of the test file. This file contains three lines (or three **newline** characters).

```
paul@rhel65:~$ cat names
Tania
Laura
Valentina
```

When **grepping** for a single character, only the lines containing that character are returned.

```
paul@rhel65:~$ grep u names
Laura
paul@rhel65:~$ grep e names
Valentina
paul@rhel65:~$ grep i names
Tania
Valentina
```

The pattern matching in this example should be very straightforward; if the given character occurs on a line, then **grep** will return that line.

21.2.2. concatenating characters

Two concatenated characters will have to be concatenated in the same way to have a match.

This example demonstrates that **ia** will match Tan**ia** but not V**a**lent**i**na and **in** will match Valent**in**a but not Ta**ni**a.

```
paul@rhel65:~$ grep a names
Tania
Laura
Valentina
paul@rhel65:~$ grep ia names
Tania
paul@rhel65:~$ grep in names
Valentina
paul@rhel65:~$
```

21.2.3. one or the other

PRCE and ERE both use the pipe symbol to signify OR. In this example we **grep** for lines containing the letter i or the letter a.

```
paul@debian7:~$ cat list
Tania
Laura
paul@debian7:~$ grep -E 'i|a' list
Tania
Laura
```

Note that we use the **-E** switch of grep to force interpretion of our string as an ERE.

We need to **escape** the pipe symbol in a BRE to get the same logical OR.

```
paul@debian7:~$ grep -G 'i|a' list
paul@debian7:~$ grep -G 'i\|a' list
Tania
Laura
```

21.2.4. one or more

The ***** signifies zero, one or more occurences of the previous and the **+** signifies one or more of the previous.

```
paul@debian7:~$ cat list2
ll
lol
lool
loool
paul@debian7:~$ grep -E 'o*' list2
ll
lol
lool
loool
paul@debian7:~$ grep -E 'o+' list2
lol
lool
loool
paul@debian7:~$
```

21.2.5. match the end of a string

For the following examples, we will use this file.

```
paul@debian7:~$ cat names
Tania
Laura
Valentina
Fleur
Floor
```

The two examples below show how to use the **dollar character** to match the end of a string.

```
paul@debian7:~$ grep a$ names
Tania
Laura
Valentina
paul@debian7:~$ grep r$ names
Fleur
Floor
```

21.2.6. match the start of a string

The **caret character (^)** will match a string at the start (or the beginning) of a line.

Given the same file as above, here are two examples.

```
paul@debian7:~$ grep ^Val names
Valentina
paul@debian7:~$ grep ^F names
Fleur
Floor
```

Both the dollar sign and the little hat are called **anchors** in a regex.

21.2.7. separating words

Regular expressions use a **\b** sequence to reference a word separator. Take for example this file:

```
paul@debian7:~$ cat text
The governer is governing.
The winter is over.
Can you get over there?
```

Simply grepping for **over** will give too many results.

```
paul@debian7:~$ grep over text
The governer is governing.
The winter is over.
Can you get over there?
```

Surrounding the searched word with spaces is not a good solution (because other characters can be word separators). This screenshot below show how to use **\b** to find only the searched word:

```
paul@debian7:~$ grep '\bover\b' text
The winter is over.
Can you get over there?
paul@debian7:~$
```

Note that **grep** also has a **-w** option to grep for words.

```
paul@debian7:~$ cat text
The governer is governing.
The winter is over.
Can you get over there?
paul@debian7:~$ grep -w over text
The winter is over.
Can you get over there?
paul@debian7:~$
```

21.2.8. grep features

Sometimes it is easier to combine a simple regex with **grep** options, than it is to write a more complex regex. These options where discussed before:

```
grep -i
grep -v
grep -w
grep -A5
grep -B5
grep -C5
```

21.2.9. preventing shell expansion of a regex

The dollar sign is a special character, both for the regex and also for the shell (remember variables and embedded shells). Therefore it is advised to always quote the regex, this prevents shell expansion.

```
paul@debian7:~$ grep 'r$' names
Fleur
Floor
```

21.3. rename

21.3.1. the rename command

On Debian Linux the **/usr/bin/rename** command is a link to **/usr/bin/prename** installed by the **perl** package.

```
paul@pi ~ $ dpkg -S $(readlink -f $(which rename))
perl: /usr/bin/prename
```

Red Hat derived systems do not install the same **rename** command, so this section does not describe **rename** on Red Hat (unless you copy the perl script manually).

There is often confusion on the internet about the rename command because solutions that work fine in Debian (and Ubuntu, xubuntu, Mint, ...) cannot be used in Red Hat (and CentOS, Fedora, ...).

21.3.2. perl

The **rename** command is actually a perl script that uses **perl regular expressions**. The complete manual for these can be found by typing **perldoc perlrequick** (after installing **perldoc**).

```
root@pi:~# aptitude install perl-doc
The following NEW packages will be installed:
  perl-doc
0 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 8,170 kB of archives. After unpacking 13.2 MB will be used.
Get: 1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main perl-do...
Fetched 8,170 kB in 19s (412 kB/s)
Selecting previously unselected package perl-doc.
(Reading database ... 67121 files and directories currently installed.)
Unpacking perl-doc (from .../perl-doc_5.14.2-21+rpi2_all.deb) ...
Adding 'diversion of /usr/bin/perldoc to /usr/bin/perldoc.stub by perl-doc'
Processing triggers for man-db ...
Setting up perl-doc (5.14.2-21+rpi2) ...

root@pi:~# perldoc perlrequick
```

21.3.3. well known syntax

The most common use of the **rename** is to search for filenames matching a certain **string** and replacing this string with an **other string**.

This is often presented as **s/string/other string/** as seen in this example:

```
paul@pi ~ $ ls
abc       allfiles.TXT  bllfiles.TXT  Scratch   tennis2.TXT
abc.conf  backup        cllfiles.TXT  temp.TXT  tennis.TXT
paul@pi ~ $ rename 's/TXT/text/' *
paul@pi ~ $ ls
abc       allfiles.text  bllfiles.text  Scratch    tennis2.text
abc.conf  backup         cllfiles.text  temp.text  tennis.text
```

And here is another example that uses **rename** with the well know syntax to change the extensions of the same files once more:

```
paul@pi ~ $ ls
abc       allfiles.text  bllfiles.text  Scratch    tennis2.text
abc.conf  backup         cllfiles.text  temp.text  tennis.text
paul@pi ~ $ rename 's/text/txt/' *.text
paul@pi ~ $ ls
abc       allfiles.txt  bllfiles.txt  Scratch   tennis2.txt
abc.conf  backup        cllfiles.txt  temp.txt  tennis.txt
paul@pi ~ $
```

These two examples appear to work because the strings we used only exist at the end of the filename. Remember that file extensions have no meaning in the bash shell.

The next example shows what can go wrong with this syntax.

```
paul@pi ~ $ touch atxt.txt
paul@pi ~ $ rename 's/txt/problem/' atxt.txt
paul@pi ~ $ ls
abc       allfiles.txt  backup        cllfiles.txt  temp.txt     tennis.txt
abc.conf  aproblem.txt  bllfiles.txt  Scratch       tennis2.txt
paul@pi ~ $
```

Only the first occurrence of the searched string is replaced.

21.3.4. a global replace

The syntax used in the previous example can be described as **s/regex/replacement/**. This is simple and straightforward, you enter a **regex** between the first two slashes and a **replacement string** between the last two.

This example expands this syntax only a little, by adding a **modifier**.

```
paul@pi ~ $ rename -n 's/TXT/txt/g' aTXT.TXT
aTXT.TXT renamed as atxt.txt
paul@pi ~ $
```

The syntax we use now can be described as **s/regex/replacement/g** where s signifies **switch** and g stands for **global**.

Note that this example used the **-n** switch to show what is being done (instead of actually renaming the file).

21.3.5. case insensitive replace

Another **modifier** that can be useful is **i**. this example shows how to replace a case insensitive string with another string.

```
paul@debian7:~/files$ ls
file1.text  file2.TEXT  file3.txt
paul@debian7:~/files$ rename 's/.text/.txt/i' *
paul@debian7:~/files$ ls
file1.txt  file2.txt  file3.txt
paul@debian7:~/files$
```

21.3.6. renaming extensions

Command line Linux has no knowledge of MS-DOS like extensions, but many end users and graphical application do use them.

Here is an example on how to use **rename** to only rename the file extension. It uses the dollar sign to mark the ending of the filename.

```
paul@pi ~ $ ls *.txt
allfiles.txt  bllfiles.txt  cllfiles.txt  really.txt.txt  temp.txt  tennis.txt
paul@pi ~ $ rename 's/.txt$/.TXT/' *.txt
paul@pi ~ $ ls *.TXT
allfiles.TXT  bllfiles.TXT    cllfiles.TXT    really.txt.TXT
temp.TXT      tennis.TXT
paul@pi ~ $
```

Note that the **dollar sign** in the regex means **at the end**. Without the dollar sign this command would fail on the really.txt.txt file.

21.4. sed

21.4.1. stream editor

The **stream editor** or short **sed** uses **regex** for stream editing.

In this example **sed** is used to replace a string.

```
echo Sunday | sed 's/Sun/Mon/'
Monday
```

The slashes can be replaced by a couple of other characters, which can be handy in some cases to improve readability.

```
echo Sunday | sed 's:Sun:Mon:'
Monday
echo Sunday | sed 's_Sun_Mon_'
Monday
echo Sunday | sed 's|Sun|Mon|'
Monday
```

21.4.2. interactive editor

While **sed** is meant to be used in a stream, it can also be used interactively on a file.

```
paul@debian7:~/files$ echo Sunday > today
paul@debian7:~/files$ cat today
Sunday
paul@debian7:~/files$ sed -i 's/Sun/Mon/' today
paul@debian7:~/files$ cat today
Monday
```

21.4.3. simple back referencing

The **ampersand** character can be used to reference the searched (and found) string.

In this example the **ampersand** is used to double the occurence of the found string.

```
echo Sunday | sed 's/Sun/&&/'
SunSunday
echo Sunday | sed 's/day/&&/'
Sundayday
```

21.4.4. back referencing

Parentheses (often called round brackets) are used to group sections of the regex so they can leter be referenced.

Consider this simple example:

```
paul@debian7:~$ echo Sunday | sed 's_\(Sun\)_\1ny_'
Sunnyday
paul@debian7:~$ echo Sunday | sed 's_\(Sun\)_\1ny \1_'
Sunny Sunday
```

21.4.5. a dot for any character

In a **regex** a simple dot can signify any character.

```
paul@debian7:~$ echo 2014-04-01 | sed 's/....-..-../YYYY-MM-DD/'
YYYY-MM-DD
paul@debian7:~$ echo abcd-ef-gh | sed 's/....-..-../YYYY-MM-DD/'
YYYY-MM-DD
```

21.4.6. multiple back referencing

When more than one pair of **parentheses** is used, each of them can be referenced separately by consecutive numbers.

```
paul@debian7:~$ echo 2014-04-01 | sed 's/\(....\)-\(..\)-\(..\)/\1+\2+\3/'
2014+04+01
paul@debian7:~$ echo 2014-04-01 | sed 's/\(....\)-\(..\)-\(..\)/\3:\2:\1/'
01:04:2014
```

This feature is called **grouping**.

21.4.7. white space

The **\s** can refer to white space such as a space or a tab.

This example looks for white spaces (\s) globally and replaces them with 1 space.

```
paul@debian7:~$ echo -e 'today\tis\twarm'
today   is      warm
paul@debian7:~$ echo -e 'today\tis\twarm' | sed 's_\s_ _g'
today is warm
```

21.4.8. optional occurrence

A question mark signifies that the previous is **optional**.

The example below searches for three consecutive letter o, but the third o is optional.

```
paul@debian7:~$ cat list2
ll
lol
lool
loool
paul@debian7:~$ grep -E 'ooo?' list2
lool
loool
paul@debian7:~$ cat list2 | sed 's/ooo\?/A/'
ll
lol
lAl
lAl
```

21.4.9. exactly n times

You can demand an exact number of times the oprevious has to occur.

This example wants exactly three o's.

```
paul@debian7:~$ cat list2
ll
lol
lool
loool
paul@debian7:~$ grep -E 'o{3}' list2
loool
paul@debian7:~$ cat list2 | sed 's/o\{3\}/A/'
ll
lol
lool
lAl
paul@debian7:~$
```

21.4.10. between n and m times

And here we demand exactly from minimum 2 to maximum 3 times.

```
paul@debian7:~$ cat list2
ll
lol
lool
loool
paul@debian7:~$ grep -E 'o{2,3}' list2
lool
loool
paul@debian7:~$ grep 'o\{2,3\}' list2
lool
loool
paul@debian7:~$ cat list2 | sed 's/o\{2,3\}/A/'
ll
lol
lAl
lAl
paul@debian7:~$
```

21.5. bash history

The **bash shell** can also interprete some regular expressions.

This example shows how to manipulate the exclamation mask history feature of the bash shell.

```
paul@debian7:~$ mkdir hist
paul@debian7:~$ cd hist/
paul@debian7:~/hist$ touch file1 file2 file3
paul@debian7:~/hist$ ls -l file1
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file1
paul@debian7:~/hist$ !l
ls -l file1
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file1
paul@debian7:~/hist$ !l:s/1/3
ls -l file3
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file3
paul@debian7:~/hist$
```

This also works with the history numbers in bash.

```
paul@debian7:~/hist$ history 6
 2089  mkdir hist
 2090  cd hist/
 2091  touch file1 file2 file3
 2092  ls -l file1
 2093  ls -l file3
 2094  history 6
paul@debian7:~/hist$ !2092
ls -l file1
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file1
paul@debian7:~/hist$ !2092:s/1/2
ls -l file2
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file2
paul@debian7:~/hist$
```

Part VI. vi

Table of Contents

Chapter 22. Introduction to vi

The **vi** editor is installed on almost every Unix. Linux will very often install **vim** (**vi improved**) which is similar. Every system administrator should know **vi(m)**, because it is an easy tool to solve problems.

The **vi** editor is not intuitive, but once you get to know it, **vi** becomes a very powerful application. Most Linux distributions will include the **vimtutor** which is a 45 minute lesson in **vi(m)**.

22.1. command mode and insert mode

The vi editor starts in **command mode**. In command mode, you can type commands. Some commands will bring you into **insert mode**. In insert mode, you can type text. The **escape key** will return you to command mode.

Table 22.1. getting to command mode

key	action
Esc	set vi(m) in command mode.

22.2. start typing (a A i I o O)

The difference between a A i I o and O is the location where you can start typing. a will append after the current character and A will append at the end of the line. i will insert before the current character and I will insert at the beginning of the line. o will put you in a new line after the current line and O will put you in a new line before the current line.

Table 22.2. switch to insert mode

command	action
a	start typing after the current character
A	start typing at the end of the current line
i	start typing before the current character
I	start typing at the start of the current line
o	start typing on a new line after the current line
O	start typing on a new line before the current line

22.3. replace and delete a character (r x X)

When in command mode (it doesn't hurt to hit the escape key more than once) you can use the x key to delete the current character. The big X key (or shift x) will delete the character left of the cursor. Also when in command mode, you can use the r key to replace one single character. The r key will bring you in insert mode for just one key press, and will return you immediately to command mode.

Table 22.3. replace and delete

command	action
x	delete the character below the cursor
X	delete the character before the cursor
r	replace the character below the cursor
p	paste after the cursor (here the last deleted character)
xp	switch two characters

22.4. undo and repeat (u .)

When in command mode, you can undo your mistakes with u. You can do your mistakes twice with . (in other words, the . will repeat your last command).

Table 22.4. undo and repeat

command	action
u	undo the last action
.	repeat the last action

22.5. cut, copy and paste a line (dd yy p P)

When in command mode, dd will cut the current line. yy will copy the current line. You can paste the last copied or cut line after (p) or before (P) the current line.

Table 22.5. cut, copy and paste a line

command	action
dd	cut the current line
yy	(yank yank) copy the current line
p	paste after the current line
P	paste before the current line

22.6. cut, copy and paste lines (3dd 2yy)

When in command mode, before typing dd or yy, you can type a number to repeat the command a number of times. Thus, 5dd will cut 5 lines and 4yy will copy (yank) 4 lines. That last one will be noted by vi in the bottom left corner as "4 line yanked".

Table 22.6. cut, copy and paste lines

command	action
3dd	cut three lines
4yy	copy four lines

22.7. start and end of a line (0 or ^ and $)

When in command mode, the 0 and the caret ^ will bring you to the start of the current line, whereas the $ will put the cursor at the end of the current line. You can add 0 and $ to the d command, d0 will delete every character between the current character and the start of the line. Likewise d$ will delete everything from the current character till the end of the line. Similarly y0 and y$ will yank till start and end of the current line.

Table 22.7. start and end of line

command	action
0	jump to start of current line
^	jump to start of current line
$	jump to end of current line
d0	delete until start of line
d$	delete until end of line

22.8. join two lines (J) and more

When in command mode, pressing **J** will append the next line to the current line. With **yyp** you duplicate a line and with **ddp** you switch two lines.

Table 22.8. join two lines

command	action
J	join two lines
yyp	duplicate a line
ddp	switch two lines

22.9. words (w b)

When in command mode, **w** will jump to the next word and **b** will move to the previous word. w and b can also be combined with d and y to copy and cut words (dw db yw yb).

Table 22.9. words

command	action
w	forward one word
b	back one word
3w	forward three words
dw	delete one word
yw	yank (copy) one word
5yb	yank five words back
7dw	delete seven words

22.10. save (or not) and exit (:w :q :q!)

Pressing the colon : will allow you to give instructions to vi (technically speaking, typing the colon will open the **ex** editor). **:w** will write (save) the file, **:q** will quit an unchanged file without saving, and **:q!** will quit vi discarding any changes. **:wq** will save and quit and is the same as typing **ZZ** in command mode.

Table 22.10. save and exit vi

command	action
:w	save (write)
:w fname	save as fname
:q	quit
:wq	save and quit
ZZ	save and quit
:q!	quit (discarding your changes)
:w!	save (and write to non-writable file!)

The last one is a bit special. With **:w! vi** will try to **chmod** the file to get write permission (this works when you are the owner) and will **chmod** it back when the write succeeds. This should always work when you are root (and the file system is writable).

22.11. Searching (/ ?)

When in command mode typing / will allow you to search in vi for strings (can be a regular expression). Typing /foo will do a forward search for the string foo and typing ?bar will do a backward search for bar.

Table 22.11. searching

command	action
/string	forward search for string

command	action
?string	backward search for string
n	go to next occurrence of search string
/^string	forward search string at beginning of line
/string$	forward search string at end of line
/br[aeio]l	search for bral brel bril and brol
/\<he\>	search for the word **he** (and not for **he**re or t**he**)

22.12. replace all (:1,$ s/foo/bar/g)

To replace all occurrences of the string foo with bar, first switch to ex mode with : . Then tell vi which lines to use, for example 1,$ will do the replace all from the first to the last line. You can write 1,5 to only process the first five lines. The s/foo/bar/g will replace all occurrences of foo with bar.

Table 22.12. replace

command	action
:4,8 s/foo/bar/g	replace foo with bar on lines 4 to 8
:1,$ s/foo/bar/g	replace foo with bar on all lines

22.13. reading files (:r :r !cmd)

When in command mode, :r foo will read the file named foo, :r !foo will execute the command foo. The result will be put at the current location. Thus :r !ls will put a listing of the current directory in your text file.

Table 22.13. read files and input

command	action
:r fname	(read) file fname and paste contents
:r !cmd	execute cmd and paste its output

22.14. text buffers

There are 36 buffers in vi to store text. You can use them with the " character.

Table 22.14. text buffers

command	action
"add	delete current line and put text in buffer a
"g7yy	copy seven lines into buffer g
"ap	paste from buffer a

22.15. multiple files

You can edit multiple files with vi. Here are some tips.

Table 22.15. multiple files

command	action
vi file1 file2 file3	start editing three files
:args	lists files and marks active file
:n	start editing the next file
:e	toggle with last edited file
:rew	rewind file pointer to first file

22.16. abbreviations

With **:ab** you can put abbreviations in vi. Use **:una** to undo the abbreviation.

Table 22.16. abbreviations

command	action
:ab str long string	abbreviate **str** to be 'long string'
:una str	un-abbreviate str

22.17. key mappings

Similarly to their abbreviations, you can use mappings with **:map** for command mode and **:map!** for insert mode.

This example shows how to set the F6 function key to toggle between **set number** and **set nonumber**. The <bar> separates the two commands, **set number!** toggles the state and **set number?** reports the current state.

```
:map <F6> :set number!<bar>set number?<CR>
```

22.18. setting options

Some options that you can set in vim.

```
:set number  ( also try :se nu )
:set nonumber
:syntax on
:syntax off
:set all  (list all options)
:set tabstop=8
:set tx   (CR/LF style endings)
:set notx
```

You can set these options (and much more) in **~/.vimrc** for vim or in **~/.exrc** for standard vi.

```
paul@barry:~$ cat ~/.vimrc
set number
set tabstop=8
set textwidth=78
map <F6> :set number!<bar>set number?<CR>
paul@barry:~$
```

22.19. practice: vi(m)

1. Start the vimtutor and do some or all of the exercises. You might need to run **aptitude install vim** on xubuntu.

2. What 3 key sequence in command mode will duplicate the current line.

3. What 3 key sequence in command mode will switch two lines' place (line five becomes line six and line six becomes line five).

4. What 2 key sequence in command mode will switch a character's place with the next one.

5. vi can understand macro's. A macro can be recorded with q followed by the name of the macro. So qa will record the macro named a. Pressing q again will end the recording. You can recall the macro with @ followed by the name of the macro. Try this example: i 1 'Escape Key' qa yyp 'Ctrl a' q 5@a (Ctrl a will increase the number with one).

6. Copy /etc/passwd to your ~/passwd. Open the last one in vi and press Ctrl v. Use the arrow keys to select a Visual Block, you can copy this with y or delete it with d. Try pasting it.

7. What does dwwP do when you are at the beginning of a word in a sentence ?

22.20. solution: vi(m)

1. Start the vimtutor and do some or all of the exercises. You might need to run **aptitude install vim** on xubuntu.

```
vimtutor
```

2. What 3 key sequence in command mode will duplicate the current line.

```
yyp
```

3. What 3 key sequence in command mode will switch two lines' place (line five becomes line six and line six becomes line five).

```
ddp
```

4. What 2 key sequence in command mode will switch a character's place with the next one.

```
xp
```

5. vi can understand macro's. A macro can be recorded with q followed by the name of the macro. So qa will record the macro named a. Pressing q again will end the recording. You can recall the macro with @ followed by the name of the macro. Try this example: i 1 'Escape Key' qa yyp 'Ctrl a' q 5@a (Ctrl a will increase the number with one).

6. Copy /etc/passwd to your ~/passwd. Open the last one in vi and press Ctrl v. Use the arrow keys to select a Visual Block, you can copy this with y or delete it with d. Try pasting it.

```
cp /etc/passwd ~
vi passwd
(press Ctrl-V)
```

7. What does **dwwP** do when you are at the beginning of a word in a sentence ?

dwwP can switch the current word with the next word.

Part VII. scripting

Table of Contents

Chapter 23. scripting introduction

Shells like **bash** and **Korn** have support for programming constructs that can be saved as **scripts**. These **scripts** in turn then become more **shell** commands. Many Linux commands are **scripts**. **User profile scripts** are run when a user logs on and **init scripts** are run when a **daemon** is stopped or started.

This means that system administrators also need basic knowledge of **scripting** to understand how their servers and their applications are started, updated, upgraded, patched, maintained, configured and removed, and also to understand how a user environment is built.

The goal of this chapter is to give you enough information to be able to read and understand scripts. Not to become a writer of complex scripts.

23.1. prerequisites

You should have read and understood **part III shell expansion** and **part IV pipes and commands** before starting this chapter.

23.2. hello world

Just like in every programming course, we start with a simple **hello_world** script. The following script will output **Hello World**.

```
echo Hello World
```

After creating this simple script in **vi** or with **echo**, you'll have to **chmod +x hello_world** to make it executable. And unless you add the scripts directory to your path, you'll have to type the path to the script for the shell to be able to find it.

```
[paul@RHEL4a ~]$ echo echo Hello World > hello_world
[paul@RHEL4a ~]$ chmod +x hello_world
[paul@RHEL4a ~]$ ./hello_world
Hello World
[paul@RHEL4a ~]$
```

23.3. she-bang

Let's expand our example a little further by putting **#!/bin/bash** on the first line of the script. The **#!** is called a **she-bang** (sometimes called **sha-bang**), where the **she-bang** is the first two characters of the script.

```
#!/bin/bash
echo Hello World
```

You can never be sure which shell a user is running. A script that works flawlessly in **bash** might not work in **ksh**, **csh**, or **dash**. To instruct a shell to run your script in a certain shell, you can start your script with a **she-bang** followed by the shell it is supposed to run in. This script will run in a bash shell.

```
#!/bin/bash
echo -n hello
echo A bash subshell `echo -n hello`
```

This script will run in a Korn shell (unless **/bin/ksh** is a hard link to **/bin/bash**). The **/etc/shells** file contains a list of shells on your system.

```
#!/bin/ksh
echo -n hello
echo a Korn subshell `echo -n hello`
```

23.4. comment

Let's expand our example a little further by adding comment lines.

```
#!/bin/bash
#
# Hello World Script
#
echo Hello World
```

23.5. variables

Here is a simple example of a variable inside a script.

```
#!/bin/bash
#
# simple variable in script
#
var1=4
echo var1 = $var1
```

Scripts can contain variables, but since scripts are run in their own shell, the variables do not survive the end of the script.

```
[paul@RHEL4a ~]$ echo $var1

[paul@RHEL4a ~]$ ./vars
var1 = 4
[paul@RHEL4a ~]$ echo $var1

[paul@RHEL4a ~]$
```

23.6. sourcing a script

Luckily, you can force a script to run in the same shell; this is called **sourcing** a script.

```
[paul@RHEL4a ~]$ source ./vars
var1 = 4
[paul@RHEL4a ~]$ echo $var1
4
[paul@RHEL4a ~]$
```

The above is identical to the below.

```
[paul@RHEL4a ~]$ . ./vars
var1 = 4
[paul@RHEL4a ~]$ echo $var1
4
[paul@RHEL4a ~]$
```

23.7. troubleshooting a script

Another way to run a script in a separate shell is by typing **bash** with the name of the script as a parameter.

```
paul@debian6~/test$ bash runme
42
```

Expanding this to **bash -x** allows you to see the commands that the shell is executing (after shell expansion).

```
paul@debian6~/test$ bash -x runme
+ var4=42
+ echo 42
42
paul@debian6~/test$ cat runme
# the runme script
var4=42
echo $var4
paul@debian6~/test$
```

Notice the absence of the commented (#) line, and the replacement of the variable before execution of **echo**.

23.8. prevent setuid root spoofing

Some user may try to perform **setuid** based script **root spoofing**. This is a rare but possible attack. To improve script security and to avoid interpreter spoofing, you need to add **--** after the **#!/bin/bash**, which disables further option processing so the shell will not accept any options.

```
#!/bin/bash -
or
#!/bin/bash --
```

Any arguments after the **--** are treated as filenames and arguments. An argument of **-** is equivalent to **--**.

23.9. practice: introduction to scripting

0. Give each script a different name, keep them for later!

1. Write a script that outputs the name of a city.

2. Make sure the script runs in the bash shell.

3. Make sure the script runs in the Korn shell.

4. Create a script that defines two variables, and outputs their value.

5. The previous script does not influence your current shell (the variables do not exist outside of the script). Now run the script so that it influences your current shell.

6. Is there a shorter way to **source** the script ?

7. Comment your scripts so that you know what they are doing.

23.10. solution: introduction to scripting

0. Give each script a different name, keep them for later!

1. Write a script that outputs the name of a city.

```
$ echo 'echo Antwerp' > first.bash
$ chmod +x first.bash
$ ./first.bash
Antwerp
```

2. Make sure the script runs in the bash shell.

```
$ cat first.bash
#!/bin/bash
echo Antwerp
```

3. Make sure the script runs in the Korn shell.

```
$ cat first.bash
#!/bin/ksh
echo Antwerp
```

Note that while first.bash will technically work as a Korn shell script, the name ending in .bash is confusing.

4. Create a script that defines two variables, and outputs their value.

```
$ cat second.bash
#!/bin/bash

var33=300
var42=400

echo $var33 $var42
```

5. The previous script does not influence your current shell (the variables do not exist outside of the script). Now run the script so that it influences your current shell.

```
source second.bash
```

6. Is there a shorter way to **source** the script ?

```
. ./second.bash
```

7. Comment your scripts so that you know what they are doing.

```
$ cat second.bash
#!/bin/bash
# script to test variables and sourcing

# define two variables
var33=300
var42=400

# output the value of these variables
echo $var33 $var42
```

Chapter 24. scripting loops

24.1. test [ ]

The **test** command can test whether something is true or false. Let's start by testing whether 10 is greater than 55.

```
[paul@RHEL4b ~]$ test 10 -gt 55 ; echo $?
1
[paul@RHEL4b ~]$
```

The test command returns 1 if the test fails. And as you see in the next screenshot, test returns 0 when a test succeeds.

```
[paul@RHEL4b ~]$ test 56 -gt 55 ; echo $?
0
[paul@RHEL4b ~]$
```

If you prefer true and false, then write the test like this.

```
[paul@RHEL4b ~]$ test 56 -gt 55 && echo true || echo false
true
[paul@RHEL4b ~]$ test 6 -gt 55 && echo true || echo false
false
```

The test command can also be written as square brackets, the screenshot below is identical to the one above.

```
[paul@RHEL4b ~]$ [ 56 -gt 55 ] && echo true || echo false
true
[paul@RHEL4b ~]$ [ 6 -gt 55 ] && echo true || echo false
false
```

Below are some example tests. Take a look at **man test** to see more options for tests.

```
[ -d foo ]              Does the directory foo exist ?
[ -e bar ]              Does the file bar exist ?
[ '/etc' = $PWD ]       Is the string /etc equal to the variable $PWD ?
[ $1 != 'secret' ]      Is the first parameter different from secret ?
[ 55 -lt $bar ]         Is 55 less than the value of $bar ?
[ $foo -ge 1000 ]       Is the value of $foo greater or equal to 1000 ?
[ "abc" < $bar ]        Does abc sort before the value of $bar ?
[ -f foo ]              Is foo a regular file ?
[ -r bar ]              Is bar a readable file ?
[ foo -nt bar ]         Is file foo newer than file bar ?
[ -o nounset ]          Is the shell option nounset set ?
```

Tests can be combined with logical AND and OR.

```
paul@RHEL4b:~$ [ 66 -gt 55 -a 66 -lt 500 ] && echo true || echo false
true
paul@RHEL4b:~$ [ 66 -gt 55 -a 660 -lt 500 ] && echo true || echo false
false
paul@RHEL4b:~$ [ 66 -gt 55 -o 660 -lt 500 ] && echo true || echo false
true
```

24.2. if then else

The **if then else** construction is about choice. If a certain condition is met, then execute something, else execute something else. The example below tests whether a file exists, and if the file exists then a proper message is echoed.

```
#!/bin/bash

if [ -f isit.txt ]
then echo isit.txt exists!
else echo isit.txt not found!
fi
```

If we name the above script 'choice', then it executes like this.

```
[paul@RHEL4a scripts]$ ./choice
isit.txt not found!
[paul@RHEL4a scripts]$ touch isit.txt
[paul@RHEL4a scripts]$ ./choice
isit.txt exists!
[paul@RHEL4a scripts]$
```

24.3. if then elif

You can nest a new **if** inside an **else** with **elif**. This is a simple example.

```
#!/bin/bash
count=42
if [ $count -eq 42 ]
then
  echo "42 is correct."
elif [ $count -gt 42 ]
then
  echo "Too much."
else
  echo "Not enough."
fi
```

24.4. for loop

The example below shows the syntax of a classical **for loop** in bash.

```
for i in 1 2 4
do
   echo $i
done
```

An example of a **for loop** combined with an embedded shell.

```
#!/bin/ksh
for counter in `seq 1 20`
do
   echo counting from 1 to 20, now at $counter
   sleep 1
done
```

The same example as above can be written without the embedded shell using the bash **{from..to}** shorthand.

```
#!/bin/bash
for counter in {1..20}
do
   echo counting from 1 to 20, now at $counter
   sleep 1
done
```

This **for loop** uses file globbing (from the shell expansion). Putting the instruction on the command line has identical functionality.

```
kahlan@solexp11$ ls
count.ksh  go.ksh
kahlan@solexp11$ for file in *.ksh ; do cp $file $file.backup ; done
kahlan@solexp11$ ls
count.ksh  count.ksh.backup  go.ksh  go.ksh.backup
```

24.5. while loop

Below a simple example of a **while loop**.

```
i=100;
while [ $i -ge 0 ] ;
do
   echo Counting down, from 100 to 0, now at $i;
   let i--;
done
```

Endless loops can be made with **while true** or **while :** , where the **colon** is the equivalent of **no operation** in the **Korn** and **bash** shells.

```
#!/bin/ksh
# endless loop
while :
do
 echo hello
 sleep 1
done
```

24.6. until loop

Below a simple example of an **until loop**.

```
let i=100;
until [ $i -le 0 ] ;
do
   echo Counting down, from 100 to 1, now at $i;
   let i--;
done
```

24.7. practice: scripting tests and loops

1. Write a script that uses a **for** loop to count from 3 to 7.

2. Write a script that uses a **for** loop to count from 1 to 17000.

3. Write a script that uses a **while** loop to count from 3 to 7.

4. Write a script that uses an **until** loop to count down from 8 to 4.

5. Write a script that counts the number of files ending in **.txt** in the current directory.

6. Wrap an **if** statement around the script so it is also correct when there are zero files ending in **.txt**.

24.8. solution: scripting tests and loops

1. Write a script that uses a **for** loop to count from 3 to 7.

```
#!/bin/bash

for i in 3 4 5 6 7
do
 echo Counting from 3 to 7, now at $i
done
```

2. Write a script that uses a **for** loop to count from 1 to 17000.

```
#!/bin/bash

for i in `seq 1 17000`
do
 echo Counting from 1 to 17000, now at $i
done
```

3. Write a script that uses a **while** loop to count from 3 to 7.

```
#!/bin/bash

i=3
while [ $i -le 7 ]
do
 echo Counting from 3 to 7, now at $i
 let i=i+1
done
```

4. Write a script that uses an **until** loop to count down from 8 to 4.

```
#!/bin/bash

i=8
until [ $i -lt 4 ]
do
 echo Counting down from 8 to 4, now at $i
 let i=i-1
done
```

5. Write a script that counts the number of files ending in **.txt** in the current directory.

```
#!/bin/bash

let i=0
for file in *.txt
do
 let i++
done
echo "There are $i files ending in .txt"
```

6. Wrap an **if** statement around the script so it is also correct when there are zero files ending in **.txt**.

```
#!/bin/bash

ls *.txt > /dev/null 2>&1
if [ $? -ne 0 ]
```

```
then echo "There are 0 files ending in .txt"
else
 let i=0
 for file in *.txt
 do
  let i++
 done
 echo "There are $i files ending in .txt"
fi
```

Chapter 25. scripting parameters

25.1. script parameters

A **bash** shell script can have parameters. The numbering you see in the script below continues if you have more parameters. You also have special parameters containing the number of parameters, a string of all of them, and also the process id, and the last return code. The man page of **bash** has a full list.

```
#!/bin/bash
echo The first argument is $1
echo The second argument is $2
echo The third argument is $3

echo \$ $$  PID of the script
echo \# $#  count arguments
echo \? $?  last return code
echo \* $*  all the arguments
```

Below is the output of the script above in action.

```
[paul@RHEL4a scripts]$ ./pars one two three
The first argument is one
The second argument is two
The third argument is three
$ 5610 PID of the script
# 3 count arguments
? 0 last return code
* one two three all the arguments
```

Once more the same script, but with only two parameters.

```
[paul@RHEL4a scripts]$ ./pars 1 2
The first argument is 1
The second argument is 2
The third argument is
$ 5612 PID of the script
# 2 count arguments
? 0 last return code
* 1 2 all the arguments
[paul@RHEL4a scripts]$
```

Here is another example, where we use **$0**. The **$0** parameter contains the name of the script.

```
paul@debian6~$ cat myname
echo this script is called $0
paul@debian6~$ ./myname
this script is called ./myname
paul@debian6~$ mv myname test42
paul@debian6~$ ./test42
this script is called ./test42
```

25.2. shift through parameters

The **shift** statement can parse all **parameters** one by one. This is a sample script.

```
kahlan@solexp11$ cat shift.ksh
#!/bin/ksh

if [ "$#" == "0" ]
 then
  echo You have to give at least one parameter.
  exit 1
fi

while (( $# ))
 do
  echo You gave me $1
  shift
 done
```

Below is some sample output of the script above.

```
kahlan@solexp11$ ./shift.ksh one
You gave me one
kahlan@solexp11$ ./shift.ksh one two three 1201 "33 42"
You gave me one
You gave me two
You gave me three
You gave me 1201
You gave me 33 42
kahlan@solexp11$ ./shift.ksh
You have to give at least one parameter.
```

25.3. runtime input

You can ask the user for input with the **read** command in a script.

```
#!/bin/bash
echo -n Enter a number:
read number
```

25.4. sourcing a config file

The **source** (as seen in the shell chapters) can be used to source a configuration file.

Below a sample configuration file for an application.

```
[paul@RHEL4a scripts]$ cat myApp.conf
# The config file of myApp

# Enter the path here
myAppPath=/var/myApp

# Enter the number of quines here
quines=5
```

And here an application that uses this file.

```
[paul@RHEL4a scripts]$ cat myApp.bash
#!/bin/bash
#
# Welcome to the myApp application
#

. ./myApp.conf

echo There are $quines quines
```

The running application can use the values inside the sourced configuration file.

```
[paul@RHEL4a scripts]$ ./myApp.bash
There are 5 quines
[paul@RHEL4a scripts]$
```

25.5. get script options with getopts

The **getopts** function allows you to parse options given to a command. The following script allows for any combination of the options a, f and z.

```
kahlan@solexp11$ cat options.ksh
#!/bin/ksh

while getopts ":afz" option;
do
 case $option in
  a)
   echo received -a
   ;;
  f)
   echo received -f
   ;;
  z)
   echo received -z
   ;;
  *)
   echo "invalid option -$OPTARG"
   ;;
 esac
done
```

This is sample output from the script above. First we use correct options, then we enter twice an invalid option.

```
kahlan@solexp11$ ./options.ksh
kahlan@solexp11$ ./options.ksh -af
received -a
received -f
kahlan@solexp11$ ./options.ksh -zfg
received -z
received -f
invalid option -g
kahlan@solexp11$ ./options.ksh -a -b -z
received -a
invalid option -b
received -z
```

You can also check for options that need an argument, as this example shows.

```
kahlan@solexp11$ cat argoptions.ksh
#!/bin/ksh

while getopts ":af:z" option;
do
 case $option in
  a)
   echo received -a
   ;;
  f)
   echo received -f with $OPTARG
   ;;
  z)
   echo received -z
   ;;
  :)
   echo "option -$OPTARG needs an argument"
   ;;
  *)
   echo "invalid option -$OPTARG"
   ;;
 esac
done
```

This is sample output from the script above.

```
kahlan@solexp11$ ./argoptions.ksh -a -f hello -z
received -a
received -f with hello
received -z
kahlan@solexp11$ ./argoptions.ksh -zaf 42
received -z
received -a
received -f with 42
kahlan@solexp11$ ./argoptions.ksh -zf
received -z
option -f needs an argument
```

25.6. get shell options with shopt

You can toggle the values of variables controlling optional shell behaviour with the **shopt** built-in shell command. The example below first verifies whether the cdspell option is set; it is not. The next shopt command sets the value, and the third shopt command verifies that the option really is set. You can now use minor spelling mistakes in the cd command. The man page of bash has a complete list of options.

```
paul@laika:~$ shopt -q cdspell ; echo $?
1
paul@laika:~$ shopt -s cdspell
paul@laika:~$ shopt -q cdspell ; echo $?
0
paul@laika:~$ cd /Etc
/etc
```

25.7. practice: parameters and options

1. Write a script that receives four parameters, and outputs them in reverse order.

2. Write a script that receives two parameters (two filenames) and outputs whether those files exist.

3. Write a script that asks for a filename. Verify existence of the file, then verify that you own the file, and whether it is writable. If not, then make it writable.

4. Make a configuration file for the previous script. Put a logging switch in the config file, logging means writing detailed output of everything the script does to a log file in /tmp.

25.8. solution: parameters and options

1. Write a script that receives four parameters, and outputs them in reverse order.

```
echo $4 $3 $2 $1
```

2. Write a script that receives two parameters (two filenames) and outputs whether those files exist.

```
#!/bin/bash

if [ -f $1 ]
then echo $1 exists!
else echo $1 not found!
fi

if [ -f $2 ]
then echo $2 exists!
else echo $2 not found!
fi
```

3. Write a script that asks for a filename. Verify existence of the file, then verify that you own the file, and whether it is writable. If not, then make it writable.

4. Make a configuration file for the previous script. Put a logging switch in the config file, logging means writing detailed output of everything the script does to a log file in /tmp.

Chapter 26. more scripting

26.1. eval

eval reads arguments as input to the shell (the resulting commands are executed). This allows using the value of a variable as a variable.

```
paul@deb503:~/test42$ answer=42
paul@deb503:~/test42$ word=answer
paul@deb503:~/test42$ eval x=\$$word ; echo $x
42
```

Both in **bash** and **Korn** the arguments can be quoted.

```
kahlan@solexp11$ answer=42
kahlan@solexp11$ word=answer
kahlan@solexp11$ eval "y=\$$word" ; echo $y
42
```

Sometimes the **eval** is needed to have correct parsing of arguments. Consider this example where the **date** command receives one parameter **1 week ago**.

```
paul@debian6~$ date --date="1 week ago"
Thu Mar  8 21:36:25 CET 2012
```

When we set this command in a variable, then executing that variable fails unless we use **eval**.

```
paul@debian6~$ lastweek='date --date="1 week ago"'
paul@debian6~$ $lastweek
date: extra operand `ago"'
Try `date --help' for more information.
paul@debian6~$ eval $lastweek
Thu Mar  8 21:36:39 CET 2012
```

26.2. (( ))

The **(( ))** allows for evaluation of numerical expressions.

```
paul@deb503:~/test42$ (( 42 > 33 )) && echo true || echo false
true
paul@deb503:~/test42$ (( 42 > 1201 )) && echo true || echo false
false
paul@deb503:~/test42$ var42=42
paul@deb503:~/test42$ (( 42 == var42 )) && echo true || echo false
true
paul@deb503:~/test42$ (( 42 == $var42 )) && echo true || echo false
true
paul@deb503:~/test42$ var42=33
paul@deb503:~/test42$ (( 42 == var42 )) && echo true || echo false
false
```

26.3. let

The **let** built-in shell function instructs the shell to perform an evaluation of arithmetic expressions. It will return 0 unless the last arithmetic expression evaluates to 0.

```
[paul@RHEL4b ~]$ let x="3 + 4" ; echo $x
7
[paul@RHEL4b ~]$ let x="10 + 100/10" ; echo $x
20
[paul@RHEL4b ~]$ let x="10-2+100/10" ; echo $x
18
[paul@RHEL4b ~]$ let x="10*2+100/10" ; echo $x
30
```

The **shell** can also convert between different bases.

```
[paul@RHEL4b ~]$ let x="0xFF" ; echo $x
255
[paul@RHEL4b ~]$ let x="0xC0" ; echo $x
192
[paul@RHEL4b ~]$ let x="0xA8" ; echo $x
168
[paul@RHEL4b ~]$ let x="8#70" ; echo $x
56
[paul@RHEL4b ~]$ let x="8#77" ; echo $x
63
[paul@RHEL4b ~]$ let x="16#c0" ; echo $x
192
```

There is a difference between assigning a variable directly, or using **let** to evaluate the arithmetic expressions (even if it is just assigning a value).

```
kahlan@solexp11$ dec=15 ; oct=017 ; hex=0x0f
kahlan@solexp11$ echo $dec $oct $hex
15 017 0x0f
kahlan@solexp11$ let dec=15 ; let oct=017 ; let hex=0x0f
kahlan@solexp11$ echo $dec $oct $hex
15 15 15
```

26.4. case

You can sometimes simplify nested if statements with a **case** construct.

```
[paul@RHEL4b ~]$ ./help
What animal did you see ? lion
You better start running fast!
[paul@RHEL4b ~]$ ./help
What animal did you see ? dog
Don't worry, give it a cookie.
[paul@RHEL4b ~]$ cat help
#!/bin/bash
#
# Wild Animals Helpdesk Advice
#
echo -n "What animal did you see ? "
read animal
case $animal in
        "lion" | "tiger")
                echo "You better start running fast!"
        ;;
        "cat")
                echo "Let that mouse go..."
        ;;
        "dog")
                echo "Don't worry, give it a cookie."
        ;;
        "chicken" | "goose" | "duck" )
                echo "Eggs for breakfast!"
        ;;
        "liger")
                echo "Approach and say 'Ah you big fluffy kitty...'."
        ;;
        "babelfish")
                echo "Did it fall out your ear ?"
        ;;
        *)
                echo "You discovered an unknown animal, name it!"
        ;;
esac
[paul@RHEL4b ~]$
```

26.5. shell functions

Shell **functions** can be used to group commands in a logical way.

```
kahlan@solexp11$ cat funcs.ksh
#!/bin/ksh

function greetings {
echo Hello World!
echo and hello to $USER to!
}

echo We will now call a function
greetings
echo The end
```

This is sample output from this script with a **function**.

```
kahlan@solexp11$ ./funcs.ksh
We will now call a function
Hello World!
and hello to kahlan to!
The end
```

A shell function can also receive parameters.

```
kahlan@solexp11$ cat addfunc.ksh
#!/bin/ksh

function plus {
let result="$1 + $2"
echo  $1 + $2 = $result
}

plus 3 10
plus 20 13
plus 20 22
```

This script produces the following output.

```
kahlan@solexp11$ ./addfunc.ksh
3 + 10 = 13
20 + 13 = 33
20 + 22 = 42
```

26.6. practice : more scripting

1. Write a script that asks for two numbers, and outputs the sum and product (as shown here).

```
Enter a number: 5
Enter another number: 2

Sum:       5 + 2 = 7
Product:   5 x 2 = 10
```

2. Improve the previous script to test that the numbers are between 1 and 100, exit with an error if necessary.

3. Improve the previous script to congratulate the user if the sum equals the product.

4. Write a script with a case insensitive case statement, using the shopt nocasematch option. The nocasematch option is reset to the value it had before the scripts started.

5. If time permits (or if you are waiting for other students to finish this practice), take a look at Linux system scripts in /etc/init.d and /etc/rc.d and try to understand them. Where does execution of a script start in /etc/init.d/samba ? There are also some hidden scripts in ~, we will discuss them later.

26.7. solution : more scripting

1. Write a script that asks for two numbers, and outputs the sum and product (as shown here).

```
Enter a number: 5
Enter another number: 2

Sum:       5 + 2 = 7
Product:   5 x 2 = 10
```

```
#!/bin/bash

echo -n "Enter a number : "
read n1

echo -n "Enter another number : "
read n2

let sum="$n1+$n2"
let pro="$n1*$n2"

echo -e "Sum\t: $n1 + $n2 = $sum"
echo -e "Product\t: $n1 * $n2 = $pro"
```

2. Improve the previous script to test that the numbers are between 1 and 100, exit with an error if necessary.

```
echo -n "Enter a number between 1 and 100 : "
read n1

if [ $n1 -lt 1 -o $n1 -gt 100 ]
then
       echo Wrong number...
       exit 1
fi
```

3. Improve the previous script to congratulate the user if the sum equals the product.

```
if [ $sum -eq $pro ]
then echo Congratulations $sum == $pro
fi
```

4. Write a script with a case insensitive case statement, using the shopt nocasematch option. The nocasematch option is reset to the value it had before the scripts started.

```
#!/bin/bash
#
# Wild Animals Case Insensitive Helpdesk Advice
#

if shopt -q nocasematch; then
  nocase=yes;
else
  nocase=no;
  shopt -s nocasematch;
fi

echo -n "What animal did you see ? "
read animal

case $animal in
```

```
  "lion" | "tiger")
    echo "You better start running fast!"
  ;;
  "cat")
    echo "Let that mouse go..."
  ;;
  "dog")
    echo "Don't worry, give it a cookie."
  ;;
  "chicken" | "goose" | "duck" )
    echo "Eggs for breakfast!"
  ;;
  "liger")
    echo "Approach and say 'Ah you big fluffy kitty.'"
  ;;
  "babelfish")
    echo "Did it fall out your ear ?"
  ;;
  *)
    echo "You discovered an unknown animal, name it!"
  ;;
esac

if [ nocase = yes ] ; then
        shopt -s nocasematch;
else
        shopt -u nocasematch;
fi
```

5. If time permits (or if you are waiting for other students to finish this practice), take a look at Linux system scripts in /etc/init.d and /etc/rc.d and try to understand them. Where does execution of a script start in /etc/init.d/samba ? There are also some hidden scripts in ~, we will discuss them later.

Part VIII. local user management

Table of Contents

Chapter 27. introduction to users

This little chapter will teach you how to identify your user account on a Unix computer using commands like **who am i**, **id**, and more.

In a second part you will learn how to become another user with the **su** command.

And you will learn how to run a program as another user with **sudo**.

27.1. whoami

The **whoami** command tells you your username.

```
[paul@centos7 ~]$ whoami
paul
[paul@centos7 ~]$
```

27.2. who

The **who** command will give you information about who is logged on the system.

```
[paul@centos7 ~]$ who
root     pts/0        2014-10-10 23:07 (10.104.33.101)
paul     pts/1        2014-10-10 23:30 (10.104.33.101)
laura    pts/2        2014-10-10 23:34 (10.104.33.96)
tania    pts/3        2014-10-10 23:39 (10.104.33.91)
[paul@centos7 ~]$
```

27.3. who am i

With **who am i** the **who** command will display only the line pointing to your current session.

```
[paul@centos7 ~]$ who am i
paul     pts/1        2014-10-10 23:30 (10.104.33.101)
[paul@centos7 ~]$
```

27.4. w

The **w** command shows you who is logged on and what they are doing.

```
[paul@centos7 ~]$ w
 23:34:07 up 31 min,  2 users,  load average: 0.00, 0.01, 0.02
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0     23:07   15.00s  0.01s  0.01s top
paul     pts/1     23:30    7.00s  0.00s  0.00s w
[paul@centos7 ~]$
```

27.5. id

The **id** command will give you your user id, primary group id, and a list of the groups that you belong to.

```
paul@debian7:~$ id
uid=1000(paul) gid=1000(paul) groups=1000(paul)
```

On RHEL/CentOS you will also get **SELinux** context information with this command.

```
[root@centos7 ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r\
:unconfined_t:s0-s0:c0.c1023
```

27.6. su to another user

The **su** command allows a user to run a shell as another user.

```
laura@debian7:~$ su tania
Password:
tania@debian7:/home/laura$
```

27.7. su to root

Yes you can also **su** to become **root**, when you know the **root password**.

```
laura@debian7:~$ su root
Password:
root@debian7:/home/laura#
```

27.8. su as root

You need to know the password of the user you want to substitute to, unless your are logged in as **root**. The **root** user can become any existing user without knowing that user's password.

```
root@debian7:~# id
uid=0(root) gid=0(root) groups=0(root)
root@debian7:~# su - valentina
valentina@debian7:~$
```

27.9. su - $username

By default, the **su** command maintains the same shell environment. To become another user and also get the target user's environment, issue the **su -** command followed by the target username.

```
root@debian7:~# su laura
laura@debian7:/root$ exit
exit
root@debian7:~# su - laura
laura@debian7:~$ pwd
/home/laura
```

27.10. su -

When no username is provided to **su** or **su -**, the command will assume **root** is the target.

```
tania@debian7:~$ su -
Password:
root@debian7:~#
```

27.11. run a program as another user

The sudo program allows a user to start a program with the credentials of another user. Before this works, the system administrator has to set up the **/etc/sudoers** file. This can be useful to delegate administrative tasks to another user (without giving the root password).

The screenshot below shows the usage of **sudo**. User **paul** received the right to run **useradd** with the credentials of **root**. This allows **paul** to create new users on the system without becoming **root** and without knowing the **root password**.

First the command fails for **paul**.

```
paul@debian7:~$ /usr/sbin/useradd -m valentina
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
```

But with **sudo** it works.

```
paul@debian7:~$ sudo /usr/sbin/useradd -m valentina
[sudo] password for paul:
paul@debian7:~$
```

27.12. visudo

Check the man page of **visudo** before playing with the **/etc/sudoers** file. Editing the **sudoers** is out of scope for this fundamentals book.

```
paul@rhel65:~$ apropos visudo
visudo               (8)  - edit the sudoers file
paul@rhel65:~$
```

27.13. sudo su -

On some Linux systems like Ubuntu and Xubuntu, the **root** user does not have a password set. This means that it is not possible to login as **root** (extra security). To perform tasks as **root**, the first user is given all **sudo rights** via the **/etc/sudoers**. In fact all users that are members of the admin group can use sudo to run all commands as root.

```
root@laika:~# grep admin /etc/sudoers
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

The end result of this is that the user can type **sudo su -** and become root without having to enter the root password. The sudo command does require you to enter your own password. Thus the password prompt in the screenshot below is for sudo, not for su.

```
paul@laika:~$ sudo su -
Password:
root@laika:~#
```

27.14. sudo logging

Using **sudo** without authorization will result in a severe warning:

```
paul@rhel65:~$ sudo su -

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for paul:
paul is not in the sudoers file.  This incident will be reported.
paul@rhel65:~$
```

The root user can see this in the **/var/log/secure** on Red Hat and in **/var/log/auth.log** on Debian).

```
root@rhel65:~# tail /var/log/secure | grep sudo | tr -s ' '
Apr 13 16:03:42 rhel65 sudo: paul : user NOT in sudoers ; TTY=pts/0 ; PWD=\
/home/paul ; USER=root ; COMMAND=/bin/su -
root@rhel65:~#
```

27.15. practice: introduction to users

1. Run a command that displays only your currently logged on user name.

2. Display a list of all logged on users.

3. Display a list of all logged on users including the command they are running at this very moment.

4. Display your user name and your unique user identification (userid).

5. Use **su** to switch to another user account (unless you are root, you will need the password of the other account). And get back to the previous account.

6. Now use **su -** to switch to another user and notice the difference.

Note that **su -** gets you into the home directory of **Tania**.

7. Try to create a new user account (when using your normal user account). this should fail. (Details on adding user accounts are explained in the next chapter.)

8. Now try the same, but with **sudo** before your command.

27.16. solution: introduction to users

1. Run a command that displays only your currently logged on user name.

```
laura@debian7:~$ whoami
laura
laura@debian7:~$ echo $USER
laura
```

2. Display a list of all logged on users.

```
laura@debian7:~$ who
laura     pts/0        2014-10-13 07:22 (10.104.33.101)
laura@debian7:~$
```

3. Display a list of all logged on users including the command they are running at this very moment.

```
laura@debian7:~$ w
 07:47:02 up 16 min,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    10.104.33.101    07:30    6.00s  0.04s  0.00s w
root     pts/1    10.104.33.101    07:46    6.00s  0.01s  0.00s sleep 42
laura@debian7:~$
```

4. Display your user name and your unique user identification (userid).

```
laura@debian7:~$ id
uid=1005(laura) gid=1007(laura) groups=1007(laura)
laura@debian7:~$
```

5. Use **su** to switch to another user account (unless you are root, you will need the password of the other account). And get back to the previous account.

```
laura@debian7:~$ su tania
Password:
tania@debian7:/home/laura$ id
uid=1006(tania) gid=1008(tania) groups=1008(tania)
tania@debian7:/home/laura$ exit
laura@debian7:~$
```

6. Now use **su -** to switch to another user and notice the difference.

```
laura@debian7:~$ su - tania
Password:
tania@debian7:~$ pwd
/home/tania
tania@debian7:~$ logout
laura@debian7:~$
```

Note that **su -** gets you into the home directory of **Tania**.

7. Try to create a new user account (when using your normal user account). this should fail. (Details on adding user accounts are explained in the next chapter.)

```
laura@debian7:~$ useradd valentina
-su: useradd: command not found
laura@debian7:~$ /usr/sbin/useradd valentina
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
```

It is possible that **useradd** is located in **/sbin/useradd** on your computer.

8. Now try the same, but with **sudo** before your command.

```
laura@debian7:~$ sudo /usr/sbin/useradd valentina
[sudo] password for laura:
laura is not in the sudoers file.  This incident will be reported.
laura@debian7:~$
```

Notice that **laura** has no permission to use the **sudo** on this system.

Chapter 28. user management

This chapter will teach you how to use **useradd**, **usermod** and **userdel** to create, modify and remove user accounts.

You will need **root** access on a Linux computer to complete this chapter.

28.1. user management

User management on Linux can be done in three complementary ways. You can use the **graphical** tools provided by your distribution. These tools have a look and feel that depends on the distribution. If you are a novice Linux user on your home system, then use the graphical tool that is provided by your distribution. This will make sure that you do not run into problems.

Another option is to use **command line tools** like useradd, usermod, gpasswd, passwd and others. Server administrators are likely to use these tools, since they are familiar and very similar across many different distributions. This chapter will focus on these command line tools.

A third and rather extremist way is to **edit the local configuration files** directly using vi (or vipw/vigr). Do not attempt this as a novice on production systems!

28.2. /etc/passwd

The local user database on Linux (and on most Unixes) is **/etc/passwd**.

```
[root@RHEL5 ~]# tail /etc/passwd
inge:x:518:524:art dealer:/home/inge:/bin/ksh
ann:x:519:525:flute player:/home/ann:/bin/bash
frederik:x:520:526:rubius poet:/home/frederik:/bin/bash
steven:x:521:527:roman emperor:/home/steven:/bin/bash
pascale:x:522:528:artist:/home/pascale:/bin/ksh
geert:x:524:530:kernel developer:/home/geert:/bin/bash
wim:x:525:531:master damuti:/home/wim:/bin/bash
sandra:x:526:532:radish stresser:/home/sandra:/bin/bash
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

As you can see, this file contains seven columns separated by a colon. The columns contain the username, an x, the user id, the primary group id, a description, the name of the home directory, and the login shell.

More information can be found by typing **man 5 passwd**.

```
[root@RHEL5 ~]# man 5 passwd
```

28.3. root

The **root** user also called the **superuser** is the most powerful account on your Linux system. This user can do almost anything, including the creation of other users. The root user always has userid 0 (regardless of the name of the account).

```
[root@RHEL5 ~]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

28.4. useradd

You can add users with the **useradd** command. The example below shows how to add a user named yanina (last parameter) and at the same time forcing the creation of the home directory (-m), setting the name of the home directory (-d), and setting a description (-c).

```
[root@RHEL5 ~]# useradd -m -d /home/yanina -c "yanina wickmayer" yanina
[root@RHEL5 ~]# tail -1 /etc/passwd
yanina:x:529:529:yanina wickmayer:/home/yanina:/bin/bash
```

The user named yanina received userid 529 and **primary group** id 529.

28.5. /etc/default/useradd

Both Red Hat Enterprise Linux and Debian/Ubuntu have a file called **/etc/default/useradd** that contains some default user options. Besides using cat to display this file, you can also use **useradd -D**.

```
[root@RHEL4 ~]# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

28.6. userdel

You can delete the user yanina with **userdel**. The -r option of userdel will also remove the home directory.

```
[root@RHEL5 ~]# userdel -r yanina
```

28.7. usermod

You can modify the properties of a user with the **usermod** command. This example uses **usermod** to change the description of the user harry.

```
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:harry potter:/home/harry:/bin/bash
[root@RHEL4 ~]# usermod -c 'wizard' harry
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:wizard:/home/harry:/bin/bash
```

28.8. creating home directories

The easiest way to create a home directory is to supply the **-m** option with **useradd** (it is likely set as a default option on Linux).

A less easy way is to create a home directory manually with **mkdir** which also requires setting the owner and the permissions on the directory with **chmod** and **chown** (both commands are discussed in detail in another chapter).

```
[root@RHEL5 ~]# mkdir /home/laura
[root@RHEL5 ~]# chown laura:laura /home/laura
[root@RHEL5 ~]# chmod 700 /home/laura
[root@RHEL5 ~]# ls -ld /home/laura/
drwx------ 2 laura laura 4096 Jun 24 15:17 /home/laura/
```

28.9. /etc/skel/

When using **useradd** the **-m** option, the **/etc/skel/** directory is copied to the newly created home directory. The **/etc/skel/** directory contains some (usually hidden) files that contain profile settings and default values for applications. In this way **/etc/skel/** serves as a default home directory and as a default user profile.

```
[root@RHEL5 ~]# ls -la /etc/skel/
total 48
drwxr-xr-x  2 root root  4096 Apr  1 00:11 .
drwxr-xr-x 97 root root 12288 Jun 24 15:36 ..
-rw-r--r--  1 root root    24 Jul 12  2006 .bash_logout
-rw-r--r--  1 root root   176 Jul 12  2006 .bash_profile
-rw-r--r--  1 root root   124 Jul 12  2006 .bashrc
```

28.10. deleting home directories

The -r option of **userdel** will make sure that the home directory is deleted together with the user account.

```
[root@RHEL5 ~]# ls -ld /home/wim/
drwx------ 2 wim wim 4096 Jun 24 15:19 /home/wim/
[root@RHEL5 ~]# userdel -r wim
[root@RHEL5 ~]# ls -ld /home/wim/
ls: /home/wim/: No such file or directory
```

28.11. login shell

The **/etc/passwd** file specifies the **login shell** for the user. In the screenshot below you can see that user annelies will log in with the **/bin/bash** shell, and user laura with the **/bin/ksh** shell.

```
[root@RHEL5 ~]# tail -2 /etc/passwd
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

You can use the usermod command to change the shell for a user.

```
[root@RHEL5 ~]# usermod -s /bin/bash laura
[root@RHEL5 ~]# tail -1 /etc/passwd
laura:x:528:534:art dealer:/home/laura:/bin/bash
```

28.12. chsh

Users can change their login shell with the **chsh** command. First, user harry obtains a list of available shells (he could also have done a **cat /etc/shells**) and then changes his login shell to the **Korn shell** (/bin/ksh). At the next login, harry will default into ksh instead of bash.

```
[laura@centos7 ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/bin/ksh
/bin/tcsh
/bin/csh
[laura@centos7 ~]$
```

Note that the **-l** option does not exist on Debian and that the above screenshot assumes that **ksh** and **csh** shells are installed.

The screenshot below shows how **laura** can change her default shell (active on next login).

```
[laura@centos7 ~]$ chsh -s /bin/ksh
Changing shell for laura.
Password:
Shell changed.
```

28.13. practice: user management

1. Create a user account named **serena**, including a home directory and a description (or comment) that reads **Serena Williams**. Do all this in one single command.

2. Create a user named **venus**, including home directory, bash shell, a description that reads **Venus Williams** all in one single command.

3. Verify that both users have correct entries in **/etc/passwd**, **/etc/shadow** and **/etc/group**.

4. Verify that their home directory was created.

5. Create a user named **einstime** with **/bin/date** as his default logon shell.

7. What happens when you log on with the **einstime** user ? Can you think of a useful real world example for changing a user's login shell to an application ?

8. Create a file named **welcome.txt** and make sure every new user will see this file in their home directory.

9. Verify this setup by creating (and deleting) a test user account.

10. Change the default login shell for the **serena** user to **/bin/bash**. Verify before and after you make this change.

28.14. solution: user management

1. Create a user account named **serena**, including a home directory and a description (or comment) that reads **Serena Williams**. Do all this in one single command.

```
root@debian7:~# useradd -m -c 'Serena Williams' serena
```

2. Create a user named **venus**, including home directory, bash shell, a description that reads **Venus Williams** all in one single command.

```
root@debian7:~# useradd -m -c "Venus Williams" -s /bin/bash venus
```

3. Verify that both users have correct entries in **/etc/passwd**, **/etc/shadow** and **/etc/group**.

```
root@debian7:~# tail -2 /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/sh
venus:x:1009:1011:Venus Williams:/home/venus:/bin/bash
root@debian7:~# tail -2 /etc/shadow
serena:!:16358:0:99999:7:::
venus:!:16358:0:99999:7:::
root@debian7:~# tail -2 /etc/group
serena:x:1010:
venus:x:1011:
```

4. Verify that their home directory was created.

```
root@debian7:~# ls -lrt /home | tail -2
drwxr-xr-x 2 serena    serena    4096 Oct 15 10:50 serena
drwxr-xr-x 2 venus     venus     4096 Oct 15 10:59 venus
root@debian7:~#
```

5. Create a user named **einstime** with **/bin/date** as his default logon shell.

```
root@debian7:~# useradd -s /bin/date einstime
```

Or even better:

```
root@debian7:~# useradd -s $(which date) einstime
```

7. What happens when you log on with the **einstime** user ? Can you think of a useful real world example for changing a user's login shell to an application ?

```
root@debian7:~# su - einstime
Wed Oct 15 11:05:56 UTC 2014 # You get the output of the date command
root@debian7:~#
```

It can be useful when users need to access only one application on the server. Just logging in opens the application for them, and closing the application automatically logs them out.

8. Create a file named **welcome.txt** and make sure every new user will see this file in their home directory.

```
root@debian7:~# echo Hello > /etc/skel/welcome.txt
```

9. Verify this setup by creating (and deleting) a test user account.

```
root@debian7:~# useradd -m test
root@debian7:~# ls -l /home/test
total 4
-rw-r--r-- 1 test test 6 Oct 15 11:16 welcome.txt
root@debian7:~# userdel -r test
root@debian7:~#
```

10. Change the default login shell for the **serena** user to **/bin/bash**. Verify before and after you make this change.

```
root@debian7:~# grep serena /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/sh
root@debian7:~# usermod -s /bin/bash serena
root@debian7:~# grep serena /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/bash
root@debian7:~#
```

Chapter 29. user passwords

This chapter will tell you more about passwords for local users.

Three methods for setting passwords are explained; using the **passwd** command, using **openssel passwd**, and using the **crypt** function in a C program.

The chapter will also discuss password settings and disabling, suspending or locking accounts.

29.1. passwd

Passwords of users can be set with the **passwd** command. Users will have to provide their old password before twice entering the new one.

```
[tania@centos7 ~]$ passwd
Changing password for user tania.
Changing password for tania.
(current) UNIX password:
New password:
BAD PASSWORD: The password is shorter than 8 characters
New password:
BAD PASSWORD: The password is a palindrome
New password:
BAD PASSWORD: The password is too similar to the old one
passwd: Have exhausted maximum number of retries for service
```

As you can see, the passwd tool will do some basic verification to prevent users from using too simple passwords. The **root** user does not have to follow these rules (there will be a warning though). The **root** user also does not have to provide the old password before entering the new password twice.

```
root@debian7:~# passwd tania
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

29.2. shadow file

User passwords are encrypted and kept in **/etc/shadow**. The /etc/shadow file is read only and can only be read by root. We will see in the file permissions section how it is possible for users to change their password. For now, you will have to know that users can change their password with the **/usr/bin/passwd** command.

```
[root@centos7 ~]# tail -4 /etc/shadow
paul:$6$ikp2Xta5BT.Tml.p$2TZjNnOYNNQKpwLJqoGJbVsZG5/Fti8ovBRd.VzRbiDSl7TEq\
IaSMH.TeBKnTS/SjlMruW8qffC0JNORW.BTW1:16338:0:99999:7:::
tania:$6$8Z/zovxj$9qvoqT8i9KIrmN.k4EQwAF5ryz5yzNwEvYjAa9L5XVXQu.z4DlpvMREH\
eQpQzvRnqFdKkVj17H5ST.c79HDZw0:16356:0:99999:7:::
laura:$6$glDuTY5e$/NYYWLxfHgZFWeoujaXSMcR.Mz.lGOxtcxFocFVJNb98nbTPhWFXfKWG\
SyYh1WCv6763Wq54.w24Yr3uAZBOm/:16356:0:99999:7:::
valentina:$6$jrZa6PVI$1uQgqR6En9mZB6mKJ3LXRB4CnFko6LRhbh.v4iqUk9MVreui1lv7\
GxHOUDSKA0N55ZRNhGHa6T2ouFnVno/0o1:16356:0:99999:7:::
[root@centos7 ~]#
```

The **/etc/shadow** file contains nine colon separated columns. The nine fields contain (from left to right) the user name, the encrypted password (note that only inge and laura have an encrypted password), the day the password was last changed (day 1 is January 1, 1970), number of days the password must be left unchanged, password expiry day, warning number of days before password expiry, number of days after expiry before disabling the account, and the day the account was disabled (again, since 1970). The last field has no meaning yet.

All the passwords in the screenshot above are hashes of **hunter2**.

29.3. encryption with passwd

Passwords are stored in an encrypted format. This encryption is done by the **crypt** function. The easiest (and recommended) way to add a user with a password to the system is to add the user with the **useradd -m user** command, and then set the user's password with **passwd**.

```
[root@RHEL4 ~]# useradd -m xavier
[root@RHEL4 ~]# passwd xavier
Changing password for user xavier.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@RHEL4 ~]#
```

29.4. encryption with openssl

Another way to create users with a password is to use the -p option of useradd, but that option requires an encrypted password. You can generate this encrypted password with the **openssl passwd** command.

The **openssl passwd** command will generate several distinct hashes for the same password, for this it uses a **salt**.

```
paul@rhel65:~$ openssl passwd hunter2
86jcUNlnGDFpY
paul@rhel65:~$ openssl passwd hunter2
Yj7mDO9OAnvq6
paul@rhel65:~$ openssl passwd hunter2
YqDcJeGoDbzKA
paul@rhel65:~$
```

This **salt** can be chosen and is visible as the first two characters of the hash.

```
paul@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
paul@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
paul@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
paul@rhel65:~$
```

This example shows how to create a user with password.

```
root@rhel65:~# useradd -m -p $(openssl passwd hunter2) mohamed
```

Note that this command puts the password in your command history!

29.5. encryption with crypt

A third option is to create your own C program using the crypt function, and compile this into a command.

```
paul@rhel65:~$ cat MyCrypt.c
#include <stdio.h>
#define __USE_XOPEN
#include <unistd.h>

int main(int argc, char** argv)
{
 if(argc==3)
   {
       printf("%s\n", crypt(argv[1],argv[2]));
   }
   else
   {
       printf("Usage: MyCrypt $password $salt\n" );
   }
  return 0;
}
```

This little program can be compiled with **gcc** like this.

```
paul@rhel65:~$ gcc MyCrypt.c -o MyCrypt -lcrypt
```

To use it, we need to give two parameters to MyCrypt. The first is the unencrypted password, the second is the salt. The salt is used to perturb the encryption algorithm in one of 4096 different ways. This variation prevents two users with the same password from having the same entry in **/etc/shadow**.

```
paul@rhel65:~$ ./MyCrypt hunter2 42
42ZrbtP1Ze8G.
paul@rhel65:~$ ./MyCrypt hunter2 33
33d6taYSiEUXI
```

Did you notice that the first two characters of the password are the **salt**?

The standard output of the crypt function is using the DES algorithm which is old and can be cracked in minutes. A better method is to use **md5** passwords which can be recognized by a salt starting with 1.

```
paul@rhel65:~$ ./MyCrypt hunter2 '$1$42'
$1$42$7l6Y3xT5282XmZrtDOF9f0
paul@rhel65:~$ ./MyCrypt hunter2 '$6$42'
$6$42$OqFFAVnI3gTSYG0yI9TZWX9cpyQzwIop7HwpG1LLEsNBiMr4w6OvLX1KDa./UpwXfrFk1i...
```

The **md5** salt can be up to eight characters long. The salt is displayed in **/etc/shadow** between the second and third $, so never use the password as the salt!

```
paul@rhel65:~$ ./MyCrypt hunter2 '$1$hunter2'
$1$hunter2$YVxrxDmidq7Xf8Gdt6qM2.
```

29.6. /etc/login.defs

The **/etc/login.defs** file contains some default settings for user passwords like password aging and length settings. (You will also find the numerical limits of user ids and group ids and whether or not a home directory should be created by default).

```
root@rhel65:~# grep ^PASS /etc/login.defs
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7
```

Debian also has this file.

```
root@debian7:~# grep PASS /etc/login.defs
#  PASS_MAX_DAYS   Maximum number of days a password may be used.
#  PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#  PASS_WARN_AGE   Number of days warning given before a password expires.
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
#PASS_CHANGE_TRIES
#PASS_ALWAYS_WARN
#PASS_MIN_LEN
#PASS_MAX_LEN
# NO_PASSWORD_CONSOLE
root@debian7:~#
```

29.7. chage

The **chage** command can be used to set an expiration date for a user account (-E), set a minimum (-m) and maximum (-M) password age, a password expiration date, and set the number of warning days before the password expiration date. Much of this functionality is also available from the **passwd** command. The **-l** option of chage will list these settings for a user.

```
root@rhel65:~# chage -l paul
Last password change                                    : Mar 27, 2014
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
root@rhel65:~#
```

29.8. disabling a password

Passwords in **/etc/shadow** cannot begin with an exclamation mark. When the second field in **/etc/passwd** starts with an exclamation mark, then the password can not be used.

Using this feature is often called **locking**, **disabling**, or **suspending** a user account. Besides **vi** (or vipw) you can also accomplish this with **usermod**.

The first command in the next screenshot will show the hashed password of **laura** in **/etc/shadow**. The next command disables the password of **laura**, making it impossible for Laura to authenticate using this password.

```
root@debian7:~# grep laura /etc/shadow | cut -c1-70
laura:$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJV
root@debian7:~# usermod -L laura
```

As you can see below, the password hash is simply preceded with an exclamation mark.

```
root@debian7:~# grep laura /etc/shadow | cut -c1-70
laura:!$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJ
root@debian7:~#
```

The root user (and users with **sudo** rights on **su**) still will be able to **su** into the **laura** account (because the password is not needed here). Also note that **laura** will still be able to login if she has set up passwordless ssh!

```
root@debian7:~# su - laura
laura@debian7:~$
```

You can unlock the account again with **usermod -U**.

```
root@debian7:~# usermod -U laura
root@debian7:~# grep laura /etc/shadow | cut -c1-70
laura:$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJV
```

Watch out for tiny differences in the command line options of **passwd**, **usermod**, and **useradd** on different Linux distributions. Verify the local files when using features like **"disabling, suspending, or locking"** on user accounts and their passwords.

29.9. editing local files

If you still want to manually edit the **/etc/passwd** or **/etc/shadow**, after knowing these commands for password management, then use **vipw** instead of vi(m) directly. The **vipw** tool will do proper locking of the file.

```
[root@RHEL5 ~]# vipw /etc/passwd
vipw: the password file is busy (/etc/ptmp present)
```

29.10. practice: user passwords

1. Set the password for **serena** to **hunter2**.

2. Also set a password for **venus** and then lock the **venus** user account with **usermod**. Verify the locking in **/etc/shadow** before and after you lock it.

3. Use **passwd -d** to disable the **serena** password. Verify the **serena** line in **/etc/shadow** before and after disabling.

4. What is the difference between locking a user account and disabling a user account's password like we just did with **usermod -L** and **passwd -d**?

5. Try changing the password of serena to serena as serena.

6. Make sure **serena** has to change her password in 10 days.

7. Make sure every new user needs to change their password every 10 days.

8. Take a backup as root of **/etc/shadow**. Use **vi** to copy an encrypted **hunter2** hash from **venus** to **serena**. Can **serena** now log on with **hunter2** as a password ?

9. Why use **vipw** instead of **vi** ? What could be the problem when using **vi** or **vim** ?

10. Use **chsh** to list all shells (only works on RHEL/CentOS/Fedora), and compare to **cat /etc/shells**.

11. Which **useradd** option allows you to name a home directory ?

12. How can you see whether the password of user **serena** is locked or unlocked ? Give a solution with **grep** and a solution with **passwd**.

29.11. solution: user passwords

1. Set the password for **serena** to **hunter2**.

```
root@debian7:~# passwd serena
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

2. Also set a password for **venus** and then lock the **venus** user account with **usermod**. Verify the locking in **/etc/shadow** before and after you lock it.

```
root@debian7:~# passwd venus
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@debian7:~# grep venus /etc/shadow | cut -c1-70
venus:$6$gswzXICW$uSnKFV1kFKZmTPaMVS4AvNA/KO27OxN0v5LHdV9ed0gTyXrjUeM/
root@debian7:~# usermod -L venus
root@debian7:~# grep venus /etc/shadow | cut -c1-70
venus:!$6$gswzXICW$uSnKFV1kFKZmTPaMVS4AvNA/KO27OxN0v5LHdV9ed0gTyXrjUeM
```

Note that **usermod -L** precedes the password hash with an exclamation mark (!).

3. Use **passwd -d** to disable the **serena** password. Verify the **serena** line in **/etc/shadow** before and after disabling.

```
root@debian7:~# grep serena /etc/shadow | cut -c1-70
serena:$6$Es/omrPE$F2Ypu8kpLrfKdW0v/UIwA5jrYyBD2nwZ/dt.i/IypRgiPZSdB/B
root@debian7:~# passwd -d serena
passwd: password expiry information changed.
root@debian7:~# grep serena /etc/shadow
serena::16358:0:99999:7:::
root@debian7:~#
```

4. What is the difference between locking a user account and disabling a user account's password like we just did with **usermod -L** and **passwd -d**?

Locking will prevent the user from logging on to the system with his password by putting a ! in front of the password in **/etc/shadow**.

Disabling with **passwd** will erase the password from **/etc/shadow**.

5. Try changing the password of serena to serena as serena.

```
log on as serena, then execute: passwd serena... it should fail!
```

6. Make sure **serena** has to change her password in 10 days.

```
chage -M 10 serena
```

7. Make sure every new user needs to change their password every 10 days.

```
vi /etc/login.defs (and change PASS_MAX_DAYS to 10)
```

8. Take a backup as root of **/etc/shadow**. Use **vi** to copy an encrypted **hunter2** hash from **venus** to **serena**. Can **serena** now log on with **hunter2** as a password ?

```
Yes.
```

9. Why use **vipw** instead of **vi** ? What could be the problem when using **vi** or **vim** ?

```
vipw will give a warning when someone else is already using that file (with vipw).
```

10. Use **chsh** to list all shells (only works on RHEL/CentOS/Fedora), and compare to **cat /etc/shells**.

```
chsh -l
cat /etc/shells
```

11. Which **useradd** option allows you to name a home directory ?

```
-d
```

12. How can you see whether the password of user **serena** is locked or unlocked ? Give a solution with **grep** and a solution with **passwd**.

```
grep serena /etc/shadow
```

```
passwd -S serena
```

Chapter 30. user profiles

Logged on users have a number of preset (and customized) aliases, variables, and functions, but where do they come from ? The **shell** uses a number of startup files that are executed (or rather **sourced**) whenever the shell is invoked. What follows is an overview of startup scripts.

30.1. system profile

Both the **bash** and the **ksh** shell will verify the existence of **/etc/profile** and **source** it if it exists.

When reading this script, you will notice (both on Debian and on Red Hat Enterprise Linux) that it builds the PATH environment variable (among others). The script might also change the PS1 variable, set the HOSTNAME and execute even more scripts like **/etc/inputrc**

This screenshot uses grep to show PATH manipulation in **/etc/profile** on Debian.

```
root@debian7:~# grep PATH /etc/profile
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
export PATH
root@debian7:~#
```

This screenshot uses grep to show PATH manipulation in **/etc/profile** on RHEL7/CentOS7.

```
[root@centos7 ~]# grep PATH /etc/profile
    case ":${PATH}:" in
                PATH=$PATH:$1
                PATH=$1:$PATH
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
[root@centos7 ~]#
```

The **root user** can use this script to set aliases, functions, and variables for every user on the system.

30.2. ~/.bash_profile

When this file exists in the home directory, then **bash** will source it. On Debian Linux 5/6/7 this file does not exist by default.

RHEL7/CentOS7 uses a small **~/.bash_profile** where it checks for the existence of **~/.bashrc** and then sources it. It also adds $HOME/bin to the $PATH variable.

```
[root@rhel7 ~]# cat /home/paul/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH
[root@rhel7 ~]#
```

30.3. ~/.bash_login

When **.bash_profile** does not exist, then **bash** will check for **~/.bash_login** and source it.

Neither Debian nor Red Hat have this file by default.

30.4. ~/.profile

When neither **~/.bash_profile** and **~/.bash_login** exist, then bash will verify the existence of **~/.profile** and execute it. This file does not exist by default on Red Hat.

On Debian this script can execute **~/.bashrc** and will add $HOME/bin to the $PATH variable.

```
root@debian7:~# tail -11 /home/paul/.profile
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

RHEL/CentOS does not have this file by default.

30.5. ~/.bashrc

The **~/.bashrc** script is often sourced by other scripts. Let us take a look at what it does by default.

Red Hat uses a very simple **~/.bashrc**, checking for **/etc/bashrc** and sourcing it. It also leaves room for custom aliases and functions.

```
[root@rhel7 ~]# cat /home/paul/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
```

On Debian this script is quite a bit longer and configures $PS1, some history variables and a number af active and inactive aliases.

```
root@debian7:~# wc -l /home/paul/.bashrc
110 /home/paul/.bashrc
```

30.6. ~/.bash_logout

When exiting **bash**, it can execute **~/.bash_logout**.

Debian use this opportunity to clear the console screen.

```
serena@deb503:~$ cat .bash_logout
# ~/.bash_logout: executed by bash(1) when login shell exits.

# when leaving the console clear the screen to increase privacy

if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
```

Red Hat Enterprise Linux 5 will simple call the **/usr/bin/clear** command in this script.

```
[serena@rhel53 ~]$ cat .bash_logout
# ~/.bash_logout

/usr/bin/clear
```

Red Hat Enterprise Linux 6 and 7 create this file, but leave it empty (except for a comment).

```
paul@rhel65:~$ cat .bash_logout
# ~/.bash_logout
```

30.7. Debian overview

Below is a table overview of when Debian is running any of these bash startup scripts.

Table 30.1. Debian User Environment

script	su	su -	ssh	gdm
~./bashrc	no	yes	yes	yes
~/.profile	no	yes	yes	yes
/etc/profile	no	yes	yes	yes
/etc/bash.bashrc	yes	no	no	yes

30.8. RHEL5 overview

Below is a table overview of when Red Hat Enterprise Linux 5 is running any of these bash startup scripts.

Table 30.2. Red Hat User Environment

script	su	su -	ssh	gdm
~./bashrc	yes	yes	yes	yes
~/.bash_profile	no	yes	yes	yes
/etc/profile	no	yes	yes	yes
/etc/bashrc	yes	yes	yes	yes

30.9. practice: user profiles

1. Make a list of all the profile files on your system.

2. Read the contents of each of these, often they **source** extra scripts.

3. Put a unique variable, alias and function in each of those files.

4. Try several different ways to obtain a shell (su, su -, ssh, tmux, gnome-terminal, Ctrl-alt-F1, ...) and verify which of your custom variables, aliases and function are present in your environment.

5. Do you also know the order in which they are executed?

6. When an application depends on a setting in $HOME/.profile, does it matter whether $HOME/.bash_profile exists or not ?

30.10. solution: user profiles

1. Make a list of all the profile files on your system.

```
ls -a ~ ; ls -l /etc/pro* /etc/bash*
```

2. Read the contents of each of these, often they **source** extra scripts.

3. Put a unique variable, alias and function in each of those files.

4. Try several different ways to obtain a shell (su, su -, ssh, tmux, gnome-terminal, Ctrl-alt-F1, ...) and verify which of your custom variables, aliases and function are present in your environment.

5. Do you also know the order in which they are executed?

```
same name aliases, functions and variables will overwrite each other
```

6. When an application depends on a setting in $HOME/.profile, does it matter whether $HOME/.bash_profile exists or not ?

```
Yes it does matter. (man bash /INVOCATION)
```

Chapter 31. groups

Users can be listed in **groups**. Groups allow you to set permissions on the group level instead of having to set permissions for every individual user.

Every Unix or Linux distribution will have a graphical tool to manage groups. Novice users are advised to use this graphical tool. More experienced users can use command line tools to manage users, but be careful: Some distributions do not allow the mixed use of GUI and CLI tools to manage groups (YaST in Novell Suse). Senior administrators can edit the relevant files directly with **vi** or **vigr**.

31.1. groupadd

Groups can be created with the **groupadd** command. The example below shows the creation of five (empty) groups.

```
root@laika:~# groupadd tennis
root@laika:~# groupadd football
root@laika:~# groupadd snooker
root@laika:~# groupadd formula1
root@laika:~# groupadd salsa
```

31.2. group file

Users can be a member of several groups. Group membership is defined by the **/etc/group** file.

```
root@laika:~# tail -5 /etc/group
tennis:x:1006:
football:x:1007:
snooker:x:1008:
formula1:x:1009:
salsa:x:1010:
root@laika:~#
```

The first field is the group's name. The second field is the group's (encrypted) password (can be empty). The third field is the group identification or **GID**. The fourth field is the list of members, these groups have no members.

31.3. groups

A user can type the **groups** command to see a list of groups where the user belongs to.

```
[harry@RHEL4b ~]$ groups
harry sports
[harry@RHEL4b ~]$
```

31.4. usermod

Group membership can be modified with the useradd or **usermod** command.

```
root@laika:~# usermod -a -G tennis inge
root@laika:~# usermod -a -G tennis katrien
root@laika:~# usermod -a -G salsa katrien
root@laika:~# usermod -a -G snooker sandra
root@laika:~# usermod -a -G formula1 annelies
root@laika:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
snooker:x:1008:sandra
formula1:x:1009:annelies
salsa:x:1010:katrien
root@laika:~#
```

Be careful when using **usermod** to add users to groups. By default, the **usermod** command will **remove** the user from every group of which he is a member if the group is not listed in the command! Using the **-a** (append) switch prevents this behaviour.

31.5. groupmod

You can change the group name with the **groupmod** command.

```
root@laika:~# groupmod -n darts snooker
root@laika:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
formula1:x:1009:annelies
salsa:x:1010:katrien
darts:x:1008:sandra
```

31.6. groupdel

You can permanently remove a group with the **groupdel** command.

```
root@laika:~# groupdel tennis
root@laika:~#
```

31.7. gpasswd

You can delegate control of group membership to another user with the **gpasswd** command. In the example below we delegate permissions to add and remove group members to serena for the sports group. Then we **su** to serena and add harry to the sports group.

```
[root@RHEL4b ~]# gpasswd -A serena sports
[root@RHEL4b ~]# su - serena
[serena@RHEL4b ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry)
[serena@RHEL4b ~]$ gpasswd -a harry sports
Adding user harry to group sports
[serena@RHEL4b ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry),522(sports)
[serena@RHEL4b ~]$ tail -1 /etc/group
sports:x:522:serena,venus,harry
[serena@RHEL4b ~]$
```

Group administrators do not have to be a member of the group. They can remove themselves from a group, but this does not influence their ability to add or remove members.

```
[serena@RHEL4b ~]$ gpasswd -d serena sports
Removing user serena from group sports
[serena@RHEL4b ~]$ exit
```

Information about group administrators is kept in the **/etc/gshadow** file.

```
[root@RHEL4b ~]# tail -1 /etc/gshadow
sports:!:serena:venus,harry
[root@RHEL4b ~]#
```

To remove all group administrators from a group, use the **gpasswd** command to set an empty administrators list.

```
[root@RHEL4b ~]# gpasswd -A "" sports
```

31.8. newgrp

You can start a **child shell** with a new temporary **primary group** using the **newgrp** command.

```
root@rhel65:~# mkdir prigroup
root@rhel65:~# cd prigroup/
root@rhel65:~/prigroup# touch standard.txt
root@rhel65:~/prigroup# ls -l
total 0
-rw-r--r--. 1 root root 0 Apr 13 17:49 standard.txt
root@rhel65:~/prigroup# echo $SHLVL
1
root@rhel65:~/prigroup# newgrp tennis
root@rhel65:~/prigroup# echo $SHLVL
2
root@rhel65:~/prigroup# touch newgrp.txt
root@rhel65:~/prigroup# ls -l
total 0
-rw-r--r--. 1 root tennis 0 Apr 13 17:49 newgrp.txt
-rw-r--r--. 1 root root   0 Apr 13 17:49 standard.txt
root@rhel65:~/prigroup# exit
exit
root@rhel65:~/prigroup#
```

31.9. vigr

Similar to vipw, the **vigr** command can be used to manually edit the **/etc/group** file, since it will do proper locking of the file. Only experienced senior administrators should use **vi** or **vigr** to manage groups.

31.10. practice: groups

1. Create the groups tennis, football and sports.

2. In one command, make venus a member of tennis and sports.

3. Rename the football group to foot.

4. Use vi to add serena to the tennis group.

5. Use the id command to verify that serena is a member of tennis.

6. Make someone responsible for managing group membership of foot and sports. Test that it works.

31.11. solution: groups

1. Create the groups tennis, football and sports.

```
groupadd tennis ; groupadd football ; groupadd sports
```

2. In one command, make venus a member of tennis and sports.

```
usermod -a -G tennis,sports venus
```

3. Rename the football group to foot.

```
groupmod -n foot football
```

4. Use vi to add serena to the tennis group.

```
vi /etc/group
```

5. Use the id command to verify that serena is a member of tennis.

```
id (and after logoff logon serena should be member)
```

6. Make someone responsible for managing group membership of foot and sports. Test that it works.

```
gpasswd -A (to make manager)
```

```
gpasswd -a (to add member)
```

Part IX. file security

Table of Contents

Chapter 32. standard file permissions

This chapter contains details about basic file security through **file ownership** and **file permissions**.

32.1. file ownership

32.1.1. user owner and group owner

The **users** and **groups** of a system can be locally managed in **/etc/passwd** and **/etc/group**, or they can be in a NIS, LDAP, or Samba domain. These users and groups can **own** files. Actually, every file has a **user owner** and a **group owner**, as can be seen in the following screenshot.

```
paul@rhel65:~/owners$ ls -lh
total 636K
-rw-r--r--. 1 paul snooker 1.1K Apr  8 18:47 data.odt
-rw-r--r--. 1 paul paul    626K Apr  8 18:46 file1
-rw-r--r--. 1 root tennis   185 Apr  8 18:46 file2
-rw-rw-r--. 1 root root       0 Apr  8 18:47 stuff.txt
paul@rhel65:~/owners$
```

User paul owns three files; file1 has paul as **user owner** and has the group paul as **group owner**, data.odt is **group owned** by the group snooker, file2 by the group tennis.

The last file is called stuff.txt and is owned by the root user and the root group.

32.1.2. listing user accounts

You can use the following command to list all local user accounts.

```
paul@debian7~$ cut -d: -f1 /etc/passwd | column
root            ntp             sam             bert            naomi
daemon          mysql           tom             rino            matthias2
bin             paul            wouter          antonio         bram
sys             maarten         robrecht        simon           fabrice
sync            kevin           bilal           sven            chimene
games           yuri            dimitri         wouter2         messagebus
man             william         ahmed           tarik           roger
lp              yves            dylan           jan             frank
mail            kris            robin           ian             toon
news            hamid           matthias        ivan            rinus
uucp            vladimir        ben             azeddine        eddy
proxy           abiy            mike            eric            bram2
www-data        david           kevin2          kamel           keith
backup          chahid          kenzo           ischa           jesse
list            stef            aaron           bart            frederick
irc             joeri           lorenzo         omer            hans
gnats           glenn           jens            kurt            dries
nobody          yannick         ruben           steve           steve2
libuuid         christof        jelle           constantin      tomas
Debian-exim     george          stefaan         sam2            johan
statd           joost           marc            bjorn           tom2
sshd            arno            thomas          ronald
```

32.1.3. chgrp

You can change the group owner of a file using the **chgrp** command.

```
root@rhel65:/home/paul/owners# ls -l file2
-rw-r--r--. 1 root tennis 185 Apr  8 18:46 file2
root@rhel65:/home/paul/owners# chgrp snooker file2
root@rhel65:/home/paul/owners# ls -l file2
-rw-r--r--. 1 root snooker 185 Apr  8 18:46 file2
root@rhel65:/home/paul/owners#
```

32.1.4. chown

The user owner of a file can be changed with **chown** command.

```
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root paul 0 2008-08-06 14:11 FileForPaul
root@laika:/home/paul# chown paul FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 paul paul 0 2008-08-06 14:11 FileForPaul
```

You can also use **chown** to change both the user owner and the group owner.

```
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 paul paul 0 2008-08-06 14:11 FileForPaul
root@laika:/home/paul# chown root:project42 FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root project42 0 2008-08-06 14:11 FileForPaul
```

32.2. list of special files

When you use **ls -l**, for each file you can see ten characters before the user and group owner. The first character tells us the type of file. Regular files get a **-**, directories get a **d**, symbolic links are shown with an **l**, pipes get a **p**, character devices a **c**, block devices a **b**, and sockets an **s**.

Table 32.1. Unix special files

first character	file type
-	normal file
d	directory
l	symbolic link
p	named pipe
b	block device
c	character device
s	socket

Below a screenshot of a character device (the console) and a block device (the hard disk).

```
paul@debian6lt~$ ls -ld /dev/console /dev/sda
crw-------   1 root root  5, 1 Mar 15 12:45 /dev/console
brw-rw----   1 root disk  8, 0 Mar 15 12:45 /dev/sda
```

And here you can see a directory, a regular file and a symbolic link.

```
paul@debian6lt~$ ls -ld /etc /etc/hosts /etc/motd
drwxr-xr-x 128 root root 12288 Mar 15 18:34 /etc
-rw-r--r--   1 root root   372 Dec 10 17:36 /etc/hosts
lrwxrwxrwx   1 root root    13 Dec  5 10:36 /etc/motd -> /var/run/motd
```

32.3. permissions

32.3.1. rwx

The nine characters following the file type denote the permissions in three triplets. A permission can be **r** for read access, **w** for write access, and **x** for execute. You need the **r** permission to list (ls) the contents of a directory. You need the **x** permission to enter (cd) a directory. You need the **w** permission to create files in or remove files from a directory.

Table 32.2. standard Unix file permissions

permission	on a file	on a directory
r (read)	read file contents (cat)	read directory contents (ls)
w (write)	change file contents (vi)	create files in (touch)
x (execute)	execute the file	enter the directory (cd)

32.3.2. three sets of rwx

We already know that the output of **ls -l** starts with ten characters for each file. This screenshot shows a regular file (because the first character is a -).

```
paul@RHELv4u4:~/test$ ls -l proc42.bash
-rwxr-xr--  1 paul proj 984 Feb  6 12:01 proc42.bash
```

Below is a table describing the function of all ten characters.

Table 32.3. Unix file permissions position

position	characters	function
1	-	this is a regular file
2-4	rwx	permissions for the **user owner**
5-7	r-x	permissions for the **group owner**
8-10	r--	permissions for **others**

When you are the **user owner** of a file, then the **user owner permissions** apply to you. The rest of the permissions have no influence on your access to the file.

When you belong to the **group** that is the **group owner** of a file, then the **group owner permissions** apply to you. The rest of the permissions have no influence on your access to the file.

When you are not the **user owner** of a file and you do not belong to the **group owner**, then the **others permissions** apply to you. The rest of the permissions have no influence on your access to the file.

32.3.3. permission examples

Some example combinations on files and directories are seen in this screenshot. The name of the file explains the permissions.

```
paul@laika:~/perms$ ls -lh
total 12K
drwxr-xr-x 2 paul paul 4.0K 2007-02-07 22:26 AllEnter_UserCreateDelete
-rwxrwxrwx 1 paul paul    0 2007-02-07 22:21 EveryoneFullControl.txt
-r--r----- 1 paul paul    0 2007-02-07 22:21 OnlyOwnersRead.txt
-rwxrwx--- 1 paul paul    0 2007-02-07 22:21 OwnersAll_RestNothing.txt
dr-xr-x--- 2 paul paul 4.0K 2007-02-07 22:25 UserAndGroupEnter
dr-x------ 2 paul paul 4.0K 2007-02-07 22:25 OnlyUserEnter
paul@laika:~/perms$
```

To summarise, the first **rwx** triplet represents the permissions for the **user owner**. The second triplet corresponds to the **group owner**; it specifies permissions for all members of that group. The third triplet defines permissions for all **other** users that are not the user owner and are not a member of the group owner.

32.3.4. setting permissions (chmod)

Permissions can be changed with **chmod**. The first example gives the user owner execute permissions.

```
paul@laika:~/perms$ ls -l permissions.txt
-rw-r--r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
paul@laika:~/perms$ chmod u+x permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxr--r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

This example removes the group owners read permission.

```
paul@laika:~/perms$ chmod g-r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx---r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

This example removes the others read permission.

```
paul@laika:~/perms$ chmod o-r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx------ 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

This example gives all of them the write permission.

```
paul@laika:~/perms$ chmod a+w permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx-w--w- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

You don't even have to type the a.

```
paul@laika:~/perms$ chmod +x permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx-wx-wx 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

You can also set explicit permissions.

```
paul@laika:~/perms$ chmod u=rw permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw--wx-wx 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

Feel free to make any kind of combination.

```
paul@laika:~/perms$ chmod u=rw,g=rw,o=r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw-rw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

Even fishy combinations are accepted by chmod.

```
paul@laika:~/perms$ chmod u=rwx,ug+rw,o=r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxrw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

32.3.5. setting octal permissions

Most Unix administrators will use the **old school** octal system to talk about and set permissions. Look at the triplet bitwise, equating r to 4, w to 2, and x to 1.

Table 32.4. Octal permissions

binary	octal	permission
000	0	---
001	1	--x
010	2	-w-
011	3	-wx
100	4	r--
101	5	r-x
110	6	rw-
111	7	rwx

This makes **777** equal to rwxrwxrwx and by the same logic, 654 mean rw-r-xr-- . The **chmod** command will accept these numbers.

```
paul@laika:~/perms$ chmod 777 permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxrwxrwx 1 paul paul 0 2007-02-07 22:34 permissions.txt
paul@laika:~/perms$ chmod 664 permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw-rw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
paul@laika:~/perms$ chmod 750 permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxr-x--- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

32.3.6. umask

When creating a file or directory, a set of default permissions are applied. These default permissions are determined by the **umask**. The **umask** specifies permissions that you do not want set on by default. You can display the **umask** with the **umask** command.

```
[Harry@RHEL4b ~]$ umask
0002
[Harry@RHEL4b ~]$ touch test
[Harry@RHEL4b ~]$ ls -l test
-rw-rw-r--  1 Harry Harry 0 Jul 24 06:03 test
[Harry@RHEL4b ~]$
```

As you can also see, the file is also not executable by default. This is a general security feature among Unixes; newly created files are never executable by default. You have to explicitly do a **chmod +x** to make a file executable. This also means that the 1 bit in the **umask** has no meaning--a **umask** of 0022 is the same as 0033.

32.3.7. mkdir -m

When creating directories with **mkdir** you can use the **-m** option to set the **mode**. This screenshot explains.

```
paul@debian5~$ mkdir -m 700 MyDir
paul@debian5~$ mkdir -m 777 Public
paul@debian5~$ ls -dl MyDir/ Public/
drwx------ 2 paul paul 4096 2011-10-16 19:16 MyDir/
drwxrwxrwx 2 paul paul 4096 2011-10-16 19:16 Public/
```

32.3.8. cp -p

To preserve permissions and time stamps from source files, use **cp -p**.

```
paul@laika:~/perms$ cp file* cp
paul@laika:~/perms$ cp -p file* cpp
paul@laika:~/perms$ ll *
-rwx------ 1 paul paul    0 2008-08-25 13:26 file33
-rwxr-x--- 1 paul paul    0 2008-08-25 13:26 file42

cp:
total 0
-rwx------ 1 paul paul 0 2008-08-25 13:34 file33
-rwxr-x--- 1 paul paul 0 2008-08-25 13:34 file42

cpp:
total 0
-rwx------ 1 paul paul 0 2008-08-25 13:26 file33
-rwxr-x--- 1 paul paul 0 2008-08-25 13:26 file42
```

32.4. practice: standard file permissions

1. As normal user, create a directory ~/permissions. Create a file owned by yourself in there.

2. Copy a file owned by root from /etc/ to your permissions dir, who owns this file now ?

3. As root, create a file in the users ~/permissions directory.

4. As normal user, look at who owns this file created by root.

5. Change the ownership of all files in ~/permissions to yourself.

6. Make sure you have all rights to these files, and others can only read.

7. With chmod, is 770 the same as rwxrwx--- ?

8. With chmod, is 664 the same as r-xr-xr-- ?

9. With chmod, is 400 the same as r-------- ?

10. With chmod, is 734 the same as rwxr-xr-- ?

11a. Display the umask in octal and in symbolic form.

11b. Set the umask to 077, but use the symbolic format to set it. Verify that this works.

12. Create a file as root, give only read to others. Can a normal user read this file ? Test writing to this file with vi.

13a. Create a file as normal user, give only read to others. Can another normal user read this file ? Test writing to this file with vi.

13b. Can root read this file ? Can root write to this file with vi ?

14. Create a directory that belongs to a group, where every member of that group can read and write to files, and create files. Make sure that people can only delete their own files.

32.5. solution: standard file permissions

1. As normal user, create a directory ~/permissions. Create a file owned by yourself in there.

```
mkdir ~/permissions ; touch ~/permissions/myfile.txt
```

2. Copy a file owned by root from /etc/ to your permissions dir, who owns this file now ?

```
cp /etc/hosts ~/permissions/
```

The copy is owned by you.

3. As root, create a file in the users ~/permissions directory.

```
(become root)# touch /home/username/permissions/rootfile
```

4. As normal user, look at who owns this file created by root.

```
ls -l ~/permissions
```

The file created by root is owned by root.

5. Change the ownership of all files in ~/permissions to yourself.

```
chown user ~/permissions/*
```

You cannot become owner of the file that belongs to root.

6. Make sure you have all rights to these files, and others can only read.

```
chmod 644 (on files)
```

```
chmod 755 (on directories)
```

7. With chmod, is 770 the same as rwxrwx--- ?

yes

8. With chmod, is 664 the same as r-xr-xr-- ?

No

9. With chmod, is 400 the same as r-------- ?

yes

10. With chmod, is 734 the same as rwxr-xr-- ?

no

11a. Display the umask in octal and in symbolic form.

```
umask ; umask -S
```

11b. Set the umask to 077, but use the symbolic format to set it. Verify that this works.

```
umask -S u=rwx,go=
```

12. Create a file as root, give only read to others. Can a normal user read this file ? Test writing to this file with vi.

```
(become root)
# echo hello > /home/username/root.txt
# chmod 744 /home/username/root.txt
(become user)
vi ~/root.txt
```

13a. Create a file as normal user, give only read to others. Can another normal user read this file ? Test writing to this file with vi.

```
echo hello > file ; chmod 744 file
```

Yes, others can read this file

13b. Can root read this file ? Can root write to this file with vi ?

Yes, root can read and write to this file. Permissions do not apply to root.

14. Create a directory that belongs to a group, where every member of that group can read and write to files, and create files. Make sure that people can only delete their own files.

```
mkdir /home/project42 ; groupadd project42
chgrp project42 /home/project42 ; chmod 775 /home/project42
```

You can not yet do the last part of this exercise...

Chapter 33. advanced file permissions

33.1. sticky bit on directory

You can set the **sticky bit** on a directory to prevent users from removing files that they do not own as a user owner. The sticky bit is displayed at the same location as the x permission for others. The sticky bit is represented by a **t** (meaning x is also there) or a **T** (when there is no x for others).

```
root@RHELv4u4:~# mkdir /project55
root@RHELv4u4:~# ls -ld /project55
drwxr-xr-x  2 root root 4096 Feb  7 17:38 /project55
root@RHELv4u4:~# chmod +t /project55/
root@RHELv4u4:~# ls -ld /project55
drwxr-xr-t  2 root root 4096 Feb  7 17:38 /project55
root@RHELv4u4:~#
```

The **sticky bit** can also be set with octal permissions, it is binary 1 in the first of four triplets.

```
root@RHELv4u4:~# chmod 1775 /project55/
root@RHELv4u4:~# ls -ld /project55
drwxrwxr-t  2 root root 4096 Feb  7 17:38 /project55
root@RHELv4u4:~#
```

You will typically find the **sticky bit** on the **/tmp** directory.

```
root@barry:~# ls -ld /tmp
drwxrwxrwt 6 root root 4096 2009-06-04 19:02 /tmp
```

33.2. setgid bit on directory

setgid can be used on directories to make sure that all files inside the directory are owned by the group owner of the directory. The **setgid** bit is displayed at the same location as the x permission for group owner. The **setgid** bit is represented by an **s** (meaning x is also there) or a **S** (when there is no x for the group owner). As this example shows, even though **root** does not belong to the group proj55, the files created by root in /project55 will belong to proj55 since the **setgid** is set.

```
root@RHELv4u4:~# groupadd proj55
root@RHELv4u4:~# chown root:proj55 /project55/
root@RHELv4u4:~# chmod 2775 /project55/
root@RHELv4u4:~# touch /project55/fromroot.txt
root@RHELv4u4:~# ls -ld /project55/
drwxrwsr-x  2 root proj55 4096 Feb  7 17:45 /project55/
root@RHELv4u4:~# ls -l /project55/
total 4
-rw-r--r--  1 root proj55 0 Feb  7 17:45 fromroot.txt
root@RHELv4u4:~#
```

You can use the **find** command to find all **setgid** directories.

```
paul@laika:~$ find / -type d -perm -2000 2> /dev/null
/var/log/mysql
/var/log/news
/var/local
...
```

33.3. setgid and setuid on regular files

These two permissions cause an executable file to be executed with the permissions of the **file owner** instead of the **executing owner**. This means that if any user executes a program that belongs to the **root user**, and the **setuid** bit is set on that program, then the program runs as **root**. This can be dangerous, but sometimes this is good for security.

Take the example of passwords; they are stored in **/etc/shadow** which is only readable by **root**. (The **root** user never needs permissions anyway.)

```
root@RHELv4u4:~# ls -l /etc/shadow
-r--------  1 root root 1260 Jan 21 07:49 /etc/shadow
```

Changing your password requires an update of this file, so how can normal non-root users do this? Let's take a look at the permissions on the **/usr/bin/passwd**.

```
root@RHELv4u4:~# ls -l /usr/bin/passwd
-r-s--x--x  1 root root 21200 Jun 17  2005 /usr/bin/passwd
```

When running the **passwd** program, you are executing it with **root** credentials.

You can use the **find** command to find all **setuid** programs.

```
paul@laika:~$ find /usr/bin -type f -perm -04000
/usr/bin/arping
/usr/bin/kgrantpty
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/fping6
/usr/bin/passwd
/usr/bin/gpasswd
...
```

In most cases, setting the **setuid** bit on executables is sufficient. Setting the **setgid** bit will result in these programs to run with the credentials of their group owner.

33.4. setuid on sudo

The **sudo** binary has the **setuid** bit set, so any user can run it with the effective userid of root.

```
paul@rhel65:~$ ls -l $(which sudo)
---s--x--x. 1 root root 123832 Oct  7  2013 /usr/bin/sudo
paul@rhel65:~$
```

33.5. practice: sticky, setuid and setgid bits

1a. Set up a directory, owned by the group sports.

1b. Members of the sports group should be able to create files in this directory.

1c. All files created in this directory should be group-owned by the sports group.

1d. Users should be able to delete only their own user-owned files.

1e. Test that this works!

2. Verify the permissions on **/usr/bin/passwd**. Remove the **setuid**, then try changing your password as a normal user. Reset the permissions back and try again.

3. If time permits (or if you are waiting for other students to finish this practice), read about file attributes in the man page of chattr and lsattr. Try setting the i attribute on a file and test that it works.

33.6. solution: sticky, setuid and setgid bits

1a. Set up a directory, owned by the group sports.

```
groupadd sports
```

```
mkdir /home/sports
```

```
chown root:sports /home/sports
```

1b. Members of the sports group should be able to create files in this directory.

```
chmod 770 /home/sports
```

1c. All files created in this directory should be group-owned by the sports group.

```
chmod 2770 /home/sports
```

1d. Users should be able to delete only their own user-owned files.

```
chmod +t /home/sports
```

1e. Test that this works!

Log in with different users (group members and others and root), create files and watch the permissions. Try changing and deleting files...

2. Verify the permissions on **/usr/bin/passwd**. Remove the **setuid**, then try changing your password as a normal user. Reset the permissions back and try again.

```
root@deb503:~# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 31704 2009-11-14 15:41 /usr/bin/passwd
root@deb503:~# chmod 755 /usr/bin/passwd
root@deb503:~# ls -l /usr/bin/passwd
-rwxr-xr-x 1 root root 31704 2009-11-14 15:41 /usr/bin/passwd
```

A normal user cannot change password now.

```
root@deb503:~# chmod 4755 /usr/bin/passwd
root@deb503:~# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 31704 2009-11-14 15:41 /usr/bin/passwd
```

3. If time permits (or if you are waiting for other students to finish this practice), read about file attributes in the man page of chattr and lsattr. Try setting the i attribute on a file and test that it works.

```
paul@laika:~$ sudo su -
[sudo] password for paul:
root@laika:~# mkdir attr
root@laika:~# cd attr/
root@laika:~/attr# touch file42
root@laika:~/attr# lsattr
---------------- ./file42
root@laika:~/attr# chattr +i file42
```

```
root@laika:~/attr# lsattr
----i------------- ./file42
root@laika:~/attr# rm -rf file42
rm: cannot remove `file42': Operation not permitted
root@laika:~/attr# chattr -i file42
root@laika:~/attr# rm -rf file42
root@laika:~/attr#
```

Chapter 34. access control lists

Standard Unix permissions might not be enough for some organisations. This chapter introduces **access control lists** or **acl's** to further protect files and directories.

34.1. acl in /etc/fstab

File systems that support **access control lists**, or **acls**, have to be mounted with the **acl** option listed in **/etc/fstab**. In the example below, you can see that the root file system has **acl** support, whereas /home/data does not.

```
root@laika:~# tail -4 /etc/fstab
/dev/sda1        /              ext3     acl,relatime    0  1
/dev/sdb2        /home/data     auto     noacl,defaults  0  0
pasha:/home/r    /home/pasha    nfs      defaults        0  0
wolf:/srv/data   /home/wolf     nfs      defaults        0  0
```

34.2. getfacl

Reading **acls** can be done with **/usr/bin/getfacl**. This screenshot shows how to read the **acl** of **file33** with **getfacl**.

```
paul@laika:~/test$ getfacl file33
# file: file33
# owner: paul
# group: paul
user::rw-
group::r--
mask::rwx
other::r--
```

34.3. setfacl

Writing or changing **acls** can be done with **/usr/bin/setfacl**. These screenshots show how to change the **acl** of **file33** with **setfacl**.

First we add **u**ser **sandra** with octal permission **7** to the **acl**.

```
paul@laika:~/test$ setfacl -m u:sandra:7 file33
```

Then we add the **g**roup tennis with octal permission **6** to the **acl** of the same file.

```
paul@laika:~/test$ setfacl -m g:tennis:6 file33
```

The result is visible with **getfacl**.

```
paul@laika:~/test$ getfacl file33
# file: file33
# owner: paul
# group: paul
user::rw-
user:sandra:rwx
group::r--
group:tennis:rw-
mask::rwx
other::r--
```

34.4. remove an acl entry

The **-x** option of the **setfacl** command will remove an **acl** entry from the targeted file.

```
paul@laika:~/test$ setfacl -m u:sandra:7 file33
paul@laika:~/test$ getfacl file33 | grep sandra
user:sandra:rwx
paul@laika:~/test$ setfacl -x sandra file33
paul@laika:~/test$ getfacl file33 | grep sandra
```

Note that omitting the **u** or **g** when defining the **acl** for an account will default it to a user account.

34.5. remove the complete acl

The **-b** option of the **setfacl** command will remove the **acl** from the targeted file.

```
paul@laika:~/test$ setfacl -b file33
paul@laika:~/test$ getfacl file33
# file: file33
# owner: paul
# group: paul
user::rw-
group::r--
other::r--
```

34.6. the acl mask

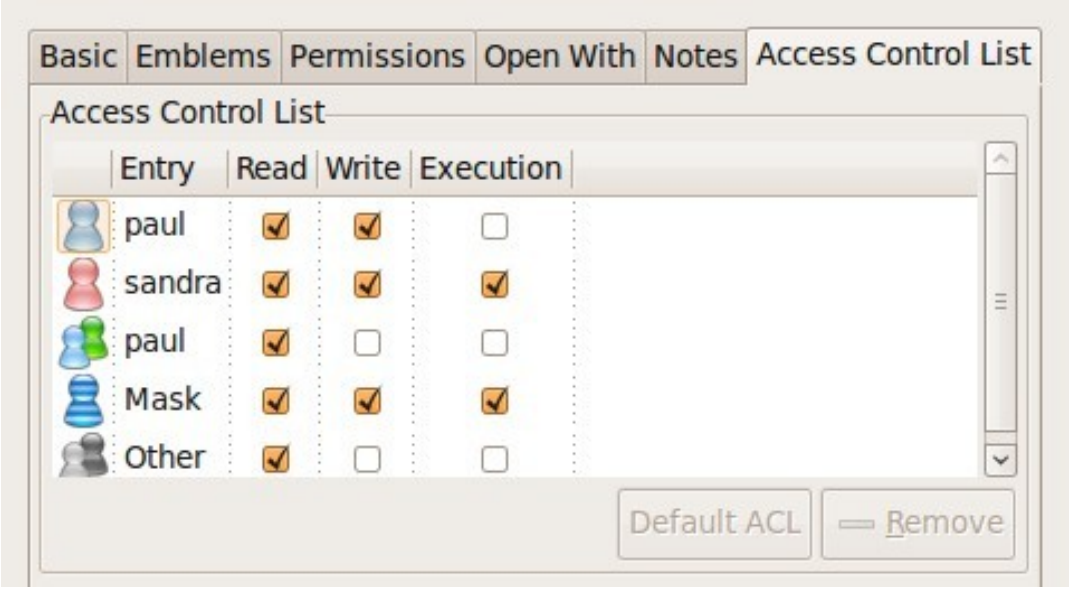
The **acl mask** defines the maximum effective permissions for any entry in the **acl**. This **mask** is calculated every time you execute the **setfacl** or **chmod** commands.

You can prevent the calculation by using the **--no-mask** switch.

```
paul@laika:~/test$ setfacl --no-mask -m u:sandra:7 file33
paul@laika:~/test$ getfacl file33
# file: file33
# owner: paul
# group: paul
user::rw-
user:sandra:rwx   #effective:rw-
group::r--
mask::rw-
other::r--
```

34.7. eiciel

Desktop users might want to use **eiciel** to manage **acls** with a graphical tool.



You will need to install **eiciel** and **nautilus-actions** to have an extra tab in **nautilus** to manage **acls**.

```
paul@laika:~$ sudo aptitude install eiciel nautilus-actions
```

Chapter 35. file links

An average computer using Linux has a file system with many **hard links** and **symbolic links**.

To understand links in a file system, you first have to understand what an **inode** is.

35.1. inodes

35.1.1. inode contents

An **inode** is a data structure that contains metadata about a file. When the file system stores a new file on the hard disk, it stores not only the contents (data) of the file, but also extra properties like the name of the file, the creation date, its permissions, the owner of the file, and more. All this information (except the name of the file and the contents of the file) is stored in the **inode** of the file.

The **ls -l** command will display some of the inode contents, as seen in this screenshot.

```
root@rhel53 ~# ls -ld /home/project42/
drwxr-xr-x 4 root pro42 4.0K Mar 27 14:29 /home/project42/
```

35.1.2. inode table

The **inode table** contains all of the **inodes** and is created when you create the file system (with **mkfs**). You can use the **df -i** command to see how many **inodes** are used and free on mounted file systems.

```
root@rhel53 ~# df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/mapper/VolGroup00-LogVol00
                     4947968  115326 4832642    3% /
/dev/hda1              26104      45   26059    1% /boot
tmpfs                  64417       1   64416    1% /dev/shm
/dev/sda1             262144    2207  259937    1% /home/project42
/dev/sdb1              74400    5519   68881    8% /home/project33
/dev/sdb5                  0       0       0    -  /home/sales
/dev/sdb6             100744      11  100733    1% /home/research
```

In the **df -i** screenshot above you can see the **inode** usage for several mounted **file systems**. You don't see numbers for **/dev/sdb5** because it is a **fat** file system.

35.1.3. inode number

Each **inode** has a unique number (the inode number). You can see the **inode** numbers with the **ls -li** command.

```
paul@RHELv4u4:~/test$ touch file1
paul@RHELv4u4:~/test$ touch file2
paul@RHELv4u4:~/test$ touch file3
paul@RHELv4u4:~/test$ ls -li
total 12
817266 -rw-rw-r--  1 paul paul 0 Feb  5 15:38 file1
817267 -rw-rw-r--  1 paul paul 0 Feb  5 15:38 file2
817268 -rw-rw-r--  1 paul paul 0 Feb  5 15:38 file3
paul@RHELv4u4:~/test$
```

These three files were created one after the other and got three different **inodes** (the first column). All the information you see with this **ls** command resides in the **inode**, except for the filename (which is contained in the directory).

35.1.4. inode and file contents

Let's put some data in one of the files.

```
paul@RHELv4u4:~/test$ ls -li
total 16
817266 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file1
817270 -rw-rw-r--  1 paul paul 92 Feb  5 15:42 file2
817268 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file3
paul@RHELv4u4:~/test$ cat file2
It is winter now and it is very cold.
We do not like the cold, we prefer hot summer nights.
paul@RHELv4u4:~/test$
```

The data that is displayed by the **cat** command is not in the **inode**, but somewhere else on the disk. The **inode** contains a pointer to that data.

35.2. about directories

35.2.1. a directory is a table

A **directory** is a special kind of file that contains a table which maps filenames to inodes. Listing our current directory with **ls -ali** will display the contents of the directory file.

```
paul@RHELv4u4:~/test$ ls -ali
total 32
817262 drwxrwxr-x   2 paul paul 4096 Feb  5 15:42 .
800768 drwx------  16 paul paul 4096 Feb  5 15:42 ..
817266 -rw-rw-r--   1 paul paul    0 Feb  5 15:38 file1
817270 -rw-rw-r--   1 paul paul   92 Feb  5 15:42 file2
817268 -rw-rw-r--   1 paul paul    0 Feb  5 15:38 file3
paul@RHELv4u4:~/test$
```

35.2.2. . and ..

You can see five names, and the mapping to their five inodes. The dot **.** is a mapping to itself, and the dotdot **..** is a mapping to the parent directory. The three other names are mappings to different inodes.

35.3. hard links

35.3.1. creating hard links

When we create a **hard link** to a file with **ln**, an extra entry is added in the directory. A new file name is mapped to an existing inode.

```
paul@RHELv4u4:~/test$ ln file2 hardlink_to_file2
paul@RHELv4u4:~/test$ ls -li
total 24
817266 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file1
817270 -rw-rw-r--  2 paul paul 92 Feb  5 15:42 file2
817268 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file3
817270 -rw-rw-r--  2 paul paul 92 Feb  5 15:42 hardlink_to_file2
paul@RHELv4u4:~/test$
```

Both files have the same inode, so they will always have the same permissions and the same owner. Both files will have the same content. Actually, both files are equal now, meaning you can safely remove the original file, the hardlinked file will remain. The inode contains a counter, counting the number of hard links to itself. When the counter drops to zero, then the inode is emptied.

35.3.2. finding hard links

You can use the **find** command to look for files with a certain inode. The screenshot below shows how to search for all filenames that point to **inode** 817270. Remember that an **inode** number is unique to its partition.

```
paul@RHELv4u4:~/test$ find / -inum 817270 2> /dev/null
/home/paul/test/file2
/home/paul/test/hardlink_to_file2
```

35.4. symbolic links

Symbolic links (sometimes called **soft links**) do not link to inodes, but create a name to name mapping. Symbolic links are created with **ln -s**. As you can see below, the **symbolic link** gets an inode of its own.

```
paul@RHELv4u4:~/test$ ln -s file2 symlink_to_file2
paul@RHELv4u4:~/test$ ls -li
total 32
817273 -rw-rw-r--  1 paul paul  13 Feb  5 17:06 file1
817270 -rw-rw-r--  2 paul paul 106 Feb  5 17:04 file2
817268 -rw-rw-r--  1 paul paul   0 Feb  5 15:38 file3
817270 -rw-rw-r--  2 paul paul 106 Feb  5 17:04 hardlink_to_file2
817267 lrwxrwxrwx  1 paul paul   5 Feb  5 16:55 symlink_to_file2 -> file2
paul@RHELv4u4:~/test$
```

Permissions on a symbolic link have no meaning, since the permissions of the target apply. Hard links are limited to their own partition (because they point to an inode), symbolic links can link anywhere (other file systems, even networked).

35.5. removing links

Links can be removed with **rm**.

```
paul@laika:~$ touch data.txt
paul@laika:~$ ln -s data.txt sl_data.txt
paul@laika:~$ ln data.txt hl_data.txt
paul@laika:~$ rm sl_data.txt
paul@laika:~$ rm hl_data.txt
```

35.6. practice : links

1. Create two files named winter.txt and summer.txt, put some text in them.
2. Create a hard link to winter.txt named hlwinter.txt.
3. Display the inode numbers of these three files, the hard links should have the same inode.
4. Use the find command to list the two hardlinked files
5. Everything about a file is in the inode, except two things : name them!
6. Create a symbolic link to summer.txt called slsummer.txt.
7. Find all files with inode number 2. What does this information tell you ?
8. Look at the directories /etc/init.d/ /etc/rc2.d/ /etc/rc3.d/ ... do you see the links ?
9. Look in /lib with ls -l...
10. Use **find** to look in your home directory for regular files that do not(!) have one hard link.

35.7. solution : links

1. Create two files named winter.txt and summer.txt, put some text in them.

```
echo cold > winter.txt ; echo hot > summer.txt
```

2. Create a hard link to winter.txt named hlwinter.txt.

```
ln winter.txt hlwinter.txt
```

3. Display the inode numbers of these three files, the hard links should have the same inode.

```
ls -li winter.txt summer.txt hlwinter.txt
```

4. Use the find command to list the two hardlinked files

```
find . -inum xyz #replace xyz with the inode number
```

5. Everything about a file is in the inode, except two things : name them!

The name of the file is in a directory, and the contents is somewhere on the disk.

6. Create a symbolic link to summer.txt called slsummer.txt.

```
ln -s summer.txt slsummer.txt
```

7. Find all files with inode number 2. What does this information tell you ?

It tells you there is more than one inode table (one for every formatted partition + virtual file systems)

8. Look at the directories /etc/init.d/ /etc/rc.d/ /etc/rc3.d/ ... do you see the links ?

```
ls -l /etc/init.d
```

```
ls -l /etc/rc2.d
```

```
ls -l /etc/rc3.d
```

9. Look in /lib with ls -l...

```
ls -l /lib
```

10. Use **find** to look in your home directory for regular files that do not(!) have one hard link.

```
find ~ ! -links 1 -type f
```

Part X. Appendices

Table of Contents

Appendix A. keyboard settings

A.1. about keyboard layout

Many people (like US-Americans) prefer the default US-qwerty keyboard layout. So when you are not from the USA and want a local keyboard layout on your system, then the best practice is to select this keyboard at installation time. Then the keyboard layout will always be correct. Also, whenever you use ssh to remotely manage a Linux system, your local keyboard layout will be used, independent of the server keyboard configuration. So you will not find much information on changing keyboard layout on the fly on linux, because not many people need it. Below are some tips to help you.

A.2. X Keyboard Layout

This is the relevant portion in /etc/X11/xorg.conf, first for Belgian azerty, then for US-qwerty.

```
[paul@RHEL5 ~]$ grep -i xkb /etc/X11/xorg.conf
        Option      "XkbModel" "pc105"
        Option      "XkbLayout" "be"
```

```
[paul@RHEL5 ~]$ grep -i xkb /etc/X11/xorg.conf
        Option      "XkbModel" "pc105"
        Option      "XkbLayout" "us"
```

When in Gnome or KDE or any other graphical environment, look in the graphical menu in preferences, there will be a keyboard section to choose your layout. Use the graphical menu instead of editing xorg.conf.

A.3. shell keyboard layout

When in bash, take a look in the /etc/sysconfig/keyboard file. Below a sample US-qwerty configuration, followed by a Belgian azerty configuration.

```
[paul@RHEL5 ~]$ cat /etc/sysconfig/keyboard
KEYBOARDTYPE="pc"
KEYTABLE="us"
```

```
[paul@RHEL5 ~]$ cat /etc/sysconfig/keyboard
KEYBOARDTYPE="pc"
KEYTABLE="be-latin1"
```

The keymaps themselves can be found in /usr/share/keymaps or /lib/kbd/keymaps.

```
[paul@RHEL5 ~]$ ls -l /lib/kbd/keymaps/
total 52
drwxr-xr-x 2 root root 4096 Apr  1 00:14 amiga
```

```
drwxr-xr-x 2 root root 4096 Apr  1 00:14 atari
drwxr-xr-x 8 root root 4096 Apr  1 00:14 i386
drwxr-xr-x 2 root root 4096 Apr  1 00:14 include
drwxr-xr-x 4 root root 4096 Apr  1 00:14 mac
lrwxrwxrwx 1 root root    3 Apr  1 00:14 ppc -> mac
drwxr-xr-x 2 root root 4096 Apr  1 00:14 sun
```

Appendix B. hardware

B.1. buses

B.1.1. about buses

Hardware components communicate with the **Central Processing Unit** or **cpu** over a **bus**. The most common buses today are **usb**, **pci**, **agp**, **pci-express** and **pcmcia** aka **pc-card**. These are all **Plag and Play** buses.

Older **x86** computers often had **isa** buses, which can be configured using **jumpers** or **dip switches**.

B.1.2. /proc/bus

To list the buses recognised by the Linux kernel on your computer, look at the contents of the **/proc/bus/** directory (screenshot from Ubuntu 7.04 and RHEL4u4 below).

```
root@laika:~# ls /proc/bus/
input  pccard  pci  usb
```

```
[root@RHEL4b ~]# ls /proc/bus/
input  pci  usb
```

Can you guess which of these two screenshots was taken on a laptop ?

B.1.3. /usr/sbin/lsusb

To list all the usb devices connected to your system, you could read the contents of **/proc/bus/usb/devices** (if it exists) or you could use the more readable output of **lsusb**, which is executed here on a SPARC system with Ubuntu.

```
root@shaka:~# lsusb
Bus 001 Device 002: ID 0430:0100 Sun Microsystems, Inc. 3-button Mouse
Bus 001 Device 003: ID 0430:0005 Sun Microsystems, Inc. Type 6 Keyboard
Bus 001 Device 001: ID 04b0:0136 Nikon Corp. Coolpix 7900 (storage)
root@shaka:~#
```

B.1.4. /var/lib/usbutils/usb.ids

The **/var/lib/usbutils/usb.ids** file contains a gzipped list of all known usb devices.

```
paul@barry:~$ zmore /var/lib/usbutils/usb.ids | head
------> /var/lib/usbutils/usb.ids <------
#
# List of USB ID's
#
# Maintained by Vojtech Pavlik <vojtech@suse.cz>
```

```
# If you have any new entries, send them to the maintainer.
# The latest version can be obtained from
#  http://www.linux-usb.org/usb.ids
#
# $Id: usb.ids,v 1.225 2006/07/13 04:18:02 dbrownell Exp $
```

B.1.5. /usr/sbin/lspci

To get a list of all pci devices connected, you could take a look at **/proc/bus/pci** or run **lspci** (partial output below).

```
paul@laika:~$ lspci
...
00:06.0 FireWire (IEEE 1394): Texas Instruments TSB43AB22/A IEEE-139...
00:08.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-816...
00:09.0 Multimedia controller: Philips Semiconductors SAA7133/SAA713...
00:0a.0 Network controller: RaLink RT2500 802.11g Cardbus/mini-PCI
00:0f.0 RAID bus controller: VIA Technologies, Inc. VIA VT6420 SATA ...
00:0f.1 IDE interface: VIA Technologies, Inc. VT82C586A/B/VT82C686/A...
00:10.0 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1....
00:10.1 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1....
...
```

B.2. interrupts

B.2.1. about interrupts

An **interrupt request** or **IRQ** is a request from a device to the CPU. A device raises an interrupt when it requires the attention of the CPU (could be because the device has data ready to be read by the CPU).

Since the introduction of pci, irq's can be shared among devices.

Interrupt 0 is always reserved for the timer, interrupt 1 for the keyboard. IRQ 2 is used as a channel for IRQ's 8 to 15, and thus is the same as IRQ 9.

B.2.2. /proc/interrupts

You can see a listing of interrupts on your system in **/proc/interrupts**.

```
paul@laika:~$ cat /proc/interrupts
      CPU0     CPU1
0:  1320048     555  IO-APIC-edge      timer
1:    10224       7  IO-APIC-edge      i8042
7:        0       0  IO-APIC-edge      parport0
8:        2       1  IO-APIC-edge      rtc
10:     3062      21  IO-APIC-fasteoi   acpi
12:      131       2  IO-APIC-edge      i8042
15:    47073       0  IO-APIC-edge      ide1
18:        0       1  IO-APIC-fasteoi   yenta
19:    31056       1  IO-APIC-fasteoi   libata, ohci1394
20:    19042       1  IO-APIC-fasteoi   eth0
21:    44052       1  IO-APIC-fasteoi   uhci_hcd:usb1, uhci_hcd:usb2,...
22:   188352       1  IO-APIC-fasteoi   ra0
```

```
23:   632444      1  IO-APIC-fasteoi   nvidia
24:     1585      1  IO-APIC-fasteoi   VIA82XX-MODEM, VIA8237
```

B.2.3. dmesg

You can also use **dmesg** to find irq's allocated at boot time.

```
paul@laika:~$ dmesg | grep "irq 1[45]"
[ 28.930069] ata3: PATA max UDMA/133 cmd 0x1f0 ctl 0x3f6 bmdma 0x2090 irq 14
[ 28.930071] ata4: PATA max UDMA/133 cmd 0x170 ctl 0x376 bmdma 0x2098 irq 15
```

B.3. io ports

B.3.1. about io ports

Communication in the other direction, from CPU to device, happens through **IO ports**. The CPU writes data or control codes to the IO port of the device. But this is not only a one way communication, the CPU can also use a device's IO port to read status information about the device. Unlike interrupts, ports cannot be shared!

B.3.2. /proc/ioports

You can see a listing of your system's IO ports via **/proc/ioports**.

```
[root@RHEL4b ~]# cat /proc/ioports
0000-001f : dma1
0020-0021 : pic1
0040-0043 : timer0
0050-0053 : timer1
0060-006f : keyboard
0070-0077 : rtc
0080-008f : dma page reg
00a0-00a1 : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
02f8-02ff : serial
...
```

B.4. dma

B.4.1. about dma

A device that needs a lot of data, interrupts and ports can pose a heavy load on the cpu. With **dma** or **Direct Memory Access** a device can gain (temporary) access to a specific range of the **ram** memory.

B.4.2. /proc/dma

Looking at **/proc/dma** might not give you the information that you want, since it only contains currently assigned **dma** channels for **isa** devices.

```
root@laika:~# cat /proc/dma
1: parport0
4: cascade
```

pci devices that are using dma are not listed in **/proc/dma**, in this case **dmesg** can be useful. The screenshot below shows that during boot the parallel port received dma channel 1, and the Infrared port received dma channel 3.

```
root@laika:~# dmesg | egrep -C 1 'dma 1|dma 3'
[   20.576000] parport: PnPBIOS parport detected.
[   20.580000] parport0: PC-style at 0x378 (0x778), irq 7, dma 1...
[   20.764000] irda_init()
--
[   21.204000] pnp: Device 00:0b activated.
[   21.204000] nsc_ircc_pnp_probe() : From PnP, found firbase 0x2F8...
[   21.204000] nsc-ircc, chip->init
```

Appendix C. License

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

The "Invariant Sections" are certain Secondary Sections whose titles

```
are designated, as being those of Invariant Sections, in the notice
that says that the Document is released under this License. If a
section does not fit the above definition of Secondary then it is not
allowed to be designated as Invariant. The Document may contain zero
Invariant Sections. If the Document does not identify any Invariant
Sections then there are none.

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

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

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

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

The "publisher" means any person or entity that distributes copies of
the Document to the public.

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

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

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either
```

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

```
   * B. List on the Title Page, as authors, one or more persons or
entities responsible for authorship of the modifications in the
Modified Version, together with at least five of the principal authors
of the Document (all of its principal authors, if it has fewer than
five), unless they release you from this requirement.
   * C. State on the Title page the name of the publisher of the
Modified Version, as the publisher.
   * D. Preserve all the copyright notices of the Document.
   * E. Add an appropriate copyright notice for your modifications
adjacent to the other copyright notices.
   * F. Include, immediately after the copyright notices, a license
notice giving the public permission to use the Modified Version under
the terms of this License, in the form shown in the Addendum below.
   * G. Preserve in that license notice the full lists of Invariant
Sections and required Cover Texts given in the Document's license
notice.
   * H. Include an unaltered copy of this License.
   * I. Preserve the section Entitled "History", Preserve its Title,
and add to it an item stating at least the title, year, new authors,
and publisher of the Modified Version as given on the Title Page. If
there is no section Entitled "History" in the Document, create one
stating the title, year, authors, and publisher of the Document as
given on its Title Page, then add an item describing the Modified
Version as stated in the previous sentence.
   * J. Preserve the network location, if any, given in the Document
for public access to a Transparent copy of the Document, and likewise
the network locations given in the Document for previous versions it
was based on. These may be placed in the "History" section. You may
omit a network location for a work that was published at least four
years before the Document itself, or if the original publisher of the
version it refers to gives permission.
   * K. For any section Entitled "Acknowledgements" or "Dedications",
Preserve the Title of the section, and preserve in the section all the
substance and tone of each of the contributor acknowledgements and/or
dedications given therein.
   * L. Preserve all the Invariant Sections of the Document,
unaltered in their text and in their titles. Section numbers or the
equivalent are not considered part of the section titles.
   * M. Delete any section Entitled "Endorsements". Such a section
may not be included in the Modified Version.
   * N. Do not retitle any existing section to be Entitled
"Endorsements" or to conflict in title with any Invariant Section.
   * O. Preserve any Warranty Disclaimers.

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

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

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version. Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity. If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
```

you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

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

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

```
8. TRANSLATION

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

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

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document
except as expressly provided under this License. Any attempt otherwise
to copy, modify, sublicense, or distribute it is void, and will
automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license
from a particular copyright holder is reinstated (a) provisionally,
unless and until the copyright holder explicitly and finally
terminates your license, and (b) permanently, if the copyright holder
fails to notify you of the violation by some reasonable means prior to
60 days after the cessation.

Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License. If your rights have been terminated and not permanently
reinstated, receipt of a copy of some or all of the same material does
not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

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

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

that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

User Guide to Using the Linux Desktop

Nah Soo Hoe and Colin Charles

Published by
the United Nations Development Programme's
Asia-Pacific Development Information Programme (UNDP-APDIP)
Kuala Lumpur, Malaysia

Web: http://www.apdip.net/
Email: info@apdip.net

Preface

This user guide is meant as an introductory guide for a user to use a modern personal computer (PC) running the Linux operating system. The main aim is to provide a self-learning guide on how to use a modern Linux desktop system. It assumes that the user has no prior knowledge of Linux or PC usage.

After going through the guide the reader should be in a position to start using a Linux desktop for both personal and office use. In particular she should be able to:

- access the Internet and use the WWW and Internet email
- manipulate and manage files, folders and the file system
- produce and print simple documents and presentation materials
- advance to become a power user by further self-learning and exploring

Linux has many distributions and sometimes the programs or tools used to perform a certain function can vary from distribution to distribution. This guide tries to be as generic as possible in the description of the features and functionalities. However, in some cases, especially some of the GUI desktop configuration tools, there is no really independent generic tool that can be used and each distribution has its own tool. In such cases, we have tried to illustrate their usage using Fedora Linux.

This guide was written on a Fedora Linux system and as such many of the screen shots reflect this. However, this should not be construed as an endorsement of this distribution of Linux over the others on the part of the authors.

Nah Soo Hoe and Colin Charles

July 2004

Chapter 1: Getting Started

In order to start using your system you will usually have to perform what is known as a user login. This procedure is necessary to identify yourself to the system. The system allows multiple users to use it concurrently and so it has to be able to identify a user in order to grant them the necessary privileges and rights to use the system and applications. Each user upon successful login will be assigned to his home directory (folder).

Some systems may have been set up so that upon power-on, a default user is logged in automatically. If this is so, then you will not be presented with a login screen or prompt as described in the section below. You may skip on to the section entitled "Basic Navigation using the Mouse on the Desktop".

Note:
*There is a special user called the **root** or **superuser** (this user is usually created during the system installation) which has unlimited access and rights to all the system files and resources. You only need to login as root if certain system level administrative tasks are to be carried out. Otherwise there is usually no need for a normal user to login as root. This is to prevent accidentally damaging the system by deleting or modifying important system files.*

Logging In

Depending on how you have set up your system, you will either have a graphical login screen or a text-based login prompt for you to perform the login process.

Fig. Graphical Login Screen

```
Fedora Core release 1
Kernel 2.4.22-1 on an i686

localhost login:
```

Fig. Text-based Login Prompt

To login, enter the username followed by the ENTER key and when the password prompt appears, enter the password followed by the ENTER key.

Starting the Graphical Desktop

If you have logged in from the graphical login screen, the graphical desktop will be started automatically for you. The graphical desktop presents a Graphical User Interface (GUI) for the user to interact with the system and run applications. If you have used the text-based screen login, you will have to start the graphical desktop manually by entering the command **`startx`** followed by the ENTER key.

```
[anita@localhost anita]$ startx
```

Fig. Starting the Graphical Desktop

Note:
*The graphical desktop that we will be using throughout most of this guide is called the **GNOME Desktop**. There is another desktop environment in popular use on Linux systems – the **KDE Desktop**. There is some coverage of KDE later, comparing the similarities and differences between GNOME and KDE although we will not be covering the KDE desktop in detail.*

For the rest of this user guide, when we refer to the graphical desktop or Desktop we shall be talking about the GNOME Desktop unless stated otherwise.

Using the Mouse on the Desktop

Proper usage of the mouse is essential in order to have a rewarding and productive experience on the graphical desktop. Most Linux graphical desktops are designed for use with a 3-button mouse. If you are using a 2-button mouse, during installation, it should have been configured to emulate the middle-button of a 3-button mouse by pressing both buttons simultaneously.

To click on a mouse the left button is depressed. (A mouse configured for a left-handed user will will need to have its right button depressed.) The right (or left button for a left-handed mouse) and middle buttons are usually used to invoke special or specific features of the GUI and instructions to do so will be explicitly given.

The term "clicking on the mouse" means that you click on the mouse once.

The term "double clicking" means that you click twice in succession on the mouse.

To "select" an item means clicking it once with the mouse.

The term "drag and drop" means that you will have to click on an item and while continuing to hold the mouse button down, drag the item to another place and on reaching its destination drop it by releasing the mouse button.

Exercises

1. Open the home folder from the Desktop.
2. Close the home folder window by clicking on the close window button at the top right-hand corner.
3. Right-click on the trash icon, view its properties and close it back.
4. Move the trash icon to another location on the desktop by using the mouse to drag it to the new location and releasing the mouse.

Main Components of the Desktop

The figure below shows a typical view of the graphical Linux desktop.

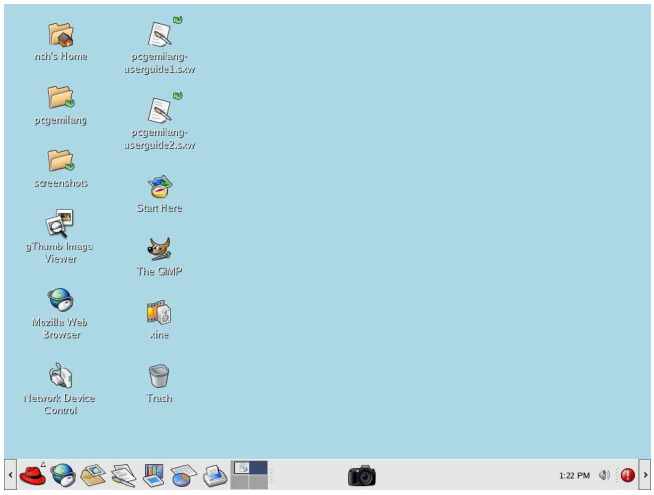


Fig. A Typical Linux Desktop

There are 3 main components on the desktop:

- the Menu System
- the Panel
- the Desktop itself

The Menu System

The main menu can be accessed by clicking on the Main Menu button located at the extreme lower left hand corner of the desktop. This may be portrayed by one of several icons depending on the desktop environment used.

Generic GNOME Main Menu button

Fedora/Red Hat Main Menu button

Clicking on this will bring up the Menu System as shown in the figure below.

Fig. The Menu System

From the Menu System you can start many of the applications installed on your Linux system. Note that the Menu System consists of a Main Menu panel and sub-menu panels. Each entry in the Menu System which has an arrow on its right means that it is an entry point to a sub-menu, and there can be sub-menus within each sub-menu. In this way applications in the Menu System can be organised and categorised for easy reference and access.

To access a the sub-menu associated with a menu entry, move the mouse and rest it on the menu entry in question and a sub-menu panel will appear.

Fig. Main Menu and Sub-menu

Clicking once on a menu entry will cause an application associated with it to be launched, i.e. executed.

The Panel

The long bar across the bottom of the desktop screen is called the Panel. The Panel contains the Main Menu icon, the application launcher icons, a notification area and applets.

Fig. The Panel

Installed by default are several application launcher icons on the Panel. Clicking on one of these will run an application. Commonly accessed applications can be added to the Panel and those that are less frequently used can be taken off.

The notification area holds alert icons so that the user can be alerted to critical messages.

Applets are small applications that run on the Panel. These usually perform useful and informative tasks like setting the sound level of the soundcard, monitoring whether the system software needs an update, etc. By default the following applets are run.

The Workspace Switcher

The graphical desktop can be regarded as a workspace drawing an analogy with the working area on a real physical tabletop. Programs are run, documents displayed and files opened on the workspace. To cut down on workspace clutter and to enable the user to organise his workspace more efficiently, the graphical desktop environment allows the usage of multiple workspaces. Each workspace can be considered as a virtual desktop.

Fig. The Workspace Switcher

By default the user has 4 desktop workspace areas to work on. The workspace switcher represents each workspace as a small square and shows the applications running in each of them. To access a workspace click on the square with the mouse.

The Taskbar

The Taskbar applet is located next to the workspace switcher and shows the titles of all the running applications in a virtual desktop (a workspace).

The Desktop Space

The Desktop space refers to the rest of the screen. It contains icons which are graphical representations of short-cuts to application launchers, file folders, files and peripheral devices like floppy disks, CD-ROM drives and printers. Double-clicking on an icon representing an application will launch or execute the application. Commonly used applications and/or files/folders are usually placed on the desktop space.

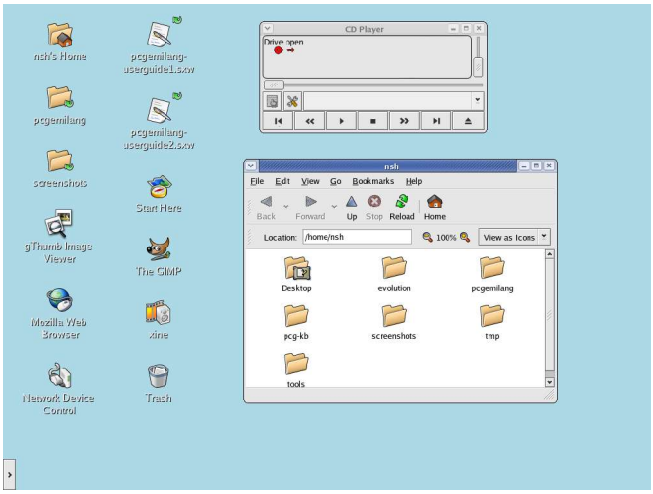


Fig. The Desktop Space

Using the Window Manager

An interactive application that is run on the graphical desktop, is usually displayed inside a window. This window can be accessed and manipulated using the window manager.

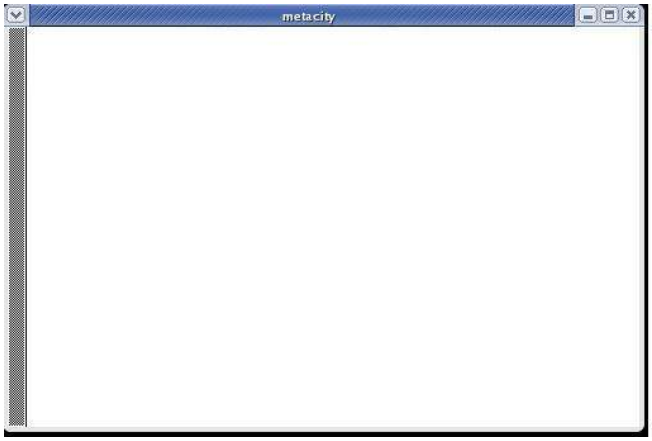


Fig. An Application Window Frame

The Titlebar

When an application is started an application window opens and a frame (see figure above) is placed around the window of the application. The top edge of the frame has a titlebar that contains the title of the application.

Fig. The Window Titlebar

At the left hand corner of the titlebar is the Window Menu button. Clicking on this button will open up the Window Menu.

Fig. The Window Menu

You can perform operations such as minimise, (un) maximise, roll up, move and resize the window from this menu. At the right edge of the titlebar there are three buttons which allow you to minimise, (un)maximise and close the window.

To perform an operation in the Window Menu, open the menu by clicking on the Window Menu button and then select the desired operation.

Manipulating the Window

Some basic instructions to manipulate the windowing system are given here. More details can be obtained by running the "Help" application on the Main Menu. After running the Help application choose:

Desktop --> Windows --> Manipulating Windows

Focus

In order for a window to receive input from the mouse and the keyboard it has to be the window "in focus". Only one window can have focus at a time. Windows that are not in focus have their titlebars greyed-out. To focus on a window place the mouse on the titlebar and click on it. (You can actually click anywhere within the window, but it

is safer to click on the titlebar since there is no possibility of accidentally clicking on some item selection or functionality of the application running within the window.) If the window is not visible you can click on the taskbar on the Panel at the bottom of the Desktop to bring into focus the window you want.

Maximise and Minimise

Maximising a window means that the window is expanded to cover the whole desktop while minimising it means that the window is taken off the desktop and it appears as an icon in the taskbar on the Panel.

You can maximise a window which has focused from the Window Menu or by clicking on the maximise button on the titlebar. You can similarly minimise the window. If the window does not have focus bring it into focus first by clicking on the taskbar on the Panel.

Resize

You can resize a window, i.e. change its size, by placing the mouse at an edge of the window and then drag the window to the desired size.

Move

To move a window to another place on the desktop, place the mouse on the titlebar and drag the window to the desired location.

Roll Up and Unroll

To roll up a window allows you to “roll up” the window frame until what can be seen of the window is just the the titlebar. You can perform this operation from the Window Menu.

To restore the window back to its original shape, select the unroll option from the Window Menu.

Close

You can close a window which has focused from the Window Menu or by clicking on the close window button on the titlebar. If the window does not have focus bring it into focus first by clicking on the taskbar on the Panel.

Note: Closing the window will terminate your application.

Move to Workspaces

The Window Menu can be used to move the application currently opened to another workspace or to all the workspaces.

Exercises

1. Open your home folder from the Main Menu. Perform the following operations on the opened window:

- maximise it
- minimise it
- resize it
- roll it up
- unroll it
- move it to another position on the Desktop
- place it on workspace 3
- close the window

Ending the Session

To end this chapter, you can exit your session on the desktop by performing what is known as a logout.

If you do not want to use the system anymore, you can turn it off by performing a system shutdown.

Logout

When you have finished working on the system, you will need to logout. Logging out will inform the system that you are no longer using the system's resources. All the files opened and programs run by you will be closed and/or stopped unless you have specifically informed the system to keep them open or running for you.

To logout, at the Main Menu select:

Main Menu --> Logout

and at the dialog window select logout and click OK

Fig. Logout Dialog Window

Shutting Down the System

When you have finished using the computer and want to power it off, you will have to perform a system shutdown.

Note:
It is very important that a proper system shutdown is performed. You should not just turn off the power switch of the computer to shut it down. Failure to observe this may lead to system software and data corruption and failure.

To shutdown, at the Main Menu, perform a logout. Then at the graphical login screen select the “Shut down” option at the bottom of the screen. Sometimes (depending on the login screen chosen) the Shutdown option is available as a sub-option under the Actions option at the bottom of the screen.

Alternatively on some systems, you may be able to perform a shut down by performing a log out operation as described in the previous section but instead of selecting “Log out”, select the “Shut down” option from the Logout dialog.

Chapter 2: Using the Desktop

Basic Desktop Customisation

The desktop can be customised to your preferences and tastes in a variety of ways. Here we will explore some basic customisations. More details can be obtained by running the "Help" application on the Main Menu. After running the Help application choose:

Desktop --> Basic Preferences

Desktop Background

The background image displayed on the desktop can be changed by running the Background application from the Main Menu (this is in the Preferences sub-menu).

Main Menu --> Preferences --> Background

To change the background image, click on the square marked "Select picture". An image selector dialog is displayed. Choose an image from the dialog. If you want to choose an image from another directory, click the Browse button. When you have chosen an image, click OK.

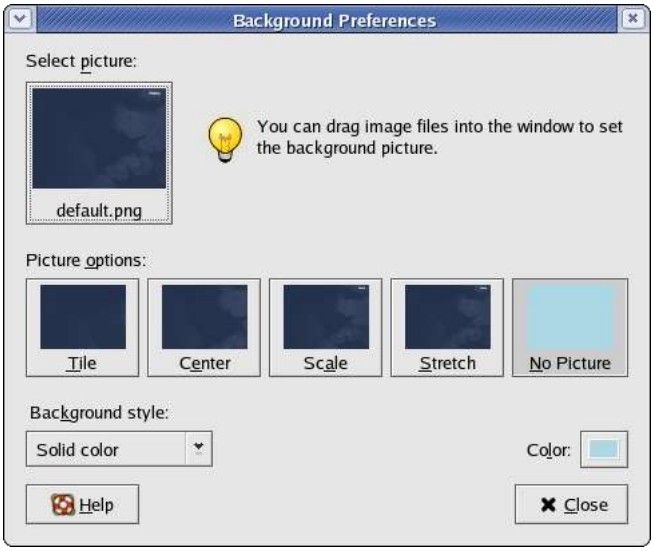


Fig. Selecting a New Background image

Desktop Themes

The desktop theme specifies the visual appearance of the panels, applets, and windows. The desktop theme may also specify the appearance of interface items in applications. For example, the theme affects the appearance of buttons, scrollbars, check boxes, and so on in the applications.

The theme used by the Desktop can be changed by running the Theme application from the Main Menu (this is in the Preferences sub-menu).

Main Menu --> Preferences --> Theme

An theme selector dialog is displayed. To change the theme, click on a new theme. The screenshot below shows a sample of some of the themes which may be available. Note that the actual themes available can vary from system to system.

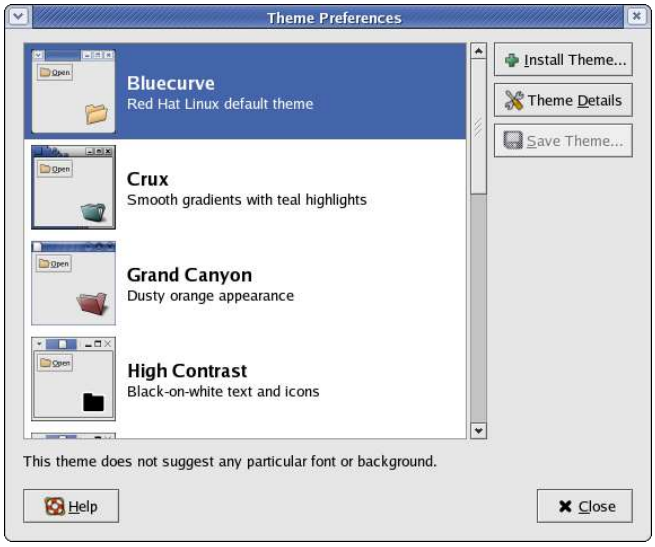


Fig. Selecting a New Theme

Default Fonts

The default fonts used to display applications and the desktop background can be changed by running the Fonts application from the Main Menu (this is in the Preferences sub-menu).

Main Menu --> Preferences --> Font

A selector dialog for the application, window title, dialog and terminal fonts are displayed. To change the font for each category of usage, click on the space listing the font.

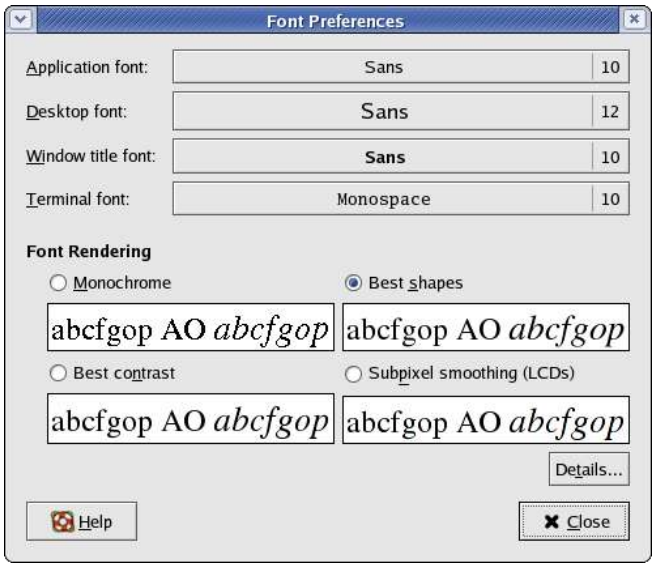


Fig. Selecting New Fonts

Menus and Toolbars

The Menus & Toolbars tool is used to customise the appearance of menus, menubars, and toolbars. Again this can be run from the Preferences sub-menu in the Main Menu.

Main Menu --> Preferences --> Menus & Toolbars

Experiment on each of the settings to get the ones preferred.

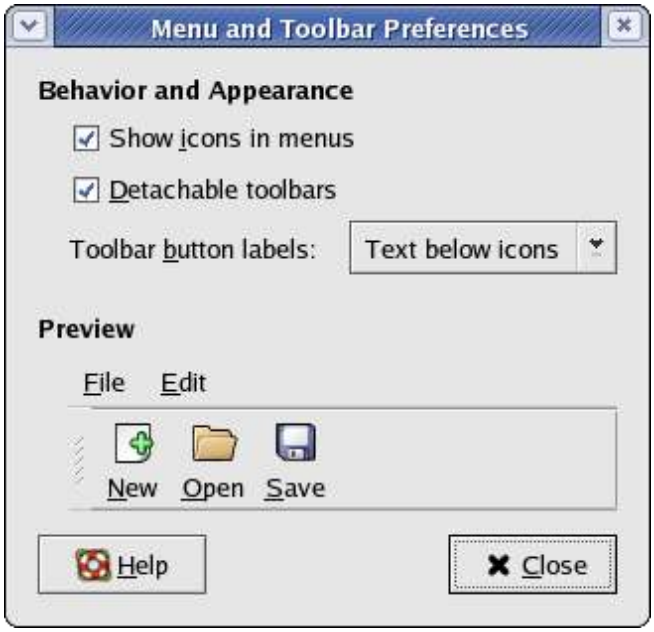


Fig. Customising Menus and Toolbars

Exercises

Experiment with each of the following desktop preferences:

- background
- themes
- fonts
- menus and toolbars

Using the Desktop

The Desktop provides a useful metaphor for a modern office work environment. So on the Desktop we find that the applications that we want to run can easily be started or launched and the information and data that we need can easily be located.

Running Applications

Applications that we can run from the Desktop are to be found either from the Main Menu (and sub-menus therein), or as icons on the Panel and the Desktop itself.

To run an application from the Main Menu, open up the menu (or sub-menu) and click on the application listed in the menu bar.

To run an application from the Panel, click once on the icon representing the application.

To run an application from the Desktop itself, double-click on the icon representing the application.

Sometimes it is more handy to have the application as an icon on the Panel or Desktop where you can launch it more easily by just clicking on it, rather than in the Main Menu, especially if it is buried deep within several sub-menus. To achieve this open the Main Menu and select the application item listed in it by clicking with the right mouse button. Click on the selection “Add this launcher to panel” and a copy of the application icon will be placed on the Panel.

To make a copy of this on the Desktop, you can drag the icon from the Panel over to the Desktop.

To delete an application icon from the Panel right-click on it and select the “Remove from Panel” option.

To delete an application icon from the Desktop, right-click on it and select the “Move to Trash” option.

The Panel

The Panel houses many useful utilities called applets. Applets are small applications that run on the Panel. As discussed in Chapter One, by default, the taskbar and the workspace switcher applets are placed and run on the Panel. Other useful applets that may be placed on the Panel include:

- clock
- sticky notes
- volume control

Placing an Applet on the Panel

To place an applet on the Panel, move the mouse over to an empty space on the Panel and right-click it. Select the item "Add to Panel" and from the sub-menus select the applet to place on the Panel. For example to place the "sticky notes" applet, select:

Add to Panel --> Accessories --> Sticky Notes

To prevent accidental removal of an applet, you can lock it on the Panel by right-clicking on its icon and selecting "Lock".

Removing an Applet from the Panel

To remove an applet, right-click on the applet icon and select "Remove from Panel". If the applet is locked, you will have to unlock it first by right-clicking on the icon and selecting "Unlock".

Configuring the Panel

To change the properties of the Panel, right-click on the Panel and select "Properties".

Fig. General Properties of the Panel

From the general properties menu you can change the orientation, size and (un)hide the Panel.

From the background properties menu, you can change the colour of the Panel as well as its visual appearance.

Launchers

Launchers allow the user a quick way to access specific resources on the system. For example if a user needs to access a specific file in one of the folders often, he can create a launcher to run an application to open the file and this launcher can be placed it on the Desktop. In this way

the resource (the file) can be accessed very quickly by just double-clicking on it.

To create a launcher on the Desktop, right-click on an empty area on the Desktop and select the item "Create Launcher". Enter the Name and the Command to run and if you want you can select an icon for it by clicking on the icon button.

Fig. Creating a Launcher

Exercises

1. The Mozilla web browser is a popular web browser application for Linux systems. It has an easily recognisable icon – a red dinosaur head.

- Locate the Mozilla application in the Main Menu and place it on the Panel as well as the Desktop.

- Launch the Mozilla application from:

    - the Desktop
    - the Main Menu
    - the Panel

2. Configure the Panel so that it has the following properties:

- a background colour of solid blue
- contains the Geyes and Screenshot (locked) applets

3. Create a launcher to run the text editor command "gedit" on the Desktop. Use an appropriate icon for the text editor.

Setting the Date and Time

It is important that the date and time are set correctly in your system. This will make it easier to manage the system resources and files and also aid in troubleshooting any problems. To set the date and time, run the Date & Time application tool from the System Settings sub-menu under the Main Menu.

Main Menu --> System Settings --> Date & Time

Alternatively you can right-click on the clock (time) applet display on the Panel and choose “Adjust Date & Time”.

A dialog box asking for the root password will appear if you are not logged in as root. This is because the system date and time are important system parameters and so only the system administrator or root is allowed to do it. After entering the correct root password, you can change the date using the displayed calender and the time in the boxes provided. Select the Time Zone tab and check to see that the timezone selected is correct. If not, correct it by selecting the appropriate timezone region. Usually the system uses the local time, so do not select the “System clock uses UTC” checkbox.

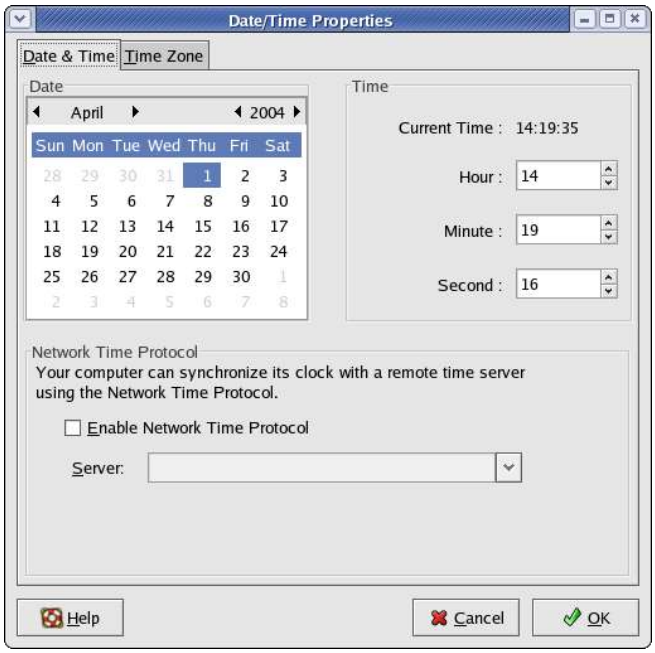


Fig. Setting the Date and Time

Click on the OK button after all is done to enable the new date and time. You can check for the new date and time by resting your mouse over the clock (time) applet display on the Panel at the bottom of the screen.

Exercises

1. Practice setting the date and time.

2. Practice changing the appearance of the desktop clock

Chapter 3: Files and Folders

The File System

One of the most powerful features of a modern computer system is the ability to store data in a form which can be easily retrieved and transported or copied across to other computer systems or media. Data created and accessed by a user in the computer is stored in what is called a *file*. This concept of an electronic file to store electronic data mimics the physical world usage of a file to store data written on paper. This mimicry is carried further by the organisation of these electronic files into electronic folders or directories. Like a physical folder, an electronic folder can contain very many files. A folder may also have sub-folders or sub-directories.

Note:
We shall be using the terms folder and directory interchangeably, unless stated otherwise.

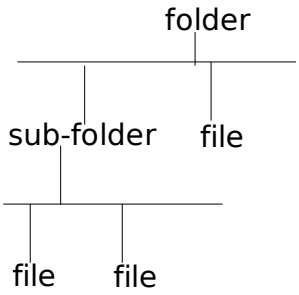


Fig. The Hierarchical File System

Files and folders can be created, copied, moved (i.e. transferred) and deleted. The folders themselves are organised in a hierarchical manner starting at the root of the file system. Each user is given a home directory and upon logging in, the user is placed in his home directory.

Ownerships and Permissions

As the system is set up to handle multiple users concurrently, it needs to have in place mechanisms with which it can control the security and privacy of the file system. It needs to be able to control access to the file system resources for each individual user. In addition it also has to be able to control access at a group level i.e. users who belong to the same group can be given certain privileges with respect to the file system operations. To be able to perform these features, associated with each file or folder are the categories:

- owner
- group
- others

and the file permissions:

- read (r)
- write (w)
- execute (x)

and the directory permissions:

- read (r)
- write (w)
- access (x)

Associated with a file are the categories owner, group and others and the permissions which these have on the file. The userid of the user who creates a file by default becomes the owner of the file. Userids on the system are assigned to one or more groups. When a userid is created on the system, a group which has the same name as the userid is also created and the new userid belongs to this group. By default this group (which has the same name as the file owner) is associated with the file. All of the other userids on the system which are not the owner of the file or belong to the group associated with the file, are placed in the category others .

By default the owner has read and write permission for a non-executable file and read,write and execute permission for an executable file. The group associated with the file has read permission for a non-executable file and read, execute rights for an executable file. The others group has read only permission for non-executable file and read, execute rights for an executable file.

The rights and ownership concepts described above apply to directories too. However since a directory cannot be executed, access rights is substituted for execution rights. Access to a directory means that the userid with the appropriate permission can descend into the directory (i.e. change directory to it).

The File Manager application described below may be used to view and modify the ownership and permissions of a file or folder. Only the owner of a file/folder can change its permissions and only the *superuser* or *root* can change the ownership of a file/folder.

Using the File Manager

As it is possible for a user to create and store hundreds and thousands of files and folders, a File Manager is needed to assist the user to manage and manipulate these files and the file system on which it resides.

In this section a brief description of the File Manager and how to use it is given. For more details you should consult the "Help" application on the Main Menu. From the Help application choose:

Desktop --> Nautilus File Manager

To start using the File Manager double-click on the home directory icon on the desktop. (This may be named "*username's* Home" where username is the username of the user currently logged in.)

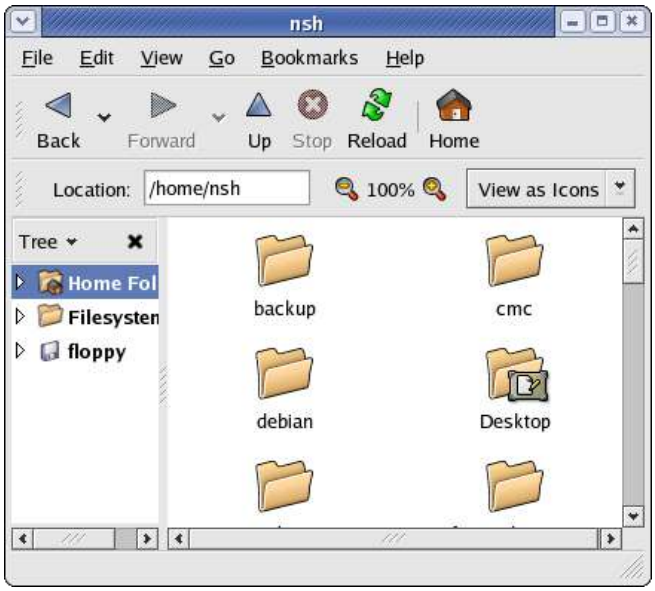


Fig. The File Manager

By default, the File Manager window consists of a side pane and a main view pane. At the top of the window just under the window titlebar, are the menu bar and the location bar.

The view pane displays the files and folders contained in the current directory that the user is in. These can be displayed

as icons (default) or changed to display them as a listing of filenames.

The side pane contains an icon that represents the current file or current folder. The side pane also contains information about the current file or current folder. A hierarchical (tree) view of the file system on the computer can be obtained from the side pane. By navigating through this, you can access files and folders outside your home directory (provided of course that you have the permission to do so).

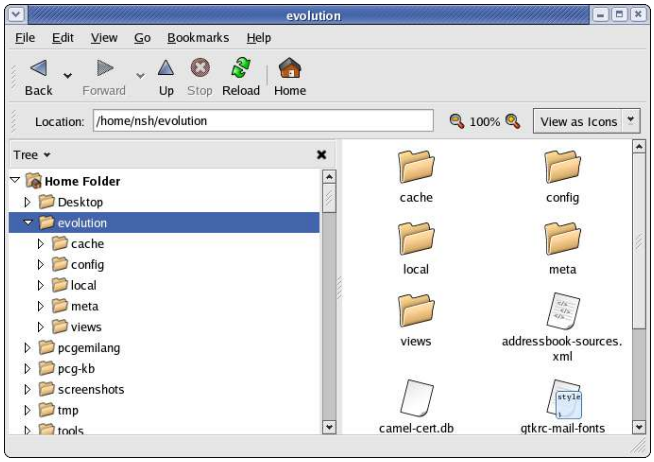


Fig. The Side Pane with a Tree View of Folders

Accessing Files and Programs

From the view pane of the File Manager, you can open files as well as run (launch) executable programs by double-clicking on the icon representing them.

Creating New Folders

To create a new folder under the folder you are currently in, move the mouse to the menu bar at the top (just beneath the titlebar of the window) and click on File and then click on Create Folder. A new folder will appear on the view pane and you will have to type in the name of the new folder.
Copy Files and Folders

To copy a file, click on the file in the view pane to select it. Then select from the menu bar at the top,

Edit --> Copy File

Next open up the folder in which you want to copy the file to and then select from the menu bar at the top,

Edit --> Paste Files

Another way to copy a file is to right click on the file icon and then select “Copy File”. Then navigate to the icon of the folder where you want the copy to be placed in and then right click on the folder icon and select “Paste Files into Folder”.

The procedures above can be done with folders too.

To copy more than one file or folder at a time, select multiple files/folders by holding down the CTRL key while clicking on the files or folders.

Moving Files and Folders

Moving a file or folder is different from copying in that a copy of the file/folder is not made, i.e. only one copy of the file/folder exists, and the file/folder is transferred from one folder to another.

To move a file, click on the file in the view pane to select it. Then select from the menu bar at the top,

Edit --> Cut File

Next open up the folder in which you want to move the file to and then select from the menu bar at the top,

Edit --> Paste Files

Another way of moving a file is to drag and drop the file into the destination folder.

The procedures above can be done with folders too.

To move more than one file or folder at a time, select multiple files/folders by holding down the CTRL key while clicking on the files or folders.

Renaming Files and Folders

To rename a file, click on the file in the view pane to select it. Then select from the menu bar at the top,

Edit --> Rename

and then type in the new name.

Alternately you can also right-click on the file and then select “Rename” .

The procedures above can be done with folders too.

Deleting Files and Folders

To delete a file, click on the file in the view pane to select it. Then select from the menu bar at the top,

Edit --> Move to Trash

Alternately you can select the file and then use the DELETE key on the keyboard to delete the file. This has the same effect as above of moving the file to the Trash folder.

It is still possible to salvage a deleted file from the Trash. To do this double-click on the Trash icon on the desktop to open up the Trash folder. Then you can move the file you want to salvage to the desired folder. Note that if you delete the file from the Trash then it cannot be recovered anymore.

The procedures above can be done with folders too.

To delete more than one file or folder at a time, select multiple files/folders by holding down the CTRL key while clicking on the files or folders.

Viewing and Modifying the Permissions of a File or Folder

To view the owner and group of a file/folder and/or to modify its permission settings, select the file/folder and select from the menu bar at the top,

File --> Properties

Click on the Permissions tab. The owner and group of the file/folder are displayed as well as the associated permissions.

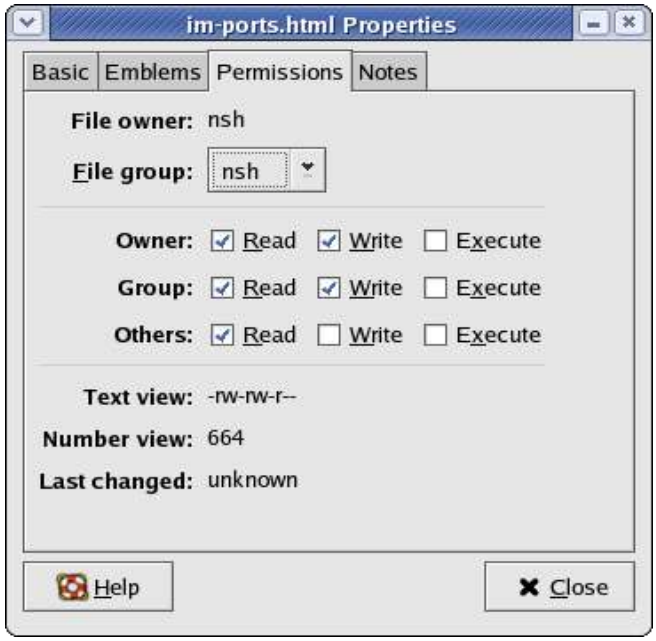


Fig. Ownership and Permission Properties of a File

To change the permissions, modify the check boxes accordingly to the new permissions desired. Click on Close to apply the changes.

Note:
It is recommended that you do not modify the default file/folder permission settings unless you know what you are doing. This is to minimise the risk of compromising the security of the files and folders.

The Trash Can

The trash can icon on the desktop contains files and folders which you have deleted using the File Manager or an application run from the desktop. As noted above it is a special folder in which is still possible to move back the deleted items to another folder so as to "undelete" them. However over time, the trash folder will be filled up as many files and folders get deleted through usage. So it is a good idea to periodically empty or delete the items found in the trash.

To empty the trash, right-click on the trash icon and select "Empty Trash". Otherwise you can open the trash by double-clicking on it and then select to delete all the items in the trash folder using the File Manager functions described above.

Exercises

1. Use the File Manager to perform the following:

- Create a new folder called *testdir* under the home directory
- Copy the following files into the folder *testdir:*
- /etc/services, /etc/hosts
- Access the *testdir* folder:
- Open the two files there to view their contents, then close them
- Rename the file *hosts* to *hosts.backup* and *services* to *services.copy*
- Move the file *hosts.backup* to the /usr/tmp directory
- Delete the file *services.copy*
- Move up to the home directory
- Delete the folder *testdir*
- Launch the application *xeyes* found in the folder */usr/X11 R6/bin*

2. Check the trash can and restore the deleted file(s) there

Using A Text Editor

Very often it is necessary to use the computer to input some text or to modify the data in a text file. While a full-fledged word processor like OpenOffice.org's Writer may be used, it can be an overkill since the sophisticated features and formatting available with a word processor are not needed in many cases. A text editor can be used instead. The Linux system comes installed with many text editors. In this section we shall be looking at how to use a text editor which comes with the GNOME Desktop - gedit.

Note:
A text file here refers to a a file which contains pure text printing characters only. Some types of files e.g. those created by a word processor, while appearing to be text-only actually contain other non-printing characters and hence are not pure-text files.

Starting Gedit

Gedit may be started from the Main Menu,

Main Menu --> Accessories --> Text Editor

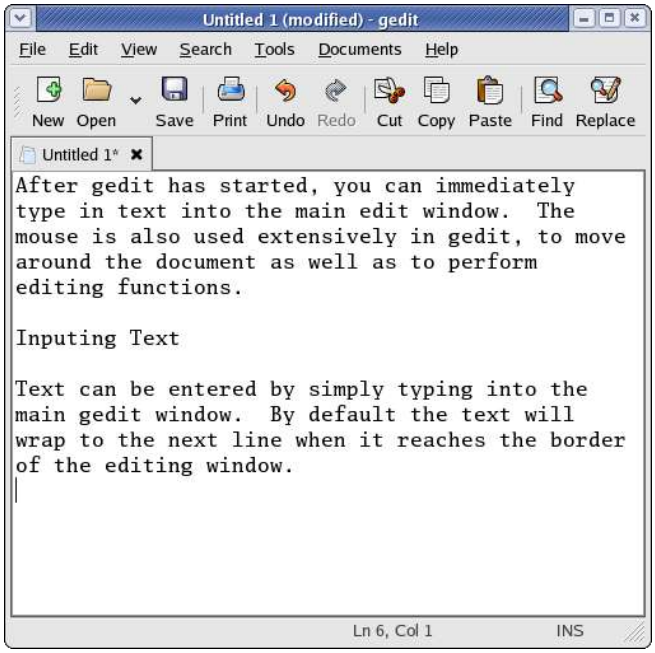


Fig The Gedit Text Editor

After gedit has started, you can immediately type in text into the main edit window. The mouse is also used extensively in gedit, to move around the document as well as to perform editing functions.

Inputing Text

Text can be entered by simply typing into the main gedit window. By default the text will wrap to the next line when it reaches the border of the editing window.

Marking Text

The mouse may be used to mark a block of text by clicking on the beginning of the block and dragging the mouse to the end of the block before releasing it.

Deleting Text

Text can be deleted one character at a time by moving the mouse to the character in question and using the DELETE and BACKSPACE keys in the normal fashion.

If there is a lot of text to be deleted, a more efficient way to delete text is to mark it first and then press the DELETE key on the keyboard.

Copy, Cut and Paste

Text can be edited by marking it first and then selecting Cut, or Copy from the buttons at the top. "Cutting" will result in the marked text being copied into a storage buffer and then deleted from the editing window, while "Copying" will result in just a copy of the text being copied into the storage buffer without the deletion of the original text. The text cut/copied in this way can then be pasted into another part of the document by moving the mouse to the desired location and selecting the Paste button at the top.

Another way of invoking these functions is to click on the Edit option on the top menu bar and selecting the desired function, e.g.

Edit --> Cut
Edit --> Copy
Edit --> Paste

Undo

After performing an editing function, e.g. deleting a block of text, you can undo the action by clicking on the Undo button at the top. The Undo function can also be invoked using the Edit menu selection at the top.

Saving Text

The text typed in can be saved by clicking on the File option on the main menu bar at the top and selecting,

File --> Save As ...

Move to the folder you want to save the file in and enter the name of the file to save.

Find and Replace

The Find button can be used to locate a text string. By default this search is not case sensitive and will also locate the text if it is part of a word. These default options can be changed from the Find dialog.

The Replace button can be used to locate a text string and replacing it with another. Again the default options of non-case sensitive and matching part of a word can be changed.

The Find and Replace functions can also be performed using the Search menu selection at the top.

Opening A File

You can edit a text document already saved on the system by opening it. From the main menu at the top choose,

File --> Open

and select the file to open.

Preferences

Preferences can be set from the Preferences menu selection,

Edit --> Preferences

The Preferences you can set include the font, colour and point size to be used, tab spacing, autowrapping, and autoindenting.

Help

More information on how to use gedit can be obtained by selecting Help from the menu at the top.

EXERCISES

1. Use the gedit text editor to type in the first page of this guide. Save the document in a file on your home folder.

2. Open the saved file and edit it by:
   - deleting the second paragraph on page one
   - adding the statement “End of Chapter” to the last page
   - save the edited document as a new file

Chapter 4: Using Common Mass Storage Peripherals

Besides the main components which make up the modern PC, there are many optional peripherals which can be attached to it and when these are properly utilised they can contribute to make the computing environment and experience more convenient, productive and pleasant. The more common peripherals include:

- printer
- scanner
- mass storage devices

In this chapter we shall be looking at how to access and use the common peripheral mass storage devices,

- floppy disk drive
- CD-ROM drive
- USB mass storage device
- CD-RW drive

Setting up and using a printer and scanner will be covered in the next chapter.

Note:
The method for accessing the mass storage peripherals described below are based on GNOME version 2.4. A new version of GNOME, version 2.6, came out while this guide is being written. In GNOME 2.6 the right-clicking on the Desktop and selecting Disks method has been deprecated. The new method is to use the Computer icon located on the Desktop. Double-click on the Computer icon, and it will display all the system devices as well as the filesystem.

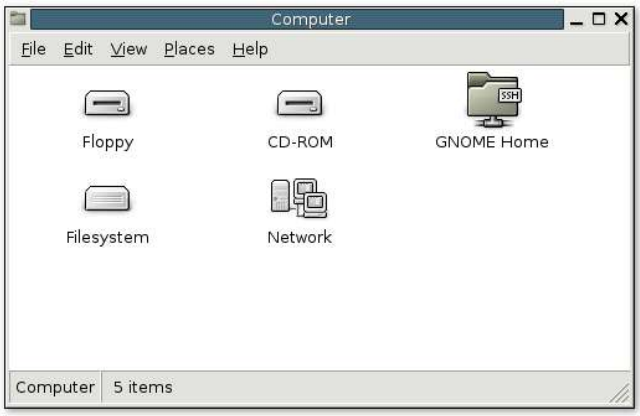


Fig. Accessing System Devices and Filesystem in GNOME 2.6

The Floppy Disk Drive

While most of the work is done using the hard disk or drive and system and user data are stored on the hard drive, sometimes you may want to transfer or copy files to another PC. One convenient way to perform this provided that the file sizes are not too large is to use a floppy disk or diskette. Diskettes are very useful as a portable storage medium for small files.

Mounting and Using the Diskette

Before a diskette can be used, you have to perform an operation called "mounting" the diskette. This is to let the system know that you are going to use and access the diskette in the floppy drive.

To mount a diskette, move the mouse to an empty area on the Desktop and right-click it. From the pop-up menu select,

Disks --> Floppy

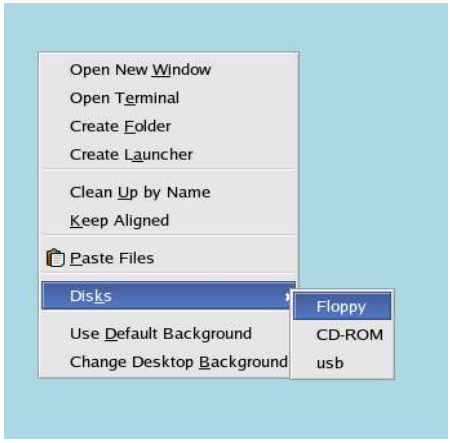


Fig. Mounting a Floppy

(Under the new version of GNOME, disk access is done by right-clicking on the Computer icon on the desktop and selecting Floppy. See the Note at the top of this chapter.)

This will mount the floppy and add a floppy icon on the desktop. Double-clicking on this will list out the files and folders in the floppy under the File Manager.

Once you have successfully mounted the diskette and listed out its directory contents under the File Manager, you can treat it like another folder to read and write files and folders. However, you have to bear in mind some differences between the floppy diskette folder and the normal folder you have been working with.

The diskette has a very low storage capacity compared with the hard disk; a floppy diskette typically will have about 1.44 MB of storage capacity only. In contrast a hard disk will have at least a few hundred MB of storage capacity at the minimum!

If the write-protect tab on the diskette is enabled, then the diskette can be used for reading only, i.e. you can read the contents of the files on it only. You cannot write to the files, and so you cannot modify and/or create new files or folders.

After using the diskette, you will need to unmount it before you take off the diskette from the floppy drive. To unmount the diskette, right-click on the floppy icon on the desktop and select "Eject". This will unmount the diskette and the floppy icon will disappear from the Desktop. Once this has happened, you can safely remove the diskette from the floppy drive.

Note:
It is important that you unmount the diskette before removing it from the floppy drive, especially if it has been mounted for writing. Failure to do so may result in incomplete data being written to the diskette and corruption of the file system on the diskette.

Formatting the Diskette

Before a diskette can be used it has to be "formatted" first. This will create the directory structures and other information needed for the system to keep track of where the data is stored on the diskette. You need to format a diskette once only. You can subsequently format it again after using it for some time but re-formating will result in the loss of the original contents of the diskette.

To format a diskette, you can use the floppy formatter selection from the Main Menu.

Main Menu --> System Tools --> Floppy Formatter

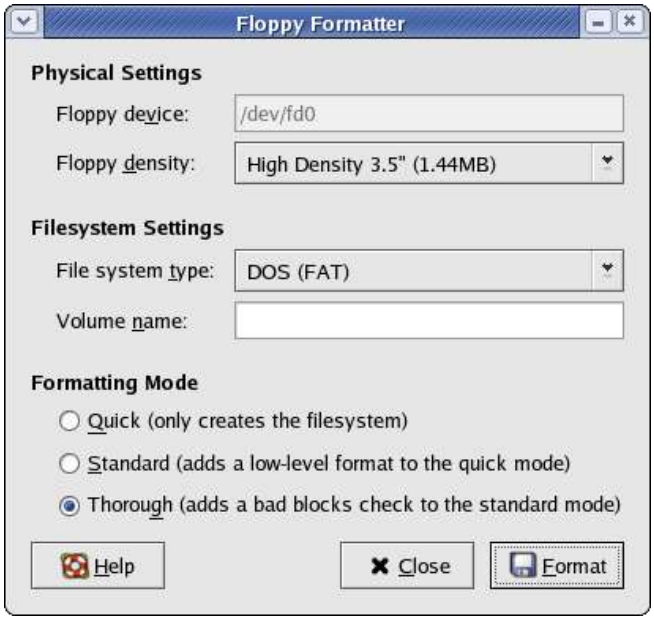


Fig. Floppy Formatter Dialog

Choose the default floppy density of 1.44 MB and the DOS (FAT) file system type. This will enable the diskette to be read on almost all commonly used operating systems including Microsoft Windows.

Exercises

1. Format a floppy diskette, mount it for read-write and copy the following files/folders found in the /etc directory to it: hosts, termcap, motd and rc.d.

2. Unmount the diskette, mount it again and copy its entire contents to the folder tmp in your home directory (create this folder if it is not there). Unmount the diskette after the copying.

The CD-ROM Drive

Nowadays the CD-ROM is widely used as a means to store and distribute data and information. A typical 5.25" CD-ROM can store up to 700 MB of data. It is thus ideal as a medium for distributing large files, multimedia games and software packages.

By default when a CD-ROM is placed in the drive it is automatically mounted and the File Manager will open it to display its directory contents. If it is not automounted, then it can be mounted by moving the mouse to an empty area on the Desktop and right-click it. From the pop-up menu select,

Disks --> CD-ROM

(Under the new version of GNOME, disk access is done by right-clicking on the Computer icon on the desktop and selecting CD-ROM. See the Note at the top of this chapter.)

This will mount the CD and add a CD icon on the desktop. Double-clicking on this will list out the files and folders in the CD-ROM under the File Manager. The File Manager can then be used to access the files and folders on the CD. Since the CD-ROM is a read-only medium, you can only read the contents of the CD and not write to it.

After using the CD, you will need to unmount it before you take it off the CD-ROM drive. To unmount the CD, right-click on the CD icon on the desktop and select "Eject". This will unmount the CD and the CD icon will disappear from the desktop.

The USB Mass Storage Device

Another popular portable storage medium is the USB mass storage device (sometimes also known as a thumb drive). Like the floppy diskette you can read and write to a thumb drive but it has the advantage of storing much more data than a diskette. Thumb drive devices of capacities 32 MB, 64, 128 MB etc. are common.

To access the thumb drive, place it in the USB slot (port) provided and move the mouse to an empty area on the desktop and right-click it. From the pop-up menu select,

Disks --> usb

Note:
This assumes that the system you are using has been set up with the name of "usb" for the thumb drive; it can be another name (customisable).

Under GNOME 2.6 (see the Note at the beginning of this chapter) the thumb drive is accessed by double-clicking on the Computer icon on the Desktop, and is refered to by the name "Flash",

Computer --> Flash

This will mount the USB thumb drive and add a thumb drive icon on the desktop. Double-clicking on this will list out the files and folders in the thumb drive under the File Manager. You can then treat it like another folder to read and write files and folders.

After finishing with the thumb drive, you will need to unmount it by right-clicking on the thumb drive icon on the desktop and selecting "Unmount Volume", before removing it from the USB port.

The CD-RW Drive

While the CD-ROM drive is very useful as a portable storage medium due to its high capacity and low cost, it suffers from the disadvantage that it is a read-only medium.

To overcome this, many PCs are sold nowadays with drives which enable you to record data onto CD-R (CD recordable) and CD-RW (CD rewritable) disks. The former refers to a CD medium which allows you to record to it only once, while with a CD-RW disk it is possible to rewrite data to it multiple times.

Note:
While a CD-RW drive supports both CR-R and CD-RW functionalities, the CD medium that you utilise determines whether you can write data to it only once (CR-R) or multiple times (CD-RW). So it is important that you buy the correct medium type for the function that you want.

CD-Creator

The File Manager has a facility which enables a user to copy files and folders very easily to a CD-RW drive. To use this feature, start the File Manager and from the menu bar at the top select,

Go --> CD Creator

and the CD Creator window will be displayed. The files and folders which are to be copied (burned) onto the CD-R(W) disk are to be placed here in this window. To do this, open up another window on the File Manager,

File --> Open New Window

In the new File Manager window, select the files and folders you want and drag and drop them into the CD Creator window. After you have finished selecting and dropping all the files and folders you want, go to the Cd Creator window and click on the "Write to CD" button at the top. A dialog box will open up and from here you can choose the writing speed, the CD name and other options. The default settings may be used if you do not know what to fill in here.

To start the burning, click on the "Write files to CD" button in the dialog box. A status window showing the status of the CD burning will be displayed.

GnomeToaster

While the CD Creator application described above is very convenient and easy to use, its functionality is mainly limited to the copying of files/folders to a CD-R(W). A more versatile CD burning application is GnomeToaster. To start it select,

Main Menu --> System Tools --> More System Tools
--> CD Writer

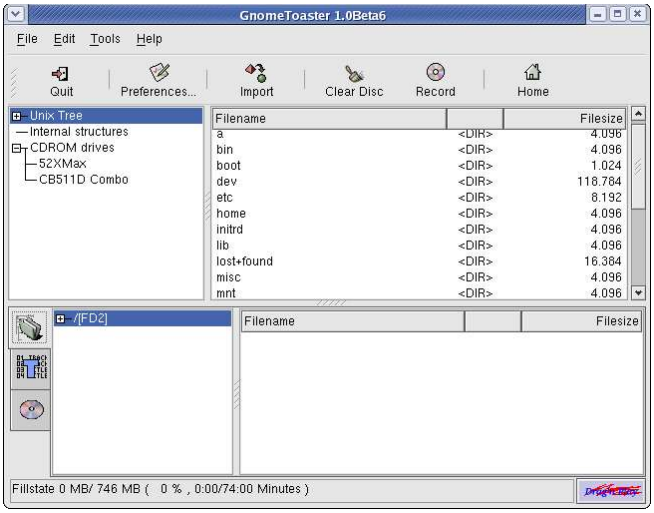


Fig GnomeToaster

Top Window - Data Sources

The GnomeToaster window has 2 main window sections. The top window shows a list of data sources - Unix Tree, Internal Structure and CDROM drives. The data sources are the sources for the data to be copied and burned into the CD-R (W).

The Unix Tree represents the file system on the computer. Clicking on the + icon of the tree will expand it to display the file system, i.e. the hierarchical tree structure comprising files and directories (folders). Clicking on a directory will cause the files in the directory to be displayed on the right panel of the top window.

Clicking on CDROM drives will expand into a listing of the CD-ROM drives present in the system and clicking on one of the drives will display the tracks in it on the right panel.

Note:
If you have a CD-ROM drive and it is not displayed on the GnomeToaster CDROM drives listing, you may have to physically enter your CD-ROM drive device information into the GnomeToaster configuration setup so that the application can recognise it. To perform this, click on the Preferences button at the top and select "CDROM and Recorder Setup". Click on the Add button at the bottom to display the setup window. Enter the CD-ROM drive data for your system. For the model and manufacturer, you can enter the information marked on your CD-ROM drive or if you do not know what they are, you can just enter any meaningful text description. The Device File entry will probably be "/dev/cdrom" while the Mountpoint entry is usually "/mnt/cdrom". For the Scsi ID use the default value of "0,0". Ensure that the boxes entitled "This Drive is a CD writer" and "Use SCSI Interface for DAE" are ***not*** *marked. Click on OK to apply the settings and click on OK again to exit the "CDROM and Recorder Setup" window.*

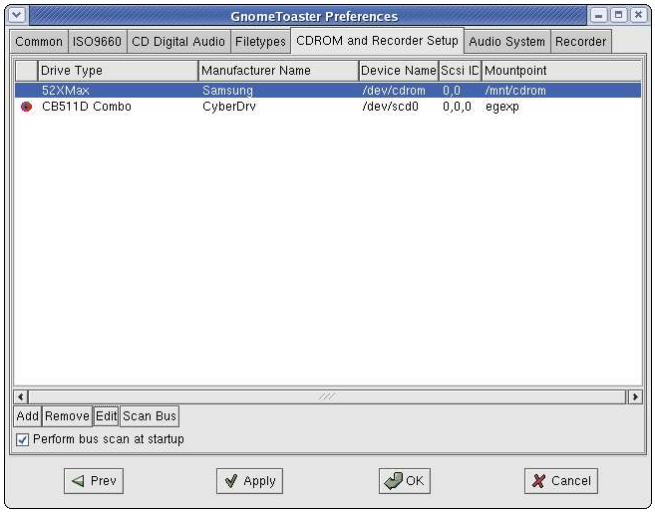


Fig GnomeToaster CDROM and Recorder Setup

The CD-ROM drive should now be visible in the CDROM drives tree listing panel. You can now use it as a source of data for burning.

Clicking on Internal Structures will display the "GnomeToaster-Filesystem" on the right panel.

Bottom Window

Writing data to the CD-R(W) involves selecting the data from the data sources window described above and dragging and dropping them into the bottom window panel. The bottom window has three possible selections – the virtual filesystem, the track editor and the recorder settings. Each of these can be selected by clicking on the icons in the left panel of the bottom window.

The virtual filesystem represents a view of the filesystem setup in a track of the CD-R(W) to be written to later. Note that this is not part of the actual filesystem on your computer and so manipulating the files here will not affect the ones in the computer filesystem. The virtual filesystem can be managed and edited by right-clicking the mouse .

The track editor shows you the tracks that you have selected to be burned to the CD-R(W). You can edit them using the track editor by right-clicking the mouse before committing them to be burned.

The recorder settings view enables you to change from the default, the various parameters for recording, e.g. recording speed etc.

FillstateStatus

At the bottom of the GnomeToaster window is the Fillstate status panel. This shows you the current space that will be taken up by the selected tracks as a percentage of the space available on the CD-R(W). This space utilisation is also displayed in megabyte (MB) units as well as the audio playing time.

Duplicating CDs

It is easy to make a duplicate of a CD using GnomeToaster. The source CD should be placed into the CD drive and the drive selected in the data source top window. The tracks in the source CD will be displayed in the right window panel. Select the tracks you want (or all of it if you are duplicating the CD) and drag them into the tracklist panel in the bottom window. The tracklist panel is displayed in the bottom window right panel when the trackeditor is selected for the bottom window.

After you have selected all the tracks you want, you should select the recorder settings parameters. This is done by clicking on the recorder settings button in the bottom window left panel. Usually you need to change only the recording speed to one which can be adequately supported

by your drive. The Multisession box should usually be off while the Fixate box turned on.

To start the actual recording (burning) of the CD click on the Record button at the top of the GnomeToaster window.

Creating a Data CD

This section will discuss how you can copy files and folders from the mounted filesystem and record them into a CD-R (W). Select the files and folders you want to copy from the Unix Tree in the Data Source window at the top and drag them into the virtual filesystem right panel in the bottom window (the virtual filesystem will have to be selected first in the bottom window). The files and folders in the virtual filesystem can be edited by right-clicking on the mouse. You can create, delete and rename files and directories (folders). You can also provide your own volume name to the data track's volume id (default name is CDROM) by selecting the root entry in the filesystem 's editor left panel and right-clicking it. Select the “Rename VOLUME/Directory” option and choose a new name for the volume.

After you have selected all the files you want, you should select the recorder settings parameters as above and then start the recording.

Writing ISO Images

It is possible to copy and store all of the data on a CD in the form of what is known as an ISO image file format. If this ISO image file is then written to a CD, the new CD will be a duplicate of the original one from which the ISO image was created. This provides a convenient way of duplicating a data CD if you can copy or download the ISO image file from another source to the filesystem on your computer.

To write ISO images to a CD-R(W), select the ISO image file from the Unix Tree filesystem data source window and drag it into the tracklist panel window in the bottom window. After that ensure that the recording settings are correct and click on the record button at the top to start the burning.

Help

There are many more options and features which GnomeToaster have, e.g. multisession burning, mixing audio and digital data, working with CD-RW media etc. The GnomeToaster documentation should be consulted for these more advanced features. The GnomeToaster User's Guide may be obtained from:

http://gnometoaster.rulez.org

Chapter 5: Using the Printer and Scanner

While the holy grail of office computing may well be the paperless office, for many users, it is often desirable and indeed sometimes necessary to be able to put on hard copy the information available on the computer. The printer allows you to do that. The printer is essentially an output device with which you are able to output text and graphics onto paper from digital data stored on the computer. To complement this, the scanner is an input device with which you can transfer text and graphics from paper to the computer and stored in the form of digital data which can then be further manipulated by other software applications.

The Printer

Printer Setup and Configuration

If you have a printer attached to your system you will need to configure and set it up before you can use it properly. To do this, click on the Print Manager icon on the Panel

Note:
To enable the system to detect your printer properly, you should turn on the printer before trying to configure it.

When you are prompted to run the printer configuration tool select OK. You will then need to enter the root password since you are now attempting to set and change some system parameters and configuration. At the printer configuration screen click on the New button to add and configure a new printer.

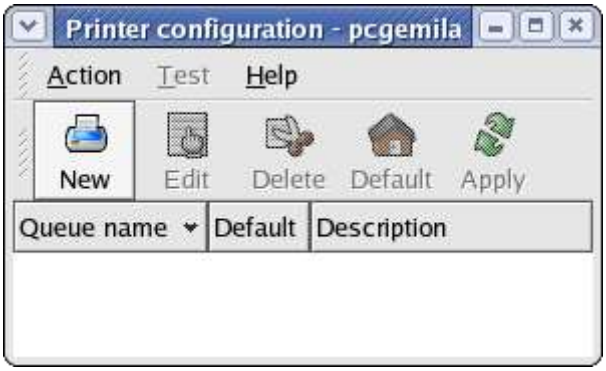


Fig. Printer Configuration

The Add new print queue screen will be displayed. Select Forward.

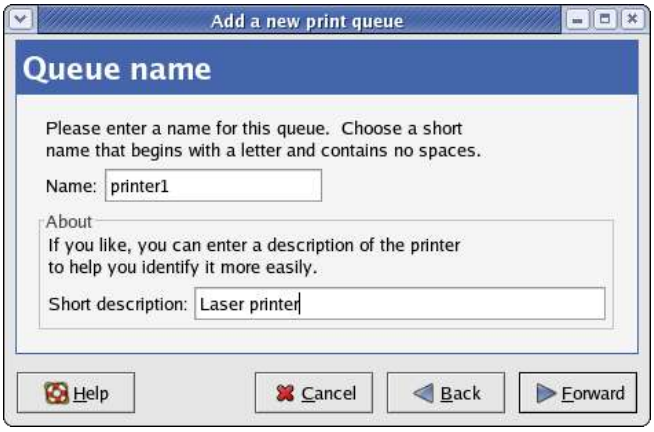


Fig. Print Queue

Fill in the details for the print queue. Enter a unique name for the printer in the Name field. This name must begin with a letter and cannot contain spaces. You may also want to enter a brief description of the printer. Enter Forward to go to the next screen.

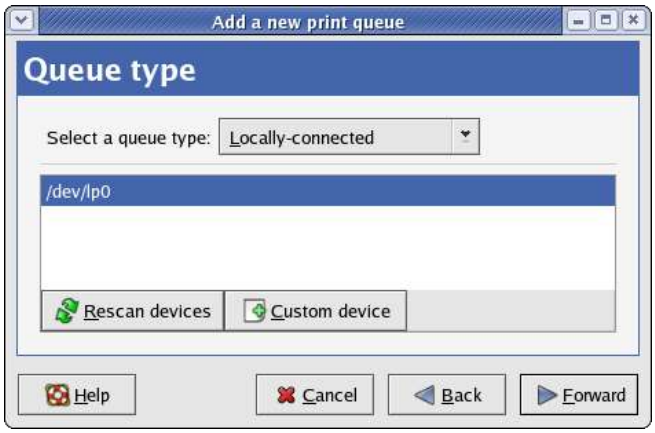


Fig. Queue Type

Select locally-connected for the queue type. Select the printer device by clicking on the device displayed on the screen. If no device is shown click on Rescan device button for the system to check for the availability of the printer device.

Note:
For a parallel printer the device name is usually /dev/lp0 and for a USB printer the name is usually /dev/usb/lp0.

Select Forward to go to the Printer model screen.

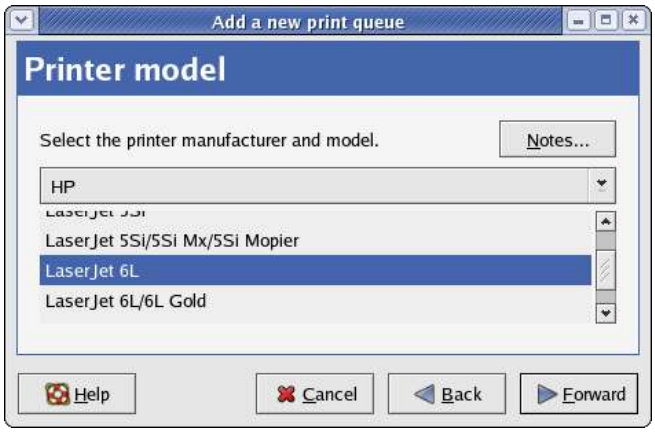


Fig. Printer Model Selection

The model of the printer should have been auto detected by the system. If it is not, click on the pull-down menu to select the manufacturer and the model of the printer. Click on Forward to go to the next screen.

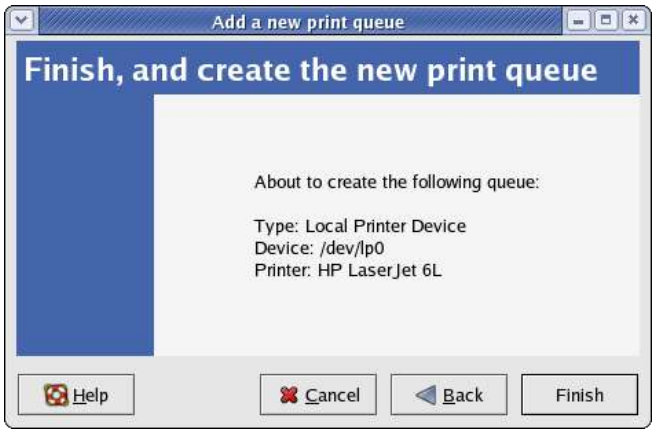


Fig. Finish and Create New Print Queue

Confirm that the printer information displayed on the screen is correct. If so click on the Finish button, otherwise click on the Back button to go back and make further changes.

After clicking Finish you will be asked whether you want to print a test page. Ensure that the printer is connected and online and answer yes to it. Check to see if the test page is printed properly. If it is, click on the Apply button in the Printer configuration screen to add this printer to the system.

After setting up the printer you can delete it or edit its properties anytime by clicking on the Print Manager icon on the Panel. The Print Manager screen will be displayed.

Fig. Print Manager

Right click on the printer in question and select Properties. Select the print queue in question and click on the Edit button to make changes or the Delete button to remove it from the system.

Note:
If you delete all the print queues associated with a printer , the printer itself will be deleted from the system too and you will have to set up a new printer again in order to print.

Managing Print Jobs

When you send something for printing from your application, the data to be printed is stored in the print spooler area and a print job is added to the print spooler queue. In this way many printing jobs can be carried out without waiting for the printer to finish printing a job first before accepting another printing job. The Print Manager is used for managing the print jobs associated with a given printer.

The Print Manager can be utilised to check on the status of the printing jobs that you have submitted to the printer and to cancel jobs still in the queue if you need to. To do this, launch the Print Manager and then double-click on the icon of the printer that you want to monitor. A list of current print jobs in the queue is displayed. To cancel a print job, highlight the job in question and then click on the Edit button on the menu bar on top and select Cancel Documents from the pull down menu.

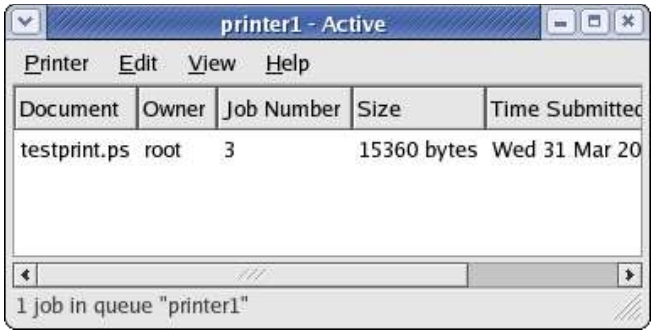


Fig: Status of Print Jobs in the Queue

Using the Printer

Normal usage of the printer from an application is straightforward. Select the printing function from your application and a print job is then automatically submitted to the printer queue.

Exercises

1. Edit the printer properties to change it to another printer

The Scanner

The scanner is a device which allows you to convert analog graphics e.g. a photograph or a printed page into digital format where it can be stored on the computer and further manipulated by appropriate software. Flat-bed scanners for personal use are quite common nowadays.

Most scanners connect to the computer using the USB, SCSI or parallel ports. To use the scanner, suitable software to control and drive it is required. The SANE backend drivers may be used to drive the scanner hardware while the XSane GUI front-end acts as the end-user interface. To check whether the scanner you have can be used, the SANE website at http://www.sane-project.org should be consulted.

Using the Scanner

You can start the scanning software on the Desktop, from the Main Menu,

Main Menu --> Graphics --> Scanning

This will invoke the XSane program and by default two windows will be displayed, the Main Xsane window and the Preview window.

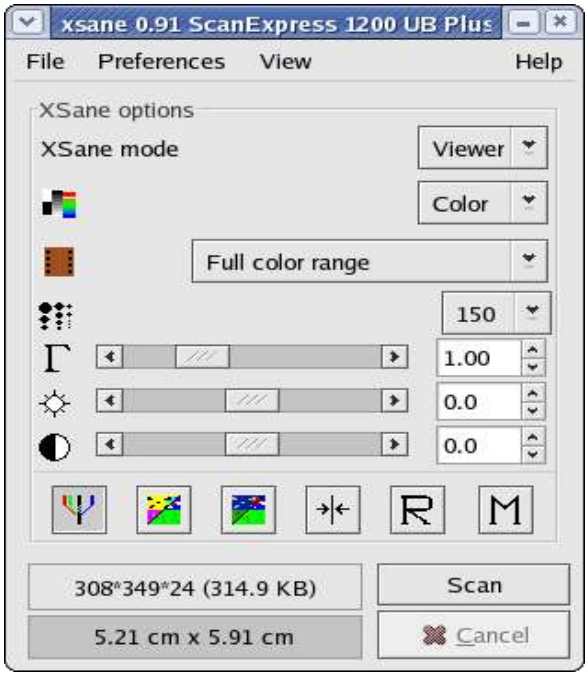


Fig XSane Main Window

There are 5 modes of viewing for this Main window – viewer, save, copy, fax, mail:

- viewer mode - an image is scanned and displayed in the viewer
- save mode - an image is scanned and saved to a file
- copy mode - an image is scanned and sent directly to a printer
- fax mode - the scanned image can be sent to a back-end fax software
- mail mode - an image is scanned and sent via email to intended recipients.

The above modes can be selected in the Main window. In addition, the following selections are also available :

Scanmode:
Selects the mode of scanning choices including colour, grayscale, lineart.

Scansource:
Selects the source of scanning, e.g.: Flatbed, Transparency, Automatic Document Feeder. This is only displayed if there is more than one possible source of input for scanning.

Scanmedium:
Selects the scan medium, e.g.: slide, standard negative or full colour range.

Scan resolution:
Selects the resolution that is to be used for scanning.

Preview

Before the actual scanning of an image takes place, the preview operation is usually carried out to preview the scanned image. To perform the preview, place the image to be scanned into the scanner and click on the "Acquire preview" button in the Preview window.

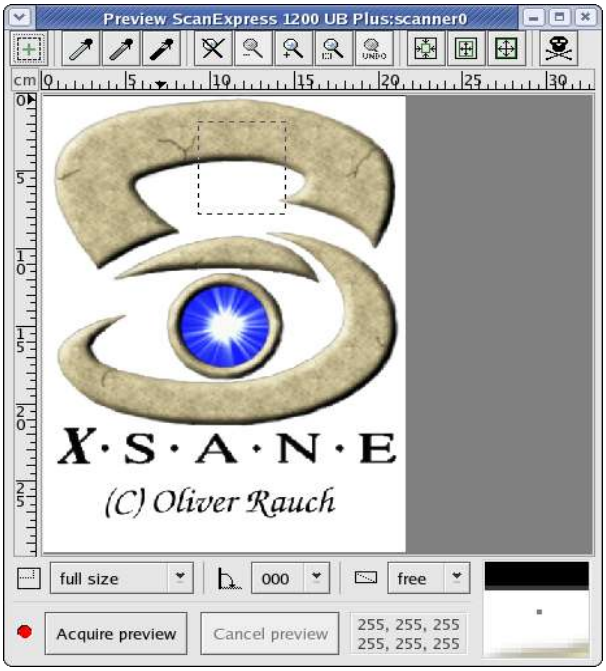


Fig Preview Window

As the scanner will scan its entire scanner bed by default, if you are scanning an image which occupies only a part of the bed you will want to select only the area which holds the image to be scanned. This can be done from the Preview window in several ways. One way is to manually select the scan area frame by pressing and holding the left mouse button to define one edge, moving the mouse so that the area to scan is completely inside the frame (which is marked by a dashed white line) and then releasing the mouse button. Another way is to select the Autoraise button and then clicking on the image to be scanned. A frame is created to mark the scan area. The mouse may be used to adjust the scan area if it is not fully correct.

Scan

After the appropriate scan area has been chosen in the Preview window, the image can be adjusted and/or enhanced for colour correction, contrast, and brightness. The gamma value, brightness and contrast can be adjusted manually in the scan window before performing the scan. In addition, the Autoenhancement button may be used to automatically enhance and sharpen the scanned image for brightness and contrast.

After a satisfactory image is seen in the preview window, perform the actual scan by clicking on the scan button in the main Scan window. The scan will be performed and the scanned image displayed in the Viewer window.

Save

The scanned image that is displayed in the Viewer window can be saved to a file by selecting the operation, File --> Save image. The file name and image format to be saved in can be chosen from the drop down menu selection.

Another way of saving is to select the Save mode, select the filename and format to save the image in and then perform the scan.

Help

More detailed information on how to use the scanner software as well as changing the default configuration can be obtained from the Help menu selection. It also contains useful general information about scanner and scanned image technologies.

EXERCISES

1. Perform a scan of an entire photograph containing several people or objects.
2. Using the same photograph as for the previous exercise, this time perform the scan for only the image of one person or object.

Chapter 6: Multimedia and Graphics Access

Multimedia Access and Players

Apart from being able to view and create graphics and other images, the PC is capable of supporting a wide range of audio and video features and facilities. These include the playing of audio CDs, audio digital files, VCD and DVD as well as multimedia games.

Note:
While support may be available in the software, the necessary hardware has to be available before these multimedia facilities can be used, e.g. a DVD drive is required to be present on the system before you can play a DVD.

By default most modern PCs come with a CD-ROM drive capable of playing audio CDs and VCDs as well as a soundcard and speakers. As such in this chapter we shall discuss how you can listen to an audio CD, play audio digital files and view VCDs.

In the following sections we shall be discussing several media players. Some of these e.g. mplayer and xine are multimedia players in that they can be used as generic players for audio CD, digital audio files, VCD and DVD.

The Volume Control Applet

Since almost all the multimedia applications require some form of audio production and mixing, it is useful if we know how to control the various channels of audio available from the sound card using the Volume Control Applet on the Panel. This may be launched by right-clicking the Volume Control icon on the Panel and selecting Open Volume Control. If it is not there you can also launch it form the Main Menu,

Main Menu --> Sound & Video --> Volume Control

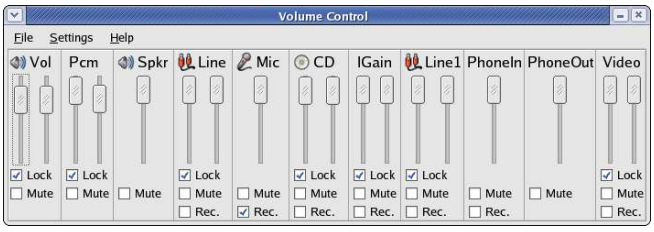


Fig Volume Control Applet

The Volume Control application is essentially an audio mixer which enables you to mix the audio for a 2-channel sound card. As can be seen from the figure above the main component for each audio channel is the "fader", represented by a control knob slider icon. The fader controls the volume of the channel. In addition each channel also has a mute, record and lock button. The mute button mutes the channel. The lock button locks the left and right channels together so that when the fader is adjusted both channels are syncronised. The rec button is a toggle to turn on or off the recording function of the soundcard on a channel.

Usually the main channel that is controlled is the Volume fader. To turn up (or down) the volume of an application, the Volume fader may be manipulated. This can also be done by clicking on the Volume Control applet on the Panel.

Playing Audio CDs

You can play an audio CD by placing the CD into the CD-ROM drive. The CD player application will be run automatically to play the CD. Ensure that your speakers are turned on so that you can hear the CD! You can also manually start the CD Player application from the Main Menu:

Main Menu --> Sound & Video --> CD Player

Fig. CD Player

You can control the CD Player just like a normal CD player with the buttons shown on the CD Player screen. These include volume control, track forward/backward, play/pause, stop, eject as well as jumping to any track on the CD directly.

Playing Digital Audio Files

The XMMS (X Multimedia System) application can be utilised to play a wide variety of digital audio file formats. These include the popular MP3 as well as the open Ogg Vorbis formats. To launch XMMS, select :

Main Menu --> Sound & Video --> Audio Player

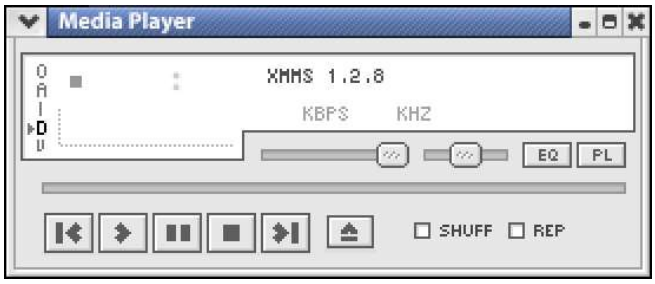


Fig. XMMS Player

Note:
XMMS may need additional plug-ins to be able to play some of the audio file formats. If these are not already installed on your system they can be downloaded from the Internet.

Again, you can control the XMMS player using the graphical knobs and buttons displayed. In addition, you can select the source of the audio files to play. To do this, right-click on the XMMS player and from the pop-up menu select Play File to select the audio file to play. If you want to be able to play an audio file from the Internet select Play Location and enter the Internet location of the file.

XMMS Skins

Skins allow you to change the appearance of an application. Note that this change is cosmetic only and the functionalities of the application are unaffected. To change the appearance of the default XMMS Player, you can install the xmms-skins package which may be included in your Linux distribution CD If it is not there then you will have to download it from the XMMS website. To change the skin of the XMMS player, start the XMMS application and then right-click on it and select,

Options --> Skin Browser

to choose the skin you want. Experiment until you find a skin to your liking.

Fig XMMS Player using a Different Skin

Playing VCD and DVD

There are several excellent applications readily available for you to use to play VCD and DVD media. Here we shall look at two of them: Xine and Mplayer. These can also play CD and digital audio music files.

Xine

The xine application may be used to play VCD and DVD. You can start xine from the Main Menu:

Main Menu --> Sound & Video --> xine

The xine user interface is highly intuitive as it resembles a normal VCD/DVD player with all the basic control knobs and buttons.

Fig. Xine Control Panel

To play a VCD place the VCD into the CD-ROM drive and click on the VCDO button on the xine control panel.

To play a DVD place the DVD into the DVD drive and click on the DVD button on the xine control panel.

The volume control can be adjusted by clicking on the volume control button.

In the event that the xine control panel interferes with the playing screen image, you can hide it by right-clicking on the panel itself and select GUI visibility. This will hide the xine control panel. To bring back the xine control panel right-click on the playing screen and select GUI visibility again.

Xine is a very powerful multimedia application with many features. It can also play audio CDs as well as digital audio files.

MPlayer

Mplayer is a popular movie player for Linux. In addition to being able to play VCD and DVD it is also able to handle a wide variety of audio and video file and streaming data formats. Hence it is useful to use Mplayer as the universal multimedia player. The Mplayer GUI can be started from the Main Menu,

Main Menu --> Sound & Video
--> More Sound and Video Applications
--> Movie Player

On start up, the main Mplayer control screen and the Mplayer video output screens are displayed.

Fig. Mplayer Control Screen

Fig. Mplayer Video Screen

The control screen enables you to control the operations of Mplayer while the video screen displays any video that is being played. In addition right-clicking on the mouse when it is inside either the control or video screens will also bring up a menu for controlling the use of Mplayer.

The Mplayer control screen enables you to perform the following:

- start, stop and pause play
- go to the next and previous stream
- jump to the first and last tracks
- adjust the sound balance
- adjust the volume and mute
- select files to play
- set up a playlist
- set the video and audio equalizer
- configure preferences
- exit the Mplayer application

All of the above functions are represented by intuitive knob, dial and button icons on the control screen making it very easy to use Mplayer.

To play a VCD, place the VCD in the CD-ROM drive, move the mouse over the control screen or video screen and right-click it. Select,

VCD --> Open disk

To play a DVD, perform the above for VCD but instead select,

DVD --> Open disk

The size of the video screen can be controlled by right-clicking on the mouse and selecting normal size, double size or full screen. When you are in the full screen mode, to return to normal size, right-click on the mouse and select normal size.

Exercises

1. Play an audio CD using the CD Player
2. Play an audio digital music using xmms
3. Play a VCD using xine and mplayer

Graphic Images Access

Very often it is useful if we are able to view graphic image files on their own, or in thumbnail fashion if there are many of them. It is also useful if there are simple tools available which will enable us to manage these image files, e.g. catalog and classify them and recall them for viewing. In this section, we shall look at several tools available on your system which provide some of these functionalities.

File Manager

The File Manager itself provides a simple and convenient means to access and view image files. To view an image file from the File Manager, navigate to the folder containing the file and double- click on it. If a folder contains image files, you may also set the "View as Image Collection" option from the View menu selection.

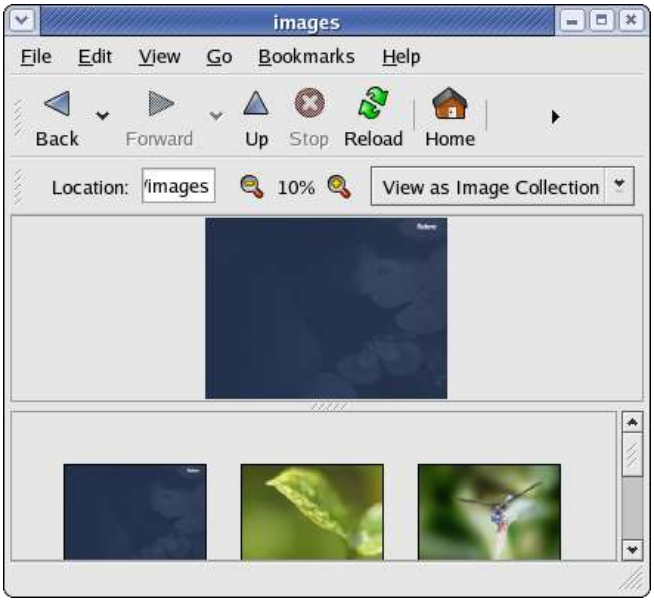


Fig. Image Collection View in File Manager

This will result in only the image files present in the folder being displayed. You can then select an individual image to view and also perform zoom in or zoom out views. If you want to view all the images in sequence, the Slide Show option may be invoked from the View menu.

Gthumb Image Viewer

The gThumb image viewer is a powerful tool for viewing, and organising graphic image files. To start this application perform:

Main Menu --> Graphics --> gThumb Image Viewer

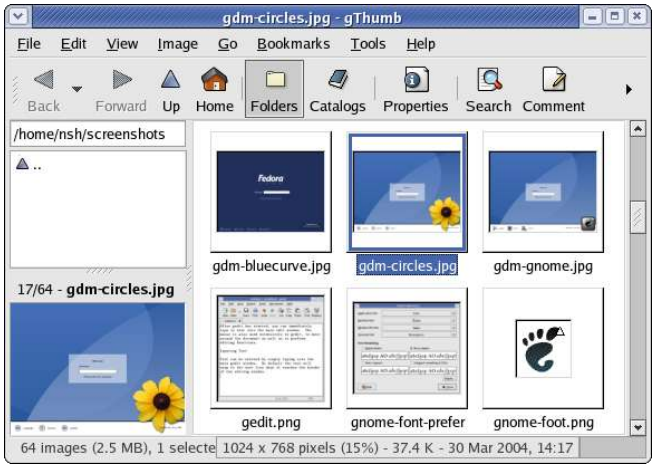


Fig. gThumb Image Viewer

On start up gThumb by default will check your home directory for any image files and if detected the gallery panel will automatically generate thumbnails of the images.

View

Once the thumbnails of the images in a folder have been generated by gThumb the simplest way to view an image is to double-click on the thumbnail and it will be displayed in full. You can also navigate to the next/previous image as well as go to full-screen viewing. The Zoom function can be used to zoom in or out of the image currently being displayed. Click on the Folders button at the top of the screen to go back to the thumbnail view. A preview of the image will be displayed in the preview area if you just select the thumbnail.

One useful feature of gThumb is the slide show. Clicking on this button will start a full screen slide show of the images in the folder.

Image Manipulation

You can also manipulate the images under gThumb. Supported functions include rotation, flipping, resizing, colour balancing, brightness control. To access these functions select Image from the menu bar at the top.

Libraries and Catalogs

gThumb allows you to organise your images in libraries and catalogs. Catalogs are logical views of a group of images and are an alternative to physical organisation in folders. A catalog enables you to group together images with a similar theme or category even though physically the image files may be in the same or different folders. For an even higher level of organisation you can create libraries and then place catalogs under a library.

Libraries and catalogs can be created and manipulated under the catalog view mode. Double-click on the Catalog icon to enter this mode. On entering this mode the folder list panel (at the top left), becomes the library/catalog list panel. From this panel you can access and navigate your catalogs and libraries.

To create a new library, select

File --> New Library

and enter the name of the library. The created library will be displayed in the catalog list panel.

To create a new catalog select

File --> New Catalog

and enter the name of the new catalog. The created catalog will be displayed in the catalog list panel.

To organise a catalog under a library, you can either create the new catalog directly under the library or create the

catalog first on its own and then move it under the library in question. The library/catalog list panel is used for navigating the library list.

To organise images into a catalog, go to the folders view mode (by double-clicking on the Folder icon), select the image(s) you want and then select

Edit --> Add to Catalog

The Choose a Catalog screen will pop up, choose a catalog from the list. If the catalog you want is under a library select the library first and then the catalog.

Convert Format

The Convert function of gThumb allows you to convert your image files from one format to another. Supported formats are:

- Portable Network Graphics (PNG)
- Joint Photographic Experts Group (JPEG, JPG)
- Tag Image File Format (TIFF)
- TARGA format (TGA)

To use the Convert function, select an image first and then from the top menu bar, select,

Tools --> Convert Format

Help

gThumb has many other useful features. The online help guide should be consulted for more details on how to use gThumb.

Exercises

Use gThumb for the following:

1. View the images in the following folder: usr/share/backgrounds/images
2. View a slide show of the images in the folder above.
3. Copy one of the images from the above folder into your home directory and scale it down to 25% of its original size and save it.

Chapter 7: Accessing the Internet

The Internet has revolutionised information usage and dissemination. It has made the global village a reality whereby almost anyone any where in the world is reachable if the person has an Internet connection. The most common way to get Internet connectivity is by using the PC, be it at home, in the place of work, the community hall or even a cybercafe.

In this chapter we shall examine some of the more common methods in which a PC can gain access to the Internet.

The Internet Service Provider (ISP)

For a personal or home user, before you can access the Internet you will need to sign up with an Internet Service Provider (ISP). The ISP usually has a network which is connected to the Internet by a permanent telecommunication link, i.e. one can view the ISP's network as part of the Internet. The ISP provides the necessary networking infrastructure to enable you to connect to its network. Thus once your PC successfully connects to the ISP's network, it can then access the resources and services provided by other computers on the Internet.

The Internet can be accessed from your system using a variety of methods, depending on the type of access methods supported by the ISP you sign up with and the type of networking devices you have installed in your system. Currently the most common methods of connectivity by a home or personal user to an ISP are:

- dial-up
- xDSL

Dial-up Connectivity

The simplest way to access the Internet is to use a dial-up telephone line connection. Almost all ISPs provide dial-up access connectivity to the Internet using the existing telephone line in the home or office. To do this you will need the availability of a telecommunication device called a "modem". Most modern PCs come with a built-in dial-up modem card or if it does not, you can purchase an external dial-up modem and use the serial port available on your PC for connection.

Before you can dial up to your ISP and access the Internet, you have to configure your system to recognise the modem and then dial the correct number to your ISP. You will need to have at hand the following information needed for the modem configuration:

- telephone number to dial to the ISP for the Internet access
- Internet access login name and password provided by the ISP

Modem Configuration

To configure your modem for Internet access, start the "Internet Configuration Wizard" tool from the Main Menu:

Main Menu --> System Tools --> Internet Configuration Wizard

You will be prompted for the root password as this is an attempt to change the system settings and so only the system administrator is allowed to perform this. Enter the root password and the Internet configuration wizard main screen will be displayed.

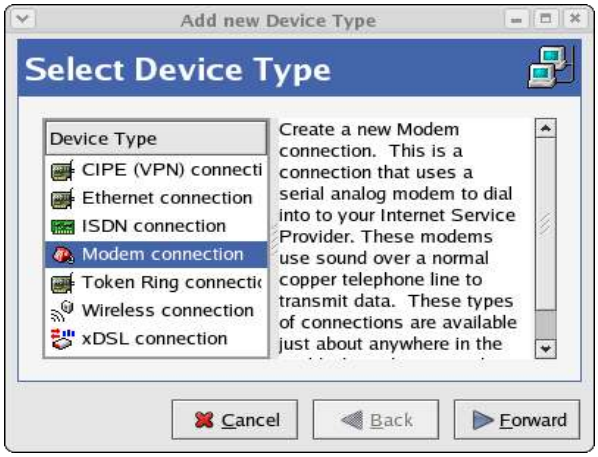


Fig. Internet Configuration Wizard

1. Select modem connection. Click on the Forward button.
2. If your system cannot detect the modem you will be prompted to enter the modem device name and related communication information. You can choose as follows:

   Modem device: /dev/modem
   Baud rate: 115200
   Flow control: Hardware (CRTSCTS)
   Modem volume: Medium

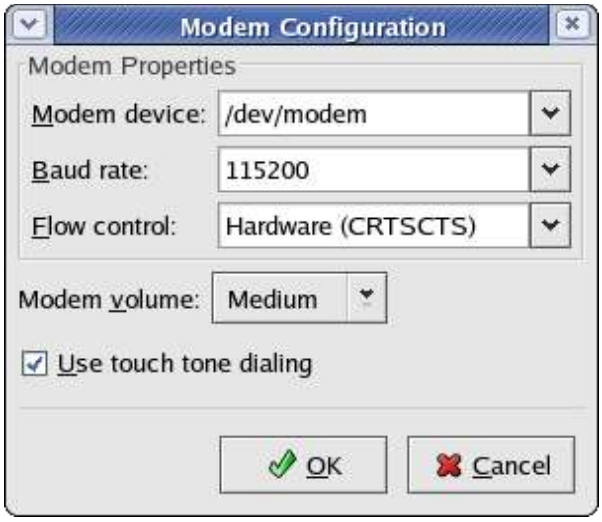


Fig. Select Modem Screen

Click on the Forward button.

3. Enter the phone number of the ISP (Internet Provider). If you need to use an area code and/or a dialling prefix to reach the ISP, you should enter them in the boxes provided, otherwise just enter the telephone number. Enter the name you want this connection to be known by, usually the name of the ISP is used (this is is just a nickname provided by you so that you can recognise this connection). Enter the login name and the password in the boxes provided.

Fig. Filling in the ISP Provider Information

Click on the Forward button.

4. For the IP Settings screen the default settings may be used if your ISP assigns IP addresses automatically (the default for most ISPs). Otherwise you will have to enable the "Statistically set IP addresses" button and enter your IP address and related information.

Fig. Filling in the IP Settings

Click on the Forward button.

5. Click on Apply to accept and end the set up.
6. The Network Configuration window will pop up; exit from it.

Activating the Modem

To test your modem and Internet connectivity, select the Network Device Control tool from the Main Menu:

Main Menu --> System Tools --> Network Device Control

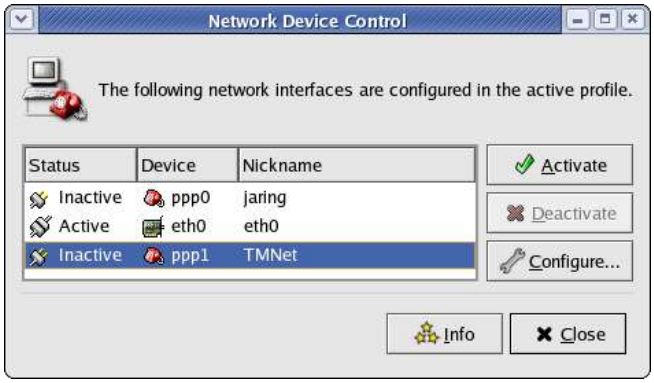


Fig. Activating the Modem

Select the profile you have set up (remember the ISP nickname you used in the set up above?) by clicking on it and then click on the "Activate" button. The modem will start to dial and connect to the ISP and after awhile upon successful login the status of the profile in the Network Devices Control screen will change to "Active" from "Inactive". You are now connected to the Internet. You can now perform the web browsing and other services discussed in the next few chapters.

To disconnect from the Internet bring up the Network Devices Control window and select the profile which was activated above and click on the "Deactivate" button. You will be disconnected from the Internet and the status of the profile will be changed to "Inactive".

Exercise

1. Perform an Internet dial-up connection using the modem on the system
2. Determine that you have Internet connectivity by accessing some well known websites
3. Disconnect from the Internet

xDSL Connectivity

The dial-up Internet connection discussed above provides ready and easy access for places which have telephone infrastructure in place. However, it has the disadvantage that the maximum data transmission speed which normal dial-up technology can provide is about 56 Kbits per second. While this speed may be adequate for email text transmission and web browsing of non-multimedia intensive web content, it is not practical for multimedia access. For heavy multimedia content access using the Internet, a high speed link in required. For the personal or home user, broadband xDSL technologies make this possible.

xDSL is an acronym used for the family of DSL (Digital Subscriber Line) technologies which enable high speed data transmission through telephone lines. There are different types of DSL and they include, ADSL, SDSL, IDSL. Collectively these are known as xDSL. ADSL (Asynchronous Digital Subscriber Line) is commonly used for the home.

xDSL Configuration

There are two main types of xDSL configuration in use and most ISPs use either one of these:

- DHCP over Ethernet
- PPoE over Ethernet

Usually if you given a login id and password for your broadband xDSL connection then you should be using the PPoE configuration.

DHCP over Ethernet

For this configuration, what is needed is just to obtain the IP configuration parameters using DHCP (Dynamic Host Configuration Protocol). To set up your xDSL customer premises equipment to perform this, the "Internet Configuration Wizard" tool from the Main Menu may be used:

Main Menu --> System Tools --> Internet Configuration Wizard

You will be prompted for the root password as this is an attempt to change the system settings and so only the system administrator is allowed to perform this. Enter the root password and the Internet configuration wizard main screen will be displayed as described in the section on dial-up modem configuration.

1. Select Ethernet connection in the select device type screen and click on the Forward button. A screen showing the detected Ethernet card(s) on your system is displayed.

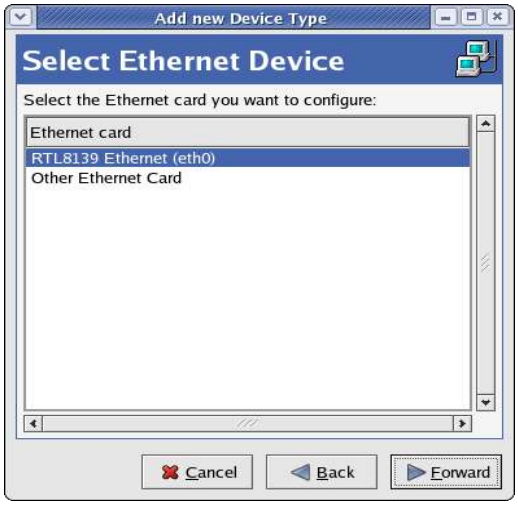


Fig. Select Ethernet Device Screen

2. Select the correct Ethernet device. (This is usually named *eth0* if you have only one Ethernet card installed on the system.) Click on the Forward button.

3. In the Configure Network Settings screen., click the button marked "Automatically obtain IP address settings with:" and ensure that "dhcp" is selected from the pull-down selection. Also check the box "Automatically obtain DNS information from provider". Click on the Forward button.

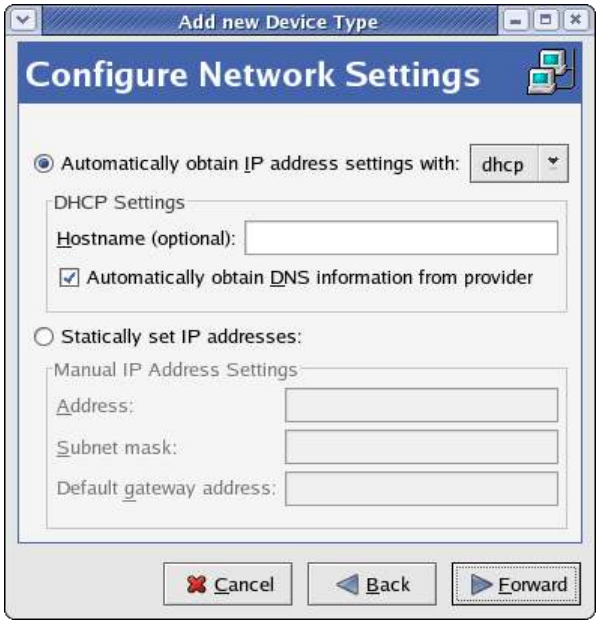


Fig. Configure Network Settings Screen

4. A summary of the configuration just entered is displayed. If something is incorrect, click on the Back button to go back and make the changes otherwise click on Apply to accept and end the set up.

5. The Network Configuration window will pop up; exit from it.

PPoE over Ethernet

For PPoE over Ethernet, start up the Internet Connection Wizard as described in the previous section and select xDSL

1. Select xDSL connection in the select device type screen and click on the Forward button. A screen showing the DSL configuration will be displayed.

Fig. Configure DSL Connection

2. The Ethernet device type will have been selected automatically (this is usually *eth0* unless you have more than one Ethernet card) and entered in the box for you. If this is incorrect select the correct Ethernet device in the pull-down selection. Enter the ISP provider name and the login name and password provided to you by the ISP. Click on the Forward button.

3. A summary of the configuration just entered is displayed. If something is incorrect, click on the Back button to go back and make the changes otherwise click on Apply to accept and end the set up.

4. The Network Configuration window will pop up; exit from it.

Activating the xDSL Link

To test your xDSL link and Internet connectivity, select the Network Device Control tool from the Main Menu:

Main Menu --> System Tools --> Network Device Control

If you are using DHCP over Ethernet, select the Ethernet device name (usually this is *eth0)* you have used in the configuration set up by clicking on it and then click on the "Activate" button.

If you are using PPoE over Ethernet, select the ISP name you used in the configuration set up by clicking on it and then click on the "Activate" button.

The link will be established after a few seconds and the status of the profile in the Network Devices Control screen will change to "Active" from "Inactive". You are now connected to the Internet.

To disconnect from the Internet bring up the Network Devices Control window. For a DHCP over Ethernet setup select the Ethernet device which was activated above and click on the "Deactivate" button. For a PPoE over Ethernet setup select the ISP name which was activated above and click on the "Deactivate" button. You will be disconnected from the Internet and the status of the profile will be changed to "Inactive".

Exercises

Perform the same Internet connectivity tests as done previously with the dial-up modem connection

Chapter 8: The World Wide Web (WWW)

The Internet has much to offer in terms of information on almost any subject matter imaginable and interaction with people and organisations from all over the world. Much of this access and interaction make use of the environment which is popularly known as the World Wide Web (WWW) or web. The WWW is an interlinked network of systems, called web servers, offering multimedia services and information. A user can access these using what is known as a web browser software.

The Mozilla Web Browser (Navigator)

Mozilla is a full-featured integrated web browser, email client, news reader and web page composer program. Using Mozilla a user can be exposed to the richness and diversity of multimedia content and services available on the WWW.

To start Mozilla, click on the Mozilla icon on the panel or launch the application from the menu system:

Main Menu --> Internet --> Mozilla Web Browser

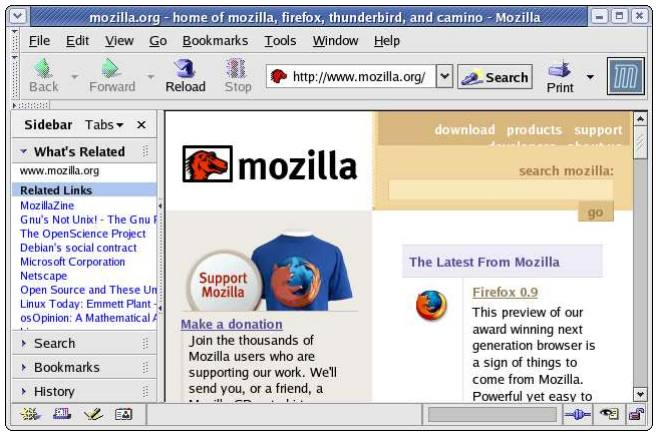


Fig. The Mozilla Web Browser

By default the web browser component of Mozilla (Navigator) will be executed and displayed. The Navigator window has the following main parts.

- the navigation toolbar
- the menu bar
- the side bar
- the display panel

(Mozilla Navigator has many features and only a a brief description of its main functionalities and features can be given here. The user should refer to the Help button on the menu bar for more details.)

The Navigation Toolbar

Fig. The Navigation Toolbar

The navigation toolbar allows you to access a website by entering its Uniform Resource Locater (URL) or more informally known as its web address, e.g. http://www.mozilla.org in the address box provided. Actually you need to enter only the name of the host i.e. "www.mozilla.org" and Mozilla is smart enough to figure out that you want to access the web server on that host.

Clicking on the arrow at the right edge of the address box will open a pull-down menu showing a history of websites visited previously. You can click on an entry in the list to select that website to access.

Also present on the navigation toolbar are the Back, Forward, Reload and Stop buttons.

The Back button enables you to go back to the previous web page displayed.

The Forward button enables you to go forward to the next web page that you have already accessed.

The Reload button forces Mozilla to re-access the website and load the current web page.

The Stop button halts the loading of a web page that is currently proceeding.

Next to the address box in the navigation toolbar is the Search button. This button enables you to perform searches for relevant web pages on the Internet by making use of a search engine. To search for some particular information, you can enter the keywords for the search into the address box and then double-click on the Search button. The results of the search will be displayed in the display panel. You can configure the search engine to use by this search button in the Mozilla Navigator configuration setup (see Configuring Mozilla below).

The Menu Bar

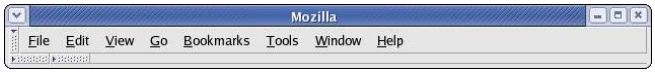


Fig The Menu Bar

The menu bar has several menu buttons. Clicking on one will open up a drop-down menu selection where selected operations can be performed.

The File button caters to the performance of file level operations like the printing and saving of web pages. the opening of web pages, files etc.

The Edit button allows you to find strings of text on the displayed page as well as to edit the Mozilla configuration to you personal preferences.

The View button allows you to control the viewing of the various toolbars as well as the zooming of text and full page display of the display panel. The HTML source code of the currently displayed page can also be viewed using the selection "Page Source" under this button.

The Go button performs similar navigational functions as the navigation toolbar described earlier. A history of previously visited sites can also be accessed by this button. The pull-down menu shows a list of previously visited websites and you can click on one of these to open up the selected web page. Under the Go button, if you select the History item,

Fig. The History Pop-up Window

a pop-up window will be displayed showing in detail the browsing history (i.e. the list of sites visited) over the last few days (by default this period is 9 days; this number is configurable, see below).

The Bookmarks button enables you to manage your bookmarks and personal folder. You can add frequently visited sites to the bookmark and/or personal folder. To manage and organise your bookmarks you can select the "Mange Bookmarks" item in the drop-down menu.

Fig. The Bookmark Manager Pop-up Window

The bookmarks can be organised into folders by dragging and dropping them into the desired folder. New folders can be created by clicking on the New Folder button at the top. Folders can have a name and description. To separate and group together related folders, a separator line can be drawn by selecting the "New Separator" button.

The Window button enables you to navigate and move among the windows opened in Mozilla. You can also use this button to move from one open Mozilla application to another, e.g. to move from the Navigator application to the Mail (email) application.

The Help button contains the Mozilla help files arranged in user-friendly web page style and format.

The Side Bar

By default a side bar is displayed on the left of the main display panel. This side bar contains some of the functions which we have discussed above from the main menu bar at the top. These include the Search, Bookmarks and History functions. The side bar also contains the "What's Related" function, which when selected will display in the side bar a list of links to webpages which contain similar topics to the page currently being displayed in the main browsing display area.

The side bar can be turned on/off by selecting from the main menu at the top,

View --> Show/Hide --> Sidebar

The Main Display Panel

This is the area where the contents of a web page are rendered and displayed. This display area can be made full screen by either selecting the View --> Full Screen selection from the top menu bar or pressing the F11 key. To disable full screen display either press F11 again or click on the unmaximise window button on the top right corner of the menu bar.

Navigation Tab Bar

Mozilla Navigator allows you to browse multiple websites within one browser window using navigational tabs. This overcomes the inconvenience of opening several windows under Mozilla to view multiple sites. To do this either choose under the menu bar:

File --> New --> Navigator Tab

or enter CONTROL-T.

If you open different web pages using this navigation tab feature, they will all be displayed under the same window. You can then use the tab bar to select between each tab screen.

Configuring Mozilla Navigator

Mozilla is highly configurable. To configure Mozilla, select from the main menu selection,

Edit --> Preferences

The categories available for configuration are listed on the left panel of the main configuration window. Clicking on the + button on the left of each category will open up further available sub-categories for configuration.

The Appearance category allows you to configure the default fonts and colours used. Here you will also be able to set the theme and select the language to be used.

The Navigator category allows you to configure the Mozilla Navigator web browser itself.

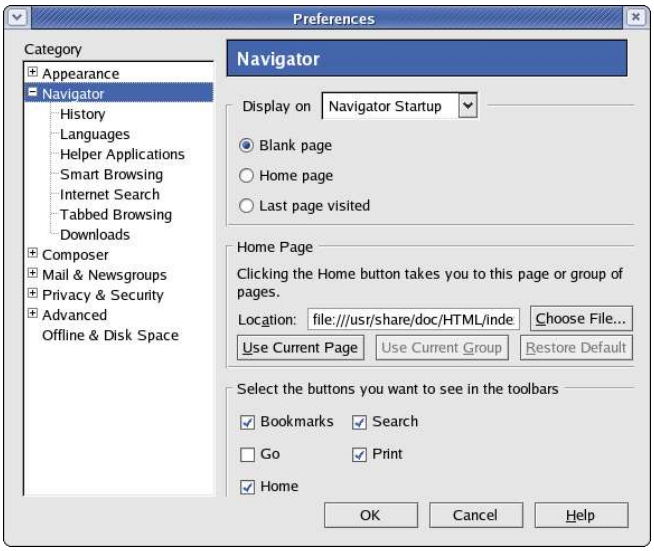


Fig. Navigator Configuration

The main Navigator configuration allows you to select the page to display when the browser is launched. If you select "Home page", the website that is designated as the home page will be displayed. This home page location is set in the Home Page location box below. If you do not want the browser to display anything on startup select "Blank"..

The History sub-category allows you to set the number of days to keep the history list for the history windows and for the location bar history.

The Languages sub-category allows you to select the language to display web pages in (where available) and also the default character coding. To add another language for web page display, click on the "Add" button and select the desired language. The listed language to use for display is in order of preference, so you will need to move your preferred language to use to the top.

The Internet Search sub-category allows you to choose which search engine you want to use for your search function.

Exercises

1. Start up the Mozilla browser and visit the websites listed below. Use the navigation tab bar feature so that you open up the websites all in the same window but under different tab pages.
   - www.mozilla.org
   - www.yahoo.com

- www.iosn.net

2. Bookmark the sites visited above.
3. Configure the Navigator so that it starts with a blank page.

Finding Information on the Internet

The Internet is a treasure house of information. Virtually information on any topic under the sun (and more!) can be found on the Internet. However, while information is easily available it may not be so easy to find the information you want. This is because the information may be available from very many sites, often in varying details and varying aspects of the same piece of information. To assist us in finding information more effectively a search engine or Internet portal may be used.

Note:
Information obtained from the Internet should be scrutinised carefully and not taken as "correct" in all cases. This is because due to its free flowing nature and easy means of access and creation, anyone can publish information on the Internet. As such, unless one is certain that the information is from an authoritative and reliable source, it should be verified by another source or means as far as possible.

Using a Portal

The term portal is used to denote a website that acts as an gateway for providing information about a subject area or group of subjects. From this portal site, information as well as links to other sites providing information about the topics in question may be found. Portals are useful starting places for new web users who do not know where and how to go and look for information about a specific topic or subject.

Many major ISPs provide portal-like information services for their subscribers so that if the latter make this their home page for their web browser, on launching their browser the ISP's portal page is opened. The subject areas covered by these ISP portals are typically subjects of general interest like shopping, local and foreign news, entertainment etc.

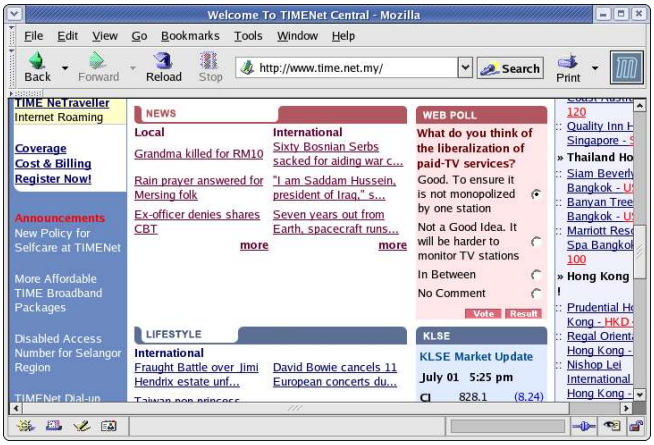


Fig. Home Page for an ISP

Other portals offering general information as well as links to other more subject-specific portals and websites include:

- Yahoo (www.yahoo.com)
- Netscape Network (www.netscape.com)
- Microsoft Network (www.msn.com)
- Angelfire (www.angelfire.lycos.com)

Using an Internet Search Engine

While portals provide a guided and categorised means to access information, sometimes we want to be more specific about the topic we want to find and portals generally are not able to provide this in a timely and efficient manner. An alternative is to utilise a search engine.

A search engine as the name implies, allows you to query it about some specific subject and it will try to retrieve links to web pages and resources which contain information about the subject matter being queried.

The popular search engines available are:

- Google (www.google.com)
- Altavista (www.altavista.com)
- Lycos (www.lycos.com)
- Yahoo (www.yahoo.com)

There are also sites which allow you to search using more than one search engines e.g.

- Search.com (www.search.com)
- Easysearcher (www.easysearcher.com)

Search Basics

While each search engine will have its own technology, its utilisation to perform a simple and basic search is essentially the same irrespective of which search engine is used.

To use a search engine for basic searches, you just type in a few descriptive words about the item/subject you are searching for. It will return to you a list of links to web pages and resources which contain all the words in the query string. Note that common words like "the", "a", "how" etc. are usually ignored by the search engine unless specifically told not to. Words are also not case sensitive unless enclosed by quotation marks. To refine and narrow down your search, you will need to add more words to the search terms you have already entered. Your new query will return a smaller subset of the pages found.

Fig. A Search Engine

The basics of using current search engines is essentially keyword matching and so it is important to be able to identify appropriate keywords so that your search is more efficient and false hits are minimised. The keywords entered should be as specific as possible in order to get better results.

More details on how to use each particular search engine are available on their respective websites and they should be consulted so that you can make efficient use of them.

Example

In this section we shall show an example of how to use a search engine.

In this example, assume that we have heard from a friend that she has been diagnosed with a foot condition in which her forefoot is in pain and there is numbness as a result of poor blood circulation in one of the toes. We also remembered her telling us that the doctor named the

condition as Freiberg disease or some name which sounds like that since the telephone line was not too clear.

Initially we try entering the following keywords, *foot pain,* in the search engine.

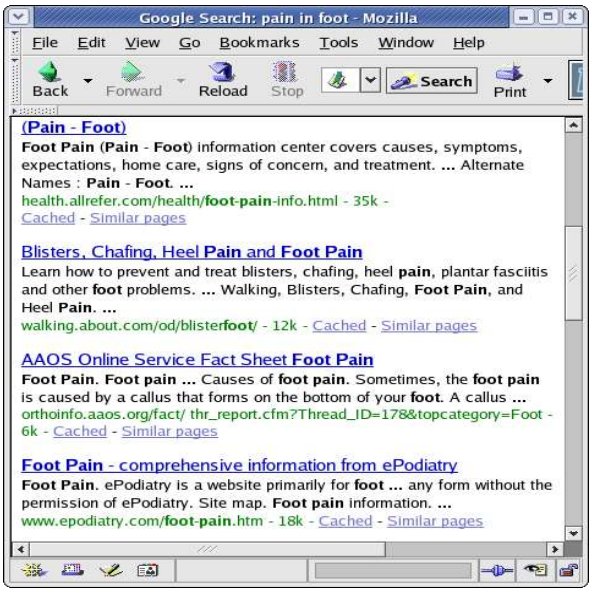


Fig. Search using Common Keywords

As can be seen from the results of the search what we got back was very general and we need to go through each of these links to check if it contains relevant information or links to relevant information. We can narrow down the search if we are more specific about where in the foot the pain is, e.g. *forefoot pain*. This will give us better results but the list of hits is still long.

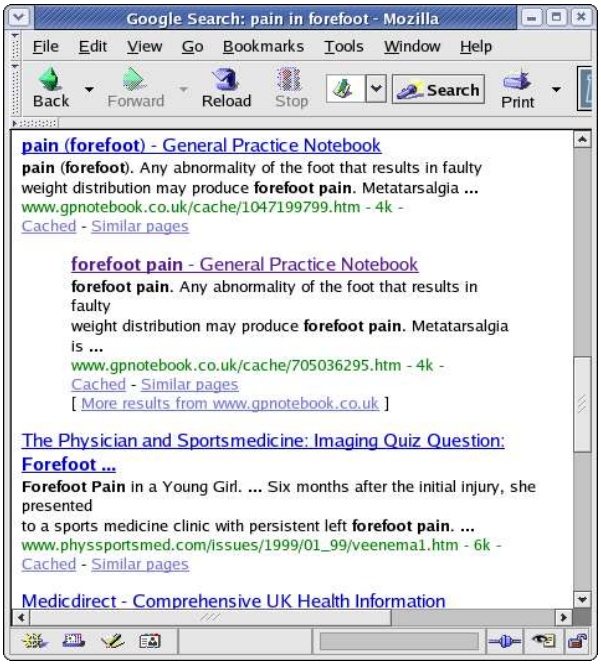


Fig. Search using Relevant Keywords

If we remember that the friend mentioned that the doctor gave the name of the condition as something sounding like Frieberg disease, we can try searching for this specific term. The search results improve immediately as can be seen below and in this particular example we have found several links which are directly related to the information we want.

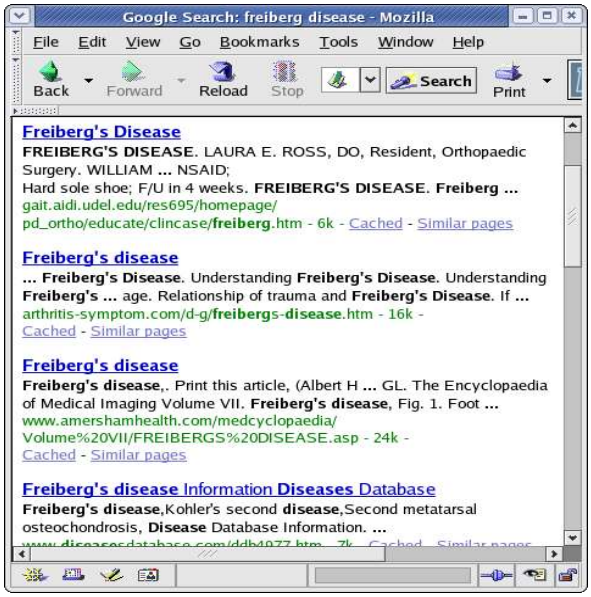


Fig. Refined Search using Specific Keywords

Chapter 9: Internet Email

Internet email has become one of the most popular applications on the Internet. An Internet experience is not complete without the usage of email. In this chapter we shall be describing two commonly used email software: Evolution and Mozilla Mail.

Email Requirements

There are two main types of Internet email systems: webmail and POP3. Webmail, as the name implies, makes use of a web browser to read, compose, send and manage your mail. POP3 email makes use of a POP3 email client to download your email from a server housing your email mailbox. The main difference between a webmail system and a POP3 email system is that for the former, generally, you have to be online to the server housing your mailbox to access and manage your mail. For the POP3 system, you need to be online to the server only to download your mail, after that you can go offline to read, reply and manage your local mail storage. You only need to go online again when you want to send out your email.

In addition to having a webmail account somewhere on the Internet, webmail needs only a web browser and Internet access to work. Examples of popular webmail services are those from Yahoo and Hotmail. Most ISPs also offer webmail services in addition to their traditional POP3 email services.

To use POP3 email you will need to have a POP3 email client software running on your computer. Both Evolution and Mozilla Mail support POP3 email. In addition, you will also need to know the name of the computer on which your POP3 mailbox is located - the POP3 server, as well as the name of the computer which allows you to send out (relay) mail through it - the SMTP server.

Note:
Of course you will need to know your Internet email address irrespective of whether you are using webmail or POP3 email!

Since the usage of webmail involves using your browser mainly, we will not dwell on this form of email anymore here. Instead we will be looking at the POP3 setup and usage of Evolution and Mozilla Mail.

Evolution

Evolution can be launched either from the icon on the panel or from the Main Menu:

Main menu --> Internet --> Evolution Email

The first time you run Evolution, you will be placed in the welcome/setup screen to configure Evolution to send and receive email using your email account. Follow the onscreen instructions to fill in the information required about your email account. Some of the important items to fill in are: described below.

Identity

For the identity screen the following have to be entered:

Email name: (fill in your name)
Email address: (fill in your email address)

Fig: Evolution Email Identity

Receiving Mail

For the receiving mail screen, the following have to be entered:

Server type: POP
Host: (fill in name of your POP3 server)
Username: (fill in username of your POP3 email account)

Fig: Evolution Email POP3 Setup

Sending Mail

For the sending mail screen the following have to be entered:

Server type: SMTP
Host: (fill in name of your SMTP server)

Fig: Evolution SMTP Setup

Time Zone

Ensure that you select the correct time zone.

Apply

After successfully filling in the configuration screens, click on the Apply button at the confirmation screen, Evolution will be started and the main Evolution screen is displayed.

Fig. Evolution Main Screen

Only a very brief description of the functionalities and features of Evolution are given here. The user should refer to the Help button on the menu bar for more details.

Inbox Folder

The Inbox houses the email sent to you which you have downloaded from the POP3 server. To see what is in your Inbox click on the Inbox icon. If you have emails in your Inbox, they will be displayed here. To read an email select it by clicking on it in the Subject Window.

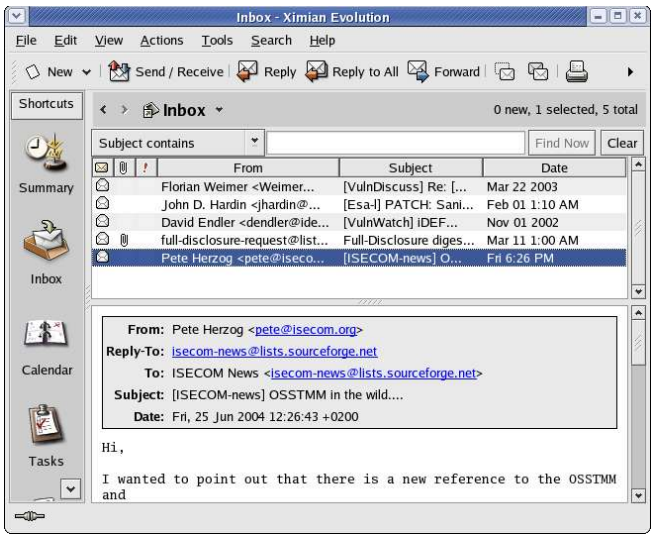


Fig. Evolution Inbox

Composing Emails

To compose an email, from the Inbox screen, click on the New button.

Enter the email address of the recipient as well as the subject. The latter should be a few words describing the contents of the mail but it should also not be too long. If you need to CC the mail to another email address, click on the View button on the top Menu bar and select CC field which will be displayed. After composing the email, you can send it.

Fig. Evolution Composing Email

Offline Mode

If you are on a dial-up connection, it is very likely that you will want to compose all the emails that you want to send offline first before actually going online to send them. To do this click on the File item on the menu bar at the top and select Work Offline

File --> Work Offline

(if the File menu shows the item Work Online then it means that you are already offline, i.e. It is a toggle between online mode and offline mode.)

In the offline mode, when you compose and send your mail they will not be sent out but rather they are stored temporarily in your Outbox folder. You can send them off later after connecting to the Internet by selecting the Send/Receive button.

Sending and Receiving Emails

To send the emails you have composed, connect to the Internet first and then select Work Online from the File menu. Click on the Send/Receive button or click on the Tools button in the menu bar and select Send/Receive. Unsent mail in your Outbox will be sent out and any incoming new mail will be placed in your Inbox.

Deleting Emails

Any mail deleted from your Inbox will be placed in the Trash folder. You should periodically empty the Trash by clicking on Actions from the menu bar and selecting Empty Trash.

Folders

Mail messages in Evolution are organised in folders similar in concept to the File Manager. To view your Evolution folders, click on the View button in the menu bar and select Folder Bar.

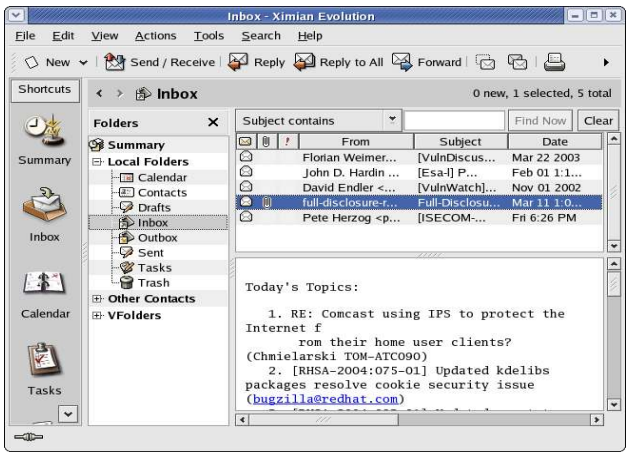


Fig: Evolution Folder View

By default the following email folders are created under the Local Folders: Inbox, Outbox, Sent , Draftsand Trash. You can create a new folder by right-clicking on Local Folders, select ing New Folder and enter the Folder name. You can move and copy mail messages between folders by right-clicking on the message in the Subject Window and selecting "Move to Folder" or "Copy to Folder".

Mozilla Mail

As described in the previous chapter, Mozilla contains an email component in addition to the web browser. This email component is called Mozilla Mail. To invoke Mozilla Mail, you can start Mozilla in the usual way and then from the main menu bar at the top select,

Window --> Mail & Newsgroups

If this is the first time you are running Mail and the email configuration has not been done, Mail will invoke the Account Wizard to take you through the configuration. This is described briefly below.

New Account Setup

In this screen choose Email account and click on Next

Identity

Fill in the information regarding your name and email address to use. Click on Next to continue.

Fig. Identity

Server Information

Enter the server information in this screen. Select POP for the type of incoming server and enter the names of the incoming (POP3) server and outgoing (SMTP) server. These hostnames will been provided to you by your mail provider or ISP. Click on Next to proceed.

Fig. Server Information

User Name

Enter the name of the id that you will need to login to retrieve your email. This name usually will be alpha or alphanumeric and should not contain blank spaces. This will have been given to you by your mail provider or ISP. Click on Next to continue.

Fig. User Name

Account Name

Enter the name by which you will want this mail account to be known as. This is meant for your own Mozilla Mail internal usage to cater to the fact that more than one email account can be set up. The default account name is the email address you entered earlier. Click on Next to continue.

Fig. Account Name

The next screen allows you to review what you have configured and if something is incorrect you can click on the Back button to go and correct it. If all is fine, click on the Finish button. If you do not want Mail to download your mail immediately after finishing this configuration, uncheck the "Download mail now" box.

Using Mozilla Mail

To start up Mozilla Mail, start Mozilla and from the main menu bar at the top select,

Window --> Mail & Newsgroups

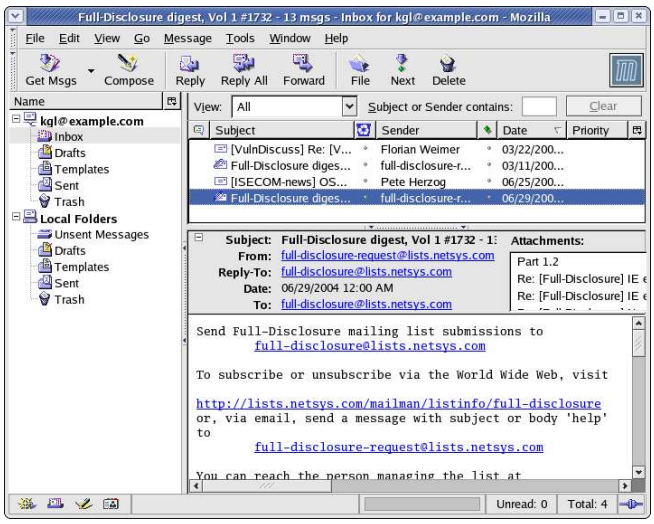


Fig. Mozilla Mail Startup Screen

The default startup screen has three main sections:

- left panel displaying mail boxes and folders
- right main mail display panel
- top panel containing the main menu bar and buttons

The left panel displays in a tree-directory format the email accounts that have been configured for the system and the mailboxes under it. The Local section contains the folders and messages that are not from any of the accounts created and associated with an Internet email account and thus considered local to the system. Note that this includes any unsent mail messages.

Reading Mail

To read any email in your inbox, select the Inbox folder under the desired mail account in the left panel. A list of the messages in your Inbox will be displayed in the top part of the right panel showing the email sender and subject header. To view the mail content click on the desired message in the top right panel and the message body will be displayed in the bottom right panel. Scroll as required to read the entire message.

Composing Mail

To compose a new email click on the Compose button at the top menu. Alternatively you can also click on the "Compose a new message" link in the right display panel. (Ensure that you have selected the appropriate email account on the system by clicking on the appropriate account name in the left display panel.)

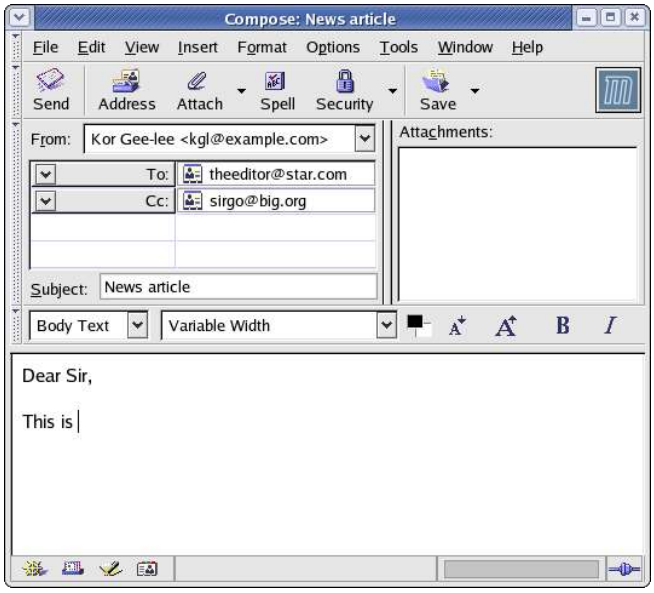


Fig. Composing Mail in Mozilla Mail

The *From* field is automatically filled in for you from the email address you provided in the setup for the account currently being used.

The *To* field has to be filled in with the email address (es) of the recipient(s). If there is more than one recipient, separate the addresses by commas or you can fill them in one line at a time by entering the RETURN key after each address.

The other email header fields like *Cc* and *Bcc* can be selected by clicking on the *To* field button.

The *Subject* field should also be filled in with an appropriate subject header. You should avoid being too verbose for the subject header.

The mail message body can be entered in the right bottom panel provided and any file attachments can be added by clicking on the Attach button at the top and then selecting the file.

Sending Mail

After composing the email, you can either send it off immediately if you are already online, send it later or keep it as a draft to be edited later.

To send it off immediately select the Send button. If the message was sent successfully it will be stored in the Sent folder.

To send it later, select from the main menu at the top,

File --> Send Later

The message will be stored under the Unsent Messages in the Local Folders section. In this way you can compose all the emails first and store them before sending them all in one go. Later on you can send all the unsent messages by selecting from the main menu at the top,

File --> Send Unsent Messages

This method of sending is especially useful if you are on a dial-up Internet connection where you can compose the messages offline and then dial-up later to send the messages.

To save it as a draft select from the main menu at the top,

File --> Save As --> Draft

The email message will be saved in the Drafts folder under the current account section. Later to edit the saved draft, select the Drafts folder under the account, click on the draft message to edit that is being displayed in the top right panel and select from the top menu,

Message --> Edit Message As New

After editing the message you can send it off as discussed above.

Receiving Mail

To receive mail, click on the Get Msgs button at the top, enter your email password (POP3 password) when asked and the email will be downloaded from your Internet mailbox and stored into the Inbox folder. Of course you will have to be online to perform this.

Deleting Mail

To delete mail messages, select the folder in which the messages are stored in, and in the top right display panel where the list of messages are displayed, select one or more messages by clicking on them and then click on the Delete button at the top.

The deleted mail is moved to the Trash folder in the account currently being used. To permanently delete them you can

either delete them from the Trash folder or click on the Trash folder and from the main menu at the top, select,

File --> Empty Trash

Folders

The various folders allow you to organise your email so that you can group and file them under appropriate folders. By default when you create a new mail account, the Inbox, Trash, Drafts and Sent folders are created automatically for you. To create a new folder perform,

File --> New --> Folder

You can file or move your messages from one folder to another by clicking on the list of messages displayed in the top right panel when the source folder is selected and dragging and dropping them into the destination folder in the usual manner.

Account Configuration

You can customise your account email settings by selecting,

Edit --> Mail & Newsgroups Account Settings

There are several important settings which you should be aware of.

Account Settings

In this main configuration screen, you can configure your name, Internet email address and organisation name to use as well as the account name which refer to these settings under Mozilla Mail.

In particular, the "Compose messages in HTML format" box should not be checked if you want to compose and send your email messages using just plain text.

Server Settings

Here you can configure the name of your POP3 mail server. You can also configure whether you want the mail to be deleted from the POP3 server after downloading them into your computer. Usually you will want them to be deleted and so you should ensure that the "Leave messages on server" box is **not** checked.

Outgoing Server (SMTP)

This is where you can configure the outgoing mail server which will process your outgoing mail.

Global Configuration

The account configuration affects only the account in question to enable you to configure multiple email accounts each with possibly its own servers.. The global configuration affects the entire mail subsystem. To access the global configuration, select,

Edit --> Preferences --> Mail & Newsgroups

Under this configuration, you can set global preferences which will affect all the accounts configured by you in Mozilla Mail. These will include the look and feel of Mail itself, the way messages are displayed, format for sending etc.

Exercises

Start up either Evolution or Mozilla Mail and perform the following:

- download your email
- read them
- reply to two of them and BCC yourself in the reply
- compose a new email
- delete spam email
- empty your trash mailbox

Using the OpenOffice.org Suite

Introduction

OpenOffice.org (OOo) is a complete office suite, featuring a word processor (Writer), a spreadsheet application (Calc), and presentation software (Impress). Besides these fundamental office applications OOo also includes a vector drawing tool (Draw), allows database access, allows the publishing of documents in the Portable Document Format (PDF) and presentations in the Flash (SWF) format!

The OOo package is fully inter-operable with the Microsoft Office suite.

Getting around the package

As a first stop for information, it is important to know how the Help system works. To get help:

Help --> Contents

The search function is very useful, and pay attention to the Options (where you can get help for the individual components in OpenOffice.org).

Setting up OpenOffice.org preferences so that it works the way you want it to is significant. The entire controls for this are available at:

Tools --> Options

Here you can setup settings like the default measurement units, font substitution, language types and many more options. Saving a document automatically is not setup by default, so turning this feature on might be helpful: you find it at the Load/Save option, under the General sub-section.

There are three important toolbars to know:

- Main toolbar – this is typically located right below the menus, and contains items like new document, save a document, exporting to PDF, copying & pasting, as well as access to the Navigator, Stylist, and Gallery.
- Object toolbar – this is right below the main toolbar, and has access to font control, and other attributes of objects.
- Function toolbar – located at the left-corner of the screen, and contains many options including quick table generation, insertion of objects, and many more.

Writer

This is a powerful tool for creating professional documents, reports, newsletters and so on – it is a word processor that allows easy integration of charts and pictures, as well as other OpenOffice.org-compatible documents. It can create everything from a simple letter to books, with professional layouts, with the use of styles.

Start it from the Main Menu by,

Main Menu --> Office --> OpenOffice.org Writer

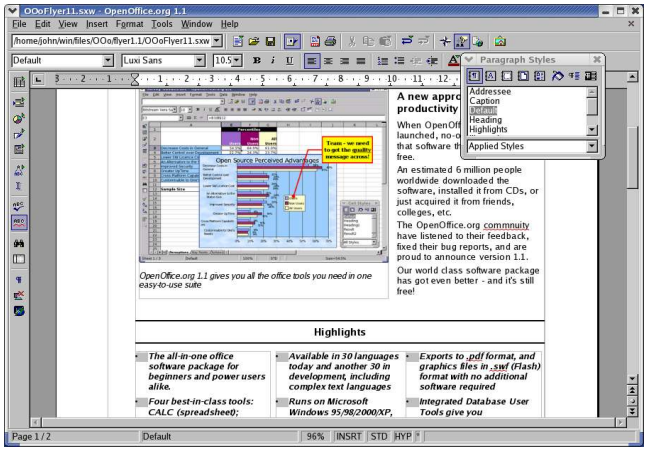


Fig. OpenOffice.org Writer

You are now presented with the word processing portion of OpenOffice.org, and the interface is rather similar to other word processing tools available. Rather than providing guidance throughout the entire package, we will just concentrate on a few tasks at hand.

Common Functions

Functions of the word processor can be controlled via the toolbars located at the top of the screen. On the first row, file actions like opening and saving files can be performed, while on the second row, changing the font, size, and style (bold, underline, or italics) are located there.

They can also be controlled by the menus that are common through packages:

- File --> New --> Text Document - creates a new empty, untitled document for you to work on.
- File --> Open - opens the file.
- File --> Close - closes the document you are working on. If changes have been made since your last save, you will be prompted to save or discard those changes.
- File --> Save - saves the document you are currently working on.
- File --> Save As... - saves an updated version of a document in a different location, with a different name, from the previously saved version.

Common Operations

For operations while writing, it is common to want to select a lot of text, copy it, maybe cut it from its current location and paste it elsewhere, or even undo an action. All this is possible with the office suite, and such options are available at the Edit menu. A few common options are:

- To copy text: select the text with the mouse, then select Edit --> Copy. Now the selected text is kept in memory for use elsewhere.
- To paste text: find the spot where text needs to be placed, place the cursor there, and then select Edit --> Paste.
- To cut text: this means that the selected text will be removed from the current location and kept in memory, to be placed elsewhere. Doing this is exactly like how a copy should be performed, except selection Edit --> Cut instead.
- To undo an action: Select Edit --> Undo. It will display the command that it is undoing at the moment.

By browsing the menu, there are also keyboard shortcuts located next to it. Once more proficient use of the package

occurs, it is much quicker to use keyboard shortcuts like Control+C for Copy, and so on.

Formatting

Formatting text is as important as writing the text, and Writer provides many formatting options, including the Stylist. Individually, you can also format the character (current selected item, or even a whole word), the paragraph, or even the page.

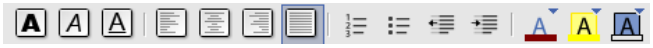


Fig. Part of the Object toolbar (Writer)

Some of the quick format options include **bold**, *italics* and underline. These options are available at the toolbar at the top of the screen.

Text alignment plays a large role in controlling how portions of the document will look. For example, an address field at the top of your letter will have such details right-aligned, while the body and rest of the base text will be left-aligned. This is all controlled by the four-icons that are located next to the bold/italics/underline icons, providing such options as: right-align, centre-align, left-align and justified. When text is justified, it looks exactly like what you're reading now! (a more professional end-to-end stretch of the text.)

Let's switch to the end of the toolbar, and notice that the options there including providing a paragraph background – which is good for highlighting a paragraph or several paragraphs of text, in colours that you choose. You can also highlight text (like you would with a highlighter and paper!), and change the font colours all with the icons there.

Aligning text by indenting it is also another feature available as part of the object toolbar. Left/right alignment of text is provided, and if text is already entered and you want to left-align it, selection of text (or having the cursor at the paragraph) must happen first, before text is indented.

Those were just quick controls. To get full control, using the **Format menu** is ideal. Controls are more varied here.

Styles

Consistency throughout a document is important – it was earlier said that writing books using OpenOffice.org is possible. So there must be a way to handle long and large documents in a consistent fashion, with similar fonts for headings, sub-headings, text, and other attributes within a document.

OpenOffice.org includes a powerful feature known as styles, and this is accessed via the Stylist (get this by hitting the F11 key, or clicking its icon on the main toolbar). Notice the floating window, which is most likely active at the “Default” style. By right-clicking on the style, there are options to modify the style, or create new custom ones.

By going to the modify option, the style can be customized via many varying attributes including spacing, alignment, font, emphasis, colour and many more. Once suitable styles have been pre-defined in the document, they can be used on existing text just by selection, and double-clicking on the style name.

Just a little bit more...

Now that the gist of the Writer package has been covered, there's just a little more to know.

Writer has a built-in spell checker. This can be accessed via:

Tools --> Spellcheck

The option to auto-spellcheck means that while typing, Writer will dynamically check your spelling, and if it detects an error, it will output a red-line at the bottom of the misspelled word. Keep in mind that the spell checking is based on the current language that is in use. This can be changed via:

Tools --> Options --> Language Settings --> Languages

Accessing word counts in the document is different to most other packages on a default install of OpenOffice.org (this can differ with several Linux distributions' offerings):

File --> Properties --> Statistics

It is under the Statistics tab that the word counts and other relevant document counts are based. On certain vendor modified distributions of OpenOffice.org, going to the Tools --> Word Count menu will allow the Statistics dialogue box to be displayed automatically.

The AutoCorrect/AutoFormat (Tools --> AutoCorrect/AutoFormat) options have replacement tables (so that CDs really are valid, and will not be changed to Cd, for instance). There are also word completion options (very useful, as the software starts thinking for you) and settings to make them more user-friendly.

The Navigator is a yet another useful tool (get this via hitting the F5 key or clicking its icon on the main toolbar), especially when dealing with larger documents. It supports jumping to bookmarks, notes, any particular object, and even creates a table-of-contents on the fly, based on the styles that are being used!

Calc

This is the spreadsheet component of the OpenOffice.org package, and contains many useful features, including an array of functions and plenty of charting options. It is fully inter-operable with Microsoft Excel, though the function separators differ in the two packages.

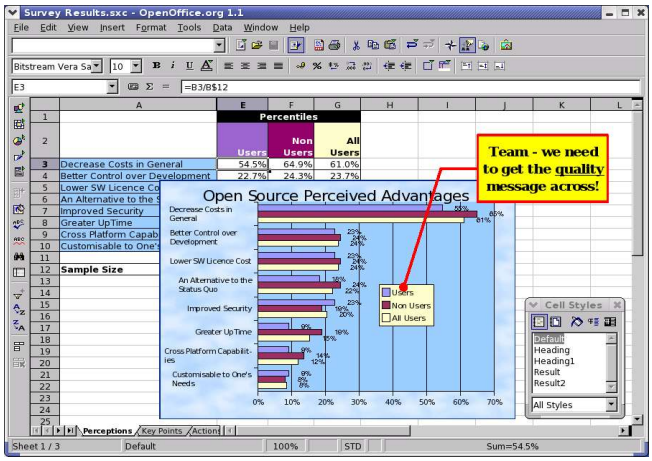


Fig. OpenOffice.org Calc

To start this, it is available via

Main Menu --> Office --> OpenOffice.org Calc

or if you already have an existing window of OpenOffice.org open,

File --> New --> Spreadsheet.

Spreadsheets contain many rows and columns, and each row and column combination is called a cell (like A1, B4, and so on). Upon inputting text into a cell, you might realize that the text is wider than the cell allows for – this can be re-sized via right-clicking the cell, and selecting the Format Cells option. There under the Alignment tab, selecting Line Break is what is required.

Formatting

Like other parts of OpenOffice.org, Calc also comes with the Stylist. But let's get around to understanding the various

differing formatting options available in this component of the package.

If there is some information that you already have created, and the area should be formatted, one particular quick and easy option is to use the AutoFormats available in Calc. This is done after selecting the area then:

Format --> AutoFormat

These are pre-defined styles that are available in Calc, and if you have created your own particular style, you can add them into your new AutoFormats.

In the object toolbar, there is an option to set the font colour within the cell. There are also options to increase/decrease the indents within a cell, and in the image below, controls for enabling:

- Currency
- Percentage
- Add/Remove significant decimal places

Fig. Part of the Object toolbar (Calc)

These are quick controls, and accessing them is as simple as clicking the icons that represent them, and automatically the cell will be formatted as stated. Not only can Borders be set easily, and cell backgrounds too, but the alignment of text within a cell can also be set. This can be either as a top aligned, centre aligned, or bottom aligned.

Now that most of the formatting options are known, it is easy to apply Styles to the spreadsheet – bring up the Stylist by hitting the F11 key, and you'll notice that cell styles (that control all elements, including formatting) and page styles can be set (the latter controlling margins, headers/footers, and borders).

Spreadsheet basics

There are a few points to note when using a spreadsheet. One of them is that calculations are performed in a left-to-right format, with algebraic ordering rules. This means it deals with brackets ("()") first, then division ("/"), multiplication ("*"), addition ("+") and finally subtraction ("-").

When applying calculations, keep in mind the range of included cells. When using a function like =SUM(), and using the argument =SUM(A1:A4), it means it looks for the sum of the cells A1, A2, A3, and A4. These operations can also be performed on non-consecutive cells, so, =SUM(A1;A4;A7) just executes the sum of cells A1, A4 and A7.

If you have used Excel before, it would be relatively common to use a comma (",") as a separator character between the parameters, however, with OpenOffice.org Calc, the separator character is a semi-colon (";"). So for the function to validate correctly, an expression such as =IF(B3>0;A1-A2;A1+A2) is correct (as opposed to replacing the ";" with ",").

Building functions

To perform calculations, spreadsheet make use of functions. Common functions include =SUM() for summation, =AVERAGE() for the average value of cells, and so on. As an aid to the novice user, OpenOffice.org provides a Function AutoPilot. This is a wizard to help build formulas, and find problems with existing expressions.

Fig. Function AutoPilot Button

Located next to the universal sum function, is the Function AutoPilot. If you click on in, a pop-up dialogue appears.

1. In the Functions tab, you can filter viewable/accessible functions via category, and the option to choose a function is shown. Use your mouse to choose a function that you plan on using.

2. Once the correct function is selected, and the action that it performs is agreeable (it is displayed on the right of the dialogue), select Next to move on.

3. Now you are allowed to input numbers. Assuming the AVERAGE function was chosen, in the number 1 field (for example), there are options to either enter a function or select a range of cells.

4. Use select a range of cells, and now a different dialogue pop's up and you can use the mouse cursor to select a range of cells. Click on OK, and you're done!

That is a very easy way to build a formula, which requires no pre-requisite knowledge about what formulas exist in Calc.

Sorting

A big part of dealing with spreadsheets involves a lot of sorting and filtering of data. To sort a dataset, selecting the active cells, then clicking

Data --> Sort

will bring a pop-up dialogue that has options for sorting the data based on the columns present, as well as if the data should be ascending or descending.

Charting

Converting data into information is a process usually accomplished well by creating graphs and charts – it is a lot easier to infer based on visual graphics, rather than lots of numbers. Calc provides a charting wizard that will allow this to be automated rather easily, with a lot of predefined settings.

1. Select the cells that are to be charted, and then go to Insert --> Chart.

2. The range is pre-selected, and certain options are provided (like where the resulting chart is); just leaving the pre-selected options will be good for the exercise. Click Next.

3. A type of chart is to be chosen – common charts include pie chart, line graphs, or even bar graphs – it all depends on the information being represented.

4. Select the defaults, and create a chart. (You should now see a bar chart created).

Now that a chart has been created, it is not static in the sense that it cannot be edited – you can control each and every aspect of how the chart looks. Double-click the chart, and you will now go into edit mode. Notice the toolbar by the left-hand side of your screen has changed? This is in direct response to it being in edit mode, and a lot of properties can be changed here.

There are plenty more features, like data filtering, scenario creation, and goal seek, which once you get more advanced with spreadsheet know-how, you will end up making use of.

Impress

No office suite is complete without a presentation piece, and OpenOffice.org shows its colours with Impress, the presentation piece of the suite. To start it, it is available at

Main Menu --> Office --> OpenOffice.org Impress

or if you already have an OpenOffice.org window open, its available at

File --> New --> Presentation.

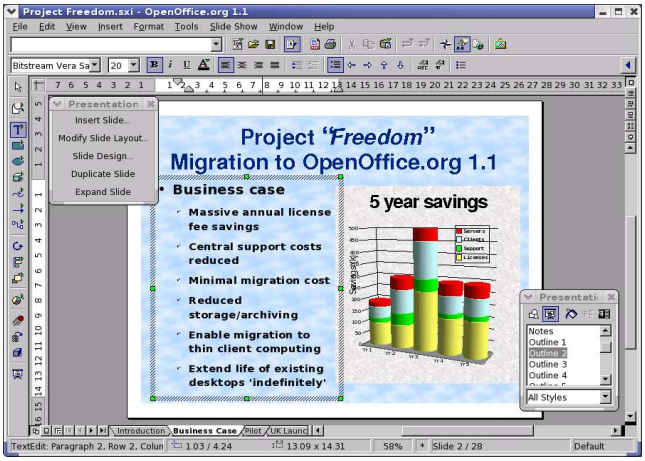


Fig. OpenOffice.org Impress

Unlike other components of OpenOffice.org, when you start Impress, you are presented with an AutoPilot, to start creating your presentation! This gives you options to start a presentation with an empty template, or even with one of the pre-defined templates. A preview dialogue is available, and once all options are selected (and Next is clicked, to move on), you get a basic presentation.

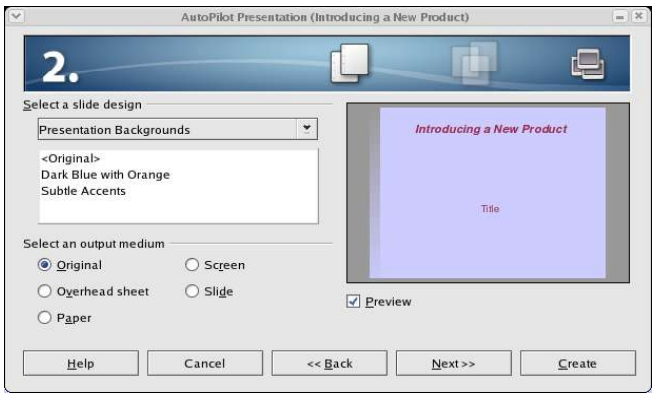


Fig. Impress AutoPilot for a new presentation

Template Management

If a big portion of time is going to be spent giving presentations, its very professional to have the presentation look like each other, in the form of a template (same logo position, copyright notices, etc...). Once a template is created (or downloaded from the Internet), you manage templates via the:

File --> Template --> Organize

Slide Design

A quick way to get slides done is via turning on:

View --> Toolbars --> Presentation

This provides a pop-up menu option that allows you to insert slides, or even modify the slide layout of the current slide.

Views

There are several views in Impress, and some have over-lapping names, but with different functionality! At the top-right-hand-corner of your screen, just above the scroll bars, you'll notice five buttons that look like what you see below.

Fig. View Buttons

The six options for workspace views are:

1. Drawing view – default, for slide design.
2. Outline view – overlook of the presentation.
3. Slide view – birds eye view to add, change, switch slides around.
4. Notes view – add speaker notes.
5. Handout view – how handouts get printed.
6. Start slideshow – run the presentation.

All the views can also be accessed via:

View --> Workspace

It is also worthwhile to note that at the bottom-left-hand-corner of your screen, where the slide tabs are displayed, there are more views to know about. You are typically located in the Slide View (same name as above, but different functionality since its on a different bar!)

Fig. More views

However, Master Views are supported and to access this view, it is the second button from the left. You can have master views of all workspace views (i.e. a master view of the slide itself, notes, and handouts). The Layer view allows layering of slides (adding and removing), and layers can be non-printing or non-displayed on screen, but printing only.

Jazzing up the presentation

Objects, like video, Java applets, music, and even other graphics can be added (embedded) to a presentation very easily. To perform this, the following menu is useful:

Insert --> Object

Keep in mind that OpenOffice.org will only play content provided all relevant plug-in's are installed. For sound playback, it assumes an already configured sound-card, otherwise it will not work.

Effects are another supported feature in Impress and consist of things like slide transitions, mouse-driven bullet-points, and even drawing animations. To get to the effects pop-up:

Slide Show --> Effects

A common effect is one where each bullet point appears upon a mouse-click. This event-driven effect is easily performed via selecting the "Appear" effect. Once that is selected, ticking the green tick (in acceptance) is necessary.

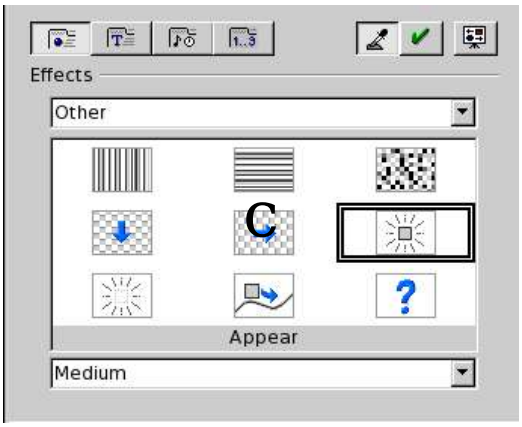


Fig. The Appear Effect

Performing slide transitions are also performed via the effects menu. However, to create animations, the menu is:

Slide Show --> Animation

Here simple animations can be created, like a bouncing ball along a line, for instance. This is done simply via:

1. Draw a circle (ball!).
2. Then draw a curve, as the path the curve should travel.
3. Select the “Move along curve” effect, and apply it (as in the figure below).
4. Now when the presentation is run, the ball will move along the line.

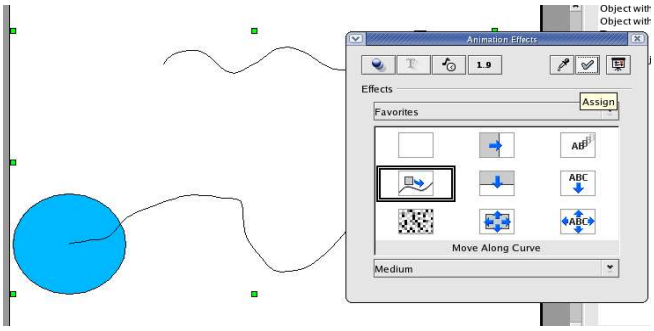


Fig. Application of moving along the curve effect

Conclusion

OpenOffice.org has the ability to be a very useful software package – it includes very powerful, free alternatives to satisfy average office suite requirements. As this is only scratching the surface, there are plenty more resources available out there, so please, use the available documentation to its fullest.

Using the OpenOffice.org Suite

Introduction

OpenOffice.org (OOo) is a complete office suite, featuring a word processor (Writer), a spreadsheet application (Calc), and presentation software (Impress). Besides these fundamental office applications OOo also includes a vector drawing tool (Draw), allows database access, allows the publishing of documents in the Portable Document Format (PDF) and presentations in the Flash (SWF) format!

The OOo package is fully inter-operable with the Microsoft Office suite.

Getting around the package

As a first stop for information, it is important to know how the Help system works. To get help:

Help --> Contents

The search function is very useful, and pay attention to the Options (where you can get help for the individual components in OpenOffice.org).

Setting up OpenOffice.org preferences so that it works the way you want it to is significant. The entire controls for this are available at:

Tools --> Options

Here you can setup settings like the default measurement units, font substitution, language types and many more options. Saving a document automatically is not setup by default, so turning this feature on might be helpful: you find it at the Load/Save option, under the General sub-section.

There are three important toolbars to know:

- Main toolbar – this is typically located right below the menus, and contains items like new document, save a document, exporting to PDF, copying & pasting, as well as access to the Navigator, Stylist, and Gallery.
- Object toolbar – this is right below the main toolbar, and has access to font control, and other attributes of objects.
- Function toolbar – located at the left-corner of the screen, and contains many options including quick table generation, insertion of objects, and many more.

Writer

This is a powerful tool for creating professional documents, reports, newsletters and so on – it is a word processor that allows easy integration of charts and pictures, as well as other OpenOffice.org-compatible documents. It can create everything from a simple letter to books, with professional layouts, with the use of styles.

Start it from the Main Menu by,

Main Menu --> Office --> OpenOffice.org Writer

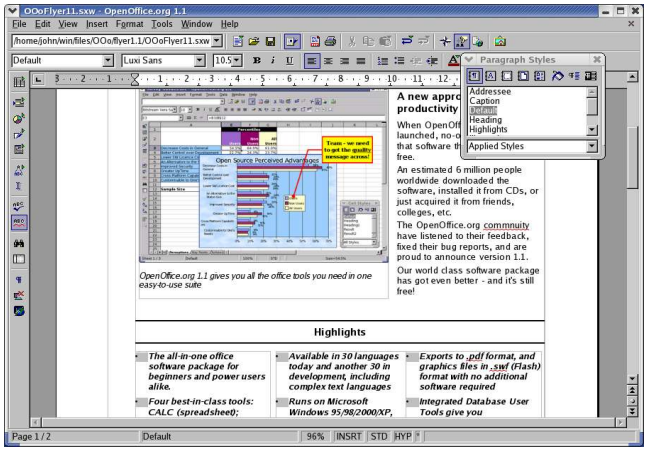


Fig. OpenOffice.org Writer

You are now presented with the word processing portion of OpenOffice.org, and the interface is rather similar to other word processing tools available. Rather than providing guidance throughout the entire package, we will just concentrate on a few tasks at hand.

Common Functions

Functions of the word processor can be controlled via the toolbars located at the top of the screen. On the first row, file actions like opening and saving files can be performed, while on the second row, changing the font, size, and style (bold, underline, or italics) are located there.

They can also be controlled by the menus that are common through packages:

- File --> New --> Text Document - creates a new empty, untitled document for you to work on.
- File --> Open - opens the file.
- File --> Close - closes the document you are working on. If changes have been made since your last save, you will be prompted to save or discard those changes.
- File --> Save - saves the document you are currently working on.
- File --> Save As... - saves an updated version of a document in a different location, with a different name, from the previously saved version.

Common Operations

For operations while writing, it is common to want to select a lot of text, copy it, maybe cut it from its current location and paste it elsewhere, or even undo an action. All this is possible with the office suite, and such options are available at the Edit menu. A few common options are:

- To copy text: select the text with the mouse, then select Edit --> Copy. Now the selected text is kept in memory for use elsewhere.
- To paste text: find the spot where text needs to be placed, place the cursor there, and then select Edit --> Paste.
- To cut text: this means that the selected text will be removed from the current location and kept in memory, to be placed elsewhere. Doing this is exactly like how a copy should be performed, except selection Edit --> Cut instead.
- To undo an action: Select Edit --> Undo. It will display the command that it is undoing at the moment.

By browsing the menu, there are also keyboard shortcuts located next to it. Once more proficient use of the package

occurs, it is much quicker to use keyboard shortcuts like Control+C for Copy, and so on.

Formatting

Formatting text is as important as writing the text, and Writer provides many formatting options, including the Stylist. Individually, you can also format the character (current selected item, or even a whole word), the paragraph, or even the page.

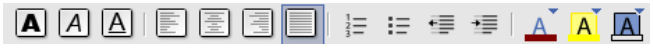


Fig. Part of the Object toolbar (Writer)

Some of the quick format options include **bold**, *italics* and underline. These options are available at the toolbar at the top of the screen.

Text alignment plays a large role in controlling how portions of the document will look. For example, an address field at the top of your letter will have such details right-aligned, while the body and rest of the base text will be left-aligned. This is all controlled by the four-icons that are located next to the bold/italics/underline icons, providing such options as: right-align, centre-align, left-align and justified. When text is justified, it looks exactly like what you're reading now! (a more professional end-to-end stretch of the text.)

Let's switch to the end of the toolbar, and notice that the options there including providing a paragraph background – which is good for highlighting a paragraph or several paragraphs of text, in colours that you choose. You can also highlight text (like you would with a highlighter and paper!), and change the font colours all with the icons there.

Aligning text by indenting it is also another feature available as part of the object toolbar. Left/right alignment of text is provided, and if text is already entered and you want to left-align it, selection of text (or having the cursor at the paragraph) must happen first, before text is indented.

Those were just quick controls. To get full control, using the **Format menu** is ideal. Controls are more varied here.

Styles

Consistency throughout a document is important – it was earlier said that writing books using OpenOffice.org is possible. So there must be a way to handle long and large documents in a consistent fashion, with similar fonts for headings, sub-headings, text, and other attributes within a document.

OpenOffice.org includes a powerful feature known as styles, and this is accessed via the Stylist (get this by hitting the F11 key, or clicking its icon on the main toolbar). Notice the floating window, which is most likely active at the “Default” style. By right-clicking on the style, there are options to modify the style, or create new custom ones.

By going to the modify option, the style can be customized via many varying attributes including spacing, alignment, font, emphasis, colour and many more. Once suitable styles have been pre-defined in the document, they can be used on existing text just by selection, and double-clicking on the style name.

Just a little bit more...

Now that the gist of the Writer package has been covered, there's just a little more to know.

Writer has a built-in spell checker. This can be accessed via:

Tools --> Spellcheck

The option to auto-spellcheck means that while typing, Writer will dynamically check your spelling, and if it detects an error, it will output a red-line at the bottom of the misspelled word. Keep in mind that the spell checking is based on the current language that is in use. This can be changed via:

Tools --> Options --> Language Settings --> Languages

Accessing word counts in the document is different to most other packages on a default install of OpenOffice.org (this can differ with several Linux distributions' offerings):

File --> Properties --> Statistics

It is under the Statistics tab that the word counts and other relevant document counts are based. On certain vendor modified distributions of OpenOffice.org, going to the Tools --> Word Count menu will allow the Statistics dialogue box to be displayed automatically.

The AutoCorrect/AutoFormat (Tools --> AutoCorrect/AutoFormat) options have replacement tables (so that CDs really are valid, and will not be changed to Cd, for instance). There are also word completion options (very useful, as the software starts thinking for you) and settings to make them more user-friendly.

The Navigator is a yet another useful tool (get this via hitting the F5 key or clicking its icon on the main toolbar), especially when dealing with larger documents. It supports jumping to bookmarks, notes, any particular object, and even creates a table-of-contents on the fly, based on the styles that are being used!

Calc

This is the spreadsheet component of the OpenOffice.org package, and contains many useful features, including an array of functions and plenty of charting options. It is fully inter-operable with Microsoft Excel, though the function separators differ in the two packages.

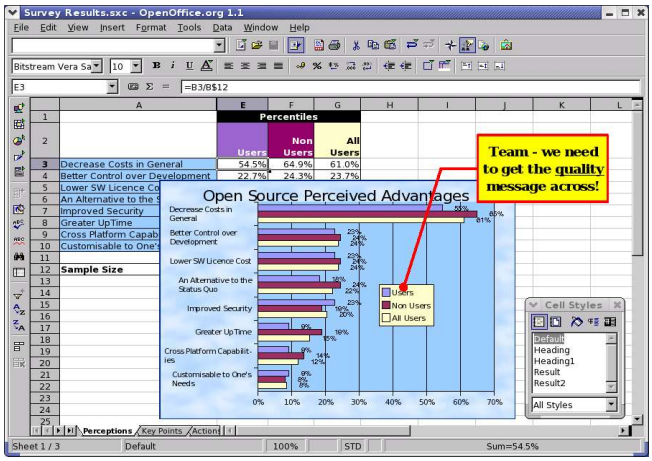


Fig. OpenOffice.org Calc

To start this, it is available via

Main Menu --> Office --> OpenOffice.org Calc

or if you already have an existing window of OpenOffice.org open,

File --> New --> Spreadsheet.

Spreadsheets contain many rows and columns, and each row and column combination is called a cell (like A1, B4, and so on). Upon inputting text into a cell, you might realize that the text is wider than the cell allows for – this can be re-sized via right-clicking the cell, and selecting the Format Cells option. There under the Alignment tab, selecting Line Break is what is required.

Formatting

Like other parts of OpenOffice.org, Calc also comes with the Stylist. But let's get around to understanding the various

differing formatting options available in this component of the package.

If there is some information that you already have created, and the area should be formatted, one particular quick and easy option is to use the AutoFormats available in Calc. This is done after selecting the area then:

Format --> AutoFormat

These are pre-defined styles that are available in Calc, and if you have created your own particular style, you can add them into your new AutoFormats.

In the object toolbar, there is an option to set the font colour within the cell. There are also options to increase/decrease the indents within a cell, and in the image below, controls for enabling:

- Currency
- Percentage
- Add/Remove significant decimal places

Fig. Part of the Object toolbar (Calc)

These are quick controls, and accessing them is as simple as clicking the icons that represent them, and automatically the cell will be formatted as stated. Not only can Borders be set easily, and cell backgrounds too, but the alignment of text within a cell can also be set. This can be either as a top aligned, centre aligned, or bottom aligned.

Now that most of the formatting options are known, it is easy to apply Styles to the spreadsheet – bring up the Stylist by hitting the F11 key, and you'll notice that cell styles (that control all elements, including formatting) and page styles can be set (the latter controlling margins, headers/footers, and borders).

Spreadsheet basics

There are a few points to note when using a spreadsheet. One of them is that calculations are performed in a left-to-right format, with algebraic ordering rules. This means it deals with brackets ("()") first, then division ("/"), multiplication ("*"), addition ("+") and finally subtraction ("-").

When applying calculations, keep in mind the range of included cells. When using a function like =SUM(), and using the argument =SUM(A1:A4), it means it looks for the sum of the cells A1, A2, A3, and A4. These operations can also be performed on non-consecutive cells, so, =SUM(A1;A4;A7) just executes the sum of cells A1, A4 and A7.

If you have used Excel before, it would be relatively common to use a comma (",") as a separator character between the parameters, however, with OpenOffice.org Calc, the separator character is a semi-colon (";"). So for the function to validate correctly, an expression such as =IF(B3>0;A1-A2;A1+A2) is correct (as opposed to replacing the ";" with ",").

Building functions

To perform calculations, spreadsheet make use of functions. Common functions include =SUM() for summation, =AVERAGE() for the average value of cells, and so on. As an aid to the novice user, OpenOffice.org provides a Function AutoPilot. This is a wizard to help build formulas, and find problems with existing expressions.

Fig. Function AutoPilot Button

Located next to the universal sum function, is the Function AutoPilot. If you click on in, a pop-up dialogue appears.

1. In the Functions tab, you can filter viewable/accessible functions via category, and the option to choose a function is shown. Use your mouse to choose a function that you plan on using.

2. Once the correct function is selected, and the action that it performs is agreeable (it is displayed on the right of the dialogue), select Next to move on.

3. Now you are allowed to input numbers. Assuming the AVERAGE function was chosen, in the number 1 field (for example), there are options to either enter a function or select a range of cells.

4. Use select a range of cells, and now a different dialogue pop's up and you can use the mouse cursor to select a range of cells. Click on OK, and you're done!

That is a very easy way to build a formula, which requires no pre-requisite knowledge about what formulas exist in Calc.

Sorting

A big part of dealing with spreadsheets involves a lot of sorting and filtering of data. To sort a dataset, selecting the active cells, then clicking

Data --> Sort

will bring a pop-up dialogue that has options for sorting the data based on the columns present, as well as if the data should be ascending or descending.

Charting

Converting data into information is a process usually accomplished well by creating graphs and charts – it is a lot easier to infer based on visual graphics, rather than lots of numbers. Calc provides a charting wizard that will allow this to be automated rather easily, with a lot of predefined settings.

1. Select the cells that are to be charted, and then go to Insert --> Chart.

2. The range is pre-selected, and certain options are provided (like where the resulting chart is); just leaving the pre-selected options will be good for the exercise. Click Next.

3. A type of chart is to be chosen – common charts include pie chart, line graphs, or even bar graphs – it all depends on the information being represented.

4. Select the defaults, and create a chart. (You should now see a bar chart created).

Now that a chart has been created, it is not static in the sense that it cannot be edited – you can control each and every aspect of how the chart looks. Double-click the chart, and you will now go into edit mode. Notice the toolbar by the left-hand side of your screen has changed? This is in direct response to it being in edit mode, and a lot of properties can be changed here.

There are plenty more features, like data filtering, scenario creation, and goal seek, which once you get more advanced with spreadsheet know-how, you will end up making use of.

Impress

No office suite is complete without a presentation piece, and OpenOffice.org shows its colours with Impress, the presentation piece of the suite. To start it, it is available at

Main Menu --> Office --> OpenOffice.org Impress

or if you already have an OpenOffice.org window open, its available at

File --> New --> Presentation.

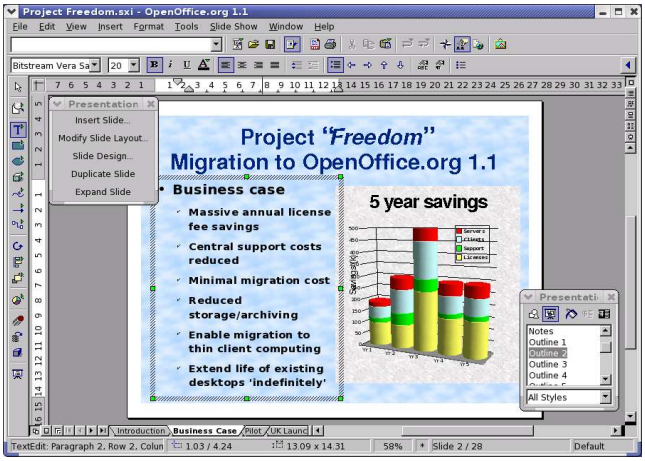


Fig. OpenOffice.org Impress

Unlike other components of OpenOffice.org, when you start Impress, you are presented with an AutoPilot, to start creating your presentation! This gives you options to start a presentation with an empty template, or even with one of the pre-defined templates. A preview dialogue is available, and once all options are selected (and Next is clicked, to move on), you get a basic presentation.

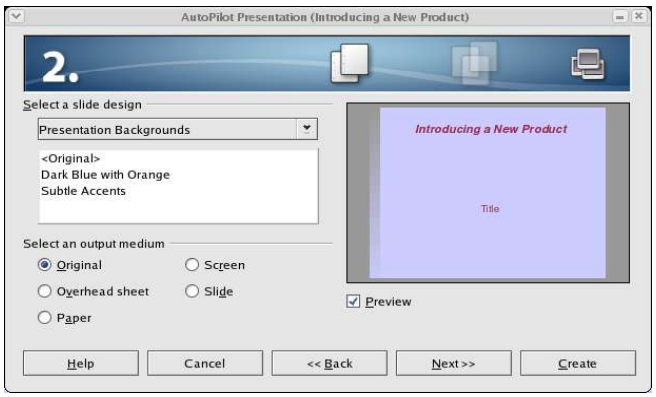


Fig. Impress AutoPilot for a new presentation

Template Management

If a big portion of time is going to be spent giving presentations, its very professional to have the presentation look like each other, in the form of a template (same logo position, copyright notices, etc...). Once a template is created (or downloaded from the Internet), you manage templates via the:

File --> Template --> Organize

Slide Design

A quick way to get slides done is via turning on:

View --> Toolbars --> Presentation

This provides a pop-up menu option that allows you to insert slides, or even modify the slide layout of the current slide.

Views

There are several views in Impress, and some have over-lapping names, but with different functionality! At the top-right-hand-corner of your screen, just above the scroll bars, you'll notice five buttons that look like what you see below.

Fig. View Buttons

The six options for workspace views are:

1. Drawing view – default, for slide design.
2. Outline view – overlook of the presentation.
3. Slide view – birds eye view to add, change, switch slides around.
4. Notes view – add speaker notes.
5. Handout view – how handouts get printed.
6. Start slideshow – run the presentation.

All the views can also be accessed via:

View --> Workspace

It is also worthwhile to note that at the bottom-left-hand-corner of your screen, where the slide tabs are displayed, there are more views to know about. You are typically located in the Slide View (same name as above, but different functionality since its on a different bar!)

Fig. More views

However, Master Views are supported and to access this view, it is the second button from the left. You can have master views of all workspace views (i.e. a master view of the slide itself, notes, and handouts). The Layer view allows layering of slides (adding and removing), and layers can be non-printing or non-displayed on screen, but printing only.

Jazzing up the presentation

Objects, like video, Java applets, music, and even other graphics can be added (embedded) to a presentation very easily. To perform this, the following menu is useful:

Insert --> Object

Keep in mind that OpenOffice.org will only play content provided all relevant plug-in's are installed. For sound playback, it assumes an already configured sound-card, otherwise it will not work.

Effects are another supported feature in Impress and consist of things like slide transitions, mouse-driven bullet-points, and even drawing animations. To get to the effects pop-up:

Slide Show --> Effects

A common effect is one where each bullet point appears upon a mouse-click. This event-driven effect is easily performed via selecting the "Appear" effect. Once that is selected, ticking the green tick (in acceptance) is necessary.

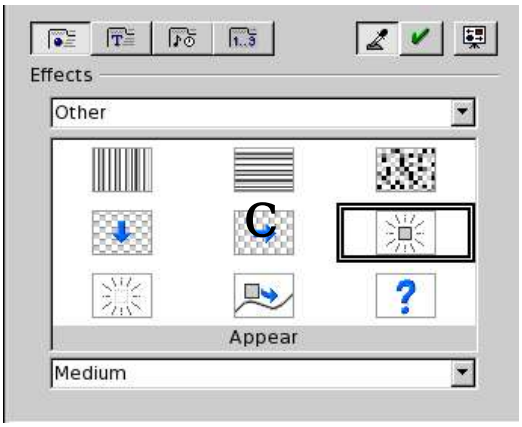


Fig. The Appear Effect

Performing slide transitions are also performed via the effects menu. However, to create animations, the menu is:

Slide Show --> Animation

Here simple animations can be created, like a bouncing ball along a line, for instance. This is done simply via:

1. Draw a circle (ball!).
2. Then draw a curve, as the path the curve should travel.
3. Select the “Move along curve” effect, and apply it (as in the figure below).
4. Now when the presentation is run, the ball will move along the line.

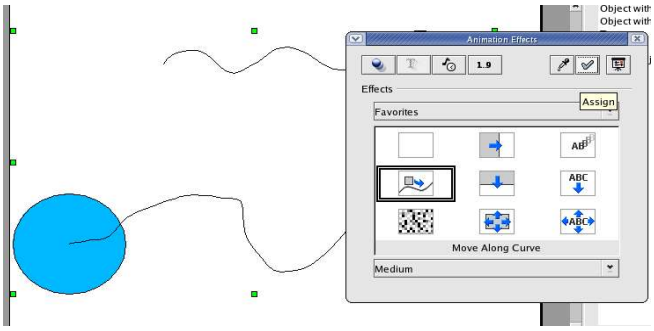


Fig. Application of moving along the curve effect

Conclusion

OpenOffice.org has the ability to be a very useful software package – it includes very powerful, free alternatives to satisfy average office suite requirements. As this is only scratching the surface, there are plenty more resources available out there, so please, use the available documentation to its fullest.

The Shell

Introduction

The Linux/Unix shell refers to a special program that allows you to interact with it by entering certain commands from the keyboard; the shell will execute the commands and display its output on the monitor. The environment of interaction is text-based (unlike the GUI-based interaction we have been using in the previous chapters) and since it is command-oriented this type of interface is termed Command Line interface or CLI. Before the advent of GUI-based computing environments, the CLI was the only way that one can interact and access a computer system.

Up until now, there was never a need to type commands into a shell; and with the modernisation and creation of a lot of newer GUI-based tools, the shell is becoming increasingly un-required to perform many tasks. But that said, the shell is a very powerful place, and a lot is achieved through it. A lot of the front-end GUI methods of doing things have similar ways and means to get done with using the shell. Professional Linux and UNIX users find the shell very powerful, and an introduction to at least the basic shell usage is useful.

Getting to a shell

Since it is most likely that you are in the graphical desktop environment now, the underlying shell that is available is not displayed. To access a shell, try the following key combination,

Control + Alt + F1

Where F1 can be replaced by F2, F3, and so on. The graphical desktop tends to run in F7 or F8, so to go back to your graphical desktop screen, just hit Control + Alt + F7. These are virtual terminals.

Alternatively, you could get to a Terminal application, so you can have a shell while your in the graphical desktop environment (this is much preferred, and will be used throughout this Chapter). To do this, go to:

Main Menu --> System Tools --> Terminal

Or right-click on the desktop, and click on the Open Terminal option. This terminal is equivalent to the virtual terminals mentioned earlier, except now you don't have to switch screens – you can just minimize or maximize the terminal (or if you're done, you can close it).

Some useful commands

Now that you are at a terminal, you might as well input some commands. For example, when you start a shell, display such as below (or similar) will be seen (and this can be configured to your liking!):

```
[-(byte@hermione)-(pts/4)-(05:34pm:05/06/2004)-]
[-(~)>
```

The cursor blinks, waiting for input. To this, some of the more used and useful commands include:

- ls – list files in the current directory.
- cd – change working directory. If your current path is /home/username/Trash for instance, typing “cd” will bring you back to /home/username.
- mkdir – make a new directory
- rmdir – delete a directory (must be empty)
- cp – invoked such as “cp currentFile newFile”, and is used to copy files.
- mv – invoked such as “mv currentLocation newLocation”. This is used to either move or rename files.
- rm – invoked such as “rm myFile”; it is used to delete files permanently.
- pwd – prints the working (current) directory.
- cat – concatenate files (can be used to join them together), and prints its output to standard output (the terminal screen). Used like: “cat myFile”.
- less – allows for file viewing in the shell, and is most useful for text files; invoked like “less myFile”.
- find – can be used to find files via the command line. Example usage could be: “find . -name toc”, which looks at the current directory (defined by “.”) for any files with the name “toc”.
- locate – picks entries from a database, that is updated regularly; invoked via “locate myFile”. Its much quicker than find (since it only searches a database), but might not be as quick to update as find (the update of the database might happen once every day only).
- date – display the current date! This can also be used to set the date of the system (but administrator privileges are required).
- history – built-in shell command for the BASH environment that shows the last run commands.

As always, these commands just begin to scratch the surface of the capability of the shell. There are thousands of such commands available on your system! And keep in mind that each and every command comes with options, that are usually executed via the *-flag* – again, the man pages list all useful commands. For instance the command

rm -i

will prompt when deleting a file, so you have to either say 'y' if you're sure, or 'n' if you do not want to delete the file.

```
[-(/tmp)> rm -i usr.bin
rm: remove regular file `usr.bin'? y
```

A Few More Concepts and Shortcuts

Now that you've seen some commands that are useful in the shell, its important to know a few more concepts. For instance, the tilde (“~”) represents the home directory, so rather than typing /home/username it can be represented via a '~'. This means less typing for you.

```
[-(~/MyOSS-Stuff/IOSN)> pwd
/home/byte/MyOSS-Stuff/IOSN
[-(~/MyOSS-Stuff/IOSN)> cd ~
[-(~)> pwd
/home/byte
```

So in that example, I was located in /home/byte/MyOSS-Stuff/IOSN, and just by issuing a “cd ~”, the shell has brought the current working directory to /home/byte.

A dot “.” means the current directory. While “../” will mean the parent directory. This can be nested to include “../../” and so on, till it reaches the top level directory /.

Input/output Redirection and Pipes

Running a command by itself with a lot of output doesn't seem all that useful. For instance, if there are many files in a directory, running a command to list the directory like,

ls /usr/bin

will result in about 2100 lines being displayed on the screen! To actually get any useful information out of it, you might want to dump the output of the ls command to a file; or maybe use a utility like less to view it. All this is possible thanks to input/output redirection and pipes.

Input redirection is performed using < or <<, while output redirection is done via > or >>. A point to note is that when using >, it just recreates the file, even if the same filename exists, while >> concatenates the output to the same file, causing it to possibly be double in size (if its the same output).

A pipe ("|") is used to pass the output of the command not to a file, or to the screen, but to the next utility. Pipes can be nested, so you can pass the data through several utilities before you can get the useful information that you want. Let's dive into some examples!

```
1. [-(/tmp)> ls /usr/bin >> usr.bin
2. [-(/tmp)> wc -l usr.bin
3. 2171 usr.bin
4. [-(/tmp)> ls /usr/bin >> usr.bin
5. [-(/tmp)> wc -l usr.bin
6. 4342 usr.bin
7. [-(/tmp)> ls /usr/bin > usr.bin
8. [-(/tmp)> wc -l usr.bin
9. 2171 usr.bin
```

Note: the line numbers are added for clarity, and are not included in the shell output!

In line 1, the output of the directory listing of /usr/bin gets placed in a file called usr.bin. On line 2, a new utility called 'wc' is used (this is used to print the number of lines in the file (as it gets passed the -l option) – its output is at line 3. The same command is then repeated on line 4, and now, the file is double the size as per line 6! That is because the >> output redirection was used, which has concatenated the two outputs together. Notice that in line 7, a single > is used, and in line 9, it shows that the file has been over-written with the new contents.

```
[-(/tmp)> ls /usr/bin | grep cancel
cancel
cancel.cups
```

The above is an example of how a pipe is used. After listing the files, the output is passed on to a utility called grep (which basically searches for a pattern, and prints the output) and the string being searched for is "cancel". It comes back with two matches. Similarly, a command like:

ls /usr/bin | less

Will place the output of the directory listing into the less pager so that it can be scrolled through easily. And for another example as to how pipes can be nested, issuing:

```
[-(/tmp)> 'ls' /usr/bin | grep auto|wc -l
19
```

sends the output of the directory listing of /usr/bin to grep, which then searches for the string "auto", and then wc prints how many times it occurs in lines.

A useful command string that a lot of systems administrators tend to use would be:

```
[root@hermione root]# tail -f /var/log/messages
Jul  5 12:04:02 hermione last message repeated 13
times
Jul  5 16:17:17 hermione last message repeated 17
times
Jul  5 16:17:28 hermione last message repeated 18
times
Jul  5 16:17:32 hermione
```

A 'tail' displays the last ten lines of the file, and the -f option means that if there are more logs, it gets displayed (via it being appended to the bottom).

Where do I get help?

Rather than get frightened off the shell, there are some sources of help, in the event that you aren't sure what you're doing in the shell.

Man Pages

These are manual pages, for each and every command that resides on your system. This is a first point of reference, and it is invoked by:

man *command-name*

e.g.

```
$ man man
```

The above runs man on itself, explaining a bit about the manual page system.

Info Pages

This is the new GNU project method of distributing manuals, and info pages are a lot more comprehensive than man pages. It is invoked by:

info *command-name*

e.g.

```
$ info info
```

The above runs info on itself, and provides some useful information as to how info can be used, and how you can navigate info documents.

Other Useful Commands (for help)

While still on the topic of help, there are a few more useful commands that you want to know about:

- whatis – invoked by "whatis package-name" and it provides information about the tool that whatis recognizes (and has in its database).

- apropos – invoked by "apropos *string*", and it provides strings matching what is located in the whatis database. This is most useful when you don't know what command you want to run, but have an idea that as to what it should be dealing with (so apropos mail should provide all sorts of mail clients that are available on your system).

Conclusion

This is the power of Linux and UNIX command lines. There is much more to learn, as there are different shells, and different shell syntaxes available. Also, regular expressions are useful, and there are plenty more utilities available, and if a liking towards the shell is taken, shell scripts can be written to perform a lot of tasks, including backing up directories and more!

Exercises

1. Open up a shell on your Desktop and perform the following:

   - find the name of the directory you are in
   - list out the contents of the current directory
   - list out the contents of the directory */usr/bin*
   - check the current date and time

2. Change directory to your home directory and make a new sub-directory there named *Temp11* and change directory to it

- copy the following files from the */etc* directory to the directory Temp11*: services, motd, fstab, hosts*
- concatenate the files copied above into one single file called *file1*
- count the number of lines present in the file *file1*
- delete the four files listed above in the directory Temp11

Package Maintenance and Update

Introduction

While a simple piece of software may consist of only a single executable file, most of the software applications available and running on your system are more complex. A typical application or utility will consist of several executable files, configuration files, documentation notes and guides and possibly even libraries too. All these files and information about where to place them in the filesystem are put together in what is refered to as a package. So when we talk about the installation or upgrading of applications, we are refering to the installation and maintenance of these packages.

There are many packaging formats available in Linux, and some are easier to use than others. In this Chapter we shall cover tarballs, the RPM packaging format (RPM) and the Debian packaging format (DEB).

Tarballs

Tarballs are the standard, and are common with file extensions such as ".tar.gz" or ".tar.bz2". This is the generic, distribution-free method of distribution software packages in the Linux world. However, tarballs are not very user-friendly; for example, to get a tarball from the Internet running, one might have to issue the following commands from the command line in a shell,

```
# bunzip2 myapp.tar.bz2
# tar -xvpf myapp.tar
# cd myapp
# ./configure
# make
# make install
```

This is a tedious task, and involves getting the software to compile before being able to run. If know-how is lacking, this method will also cause a lot of grief, as sometimes during the "configure" stage, dependencies to get it running aren't met.

This is the aim of package management formats like RPM and DEB – to ease the burden of dependency resolution, so that the end-user will just install the software with ease, and if dependencies are required, they get installed along.

Keeping up-to-date

On Red Hat Linux/Fedora Core systems, there is a graphical front-end called up2date. It is invoked by the little icon at the bottom of the notification area (nearby where the clock is located).

Fig. Up2date Icon

It can also be accessed via,

Main Menu --> System Tools -->
Red Hat Network

When you run it, it will require that you enter the root password (as this effects the system, administrative rights are required). Once that is entered, an image like the one below is displayed.

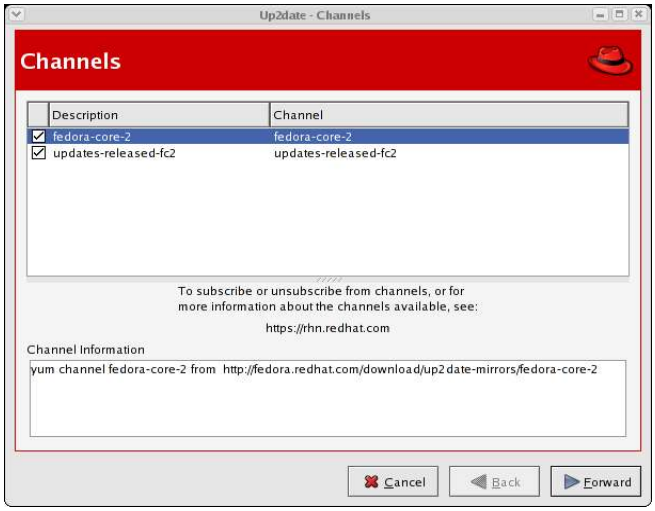


Fig. Up2date Channels Screen

Just click Next (make sure the Internet connection is enabled, if it is not enable it before proceeding further) – it will contact the online servers and find packages that are installed and need updating, and it will prompt you along the way. Once it is done, your system will be updated (and the blue icon with a tick will be displayed). If your system is not updated or the packages not kept up to date, a red icon with an exclamation mark will be displayed and it will be blinking.

On a Debian GNU/Linux system, a tool known as "apt-get" is available on the command line. On a default install, that is all that is provided, however, a good GUI front-end to it is Synaptic, which can be downloaded from the Internet via,

```
# apt-get install synaptic
```

Fedora also comes with another updating tool known as yum, and this can be invoked via the command line such as,

```
# yum update
```

To upgrade your current system,

```
# yum upgrade
```

can be invoked.

Keep in mind that keeping an updated system is very important, as when security holes or bugs are found in software and get fixed, you will always be kept abreast of such developments. A non-updated system can be an insecure system, and that is not good practice.

Installing new packages

If a package is available on your Red Hat Linux or Fedora Core CDROM, there's an Add/Remove Applications application that is useful. It is invoked via,

Main Menu --> System Settings -->
Add/Remove Applications

It will ask you for the root password, and once that is provided, it will display all applications that may be installed. Once you have ticked the applications that you want installed, you just need to click "Update" to install. Change the discs as you are prompted, and once this is done, you will have the software installed.

However, in the open source world where applications change quite often, and fixes are posted, this method might mean you get out-dated software. This is where tools like yum and apt come into play.

To search the yum database for a piece of software, you can invoke,

```
# yum search xargs
```

where xargs is an example of an application that needs to be installed. Yum will report if it finds xargs, and if its successful, performing,

```
# yum install xargs
```

will be all that is required. If xargs calls for any dependencies, it will be resolved automatically, and those packages get pulled in automatically too.

This is similar with Debian and apt.

```
# apt-cache search xargs
# apt-get install xargs
```

If you want to install a downloaded RPM or DEB file manually, it can be performed like,

```
# rpm -ivh xargs.rpm
```

or

```
# dpkg -i xargs.deb
```

And if you're manually upgrading a package, use,

```
# rpm -Uvh xargs.rpm
```

The above command will upgrade the package if it is already installed or install it if it is not. To perfrom an upgrade only if the package is curently installed, use,

```
# rpm -Fvh xargs.rpm
```

There are many more options to pass to the rpm, dpkg, yum, apt-get and apt-cache tools, and the best way to learn more, would be to read their manual pages. It is also worthy to note that apt-get is available for RPM-based systems, so versions for Red Hat Linux or Fedora Core (or even SuSE or Mandrake) are available as a download from the Internet.

Chapter 13: Getting More Info (and Help!)

The previous chapters have provided a guide on how to use the graphical desktop of a typical Linux system. However, they have just scratched the surface of the features and functions of the Desktop environment and the applications therein. In this section we shall look at the resources available to a user to get more information and help.

Online Documentation

Much of the details on how to use and exploit further the software available is available as online documentation on the system itself. The online documentation is available in two types, the *Help* from the Main Menu and/or applications and the text-based Unix-style *man* and *info* commands.

Desktop Help

The Desktop Help can be invoked from the Main Menu,

Main Menu --> Help

Invoking this will display the screen below.

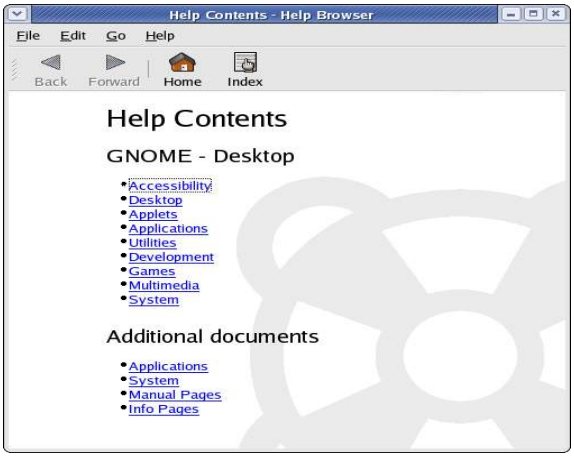


Fig. GNOME Help Screen

The Help content is divided into several main categories. So you will need to select the appropriate category to view the help content of interest. Most of the information on how to use the Desktop can be found from the Help here. For example, to view the help information on the File Manage, select,

Desktop --> Nautilus File Manager

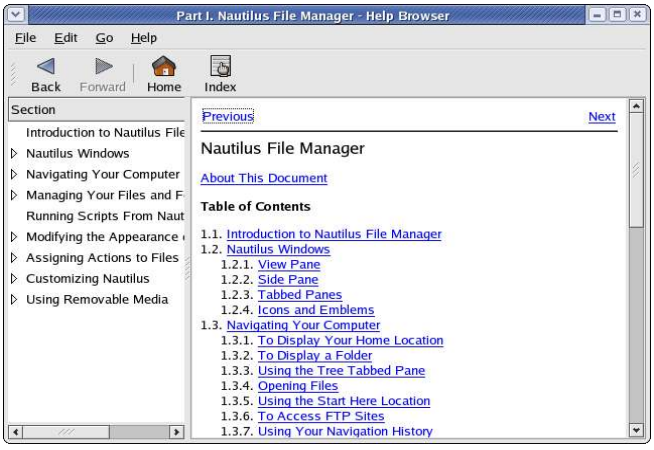


Fig. File Manager Help Screen

Help Selection in Applications

Most of the Desktop applications have a Help button in their main menubar at the top. Selecting this will give you more information on how to use the application. The Help screen for the OpenOffice.org Writer application is displayed below.

Fig. OpenOffice.org Writer Help Screen

Man and Info Pages

As discussed in Chapter 11, from the command line interface using a Shell, it is possible to access a comprehensive help system on the commands available via the *man* and *info* commands. For example, to find out more on how to use the directory listing command, *ls*, open up a shell (see Chapter 11) and at the command prompt enter,

```
$ man ls
```

More detailed information on certain commands may be found using the *info* command, e.g.

```
$ info ls
```

To learn how to use the *man* and *info* commands, make use of these commands themselves e.g.

```
$ man info
$ man man
$ info info
$ info man
```

The Internet (WWW)

There is a lot of information available on the WWW on all the software available on the system. These may be classified broadly as follows:

- Websites of specific software projects
- Websites of specific Linux distributions and/or vendors
- General Linux websites
- General Open Source websites

Websites of Specific Software

Below are links to the websites of the software applications discussed in this guide.

GNOME – www.gnome.org
KDE – www.kde.org
The Freedesktop Project – www.freedesktop.org
OpenOffice.org – www.openoffice.org
Mozilla – www.mozilla.org
Ximian Evolution - www.novell.com/products/evolution/
gToaster – gnometoaster.rulez.org
Sane - www.sane-project.org
XSane – www.xsane.org
MPlayer - www.mplayerhq.hu
Xine – xinehq.de

XMMS - www.xmms.org
gThumb – gthumb.sourceforge.net

Linux Distributions and/or Vendors

Links to specific Linux distributions and vendors are listed below. In particular the website Distrowatch should be consulted for information and links to the hundreds of Linux distributions available today.

Fedora Linux – fedora.redhat.com
Debian Linux – www.debian.org
Slackware Linux – www.slackware.org
Redhat Linux – www.redhat.com
SuSE Linux – www.suse.com
Mandrake Linux – www.linux-mandrake.com
...
...
... many. many, many more
... (for links and information on many Linux and other OSS operating system distributions see the Distrowatch website below)
Distrowatch – www.distrowatch.org

General Linux Websites

Resources catering to new Linux users can be found in many of the website links below.

Linux Online – www.linux.org
Linux.com – www.linux.com
Linux.net – www.linux.net
Linux Headquarters – www.linuxhq.com
LinuxHQ.org – www.linuxhq.org
Linxu Today – www.linuxtoday.com
The Linux Documentation project – www.tldp.org
Linuxquestions.org - www.linuxquestions.org
The Linux Standard Base Project – www.linuxbase.org
Linux Journal – www.linuxjournal.com
Linux Gazette - www.linuxgazette.com
Linux Compatible - www.linuxcompatible.org

Free and Open Source Software Websites

In this section, general information on Open Source and Free Software may be obtained as well as news and updates.

The Open Source Initiative – www.opensource.org
The Free Software Foundation – www.fsf.org
Sourceforge – sourceforge.net
Freshmeat – freshmeat.net
Newsforge – www.newsforge.com
Open Source Development Network – www.osdn.com
Slashdot – slashdot.org
International Open Source Network – www.iosn.net
The Asian Open Source Centre – www.asiaosc.org
OSNews - www.osnews.com

Appendix: KDE (The K Desktop Environment)

Introduction

The focus for the large part of this guide has been with the GNOME Desktop. However, there is another popular graphical desktop environment out there known as the K Desktop Environment, affectionately known as KDE. It is included with most systems, and has a strong user-base, just like the GNOME Desktop. KDE offers an alternative desktop computing experience in that while the applications should all function in the same manner irrespective of the desktop environment chosen, the look and feel of the graphical desktop are different. Desktop-specific tools and applets may also be different from one environment to the next.

This section will briefly introduce KDE as well as highlight some of the more important differences from the user's perspective between the KDE and GNOME Desktops.

Login into KDE

To login to the KDE Desktop rather than the GNOME Desktop, at the graphical login screen, click on Sessions at the bottom, and then select the KDE option. Enter your username and password – there might be a pop-up warning asking if the change is for the current session or for all future sessions (this can be permanently changed using the programs "system-config-switchdesk" on a Fedora/Red Hat system and "switchdesk" on a Debian system.).

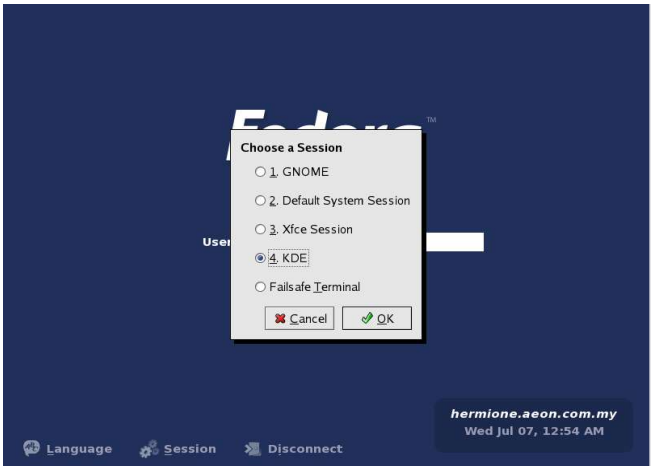


Fig. Choosing KDE at the Login Screen

After login, the KDE Desktop is displayed and as can be seen below is rather similar, but not identical, in appearance to the GNOME Desktop shown in Chapter 1.

The KDE Desktop

The KDE Desktop has similar components to the GNOME Desktop and their functionalities and usage do not differ much. So on the desktop we find the following familiar components: the Menu System, the Panel, the Desktop itself.

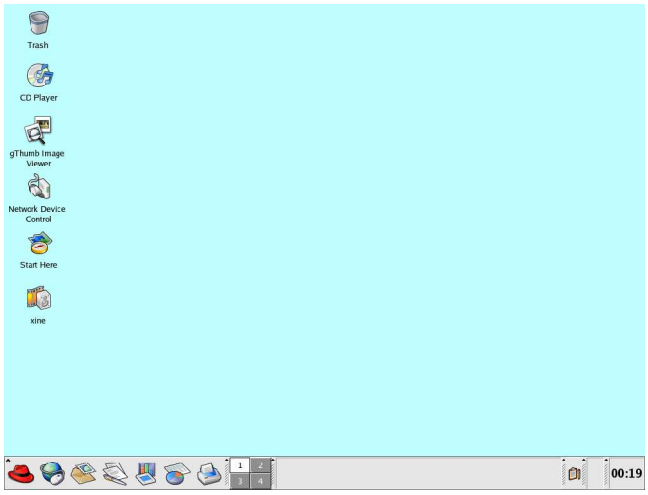


Fig. The KDE Desktop (Fedora Core)

On closer examination of the Desktop there are some subtle differences, For example, one of the things that set KDE apart from GNOME, would be the icon set. In contrast to GNOME, there is no "Computer" icon, but just some for your devices, and the usual "Start Here" set. KDE on systems other than Red Hat Linux or Fedora Core, will look a lot different, as the themes can be configured otherwise.

Fig. KDE Desktop Icon Set

Another item that you will notice is that if you right-click on the Desktop, the right-click pop-up menu differs from the GNOME offering. It allows you to create new documents, edit bookmarks (which are a handy feature to jump to places quickly with just a few clicks of the mouse button), and of course, run a command.

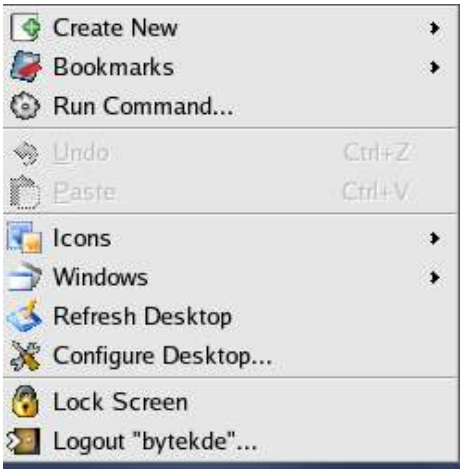


Fig. Right-click Menu in KDE

By clicking on the Main Menu, again, differences will be prevalent. But notice that the categories of applications are similar? It is just the look and feel, that seems to differ a little.

Fig. KDE Menus

Control Center

There is a central place where all the Desktop and system configuration can be set and viewed – the KDE Control Center.

Fig. KDE Control Center

The Control Center can be launched from the Main Menu,

Main Menu --> Control Center

This central place for configuration makes it very convenient for users and administrators alike. Using the Control Center is easy. To configure a particular setting or parameter, open up the section it is under and select the item to configure. For example, to change the background colour, select,

Appearance & Theme --> Background

Klipper – Clipboard Applet

Klipper is a unique feature of KDE – it provides for clipboard access in the GUI application environment. It allows a multitude of copying and pasting options, and works well between all applications. Using Klipper, one can cut and paste text seamlessly between applications running on KDE. To place Klipper on the Panel, right-click on the Panel,

Add --> Applet --> Klipper

Fig. Klipper

Konqueror - File Manager and Web Browser

Lastly, one major difference between KDE and GNOME is the file manager. In KDE, Konqueror is the default file manager. Konqueror provides all the functionalities one will expect from a modern file manager, including navigation of the filesystem, file/folder copying, renaming, deletion and creation and application launching.

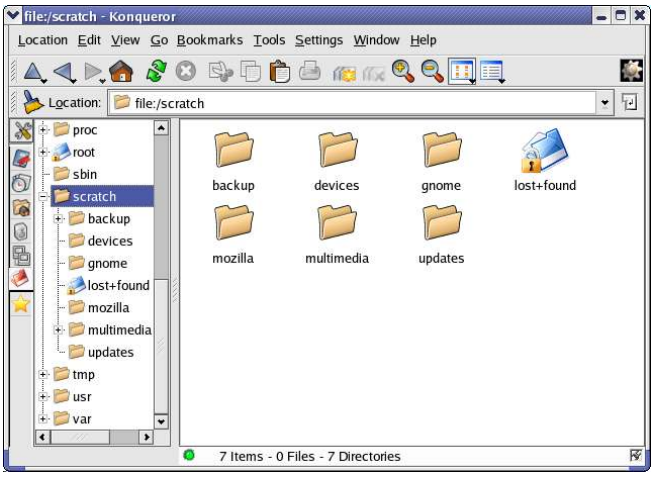


Fig. Konqueror File Manager

It is also able to display graphic image files and generate an image gallery web page from them. In addition. Konqueror is a standards-compliant web-browser and is perfectly capable of browsing the WWW on the Internet - just enter the website to go to in the Konqueror location bar.

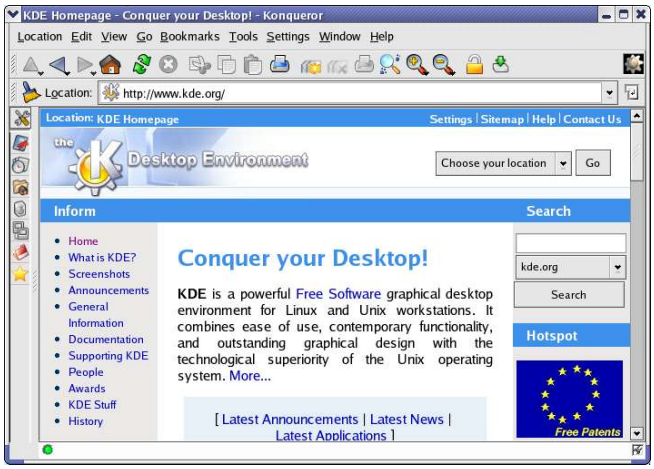


Fig. Konqueror Web Browser

To learn more about the many features of Konqueror, see the online Konqueror documentation which can be invoked by selecting from the Konqueror main toolbar,

Help --> Konqueror Handbook

The Help Center

One of the best ways to learn about how to use KDE effectively is through its online help documentation – the Help Center. This can be invoked from the Main Menu,

Main Menu --> Help

The Help Center covers the graphical desktop usage and configuration as well as the KDE applets and applications. Standard Unix/Linux manual and info pages can be accessed from here too.

The KDE Help Center should be consulted for more information about how to use KDE.

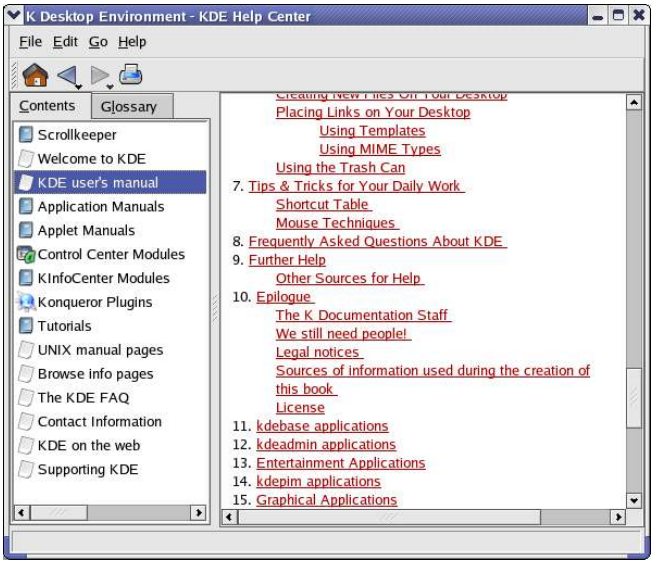


Fig. The KDE Help Center

Command-line Bootcamp

Keith Bradnam

UC Davis Genome Center

Version 1.02 — 2015–12–03

Introduction

This 'bootcamp' is intended to provide the reader with a basic overview of essential Unix/Linux commands that will allow them to navigate a file system and move, copy, edit files. It will also introduce a brief overview of some 'power' commands in Unix. It was orginally developed as part of a Bioinformatics Core Workshop taught at UC Davis (Using the Linux Command-Line for Analysis of High Throughput Sequence Data).

Why Unix?

The Unix operating system has been around since 1969. Back then there was no such thing as a graphical user interface. You typed everything. It may seem archaic to use a keyboard to issue commands today, but it's much easier to automate keyboard tasks than mouse tasks. There are several variants of Unix (including Linux), though the differences do not matter much for most basic functions.

Increasingly, the raw output of biological research exists as *in silico* data, usually in the form of large text files. Unix is particularly suited to working with such files and has several powerful (and flexible) commands that can process your data for you. The real strength of learning Unix is that most of these commands can be combined in an almost unlimited fashion. So if you can learn just five Unix commands, you will be able to do a lot more than just five things.

Typeset Conventions

Command-line examples that you are meant to type into a terminal window will be shown indented in a constant-width font, e.g.

```
ls -lrh
```

Sometimes the accompanying text will include a reference to a Unix command. Any such text will also be in a constant-width, boxed font. E.g.

> Type the `pwd` command again.

From time to time this documentation will contain web links to pages that will help you find out more about certain Unix commands. Usually, the *first* mention of a command or function will be a hyperlink to Wikipedia. Important or critical points will be styled like so:

> ***This is an important point!***

Assumptions

The lessons from this point onwards will assume very little apart from the following:

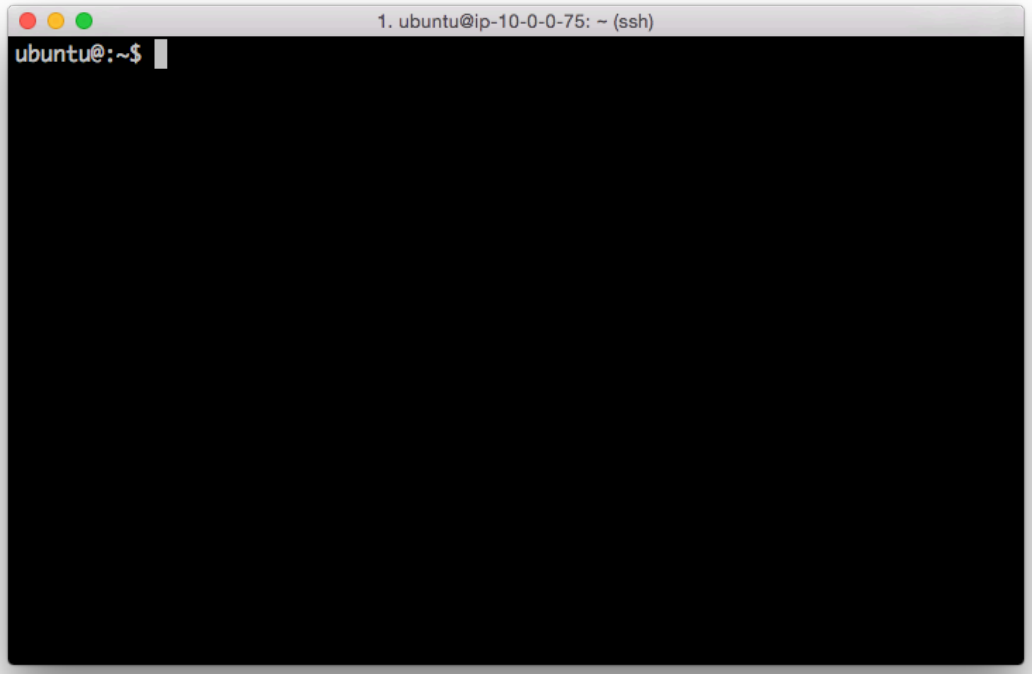
1. You have access to a Unix/Linux system
2. You know how to launch a terminal program on that system
3. You have a home directory where you can create/edit new files

In the following documentation, we will also assume that the logged in user has a username ‘ubuntu’ and the home directory is located at `/home/ubuntu`.

1. The Terminal

A *terminal* is the common name for the program that does two main things. It allows you to type input to the computer (i.e. run programs, move/view files etc.) and it allows you to see output from those programs. All Unix machines will have a terminal program available.

Open the terminal application. You should now see something that looks like the following:



Terminal application

There will be many situations where it will be useful to have multiple terminals open and it will be a matter of preference as to whether you want to have multiple windows, or one window with multiple tabs (there are typically keyboard shortcuts for switching between windows, or moving between tabs).

2. Your first Unix command

It's important to note that you will always be *inside* a single directory when using the terminal. The default behavior is that when you open a new terminal you start in your own *home* directory (containing files and directories that only you can modify). To see what files and directories are in our home directory, we need to use the ls command. This command lists the contents of a directory. If we run the `ls` command we should see something like:

```
ubuntu@:~$ ls
command_line_course  linux_bootcamp
ubuntu@:~$
```

There are four things that you should note here:

1. You will probably see different output to what is shown here, it depends on your computer setup. Don't worry about that for now.
2. The `ubuntu@:~$` text that you see is the Unix command prompt. In this case, it contains a user name ('ubuntu') and the name of the current directory ('~', more on that later). Note that the command prompt might not look the same on different Unix systems. In this case, the `$` sign marks the end of the prompt.
3. The output of the `ls` command lists two things. In this case, they are both directories, but they could also be files. We'll learn how to tell them apart later on. These directories were created as part of a specific course that used this bootcamp material. You will therefore probably see something very different on your own computer.
4. After the `ls` command finishes it produces a new command prompt, ready for you to type your next command.

The `ls` command is used to list the contents of *any* directory, not necessarily the one that you are currently in. Try the following:

```
ubuntu@:~$ ls /data
bioinfo.course.data  command_line_course  galaxy  lost+found  refs

ubuntu@:~$ ls /etc/perl
CPAN  Net  XML
```

3: The Unix tree

Looking at directories from within a Unix terminal can often seem confusing. But bear in mind that these directories are exactly the same type of folders that you can see if you use any graphical file browser. From the *root* level (`/`) there are usually a dozen or so directories. You can treat the root directory like any other, e.g. you can list its contents:

```
ubuntu@:~$ ls /
bin   dev   initrd.img      lib64       mnt   root  software  tmp  vmlinuz
boot  etc   initrd.img.old  lost+found  opt   run   srv       usr  vmlinuz.old
data  home  lib             media       proc  sbin  sys       var
```

You might notice some of these names appearing in different colors. Many Unix systems will display files and directories differently by default. Other colors may be used for special types of files. When you log in to a computer you are working with your files in your home directory, and this is often inside a directory called ‘users’ or ‘home’.

4: Finding out where you are

There may be many hundreds of directories on any Unix machine, so how do you know which one you are in? The command pwd will Print the Working Directory and that's pretty much all this command does:

```
ubuntu@:~$ pwd
/home/ubuntu
```

When you log in to a Unix computer, you are typically placed into your *home* directory. In this example, after we log in, we are placed in a directory called 'ubuntu' which itself is a *subdirectory* of another directory called 'home'. Conversely, 'users' is the *parent* directory of 'clmuser'. The first forward slash that appears in a list of directory names always refers to the top level directory of the file system (known as the root directory). The remaining forward slash (between 'home' and 'ubuntu') delimits the various parts of the directory hierarchy. If you ever get 'lost' in Unix, remember the `pwd` command.

As you learn Unix you will frequently type commands that don't seem to work. Most of the time this will be because you are in the wrong directory, so it's a really good habit to get used to running the `pwd` command a lot.

5: Making new directories

If we want to make a new directory (e.g. to store some work related data), we can use the mkdir command:

```
ubuntu@:~$ mkdir Learning_unix
ubuntu@:~$ ls
command_line_course  Learning_unix  linux_bootcamp
```

6: Getting from ‘A’ to ‘B’

We are in the home directory on the computer but we want to to work in the new `Learning_unix` directory. To change directories in Unix, we use the cd command:

```
cd Learning_unix
ubuntu@:~/Learning_unix$
```

Notice that — on this system — the command prompt has expanded to include our current directory. This doesn’t happen by default on all Unix systems, but you should know that you can configure what information appears as part of the command prompt.

Let’s make two new subdirectories and navigate into them:

```
ubuntu@:~/Learning_unix$ mkdir Outer_directory
ubuntu@:~/Learning_unix$ cd Outer_directory
ubuntu@:~/Learning_unix/Outer_directory$

ubuntu@:~/Learning_unix/Outer_directory$ mkdir Inner_directory
ubuntu@:~/Learning_unix/Outer_directory$ cd Inner_directory/
ubuntu@:~/Learning_unix/Outer_directory/Inner_directory$
```

Now our command prompt is getting quite long, but it reveals that we are three levels beneath the home directory. We created the two directories in separate steps, but it is possible to use the `mkdir` command in way to do this all in one step.

Like most Unix commands, `mkdir` supports *command-line options* which let you alter its behavior and functionality. Command-like options are — as the name suggests — optional arguments that are placed after the command name. They often take the form of single letters (following a dash). If we had used the `-p` option of the `mkdir` command we could have done this in one step. E.g.

```
mkdir -p Outer_directory/Inner_directory
```

Note the spaces either side the `-p` !

7: The root directory

Let's change directory to the root directory, and then into our home directory

```
ubuntu@:~/Learning_unix/Outer_directory/Inner_directory$ cd /
ubuntu@:/$ cd home
ubuntu@:/home$ cd ubuntu
ubuntu@:~$
```

In this case, we may as well have just changed directory in one go:

```
cd /home/ubuntu/
```

The leading `/` is incredibly important. The following two commands are very different:

```
cd /home/ubuntu/
cd home/ubuntu/
```

The first command says go the `unbuntu` directory that is beneath the `home` directory that is at the top level (the root) of the file system. There can only be one `/home/ubuntu` directory on any Unix system.

The second command says go to the `unbuntu` directory that is beneath the `home` directory that is located wherever I am right now. There can potentially be many `home/ubuntu` directories on a Unix system (though this is unlikely).

Learn and understand the difference between these two commands.

8: Navigating upwards in the Unix filesystem

Frequently, you will find that you want to go 'upwards' one level in the directory hierarchy. Two dots `..` are used in Unix to refer to the *parent* directory of wherever you are. Every directory has a parent except the root level of the computer. Let's go into the `Learning_unix` directory and then navigate up two levels:

```
ubuntu@:~$ cd Learning_unix/
ubuntu@:~/Learning_unix$ cd ..
ubuntu@:~$ cd ..
ubuntu@:/home$
```

What if you wanted to navigate up *two* levels in the file system in one go? It's very simple, just use two sets of the `..` operator, separated by a forward slash:

```
cd ../..
```

9: Absolute and relative paths

Using `cd ..` allows us to change directory *relative* to where we are now. You can also always change to a directory based on its *absolute* location. E.g. if you are working in the `/home/ubuntu/Learning_unix` directory and you then want to change to the `/tmp` directory, then you could do either of the following:

```
$ cd ../../../tmp
```

or…

```
$ cd /tmp
```

They both achieve the same thing, but the 2nd example requires that you know about the full *path* from the root level of the computer to your directory of interest (the 'path' is an important concept in Unix). Sometimes it is quicker to change directories using the relative path, and other times it will be quicker to use the absolute path.

10: Finding your way back home

Remember that the command prompt shows you the name of the directory that you are currently in, and that when you are in your home directory it shows you a tilde character (`~``) instead? This is because Unix uses the tilde character as a short-hand way of specifying a home directory.

See what happens when you try the following commands (use the `pwd` command after each one to confirm the results if necessary):

```
cd /
cd ~
cd
```

Hopefully, you should find that `cd` and `cd ~` do the same thing, i.e. they take you back to your home directory (from wherever you were). You will frequently want to jump straight back to your home directory, and typing `cd` is a very quick way to get there.

You can also use the `~` as a quick way of navigating into subdirectories of your home directory when your current directory is somewhere else. I.e. the quickest way of navigating from the root directory to your `Learning_unix` directory is as follows:

```
ubuntu@:~$ cd /
ubuntu@:/$ cd ~/Learning_unix
```

11: Making the `ls` command more useful

The `..` operator that we saw earlier can also be used with the `ls` command, e.g. you can list directories that are 'above' you:

```
ubuntu@:~/Learning_unix$ cd ~/Learning_unix/Outer_directory/
ubuntu@:~/Learning_unix/Outer_directory$ ls ../../
command_line_course  Learning_unix  linux_bootcamp
```

Time to learn another useful command-line option. If you add the letter 'l' to the `ls` command it will give you a longer output compared to the default:

```
ubuntu@:~/Learning_unix$ ls -l /home
total 12
drwxr-xr-x 8 galaxy galaxy 4096 Apr  2 22:47 galaxy
drwxr-xr-x 3 root   root   4096 Mar 16 23:06 nate
drwxr-xr-x 9 ubuntu ubuntu 4096 Jun 15 02:07 ubuntu
```

For each file or directory we now see more information (including file ownership and modification times). The 'd' at the start of each line indicates that these are directories. There are many, many different options for the `ls` command. Try out the following (against any directory of your choice) to see how the output changes.

```
ls -l
ls -R
ls -l -t -r
ls -lh
```

Note that the last example combine multiple options but only use one dash. This is a very common way of specifying multiple command-line options. You may be wondering what some of these options are doing. It's time to learn about Unix documentation….

12: Man pages

If every Unix command has so many options, you might be wondering how you find out what they are and what they do. Well, thankfully every Unix command has an associated ‘manual’ that you can access by using the `man` command. E.g.

```
man ls
man cd
man man # yes even the man command has a manual page
```

When you are using the man command, press `space` to scroll down a page, `b` to go back a page, or `q` to quit. You can also use the up and down arrows to scroll a line at a time. The man command is actually using another Unix program, a text viewer called `less`, which we’ll come to later on.

13: Removing directories

We now have a few (empty) directories that we should remove. To do this use the rmdir command, this will only remove empty directories so it is quite safe to use. If you want to know more about this command (or any Unix command), then remember that you can just look at its man page.

```
ubuntu@:~$ cd ~/Learning_unix/Outer_directory/
ubuntu@:~/Learning_unix/Outer_directory$ rmdir Inner_directory/
ubuntu@:~/Learning_unix/Outer_directory$ cd ..
ubuntu@:~/Learning_unix$ rmdir Outer_directory/
ubuntu@:~/Learning_unix$ ls
ubuntu@:~/Learning_unix$
```

*** Note, you have to be outside a directory before you can remove it with `rmdir` ***

14: Using tab completion

Saving keystrokes may not seem important, but the longer that you spend typing in a terminal window, the happier you will be if you can reduce the time you spend at the keyboard. Especially, as prolonged typing is not good for your body. So the best Unix tip to learn early on is that you can tab complete the names of files and programs on most Unix systems. Type enough letters that uniquely identify the name of a file, directory or program and press tab…Unix will do the rest. E.g. if you type 'tou' and then press tab, Unix should autocomplete the word to 'touch' (this is a command which we will learn more about in a minute). In this case, tab completion will occur because there are no other Unix commands that start with 'tou'. If pressing tab doesn't do anything, then you have not have typed enough unique characters. In this case pressing tab *twice* will show you all possible completions. This trick can save you a LOT of typing!

Navigate to your home directory, and then use the `cd` command to change to the `Learning_unix` directory. Use tab completion to complete directory name. If there are no other directories starting with 'L' in your home directory, then you should only need to type 'cd' + 'L' + 'tab'.

> ***Tab completion will make your life easier and make you more productive!***

Another great time-saver is that Unix stores a list of all the commands that you have typed in each login session. You can access this list by using the history command or more simply by using the up and down arrows to access anything from your history. So if you type a long command but make a mistake, press the up arrow and then you can use the left and right arrows to move the cursor in order to make a change.

15: Creating empty files with the touch command

The following sections will deal with Unix commands that help us to work with files, i.e. copy files to/from places, move files, rename files, remove files, and most importantly, look at files. First, we need to have some files to play with. The Unix command touch will let us create a new, empty file. The touch command does other things too, but for now we just want a couple of files to work with.

```
ubuntu@:~$ cd Learning_unix/
ubuntu@:~/Learning_unix$ touch heaven.txt
ubuntu@:~/Learning_unix$ touch earth.txt
ubuntu@:~/Learning_unix$ ls
earth.txt  heaven.txt
```

16: Moving files

Now, let's assume that we want to move these files to a new directory ('Temp'). We will do this using the Unix mv (move) command. Remember to use tab completion:

```
ubuntu@:~/Learning_unix$ mkdir Temp
ubuntu@:~/Learning_unix$ mv heaven.txt Temp/
ubuntu@:~/Learning_unix$ mv earth.txt Temp/
ubuntu@:~/Learning_unix$ ls
Temp
ubuntu@:~/Learning_unix$ ls Temp/
earth.txt  heaven.txt
```

For the `mv` command, we always have to specify a source file (or directory) that we want to move, and then specify a target location. If we had wanted to we could have moved both files in one go by typing any of the following commands:

```
mv *.txt Temp/
mv *t Temp/
mv *ea* Temp/
```

The asterisk `*` acts as a wild-card character, essentially meaning 'match anything'. The second example works because there are no other files or directories in the directory that end with the letters 't' (if there was, then they would be moved too). Likewise, the third example works because only those two files contain the letters 'ea' in their names. Using wild-card characters can save you a lot of typing.

The '?' character is also a wild-card but with a slightly different meaning. See if you can work out what it does.

17: Renaming files

In the earlier example, the destination for the `mv` command was a directory name (Temp). So we moved a file from its source location to a target location, but note that the target could have also been a (different) file name, rather than a directory. E.g. let's make a new file and move it whilst renaming it at the same time:

```
ubuntu@:~/Learning_unix$ touch rags
ubuntu@:~/Learning_unix$ ls
rags  Temp
ubuntu@:~/Learning_unix$ mv rags Temp/riches
ubuntu@:~/Learning_unix$ ls Temp/
earth.txt  heaven.txt  riches
```

In this example we create a new file ('rags') and move it to a new location and in the process change the name (to 'riches'). So `mv` can rename a file as well as move it. The logical extension of this is using `mv` to rename a file without moving it (you have to use `mv` to do this as Unix does not have a separate 'rename' command):

```
ubuntu@:~/Learning_unix$ mv Temp/riches Temp/rags
```

18: Moving directories

It is important to understand that as long as you have specified a ‘source’ and a ‘target’ location when you are moving a file, then it doesn’t matter what your *current* directory is. You can move or copy things within the same directory or between different directories regardless of whether you are in any of those directories. Moving directories is just like moving files:

```
ubuntu@:~/Learning_unix$ mv Temp/riches Temp/rags
ubuntu@:~/Learning_unix$ mkdir Temp2
ubuntu@:~/Learning_unix$ mv Temp2 Temp
ubuntu@:~/Learning_unix$ ls Temp/
earth.txt  heaven.txt  rags  Temp2
```

This step moves the Temp2 directory inside the Temp directory. Try creating a ‘Temp3’ directory inside ‘Learning_unix’ and then `cd` to `/tmp`. Can you move `Temp3` inside `Temp` without changing directory?

19: Removing files

You've seen how to remove a directory with the `rmdir` command, but `rmdir` won't remove directories if they contain any files. So how can we remove the files we have created (inside `Learning_Unix/Temp`)? In order to do this, we will have to use the rm (remove) command.

> ***Please read the next section VERY carefully. Misuse of the `rm` command can lead to needless death & destruction***

Potentially, `rm` is a very dangerous command; if you delete something with `rm`, you will not get it back! It is possible to delete everything in your home directory (all directories and subdirectories) with `rm`, that is why it is such a dangerous command.

Let me repeat that last part again. It is possible to delete EVERY file you have ever created with the `rm` command. Are you scared yet? You should be. Luckily there is a way of making `rm` a little bit safer. We can use it with the `-i` command-line option which will ask for confirmation before deleting anything (remember to use tab-completion):

```
ubuntu@:~/Learning_unix$ cd Temp
ubuntu@:~/Learning_unix/Temp$ ls
earth.txt  heaven.txt  rags  Temp2
ubuntu@:~/Learning_unix/Temp$ rm -i earth.txt heaven.txt rags
rm: remove regular empty file ‘earth.txt’? y
rm: remove regular empty file ‘heaven.txt’? y
rm: remove regular empty file ‘rags’? y
ubuntu@:~/Learning_unix/Temp$ ls
Temp2
```

We could have simplified this step by using a wild-card (e.g. `rm -i *.txt`) or we could have made things more complex by removing each file with a separate `rm` command. Let's finish cleaning up:

```
rmdir Temp2/Temp3
rmdir Temp2
cd ..
rmdir Temp
```

20: Copying files

Copying files with the cp (copy) command is very similar to moving them. Remember to always specify a source and a target location. Let's create a new file and make a copy of it:

```
ubuntu@:~/Learning_unix$ touch file1
ubuntu@:~/Learning_unix$ cp file1 file2
ubuntu@:~/Learning_unix$ ls
file1  file2
```

What if we wanted to copy files from a different directory to our current directory? Let's put a file in our home directory (specified by `~` remember) and copy it to the current directory (`Learning_unix`):

```
ubuntu@:~/Learning_unix$ touch ~/file3
ubuntu@:~/Learning_unix$ ls ~
command_line_course  file3  Learning_unix  linux_bootcamp
ubuntu@:~/Learning_unix$ cp ~/file3 .
ubuntu@:~/Learning_unix$ ls
file1  file2  file3
```

This last step introduces another new concept. In Unix, the current directory can be represented by a `.` (dot) character. You will mostly use this only for copying files to the current directory that you are in. Compare the following:

```
ls
ls .
ls ./
```

In this case, using the dot is somewhat pointless because `ls` will already list the contents of the current directory by default. Also note how the trailing slash is optional. You can use `rm` to remove the temporary files.

21: Copying directories

The `cp` command also allows us (with the use of a command-line option) to copy entire directories. Use `man cp` to see how the `-R` or `-r` options let you copy a directory *recursively*.

22: Viewing files with less

So far we have covered listing the contents of directories and moving/copying/deleting either files and/or directories. Now we will quickly cover how you can look at files. The less command lets you view (but not edit) text files. We will use the echo command to put some text in a file and then view it:

```
ubuntu@:~/Learning_unix$ echo "Call me Ishmael."
Call me Ishmael.
ubuntu@:~/Learning_unix$ echo "Call me Ishmael." > opening_lines.txt
ubuntu@:~/Learning_unix$ ls
opening_lines.txt
ubuntu@:~/Learning_unix$ less opening_lines.txt
```

On its own, `echo` isn't a very exciting Unix command. It just echoes text back to the screen. But we can redirect that text into an output file by using the `>` symbol. This allows for something called file redirection.

> ***Careful when using file redirection (>), it will overwrite any existing file of the same name***

When you are using less, you can bring up a page of help commands by pressing `h`, scroll forward a page by pressing `space`, or go forward or backwards one line at a time by pressing `j` or `k`. To exit less, press `q` (for quit). The `less` program also does about a million other useful things (including text searching).

23: Viewing files with cat

Let's add another line to the file:

```
ubuntu@:~/Learning_unix$ echo "The primroses were over." >> opening_lines.txt
ubuntu@:~/Learning_unix$ cat opening_lines.txt
Call me Ishmael.
The primroses were over.
```

Notice that we use `>>` and not just `>`. This operator will **append** to a file. If we only used `>`, we would end up overwriting the file. The cat command displays the contents of the file (or files) and then returns you to the command line. Unlike `less` you have no control on how you view that text (or what you do with it). It is a very simple, but sometimes useful, command. You can use `cat` to quickly combine multiple files or, if you wanted to, make a copy of an existing file:

```
cat opening_lines.txt > file_copy.txt
```

24: Counting characters in a file

```
ubuntu@:~/Learning_unix$ ls
opening_lines.txt

ubuntu@:~/Learning_unix$ ls -l
total 4
-rw-rw-r-- 1 ubuntu ubuntu 42 Jun 15 04:13 opening_lines.txt

ubuntu@:~/Learning_unix$ wc opening_lines.txt
 2  7 42 opening_lines.txt

ubuntu@:~/Learning_unix$ wc -l opening_lines.txt
2 opening_lines.txt
```

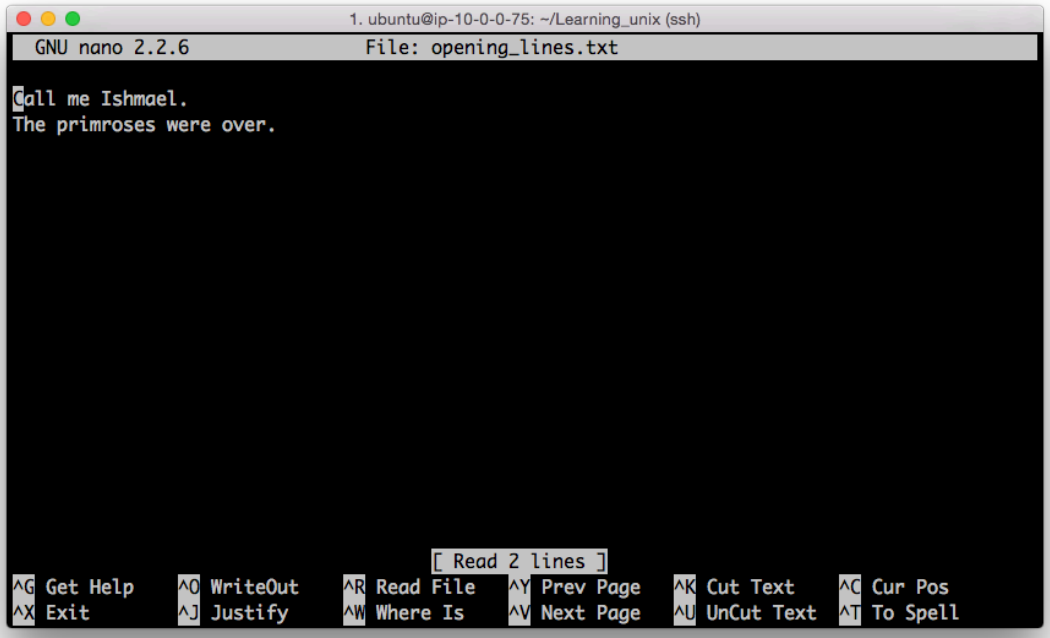
The `ls -l` option shows us a long listing, which includes the size of the file in bytes (in this case '42'). Another way of finding this out is by using Unix's wc command (word count). By default this tells you many lines, words, and characters are in a specified file (or files), but you can use command-line options to give you just one of those statistics (in this case we count lines with `wc -l`).

25: Editing small text files with nano

Nano is a lightweight editor installed on most Unix systems. There are many more powerful editors (such as ‘emacs’ and ‘vi’), but these have steep learning curves. Nano is very simple. You can edit (or create) files by typing:

```
nano opening_lines.txt
```

You should see the following appear in your terminal:



The bottom of the nano window shows you a list of simple commands which are all accessible by typing ‘Control’ plus a letter. E.g. Control + X exits the program.

26: The $PATH environment variable

One other use of the `echo` command is for displaying the contents of something known as *environment variables*. These contain user-specific or system-wide values that either reflect simple pieces of information (your username), or lists of useful locations on the file system. Some examples:

```
ubuntu@:~/Learning_unix$ echo $USER
ubuntu
ubuntu@:~/Learning_unix$ echo $HOME
/home/ubuntu
ubuntu@:~/Learning_unix$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

The last one shows the content of the `$PATH` environment variable, which displays a — colon separated — list of directories that are expected to contain programs that you can run. This includes all of the Unix commands that you have seen so far. These are files that live in directories which are run like programs (e.g. `ls` is just a special type of file in the `/bin` directory).

Knowing how to change your $PATH to include custom directories can be necessary sometimes (e.g. if you install some new bioinformatics software in a non-standard location).

27: Matching lines in files with grep

Use `nano` to add the following lines to `opening_lines.txt`:

```
Now is the winter of our discontent.
All children, except one, grow up.
The Galactic Empire was dying.
In a hole in the ground there lived a hobbit.
It was a pleasure to burn.
It was a bright, cold day in April, and the clocks were striking thirteen.
It was love at first sight.
I am an invisible man.
It was the day my grandmother exploded.
When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow.
Marley was dead, to begin with.
```

You will often want to search files to find lines that match a certain pattern. The Unix command grep does this (and much more). The following examples show how you can use grep's command-line options to:

- show lines that match a specified pattern
- ignore case when matching (`-i`)
- only match whole words (`-w`)
- show lines that don't match a pattern (`-v`)
- Use wildcard characters and other patterns to allow for alternatives (`*` , `.` , and `[]`)

```
grep was opening_lines.txt
The Galactic Empire was dying.
It was a pleasure to burn.
It was a bright, cold day in April, and the clocks were striking thirteen.
It was love at first sight.
It was the day my grandmother exploded.
When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow.
Marley was dead, to begin with.

grep -v was opening_lines.txt
Call me Ishmael.
The primroses were over.
Now is the winter of our discontent.
All children, except one, grow up.
In a hole in the ground there lived a hobbit.
I am an invisible man.

grep all opening_lines.txt
Call me Ishmael.

grep -i all opening_lines.txt
Call me Ishmael.
All children, except one, grow up.

grep in opening_lines.txt
```

```
Now is the winter of our discontent.
The Galactic Empire was dying.
In a hole in the ground there lived a hobbit.
It was a bright, cold day in April, and the clocks were striking thirteen.
I am an invisible man.
Marley was dead, to begin with.

grep -w in opening_lines.txt
In a hole in the ground there lived a hobbit.
It was a bright, cold day in April, and the clocks were striking thirteen.

grep -w o.. opening_lines.txt
Now is the winter of our discontent.
All children, except one, grow up.

grep [aeiou]t opening_lines.txt
In a hole in the ground there lived a hobbit.
It was love at first sight.
It was the day my grandmother exploded.
When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow.
Marley was dead, to begin with.

grep -w -i [aeiou]t opening_lines.txt
It was a pleasure to burn.
It was a bright, cold day in April, and the clocks were striking thirteen.
It was love at first sight.
It was the day my grandmother exploded.
When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow.
```

28: Combining Unix commands with pipes

One of the most poweful features of Unix is that you can send the output from one command or program to any other command (as long as the second commmand accepts input of some sort). We do this by using what is known as a pipe. This is implemented using the 'l' character (which is a character which always seems to be on different keys depending on the keyboard that you are using). Think of the pipe as simply connecting two Unix programs. Here's an example which introduces some new Unix commands:

```
ubuntu@:~/Learning_unix$ grep was opening_lines.txt | wc -c
316

ubuntu@:~/Learning_unix$
grep was opening_lines.txt | sort | head -n 3 | wc -c
130
```

The first use of `grep` searches the specified file for lines matching 'was', it sends the lines that match through a pipe to the `wc` program. We use the `-c` option to just count characters in the matching lines (316).

The second example first sends the output of `grep` to the Unix `sort` command. This sorts a file alphanumerically by default. The sorted output is sent to the `head` command which by default shows the first 10 lines of a file. We use the `-n` option of this command to only show 3 lines. These 3 lines are then sent to the `wc` command as before.

> ***Whenever making a long pipe, test each step as you build it!***

Miscellaneous Unix power commands

The following examples introduce some other Unix commands, and show how they could be used to work on a fictional file called `file.txt`. Remember, you can always learn more about these Unix commands from their respective man pages with the `man` command. These are not all real world cases, but rather show the diversity of Unix command-line tools:

- View the penultimate 10 lines of a file (using head and tail commands):

```
tail -n 20 file.txt | head
```

- Show lines of a file that begin with a start codon (ATG) (the `^` matches patterns at the start of a line):

```
grep "^ATG" file.txt
```

- Cut out the 3rd column of a tab-delimited text file and sort it to only show unique lines (i.e. remove duplicates):

```
cut -f 3 file.txt | sort -u
```

- Count how many lines in a file contain the words 'cat' or 'bat' (`-c` option of `grep` counts lines):

```
grep -c '[bc]at' file.txt
```

- Turn lower-case text into upper-case (using tr command to 'transliterate'):

```
cat file.txt | tr 'a-z' 'A-Z'
```

- Change all occurences of 'Chr1' to 'Chromosome 1' and write changed output to a new file (using sed command):

```
cat file.txt | sed 's/Chr1/Chromosome 1/' > file2.txt
```

Version history

2015–06–14 - Version 1.0, adapted from Unix and Perl for Biologists Primer
2015–06–24 - Version 1.01: clarified that this material is assuming user name is ’ubuntu’and made other minor clarifications (such as what this material was first produced for).
2015–12–03 - Version 1.02: changed license to CC BY (from CC BY-NC-SA)