Ubuntu Server

Succinctly®

by José Roberto Olivas Mendoza

Syncfusion | Technology Resource Portal

Ubuntu Server Succinctly

By
José Roberto Olivas Mendoza

Foreword by Daniel Jebaraj

Important licensing information. Please read.

Technical Reviewer: Jason Cannon

Copy Editor: Courtney Wright

Acquisitions Coordinator: Hillary Bowling, online marketing manager, Syncfusion, Inc.

Proofreader: Graham High, senior content producer, Syncfusion, Inc.

Table of Contents

The Story behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the Internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

Free forever

Syncfusion will be working to produce books on several topics. The books will always be free. Any updates we publish will also be free.

Free? What is the catch?

There is no catch here. Syncfusion has a vested interest in this effort.

As a component vendor, our unique claim has always been that we offer deeper and broader frameworks than anyone else on the market. Developer education greatly helps us market and sell against competing vendors who promise to “enable AJAX support with one click,” or “turn the moon to cheese!”

Let us know what you think

If you have any topics of interest, thoughts, or feedback, please feel free to send them to us at succinctly-series@syncfusion.com.

We sincerely hope you enjoy reading this book and that it helps you better understand the topic of study. Thank you for reading.

Please follow us on Twitter and “Like” us on Facebook to help us spread the word about the *Succinctly* series!

About the Author

I'm an IT businesses entrepreneur, software developer, and huge technology fan. My company went to market in 1990, focused in custom software development. We started with COBOL as our main programming language, and since then we've been evolving up to .NET and Microsoft Office technologies. Since the moment we came up in the IT market arena, Microsoft technologies were the main core of our business…until a few years ago.

Ubuntu appeared when some of our customers asked for better performance of the RDBMS that they were using. But they were clear about this: "We don't want to spend money in server and RDBMS software licensing." That requirement brought two players to our game: PostgreSQL and Linux.

At that time, we had no consultants focused in any Linux distribution available in the market, and the time needed to acquire enough implementing skills was beyond our deadline for many of those Linux distributions.

After intensive research, we found Ubuntu, and we chose it because the distribution has a focus on providing a solid desktop and server experience. That experience made our consultants' transition from Windows to Linux easier. At the same time, it allowed us to get into the Linux market quickly, expanding our solutions portfolio.

Recently, we've been thinking that selecting Ubuntu as our main Linux distribution was the best choice we made. Canonical, the company that delivers Ubuntu, has made a huge effort to ensure more hardware and software can work optimally with the Ubuntu platform, and with its partners' collaboration, has taken Ubuntu to smartphones and IoT. This tells us we can count on using Ubuntu as part of our solutions portfolio for many years to come.

Who Is This Book For?

This book is written primarily for IT professionals who want to learn about making an Ubuntu Server implementation. The book starts with a brief story about the vision of Canonical, the company that delivers Ubuntu, and how Ubuntu came into the IT market. It also shows some of the promises and goals that motivated the Ubuntu project.

The first chapter covers the Ubuntu installation process, including hardware requirements. The rest of the book is intended to show how to implement several features that are expected in a server operating system. Themes like security and remote administration are discussed, too.

Then, I'll explain how to make Ubuntu and Windows work together in harmony, integrating network resources such as files and printing. We'll also explain how to turn Ubuntu into a database server, using PostgreSQL.

Finally, I'll explain the concept of the desktop experience in Ubuntu Server and the process needed to install this feature in the server.

Ubuntu Server 15.10 was used for the purposes of this book. Regarding the scripting section, a series of script samples can be downloaded from the Syncfusion Bitbucket account.

I hope that by the end of this book, IT professionals who aren't involved with Linux can deliver an Ubuntu Server installation, take advantage of its capabilities, and benefit from using it in their projects.

Chapter 1 Introduction

What is Ubuntu?

A bunch of people from the Debian, GNOME, and GNU Arch projects were gathered by Mark Shuttleworth in April 2004. He asked them if developing a better operating system was possible. They answered, “Yes.” Then, in the way of a brainstorm, he asked them what it would look like and a description of the community that would build that OS.

The group worked with Shuttleworth to answer these questions, and then they decided to try to make the vision into a reality. They named themselves the Warthogs and gave themselves a six-month deadline to build a proof-of-concept OS. They named that first release the Warty Warthog, assuming that their first product would have its warts. Then they got down to business.

The vision became Ubuntu

Far from being warty, the Warty Warthog exceeded the group’s expectations and most optimistic predictions. Within six months, Ubuntu was in the number one spot on several popular rankings of GNU/Linux distributions. Ubuntu has demonstrated the most explosive growth of any GNU/Linux distribution in recent memory and an impressive continued growth of any free or open-source software project in history.

Today, millions of people are using Ubuntu, and many of these users give back to the Ubuntu community by developing documentation, translations, and code. By doing this, these users make Ubuntu improvements every day, and a huge community—both online and in local communities—makes Ubuntu growth unchecked.

What does Ubuntu mean?

The name Ubuntu was taken from a concept and term found in several South African languages, and refers to an ideology or ethic that could be translated as “humanity toward others.”

Mark Shuttleworth liked Ubuntu as a name for the project for several reasons. For one, he’s from South Africa. Secondly, the project emphasizes relationships with others, simultaneously providing a framework for a profound type of community and sharing. This term represented the side of free software that the team wanted to share with the world.

Ubuntu promises and goals

The Ubuntu project is based on a series of philosophical goals that are summarized in the following sentences (the complete text is available at Ubuntu’s website):

- **Philosophy**: Our work is driven by a software freedom concept that aims to spread and bring the benefits of software to all parts of the world. At the core of Ubuntu

philosophy are these ideas: Every computer user should have the freedom to download, run, copy, distribute, study, share, change, and improve their software for any purpose, without paying license fees. Every computer user should be able to user their software in the language of their choice. Every computer user should be given every opportunity to use software, even if they have a disability.

- **Free software**: The “free” in free software is used primarily in reference to freedom and not in price, although they’re committed to not charging for Ubuntu. The most important thing about Ubuntu is that it confers rights of software freedom to the people who install and use it.
- **Open source**: Open source is a term created in 1998 to remove the ambiguity in the word “free.” The open source definition describes open software as a software project that allows access to source code, free redistribution, and modifications to create derived works that can be distributed under the same terms as the license of the original software.

Sustaining and going beyond the vision

The group that plays a significant role in driving the Ubuntu project is Canonical Ltd. Canonical is a company founded by Mark Shuttleworth with the primary goal of developing and supporting the Ubuntu distribution. Many of the core developers on Ubuntu work full time or part time for Canonical Ltd. This funding by the company allows Ubuntu to make the type of support commitment that it does, because this subset of developers has paid jobs. In this way, Canonical ensures that Ubuntu’s bottom-line commitments are kept.

Canonical does not fund all Ubuntu work, and maybe it couldn’t. Canonical can release a distribution every six months, but the distribution is made much better by the contributions from the community of users. Many of the features, translations, documentation, and much more are created outside Canonical. In that way, Canonical focuses on essential work and ensures that the deadlines are met. Besides, Canonical’s staff is sprinkled across the globe, so no proper office is necessary, and that cuts down on the costs of infrastructure and daily expenses.

So, if downloading Ubuntu is free of charge, how does Canonical make money? At this time, Canonical hasn’t been making money. The main funding comes from Mark Shuttleworth and some people’s donations. However, Canonical is looking toward making Ubuntu profitable, and their key revenue comes from the following services:

- Support services, mostly for businesses.
- Contracting services for businesses, such as OEMs like Dell, to integrate Ubuntu in their hardware solutions.
- Ubuntu Software Center’s paid section (Canonical takes a cut of purchases).
- The Canonical Store, which sells physical Ubuntu branded items.
- Amazon referrals. When a search is made in the Ubuntu Dash, the user may see Amazon products (unless they are turned off). Ubuntu takes a cut of these.

Chapter summary

This introduction looked into the vision of a better Linux distribution and OS. Also, it provided a brief history about Ubuntu, the project which came from that vision. It also showed how Ubuntu is sustained with the main role played by Canonical and the huge collaboration of the community of users.

Chapter 2 Installing Ubuntu Server

Getting started

Hardware requirements

Ubuntu 15.10 is available for the three major architectures: Intel x86, AMD64, and ARM. The following table lists the recommended hardware specifications.

Table 1: Recommended Minimum Requirements

Install type	CPU	RAM	Hard Drive Space	
			Base System	All Tasks Installed
Server (Standard)	1 Gigahertz	512 Megabytes	1 Gigabyte	1.75 Gigabytes
Server (Minimal)	300 Megahertz	192 Megabytes	700 Megabytes	1.4 Gigabytes

Note: Depending on the user needs, Ubuntu might manage with less than what is suggested. However, most users risk being frustrated if they ignore these suggestions.

Getting Ubuntu

Ubuntu can be downloaded from the Ubuntu website as an ISO image file.

Once the download is complete, the file must be burned onto a CD or DVD. Also, the user needs to ensure that the CD/DVD drive is set as the first boot device in the system that will be used for installation.

Note: Setting boot device priority depends on the type of BIOS embedded into each device, but commonly this feature is identified as boot priority or boot order.

Booting from installation disc

First, the disc containing the Ubuntu Sever installation must be inserted in the system's CD/DVD drive. Then, the system needs to be rebooted. Once this has been done, the following screen will appear.

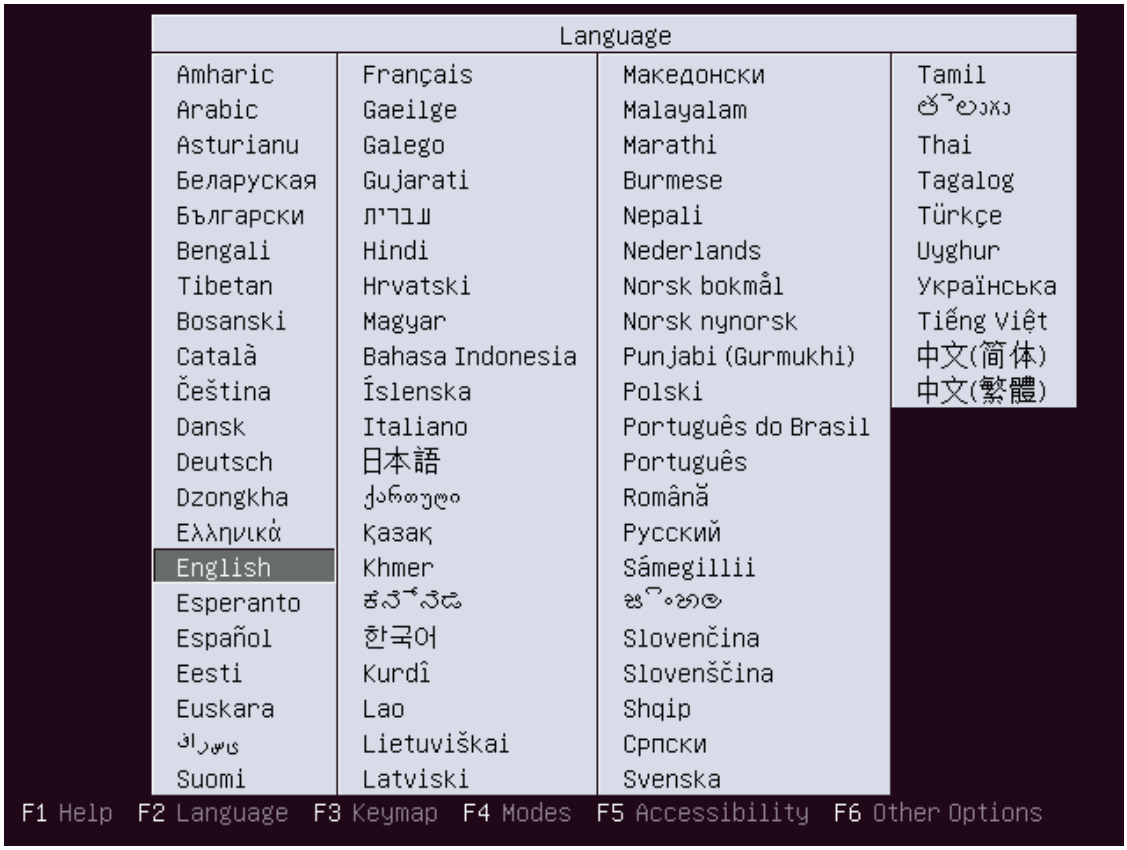

Figure 1: Screen for Choosing Installation Language

As shown in the previous figure, the Ubuntu installation program asks for the language that will be used for the entire installation process. Choose **English** and press Enter to show the installation menu.

The installation menu

This menu shows a group of options that allow installation of Ubuntu Server in the system, including a rescue alternative for broken systems. Choices for checking hard drive defects and testing system memory are also included.

Here is a brief explanation of each option that appears in the menu:

- **Install Ubuntu Server**: Starts the Ubuntu Server installation process.
- **Multiple server install with MAAS**: Allows you to install several servers using MAAS (Metal as a Service).
- **Check disc for defects**: Runs a utility to check the destination hard drive for bad sectors or physical defects.
- **Test memory**: Runs a utility to check system RAM integrity.
- **Boot from first hard disk**: Exits the installation process and tries to boot the system from the first hard drive found in it.

- **Rescue a broken system**: Initiates the System Rescue utility to fix problems found in a previous installation.

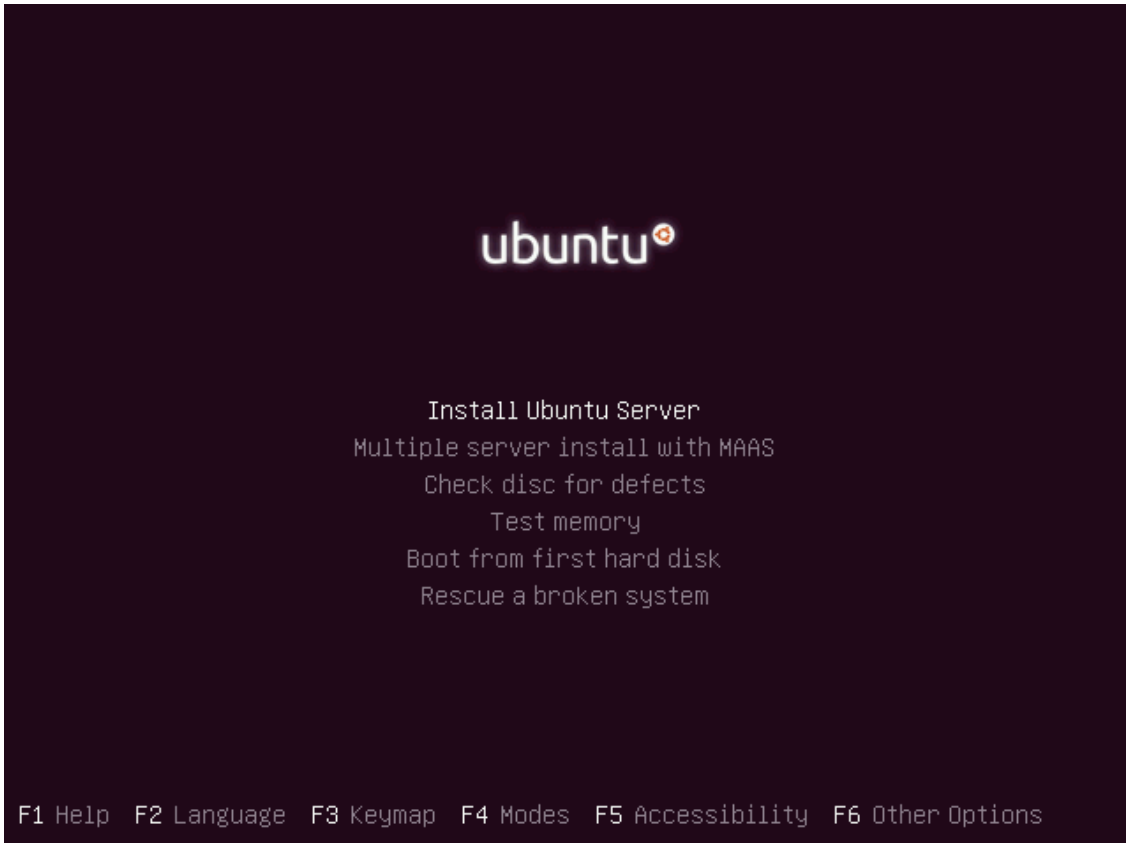

Figure 2: Screen for Installation Menu

Select **Install Ubuntu Server** to start the installation process, which is described in the following section.

The installation process

Language selection

The first step in this process is the selection of the default language that Ubuntu will use, both for installation and for the user interface once Ubuntu is installed.

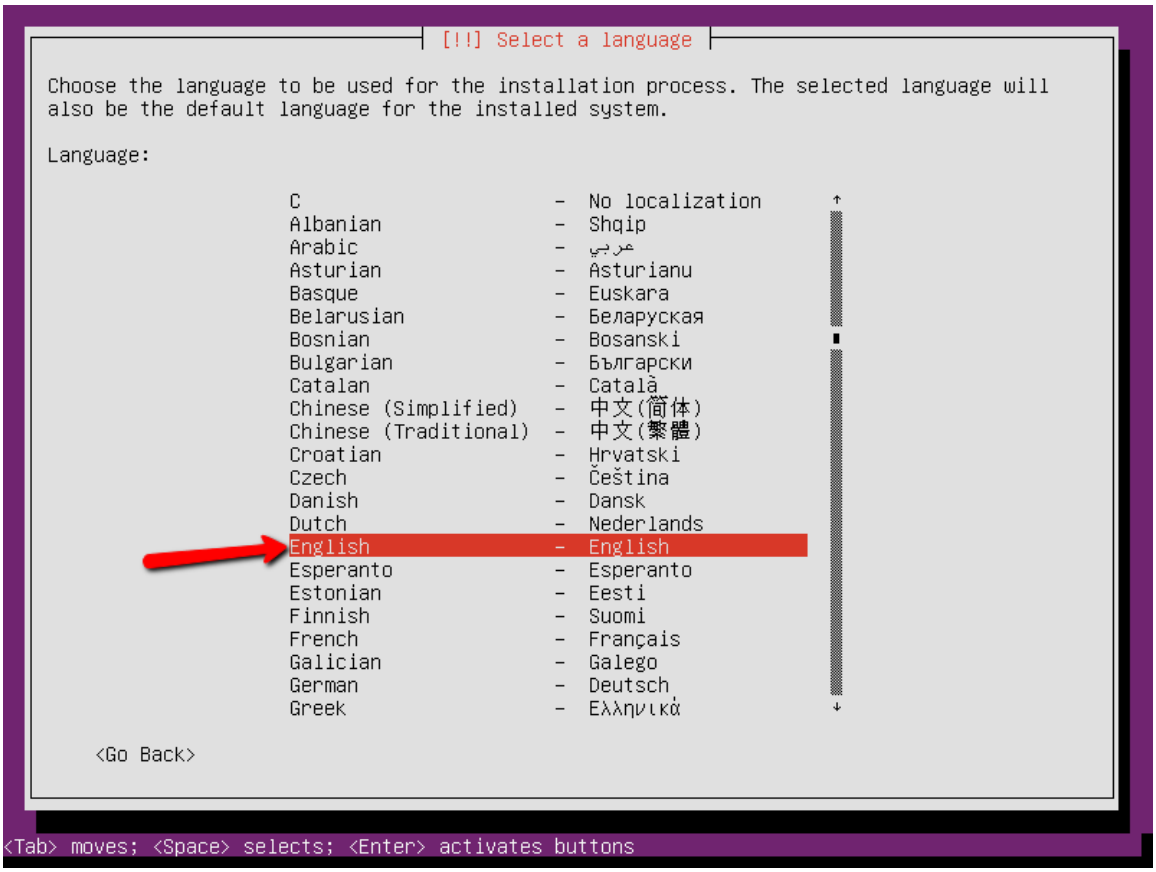

Figure 3: Language Selection Screen

The following table describes the keys you need to use to work with the installation interface.

Table 2: Installation Interface Navigation Keys

Key	Action
Tab	Moves between the interface elements shown on the screen.
Space bar	Selects an interface element and performs the action associated with it.
Enter	Activates any button found on the screen, or selects an element from a list.

To continue the process, press Enter.

Selecting geographical location

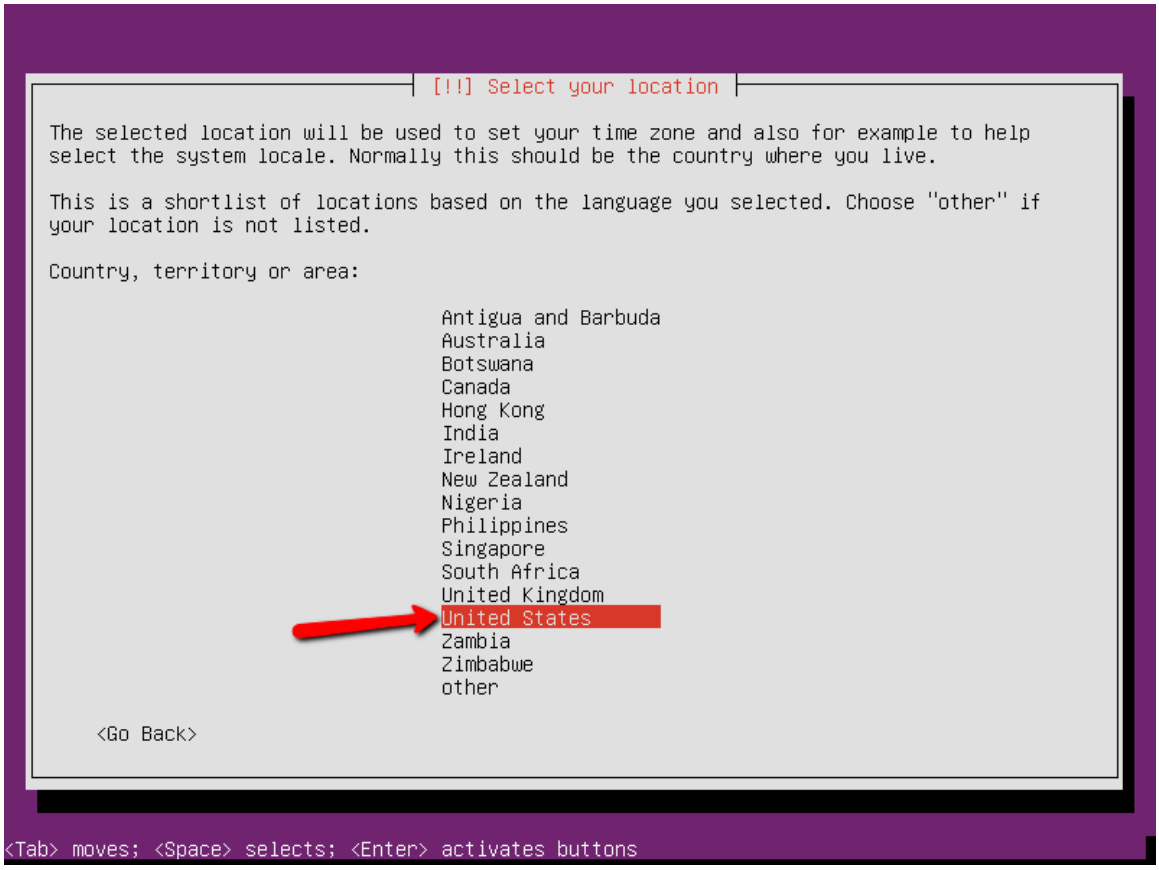

Figure 4: Selecting Geographical Location

This screen allows you to choose the geographical location in which the system will be installed, and for the purpose of this book, we'll select **United States**.

Configuring the keyboard layout

Now the keyboard layout must be selected to make sure it works properly during the installation process and after the system is installed. The configuration can be made automatically (by pressing a series of keys) or by selecting the keyboard type from a list. This depends on the **Yes** or **No** selection, as shown in the following screenshot.

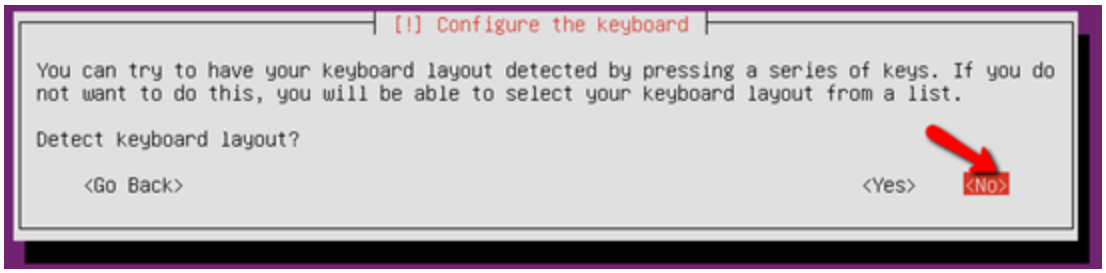

Figure 5: Configuring the Keyboard

To manually choose the keyboard type, press Enter to select **No**.

Selecting the country of origin

Figure 6: Configuring the Keyboard: Country of Origin

Next, the installation program asks for the country of origin for the keyboard attached to the computer. Select **English (US)** as the country of origin to match the geographical location previously chosen.

Selecting keyboard layout

The final step for keyboard configuration consists of selecting the proper keyboard layout, based on the physical distribution of the keyboard attached to the computer. According to the geographical location and the country of origin previously chosen, a set of English variations will be shown on the screen. It will be assumed that English (US) is the matching layout for the keyboard plugged into the computer.

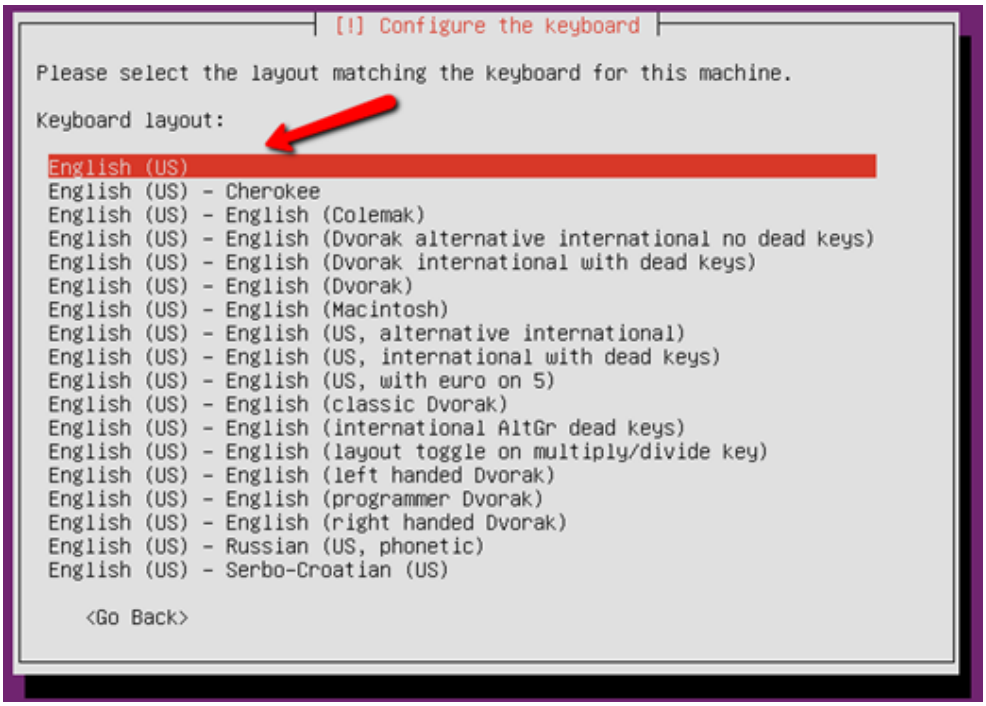

Figure 7: Configuring the Keyboard: Keyboard Layout

Detecting system hardware

At this point, the installation program proceeds to check the system to detect all hardware components installed. This includes CD/DVD-ROM drives, USB interfaces, CPU clock info, the motherboard's chipset, and network hardware. This part of the process is shown in the following screenshots.

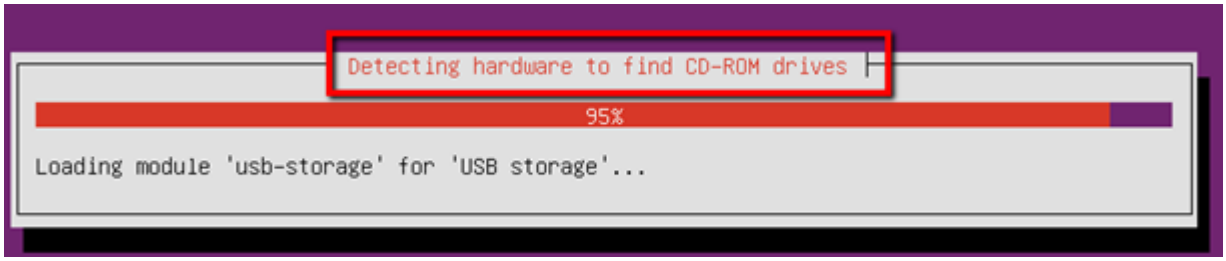

Figure 8: Detecting CD-ROM Drives

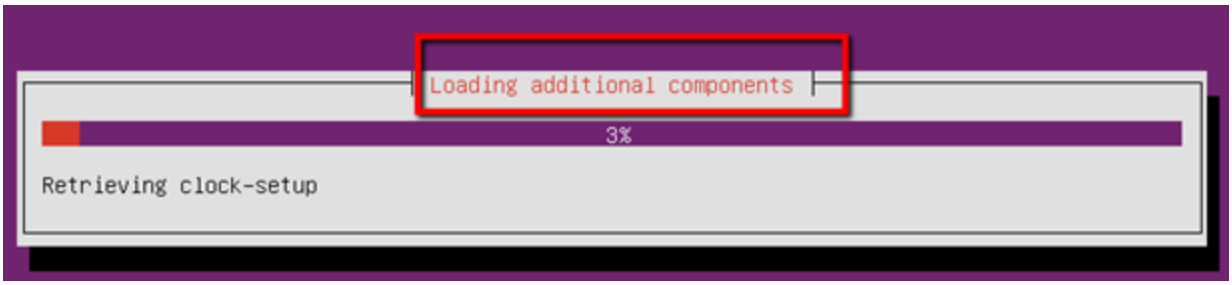

Figure 9: Detecting Additional Components

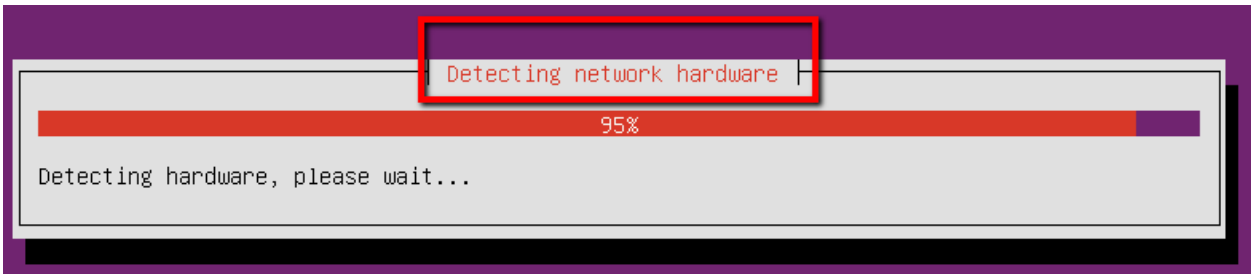

Figure 10: Detecting Network Hardware

Configuring the server for networking

Setting up a hostname

Every computer must be identified with a name for networking purposes. This name is known as the hostname, and must be unique for each computer connected to the network. The installation program suggests that the network administrator should be consulted to assign a correct hostname to the system. In case of building a home network, a name that identifies the main purpose for the computer to be set up is recommended (e.g., mailserver).

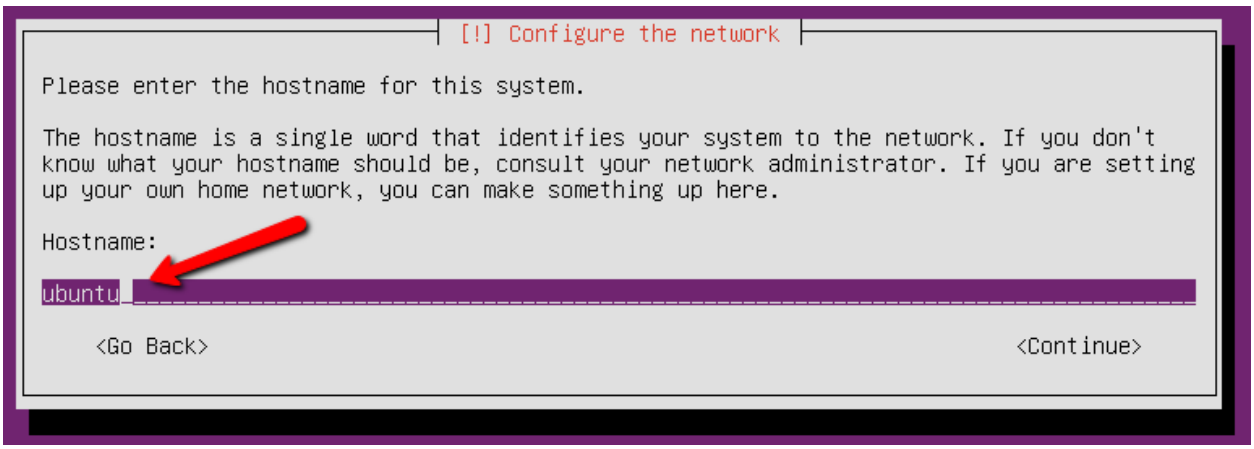

Figure 11: Assigning a Hostname to the System

Setting up a user account

All operating systems derived from UNIX (all Linux distributions, including Ubuntu, fall into this definition) have a root account. This account is the most privileged on the system, giving the person who uses it the ability to carry out all facets of system administration, such as adding accounts, changing user passwords, browsing files, installing software, and more. However, the system assumes that the user knows what he or she is doing, and will perform exactly every task the user requests. Handled without the proper care, the user could wipe out some critical system files and, as a result, make the system crash.

Fortunately, the Ubuntu installation program takes this into account, and asks to set up a user account that will be used instead of the root account. So, every time the user logs into the server, they will have non-administrative privileges, avoiding the possibility of system-fatal accidents.

The user account setup process has two steps. First, the installation program asks for the user's real name. This name will be used by any program that needs to display the user's full name, such as email clients. Then, the process asks for a plain user name, which can be an abbreviation of the full name or other name that's easy to remember.

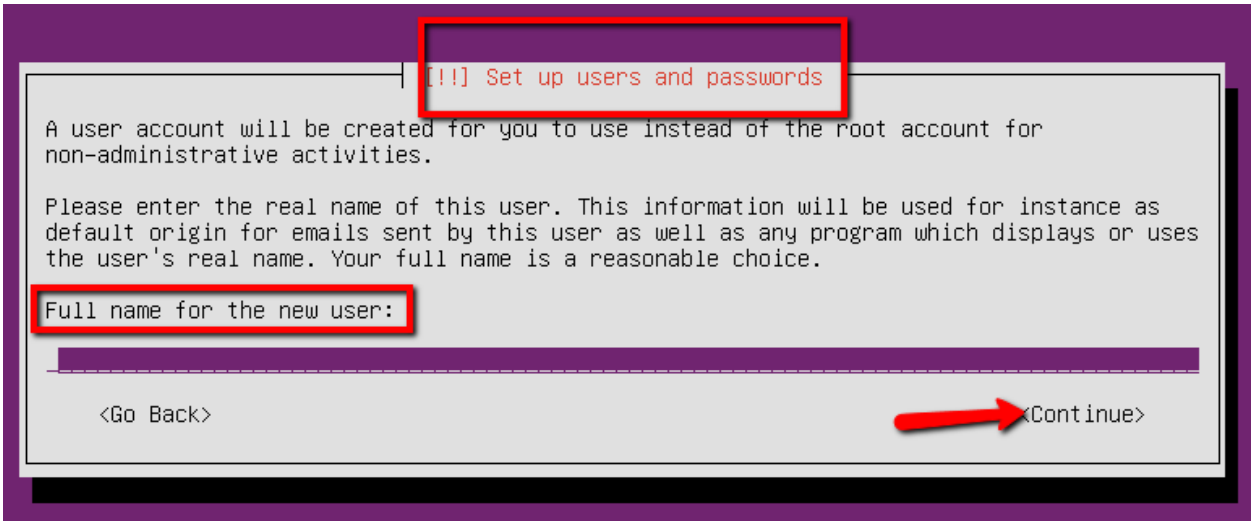

Figure 12: User Account: Entering the Full Name

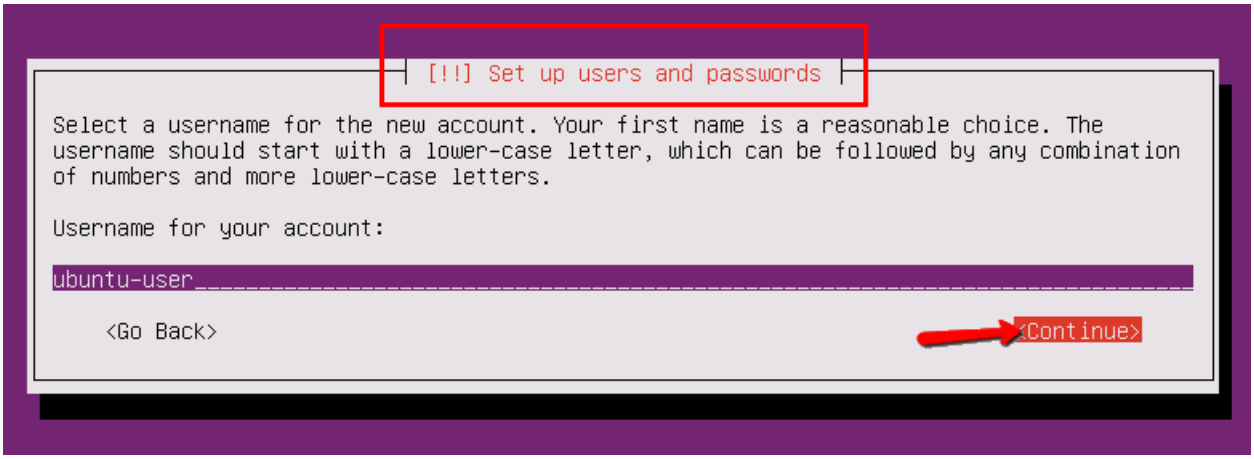

Figure 13: User Account: Entering the Plain User Name

The second step is assigning a password to the user account. A good mixture of letters, numbers, and punctuation is suggested, as well as changing the password at regular intervals.

Type a password and press Enter. Once this is done, a second dialog box will appear asking for the confirmation of the previous entry typed. If both entries match, the process will continue.

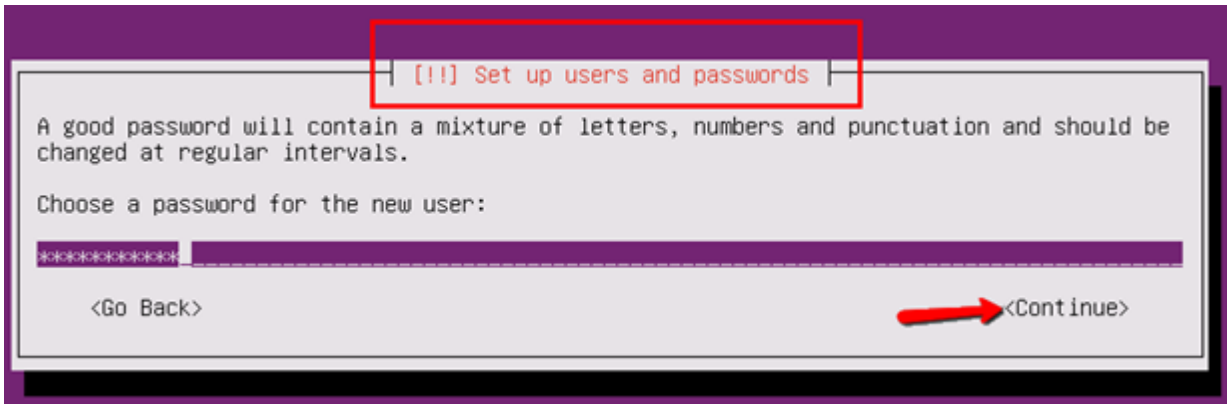

Figure 14: User Account: Entering Password

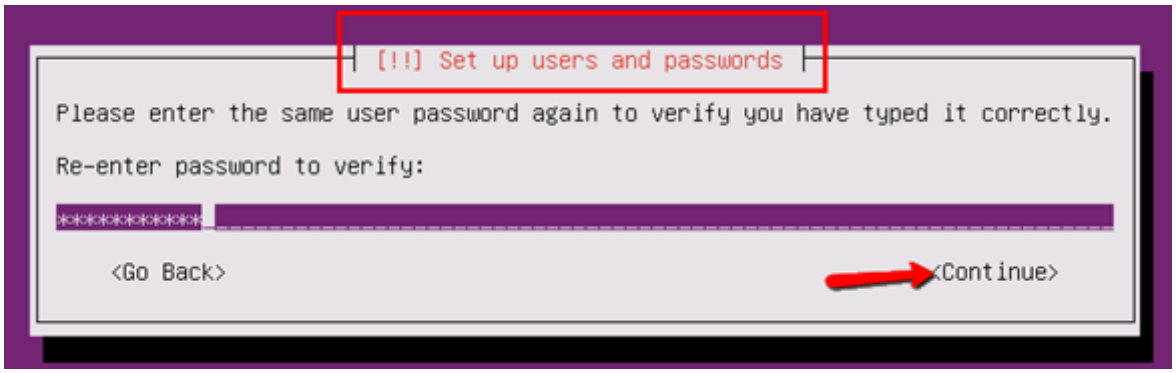

Figure 15: User Account: Re-entering Password for Verification

Now, the installation program wants to know if the user's home directory should be encrypted to keep its contents private from other persons. The home directory is where documents, pictures, music, and pretty much everything else can be stored. The user's application settings are stored here, too.

The suggestion is to keep the home directory unencrypted so that data can be easily recovered from a possible corruption.

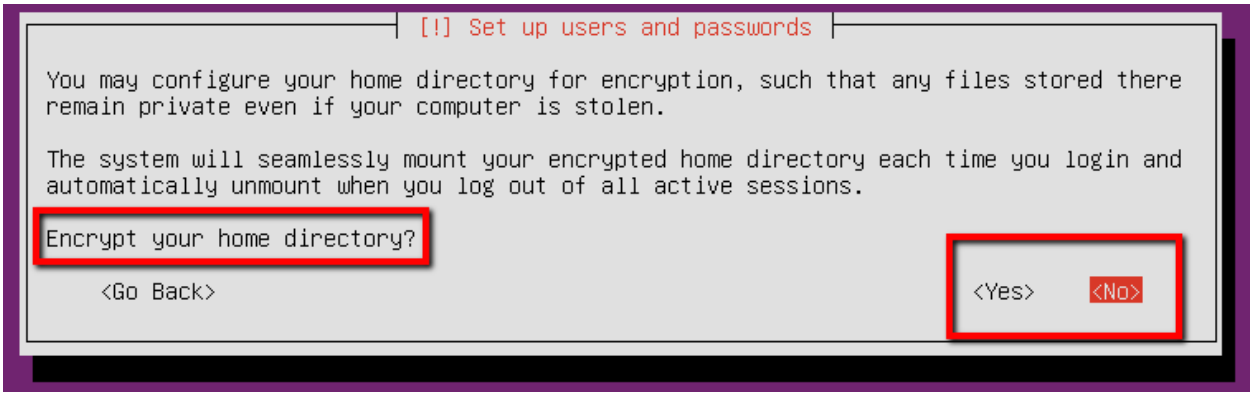

Figure 16: Asking for Home Directory Encryption

Configuring the clock

At this point, the installation program asks for clock configuration. A dialog box appears for selecting the proper time zone according to the physical location in which the system is being deployed. Choose the appropriate time zone with the arrow keys and then press Enter to continue the process.

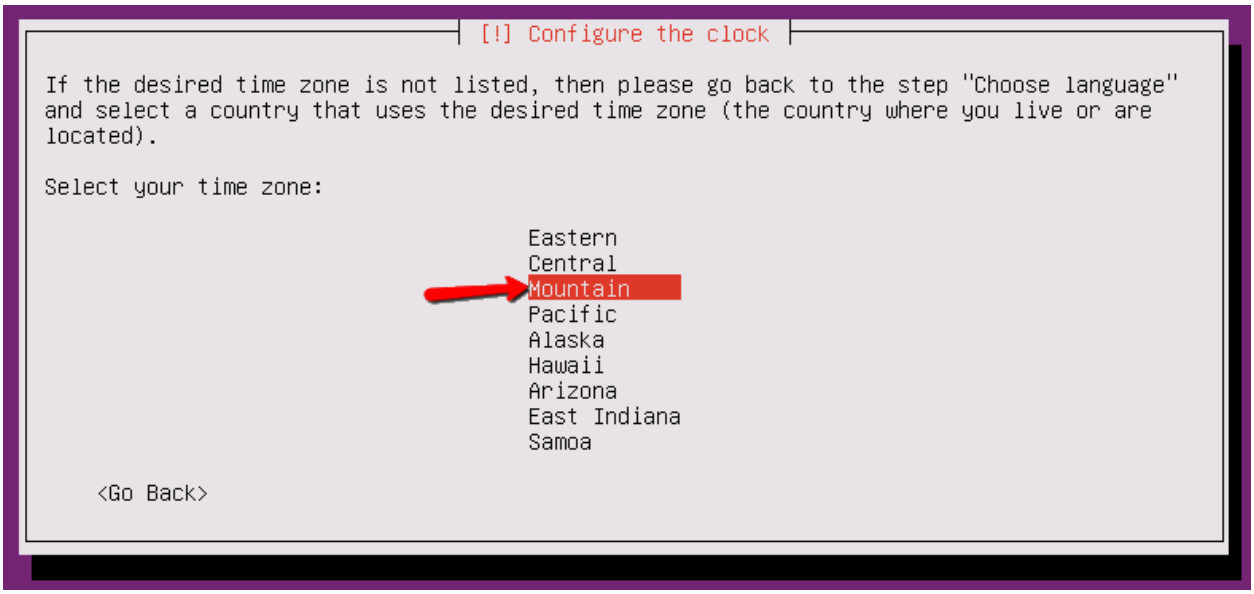

Figure 17: Time Zone Selection Dialog Box

Hard disk partitioning

Disk partitioning is the creation of one or more regions on a hard disk or other kind of storage so that an operating system can manage data in each region separately. Partitioning is the first step for preparing a disk before any files or directories have been created. Once the partitions are created, the disk stores the information about their locations and sizes in a special area known as the partition table. This partition table is read by the operating system before any other part of the disk. Each partition appears in the operating system as a distinct "logical" disk that uses part of the actual disk.

Identifying hard disks in Ubuntu

For a user who is making a transition from Windows, the first thing to understand is that there are no letters (e.g., C or D) to identify a hard disk in Ubuntu. The naming convention for a hard drive starts with a slash and the **dev** abbreviation. This abbreviation stands for "device." Then, **dev** is followed by another slash and the **sd** abbreviation, which stands for SCSI (Small Computer System Interface) device.

An example of a hard disk identification would be **/dev/sd**, but what if the computer has more than one hard disk? This issue is solved by adding a letter at the end of the hard disk identification, starting with **a**. So, for the first drive found in the computer, its identification name would be **/dev/sda**. This is equal to hard disk 0 in Windows. For the second disk, it would be **/dev/sdb**, and so on.

File systems and partitions in Ubuntu

Once a disk partition is created, it needs to be formatted before it can be used to store data. This process includes formatting the partition with a file system. A file system is a way in which files are named and placed logically, for storage and retrieval. This way also takes into account naming conventions, including the maximum number of characters in the name, which characters can be used to name files, and the format for specifying the path to a file through the storage structure.

There are several file systems available for Ubuntu (in fact, for every Linux distribution). The ones used the most are Ext4, Ext3, and Ext2.

In addition to these file systems, there's another type called swap. It's used to increase the amount of total RAM (physical RAM plus virtual memory).

Note: For those who are moving from Windows: The file system used by Windows is named NTFS (New Technology File System). Ubuntu can handle NTFS partitions for storage and retrieval purposes only.

Mount points

A mount point is a directory (typically empty) in the current accessible file system that a hard drive is using, and is used to mount an additional file system on it (logically attached). This

mount point becomes the root directory of the newly added file system, and the file system becomes accessible from that directory.

The most common mount points used on Ubuntu are summarized in the following table.

Table 3: Common Mount Points in Ubuntu

Mount Point	Purpose
/	Root; the equivalent of the C drive in Windows.
/boot	The partition on which the boot loader files are stored.
/home	The partition that will store the user's files and directories.
swap	Used to increase the amount of RAM (virtual memory).

Partitioning and mount point creation

The best way to accomplish this task is by choosing the **Guided Partitioning** option from the partitioning menu. In this case, all common partitions will be created automatically, and the LVM (Logical Volume Manager) will be set up. Press Enter, and the hard disk selection menu will pop up asking for a disk to be used for partitioning.

Once the desired disk is selected, a dialog box will be shown, asking to save the partitioning scheme created by the installation program.

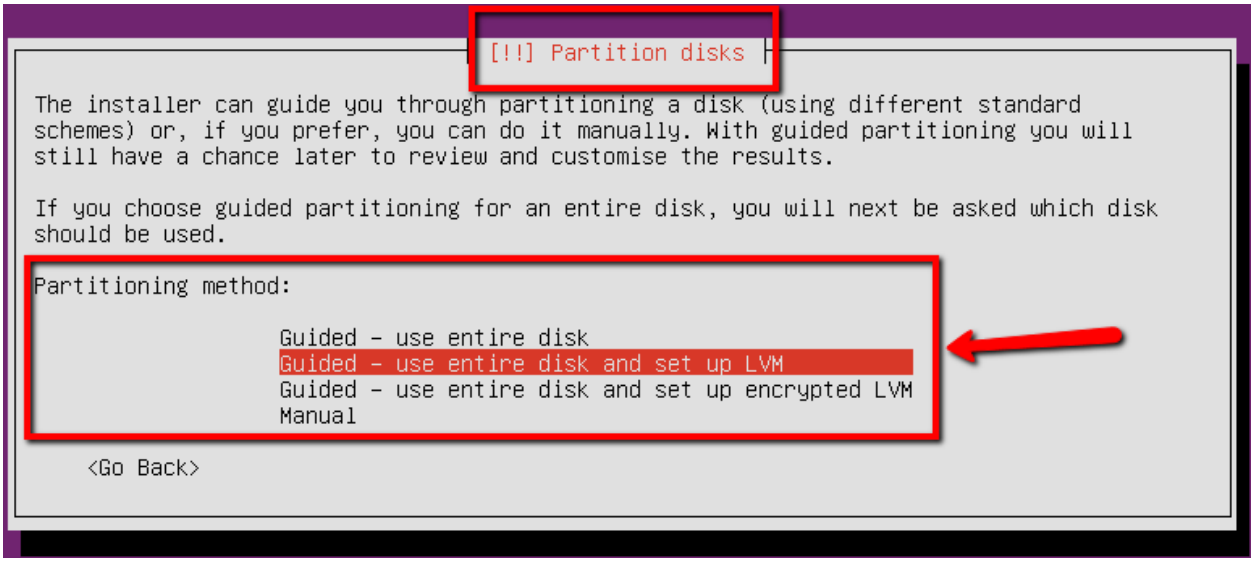

Figure 18: Selection of Guided Partitioning Option

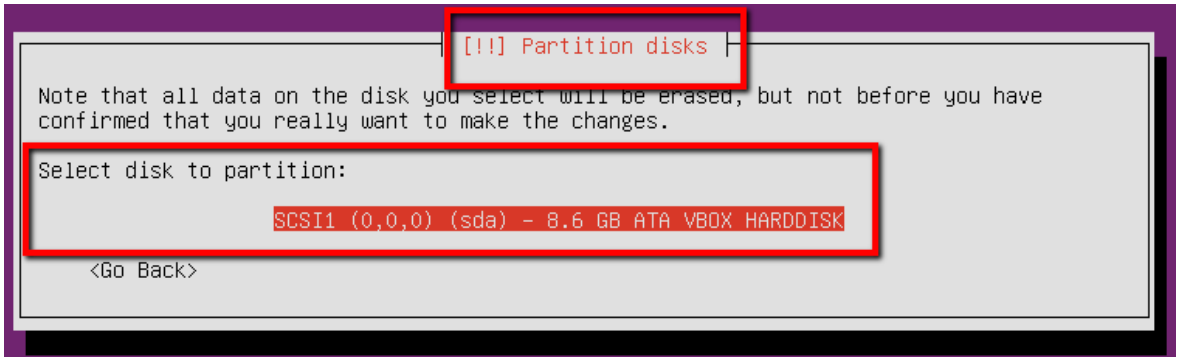

Figure 19: Hard Disk Selection

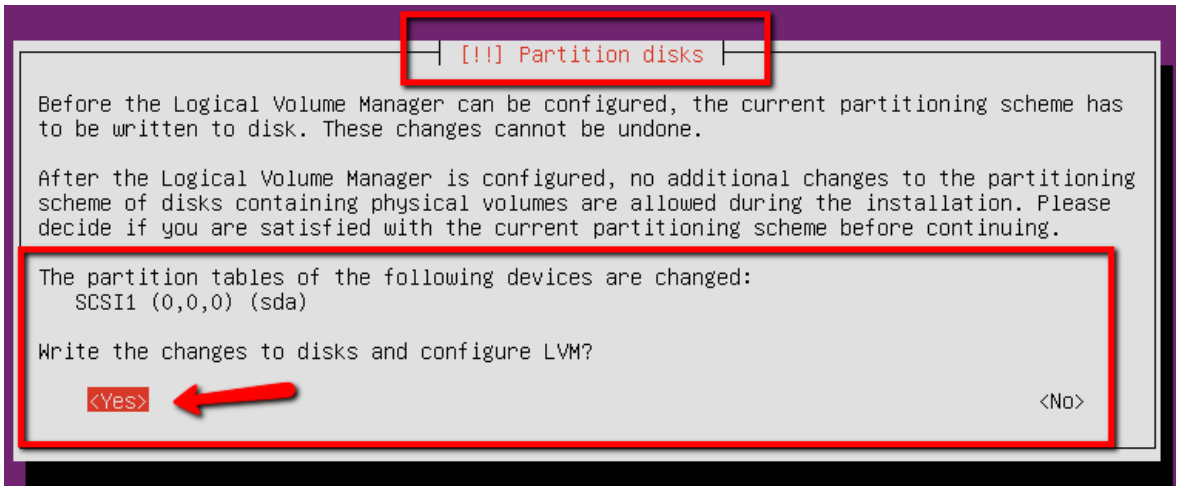

Figure 20: Dialog Box for Saving Changes

Before saving the partitioning scheme on hard disk, the installation program shows it in a dialog box. Press Enter to select **Yes**, and the changes will be written to the disk.

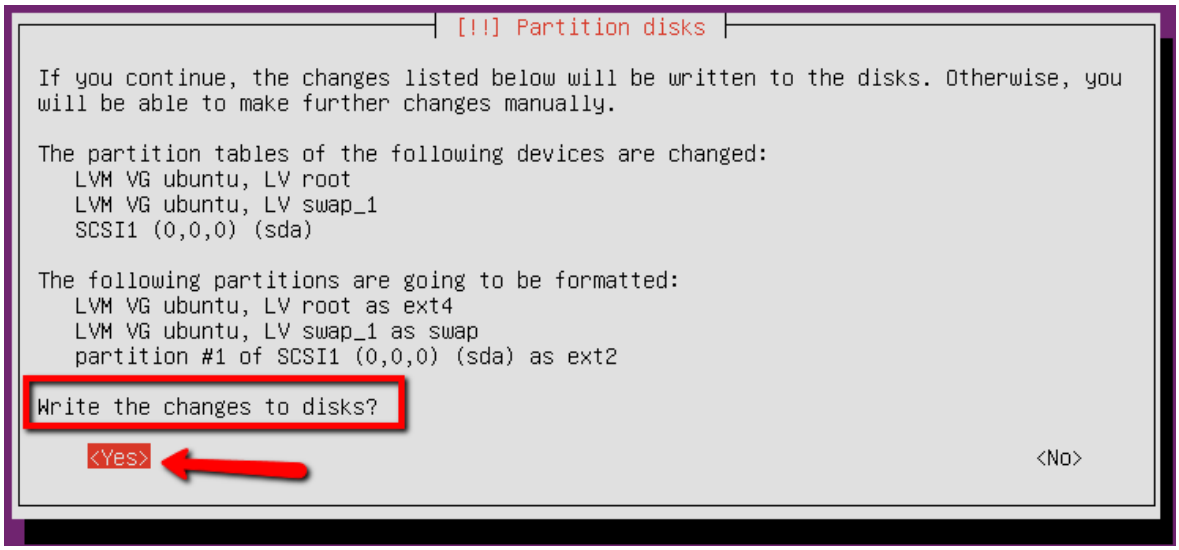

Figure 21: Partition Scheme Dialog Box

Now, the installation program copies the files needed to configure the system and install devices.

Configuring the package manager

The package manager is a management system that deals with installing, upgrading, configuring, and removing software. In addition to providing access to an organized base of over 45,000 software packages, it also facilitates feature dependency resolution capabilities and software update checking.

The package manager needs an Internet connection to download possible upgrades and additional software requested by the installation. At this point, a dialog box appears asking for an HTTP proxy to access the outside. The most common scenario doesn't need a proxy, so this entry should be empty.

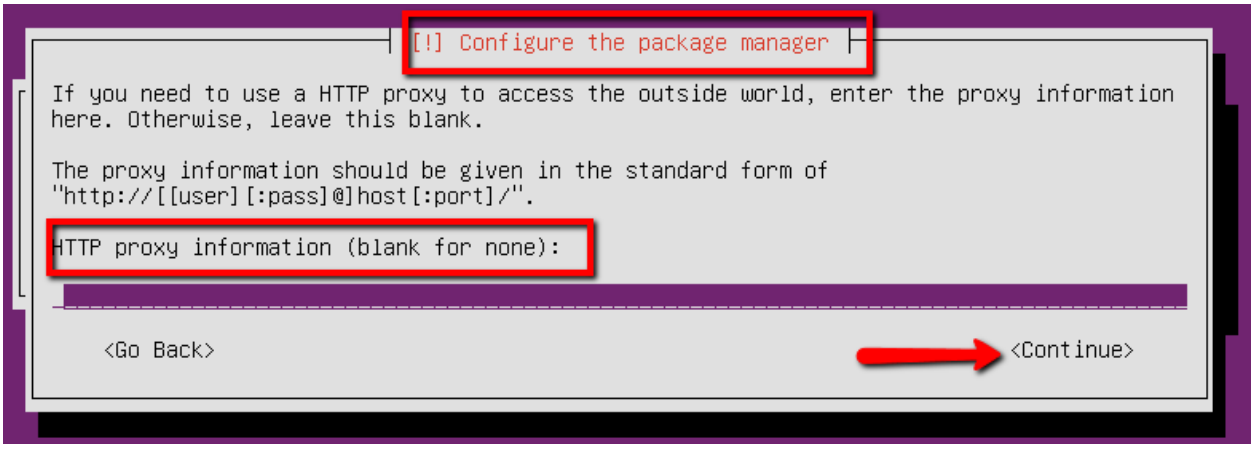

Figure 22: Proxy Configuration Dialog Box

Configuring update management

Ubuntu is continuously evolving. Updates for the software are available from its website, and usually they need to be applied manually by using management tools. However, you can choose to install these updates automatically. In this case, the system will install any software update that it finds on Ubuntu's website.

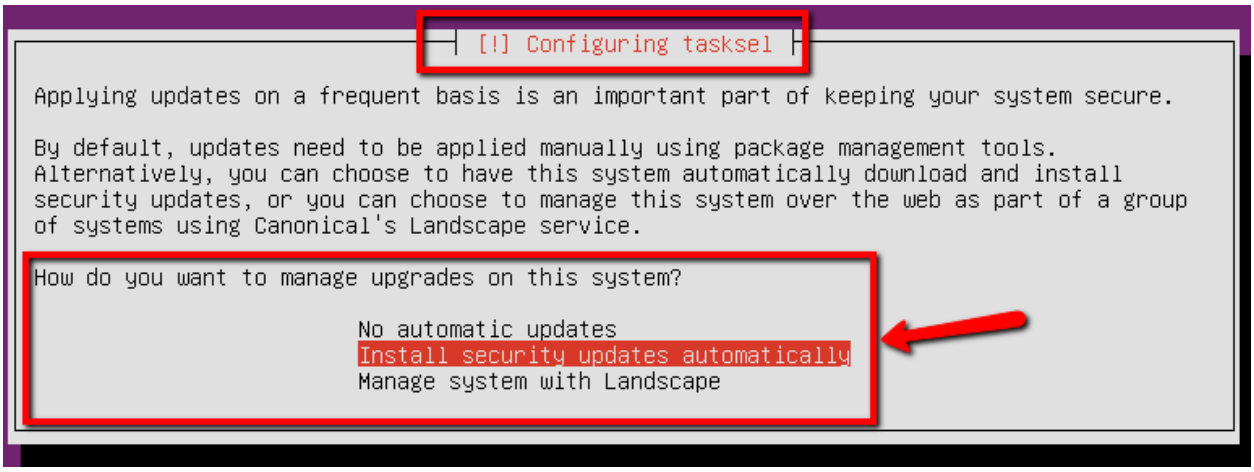

Figure 23: Update Configuration Dialog Box

Install additional software

At this time, the core software of the system is installed. It is possible to install one or more predefined collections of software prior to the installation process ending. You can install software at any time later, so I recommended not making any selection at all at this point.

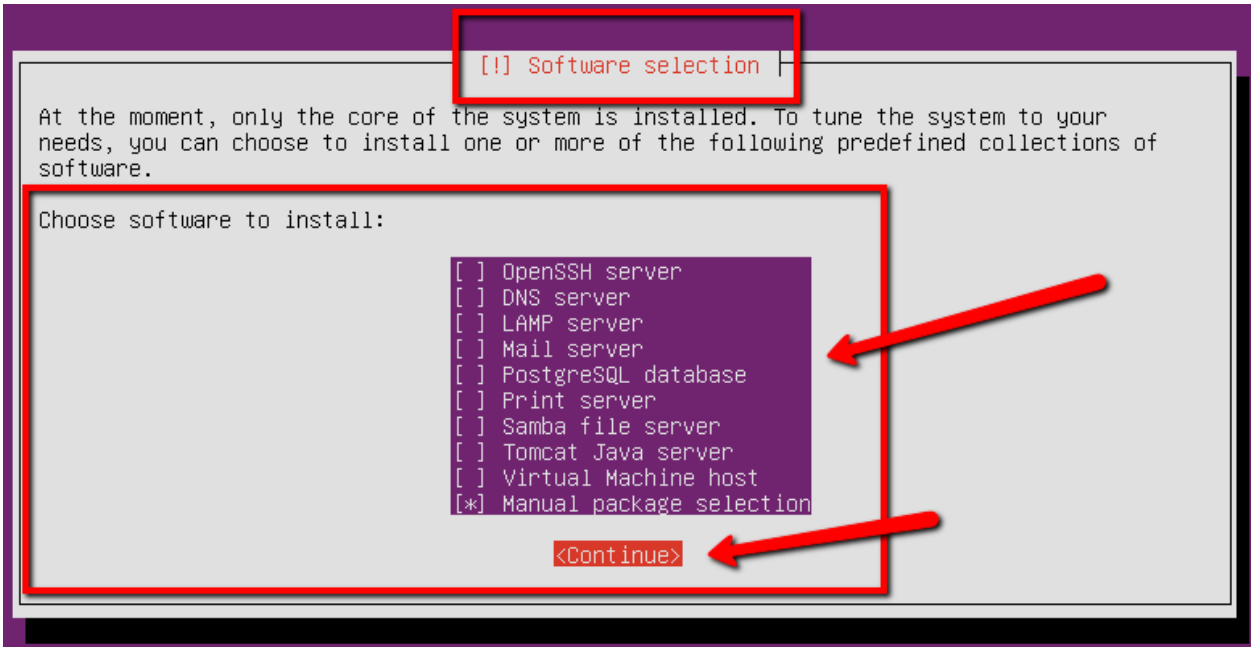

Figure 24: Software Selection Dialog Box

Installing the GRUB boot loader

A boot loader is a small piece of software that places the operating system of a computer into memory. When the computer is powered up or restarted, the basic input/output system (BIOS) performs some initial tests and then transfers control to the Master Boot Record (MBR), where the boot loader is found. GRUB should be installed to manage the Ubuntu booting process.

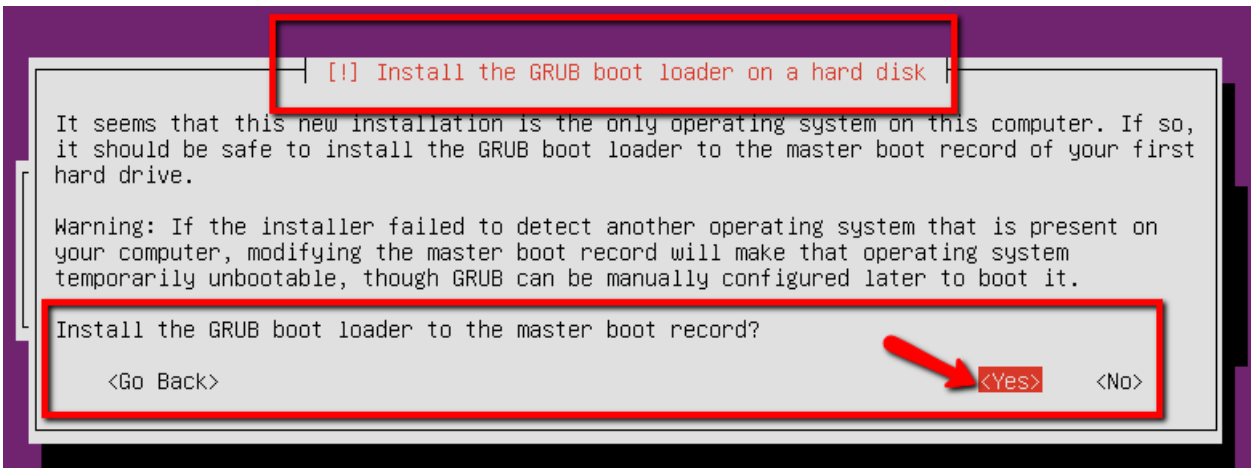

Figure 25: Selecting GRUB Boot Loader Installation

Finishing the installation

This is the final step of the installation process. Once the GRUB boot loader is installed, a series of system files are copied to the disk and a dialog box appears. It indicates that all the installation media used for the process must be removed from the computer. Once this is done, the computer must be restarted. Now you can log into Ubuntu Server with the username and password created during installation.

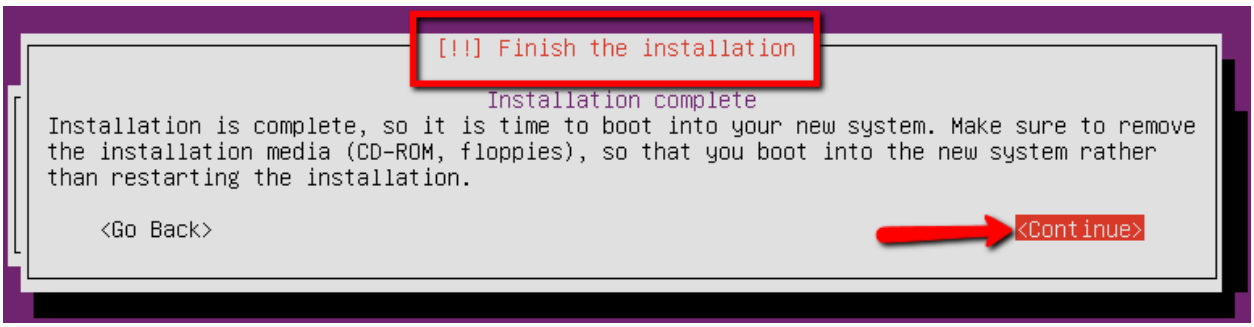

Figure 26: Finishing Installation Dialog Box

Chapter summary

This chapter focused on the Ubuntu Server installation process. It started with some explanation about the hardware requirements for installing Ubuntu properly. Concepts like disk partitioning, mount points, and boot loader were discussed, while the installation process was explained step by step. Also, information about the root user was briefly discussed. If you've made it through this chapter, a useful working copy of Ubuntu Server should now be deployed.

Chapter 3 Beginning with Ubuntu Server

Logging in

Ubuntu Server is loaded once the computer is powered on or restarted. After performing a series of tasks, the login screen appears.

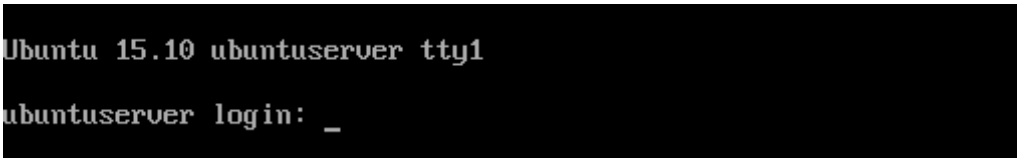

Figure 27: Ubuntu Server login screen

The username and password chosen during installation must be entered in order to start an Ubuntu Server session. When the session begins, the command prompt appears on the screen.

Figure 28: Ubuntu Server command prompt

The command prompt has the following parts: The user name, an **@** (at) symbol, the hostname given to the computer during installation, the working directory (in this case the home directory, indicated by the ~ symbol), and a dollar sign (**$**), which indicates that a non-root user is logged in. The cursor is placed after the **$** symbol appearing at the end, waiting for a command entered by the user. Some concepts, like home directory or non-root user, will be explained later.

The sudo command

When Ubuntu Server was installed, a username and password were supplied to log into a session after installation. This user account lacks administrative privileges, so any attempt to execute a command that requires these privileges will be denied. But what if an administrative task must be done? The **sudo** command comes to save the day!

sudo stands for “super user do” and it’s generally pronounced as “sue dough.” When the user prefixes any command with **sudo**, it will run with elevated privileges. Elevated privileges are required to perform certain administrative tasks like shutting down or rebooting the computer.

For those who are used to Windows, **sudo** is very similar to the WUAC (Windows User Account Control) dialog box that pop ups when you try to do anything important. This dialog box asks the user if they wish to continue, and if the answer is “yes,” the task is performed.

The story is more dramatic in Ubuntu; things might behave quite strangely without the proper permissions. A configuration file may not save correctly, or a program previously installed might

simply refuse to run, but all that is needed to do is ask for permission. And that's when the **sudo** command comes into play.

When the user tries to issue the following command:

Code Listing 1

```
$ reboot
```

This command fails, citing: "must be superuser." In this case, superuser refers to the root user, which holds all necessary permissions to perform any task. To avoid this failure, the **sudo** command is used, as shown in the following snippet.

Code Listing 2

```
$ sudo reboot
```

sudo asks for the current user password, and if it is supplied correctly, then executes the command indicated.

Note: The $ sign, which appears here and will be appear in every given sample, refers to the Ubuntu Server command prompt. The user does not have to type it when entering commands.

Why is sudo better?

sudo is the best and safest way to get elevated privileges because the user doesn't need to know the root password. Every time **sudo** is executed, the user's password is required.

Let's look at another way of doing things. The switch user command **su** will ask the person who uses the computer for the root password. Then, it gives that person a superuser prompt identified by the **#** symbol. That **#** symbol means "Danger! You're logged in as root!" The first command that this person issues may go well. But their forgetfulness will cause them to remain logged in as root. One bad typo, and bam! The entire hard drive is erased instead of that fake file that was downloaded. With **sudo**, that person has to enter **sudo** every time a command is issued. Thus, they don't have to remember to switch back to regular user mode, and fewer accidents will happen.

The sudoers file

This file controls who can use the **sudo** command to gain elevated privileges. Generally, it is located at **/etc/sudoers**. The best and safest way to edit this file is by using the following command.

Code Listing 3

```
$ sudo visudo -f /etc/sudoers
```

This command will start the VI editor with elevated privileges so the user can edit the file and save it. The editor also will put a file lock on the sudoers file, avoiding edits by anybody else until it is saved. Also, before saving the file, the editor will parse it, looking for simple errors.

By default, the **visudo** command will edit /etc/sudoers, so the following sample is also valid for editing this file.

Code Listing 4

```
$ sudo visudo
```

The default sudoers file is configured to grant elevated privileges for the superuser (root) and for all members of the group admin. Also, all members of the group sudo can execute any command. Any user outside the admin and sudo groups cannot use the **sudo** command to gain elevated privileges. This default configuration is shown in the following figure.

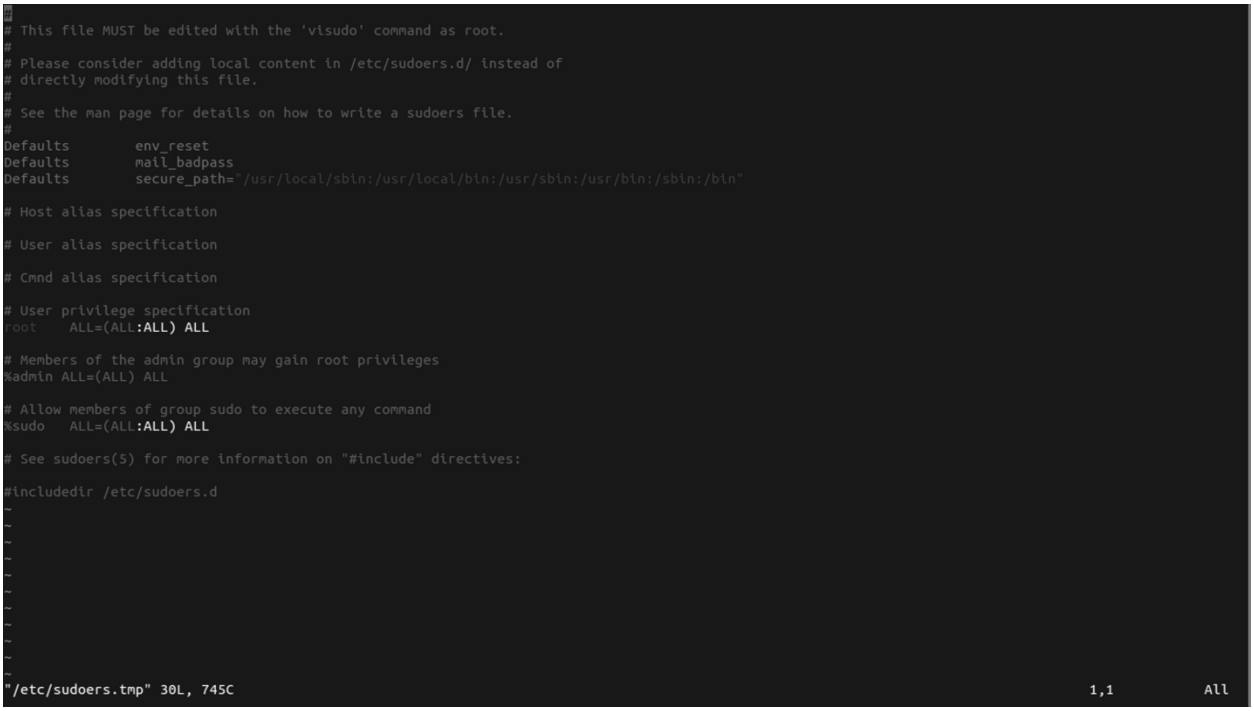

Figure 29: The sudoers Default File

The sudoers file is composed of aliases (basically variables) and user specifications, which control who can run what.

Aliases

There are four kinds of aliases: **User_Alias**, **Runas_Alias**, **Host_Alias**, and **Cmnd_Alias**. An alias definition must be stored in the file using the following form.

Code Listing 5

```
Alias_Type NAME = item1, item2…
```

Alias_Type is one of **User_Alias**, **Runas_Alias**, **Host_Alias**, or **Cmnd_Alias**. A name is a string of uppercase letters, numbers, and underscores starting with an uppercase letter. Several aliases of the same type can be placed on one line by separating them with colons (**:**) using the following form.

Code Listing 6

```
Alias_Type NAME1 = item1, item2… : NAME2 = item3
```

User aliases

User aliases allow you to specify groups of users. Usernames, system groups (prefixed by a **%**), and netgroups (prefixed by a **+**) can be declared, as shown in the following sample.

Code Listing 7

```
# Everybody in the system group "admin" is covered by the alias ADMINS
 User_Alias ADMINS = %admin
 # The users "tom", "dick", and "harry" are covered by the USERS alias
 User_Alias USERS = tom, dick, harry
 # The users "tom" and "mary" are in the WEBMASTERS alias
 User_Alias WEBMASTERS = tom, mary
 # You can also use "!" to exclude users from an alias
 # This matches anybody in the USERS alias who isn't in WEBMASTERS or ADMINS
aliases
 User_Alias LIMITED_USERS = USERS, !WEBMASTERS, !ADMINS
```

Runas aliases

Runas aliases are almost the same as user aliases, but you are allowed to specify users by user IDs (UIDs). For example:

Code Listing 8

```
# Note the hash (#) on the following line indicates a uid, not a comment.
Runas_Alias ROOT = #15
# This is for all the admin users similar to the User_Alias of ADMINS set
earlier
# with the addition of "root"
```

```
Runas_Alias ADMINS = %admin, root
```

In the previous example, the UID 15 is specified as a member of the **ROOT** alias. Then an alias **ADMINS** is defined with the admin group and the root user. Assuming the sudoers file has a command definition in which one or more commands need to be executed with superuser privileges, and that definition grants these privileges for both aliases (**ROOT** and **ADMINS**), the user with a UID of 15 will get superuser privileges.

Host aliases

A host alias is a list of hostnames, IP addresses, networks, and netgroups (prefixed with a **+**). If a netmask is not specified with a network, the netmask of the host's Ethernet interface(s) will be used when matching.

Code Listing 9

```
# This is all the servers present in the network
 Host_Alias SERVERS = 192.168.0.1, 192.168.0.2, server1
 # This is the whole network
 Host_Alias NETWORK = 192.168.0.0/255.255.255.0
 # And this is every machine in the network that is not a server
 Host_Alias WORKSTATIONS = NETWORK, !SERVER
 # This could have been done in one step with
 # Host_Alias WORKSTATIONS = 192.168.0.0/255.255.255.0, !SERVERS
 # But this method may be clearer.
```

Command aliases

Command aliases are lists of commands and directories. You can use this to specify a group of commands. If you specify a directory, it will include any file within that directory, but not in any subdirectories.

Code Listing 10

```
# All the shutdown commands
 Cmnd_Alias SHUTDOWN_CMDS = /sbin/poweroff, /sbin/reboot, /sbin/halt
 # Printing commands
 Cmnd_Alias PRINTING_CMDS = /usr/sbin/lpc, /usr/sbin/lprm
 # Admin commands
 Cmnd_Alias ADMIN_CMDS = /usr/sbin/passwd, /usr/sbin/useradd,
/usr/sbin/userdel, /usr/sbin/usermod, /usr/sbin/visudo
 # Web commands
 Cmnd_Alias WEB_CMDS = /etc/init.d/apache2
```

User specifications

User specifications are where the sudoers file sets who can run what as who. It is the key part of the file, and all the aliases have just been set up for this very point. If this was a film, this is

where all the key threads of the story come together in the glorious unveiling before the climatic ending. Basically, it is important, and without this, any prior setting in the file doesn't make sense.

A user specification is in the following format:

Code Listing 11

```
<user list> <host list> = <operator list> <tag list>: <command list>
```

The user list is a list of users or a user alias that has already been set, the host list is a list of hosts or a host alias, the operator list is a list of users they must be running as or a **Runas_alias**, and the command list is a list of commands or a **Cmnd_alias**.

The tag list allows you to set special things for each command. The **PASSWD** and **NOPASSWD** clauses can be used to specify whether the user must enter a password or not. **NOEXEC** can also be used to prevent any programs launching shells themselves (once a program is running with **sudo** as root, it has full root privileges, so it could launch a root shell to circumvent any restrictions in the sudoers file).

For example:

Code Listing 12

```
# This lets the webmasters run all the web commands on the machine
# "webserver" provided they give a password
WEBMASTERS webserver= WEB_CMDS
 # This lets the admins run all the admin commands on the servers
 ADMINS SERVERS= ADMIN_CMDS
 # This lets all the USERS run admin commands on the workstations provided
 # they give the root password or an admin password (using "sudo -u
<username>")
 USERS WORKSTATIONS=(ADMINS) ADMIN_CMDS
 # This lets "harry" shut down his own machine without a password
 harry harrys-machine= NOPASSWD: SHUTDOWN_CMDS
 # And this lets everybody print without requiring a password
 ALL ALL=(ALL) NOPASSWD: PRINTING_CMDS
```

Shutting down the computer without a password

Every time a shutdown command is issued using **sudo**, the password for the current user is required to execute it. This can be annoying for the server administrator. There's a way to prevent password requirement for shutdown commands, and this is accomplished by editing the sudoers file.

For security reasons, making a backup copy of the default sudoers file is suggested. The following command can do this task.

Code Listing 13

```
$ sudo cp /etc/sudoers /etc/sudoers.bak
```

cp is the command for copying files or directories. In this case, the copy of default sudoers file will be stored in **sudoers.bak**. The **sudo** command is used to avoid an "access denied" error when **cp** attempts to copy the sudoers file.

Note: The first time the `sudo` command is issued, Ubuntu will ask for the password belonging to the user logged in. After that, the next time sudo is executed, sudo won't ask for the user password again until a 15-minute period has elapsed. This behavior can be changed using the `timestamp_timeout` option in the sudoers file (e.g., `timestamp_timeout = 3` will shorten the timeout period to three minutes).

The following command will open the file for editing.

Code Listing 14

```
$ sudo visudo
```

There are two modes in visudo editor: Command Mode, which is accessed pressing the Esc key, and Insert Mode, which is accessed pressing the I key. The user is always in one of them.

To move around the sudoers file, the user must press the Esc key and use the arrow keys to place the cursor where changes need to be made. Then, pressing the I key will switch visudo to Insert Mode where text can be entered.

First, a **Cmnd_Alias** needs to be created in order to specify which commands must be considered for shutting down the computer.

Code Listing 15

```
#Add this line below "# Cmnd alias specification" section
Cmnd_Alias SHUTDOWN_CMDS = /sbin/poweroff, /sbin/halt, /sbin/reboot
```

In the previous example, a **SHUTDOWN_CMDS** alias is created. It groups the commands **poweroff**, **halt**, and **reboot**, which are located in the /sbin directory and perform all operations for shutting down the computer.

Then, a user specification must be created to assign **SHUTDOWN_CMDS** to the user who is responsible for server administration.

Code Listing 16

```
#Add this after the "%admin ALL = (ALL) ALL" line
<username> ALL=(ALL) NOPASSWD: SHUTDOWN_CMDS
```

In this case, **<username>** must be replaced with the username of the person responsible for server administration. This specification indicates that every time **sudo** attempts to execute any command from the **SHUTDOWN_CMDS** alias and the username corresponds to the server administrator, no password will be required.

Once the previous lines are added, the sudoers file should look like the following figure.

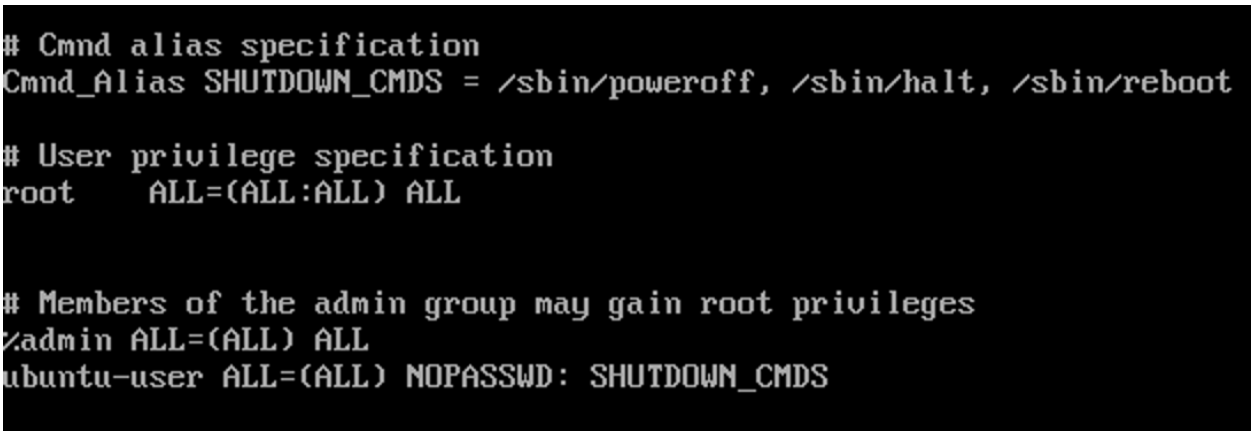

Figure 30: The sudoers File After Being Edited

To save all changes made, press Esc to enter Command Mode. Then, type **:w** and press Enter to save changes to the disk. To exit visudo, type **:q** and press Enter. Typing **:q!** and pressing Enter will exit visudo without saving any changes.

At this moment, if you issue a **sudo** command with any of those specified in the **SHUTDOWN_CMDS** alias, you won't need to provide a password, and the command will be executed as expected.

To shut down the computer server administrator, you can type the following:

Code Listing 17

```
$ sudo poweroff
```

Tip: A backup copy of any file should be made before making modifications to it.

Chapter summary

A description about logging in to Ubuntu Server was shown at the beginning of this chapter. The rest of the chapter focused on how the **sudo** command works. **sudo** is the most important command in Ubuntu, and is used to grant superuser privileges to any command that is executed along with it. It's the best option for doing administrative tasks because every time such a task ends its execution, **sudo** revokes the elevated privileges and brings the system to a normal user state. This forces the use of **sudo** for every administrative command, helping you avoid fatal accidents. When the **sudo** command is used, the system asks for the active user password.

This behavior can be modified using the sudoers file. The sudoers file tells the **sudo** command which users can gain superuser privileges, and in some cases, how certain commands must be executed, and by whom. The chapter ended with an example that explains how to shut down the computer with no password request by the **sudo** command.

Chapter 4 Managing Files and Directories

Managing the file system is probably the most fundamental skill a user needs in order to get an idea of what is around and how to manage it when logged in. At this point, all the tasks to control the operating system have been made from the command line. So, this chapter attempts to get the user up to speed with the basics.

File and directory names in Ubuntu Server

Like in any other operating system, every file or directory stored in the disk must be identified with a name. In Ubuntu Server, the names for each file or directory should comply with the following rules, according to the best file-naming practices, in order to ensure cross-platform operations:

- All file or directory names are case sensitive. So, a file named *Locations* is different from a file with the name *locations*.
- Upper and lowercase letters, numbers, and the dot (.) and underscore (_) symbols can be used.
- Spaces can be used, but it's better to avoid them.
- An extension can be used to identify similar files. This extension is placed after the file name, preceded by a dot (.)
- The name for a file or directory can be a maximum of 255 bytes long.
- A filename must be unique inside its directory.
- The following characters should be avoided in file names: / < > | : & * ?

Exploration and navigation

This section shows the basic commands that allow the user to know where exactly they are in the filesystem, and how to move around in it.

Finding where the user is

Once the user is logged in the server, he or she is typically dropped into the user's home directory since this is the location where the user has full dominion. If the user's home directory doesn't exist, the user is dropped into the root (/) directory.

To find out where the home directory is in relationship with the rest of the filesystem, the following command must be issued.

Code Listing 18

```
$ pwd
```

Then, the operating system will return something like the following:

/home/ubuntu-user

The home directory is named after the user account, and is stored within the directory called **/home**. For the previous example, the name returned by the operating system indicates that the user logged in is named **ubuntu-user**.

Showing the contents of directories

To display a directory's contents, type the **ls** command, as shown in the following sample.

Code Listing 19

```
$ ls /usr/share
```

The operating system will return a list with the **/usr/share** directory contents, which can be files, other directories contained within it, or a combination of both. For the previous example, the contents may be displayed like the following.

Code Listing 20

```
adduser             groff                           pam-configs
applications        grub                            perl
apport              grub-gfxpayload-lists           perl5
apps                hal                             pixmaps
apt                 i18n                            pkgconfig
aptitude            icons                           polkit-1
apt-xapian-index    info                            popularity-contest
. . .
```

The previous list shows only the names of files or directories. If further information about the directory contents is desired, you must use the **-l** flag.

Code Listing 21

```
$ ls /usr/share -l
```

The list obtained displays plenty information, and may look like the following example.

Code Listing 22

```
total 440
drwxr-xr-x   2 root root  4096 Apr 17  2014 adduser
drwxr-xr-x   2 root root  4096 Sep 24 19:11 applications
drwxr-xr-x   6 root root  4096 Oct  9 18:16 apport
```

```
drwxr-xr-x   3 root root  4096 Apr 17  2014 apps
drwxr-xr-x   2 root root  4096 Oct  9 18:15 apt
drwxr-xr-x   2 root root  4096 Apr 17  2014 aptitude
drwxr-xr-x   4 root root  4096 Apr 17  2014 apt-xapian-index
drwxr-xr-x   2 root root  4096 Apr 17  2014 awk
. . .
```

The first block describes the file type and permissions. Each subsequent column, separated by white space, describes the owner, group owner, item size (in bytes), last modification date and time, and the name of the item. For the previous example, the **d** at the beginning of the first block means "directory." If a normal file were listed, this character could be a dash (**-**). The following list shows the file types available.

- **d**: Directory; a container which holds other files or directories.
- **l**: Symbolic link; a pointer that redirects to another file or directory.
- **p**: Named pipe; a special instance of a file that has no contents on the filesystem.
- **s**: Socket; a special file type which allows inter-process networking using the file system's access control protection.
- **b**: Block device; the name to identify a storage device (e.g., /dev/sda for hard disk 0).
- **c**: Character device; a special file that provides unbuffered and direct access to a hardware device (e.g., **/dev/input/tty** for the terminal that is running).

File permissions in Ubuntu

The **-l** flag of the **ls** command displays a block of 10 characters at the beginning of each item returned. The first character indicates the file type, such as a directory (**d**) or a normal file (**-**). The other nine characters refer to the permissions for that file. These characters can be split into three blocks. Each block indicates which permissions are granted for the file "owner," the "group," and "other" users, in that order. An explanation for these kinds of users is presented in Table 4.

Table 4: User Types in Ubuntu

User Type	Description
owner	The username of the person who owns the file or directory. By default, the user who creates the file or directory will become its owner.
group	The user group that owns the file or folder. All users who belong to the same group as the user who created the file or directory will have the same access permissions to that file or folder.
other	A user who isn't the owner of the file or directory and doesn't belong to the same group the file or folder does.

Regarding file permissions, there are three types:

Table 5: Permission Types in Ubuntu

Permission type	Description
r	The user can read a file or directory.
w	The user can write, overwrite, append-to, or delete a file or directory.
x	The user can “run” a file. For example, a script or program.

If an item of the list displayed by the **ls** command shows a block like **drwxr-xr-x**, it means that this item is a directory (**d**) with read (**r**), write (**w**), and execution (**x**) permissions for the owner; the group in this case has read (**r**) and execution (**x**) permissions, and other users have read (**r**) and execution (**x**) permissions, too.

Wildcard characters

A wildcard character is used to add criteria for displaying the content of a directory. The following wildcard characters are used in Ubuntu:

- Asterisk (*****): Used to represent 0 or more occurrences of any character, starting from the point at which asterisk appears. For example, if the user enters **lop***, this means that all files or directories that have a name starting with *lop* will be considered, no matter which characters appear after lop.
- Question mark (**?**): Used to represent a single unknown character in the point where it appears. For example, if the user enters **mem*.tx?**, this means all files that have a name starting with *mem*, no matter which characters appear before the dot, and with an extension starting with *tx*, no matter which character appear after the *x*, will be considered.
- Brackets (**[]**): Used to specify a range. For example, if the user wants to list all JPG filenames beginning with an image and a number between 10 and 30, **image[10-30].jpg** should be entered after the **ls** command.
- Curly brackets (**{}**): Used to separate terms, when each term must be the name of something or a wildcard. For example, if the user enters **{*.pdf,*.png}** after the **ls** command, all files with *pdf* or *png* extensions will be listed.

Moving around the filesystem

To move around the filesystem, the user needs to change the current location (usually the home directory) to any other that exists in the system. This can be accomplished using the **cd** command. In the following example, the user moves to the **/bin** directory.

Code Listing 23

```
$ cd /bin
```

The directory name provided in the previous example is known as an *absolute path*. In Ubuntu, every file and directory is under the top-most directory, which is called the “root” directory. This

directory is referenced by a single, leading forward slash (**/**). An absolute path points to the location of a directory in relation to this top-most directory. This avoids ambiguous references when referring to a directory anywhere in the filesystem.

An alternative is the use of *relative paths*. These refer to a directory in relation to the current directory (the directory in which the user is located). The use of relative paths makes navigation easier, especially when the directories are close in hierarchy. In relative paths, any directory within the current directory can be referenced by name without a leading forward slash. Assuming the user is in the **/usr/share** directory and a "locale" directory exists within it, the following command changes the location to the locale directory.

Code Listing 24

```
$ cd locale
```

Likewise, moving to multiple directory levels with relative paths is possible by providing the portion of the path that comes after the current directory. Again, assuming the user is in the **/usr/share** directory and a **docs** directory exists within a common directory, changing to that directory can be accomplished by typing the following command.

Code Listing 25

```
$ cd common/docs
```

Moving to the previous directory in relation to its hierarchy is also possible. This directory is called the *parent directory*. The following command changes the current location to the parent directory.

Code Listing 26

```
$ cd ..
```

The double dot indicator after the **cd** command refers to the parent directory. Another special indicator is a single dot, which refers to the current directory.

Finally, issuing the **cd** command without providing a directory name takes the user back to the home directory.

Code Listing 27

```
$ cd
```

Viewing files

In contrast to some operating systems, Ubuntu and other Unix-like operating systems rely on plain text files for vast portions of the system. This section discusses different ways to view these files.

One way to view files is by using the **less** command. This command is known as a "pager," because it allows the user to scroll through pages for a file. Unlike the previous commands discussed, which return to the command line immediately after execution, **less** is an application that will continue to run and occupy the screen until the user exits.

To open the **/etc/services** file, which is a configuration file with service information that the system knows about, the following command must be issued.

Code Listing 28

```
$ less /etc/services
```

less will open the file, and the portion of the document that fits in the screen will be displayed.

Code Listing 29

```
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.
#
# Updated from http://www.iana.org/assignments/port-numbers and other
# sources like http://www.freebsd.org/cgi/cvsweb.cgi/src/etc/services .
# New ports will be added on request if they have been officially assigned
# by IANA and used in the real-world or are needed by a debian package.
# If you need a huge list of used numbers please install the nmap package.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
. . .
```

The following table shows the keys that allow you to navigate through the file.

Table 6: less Navigation Keys

Key	Description
Up Arrow and Down Arrow keys	Scroll the file one line up or down.
Page Up	Scrolls the file one page backward.
Page Down	Scrolls the file one page forward.
/	Allows text searching (e.g., **/mail** looks for "mail" in the file).
n (lowercase)	Gets the next result for a text search previously entered with the **/**

Key	Description
	key.
N (uppercase)	Gets the previous result for a text search previously entered with the **/** key.
q	Exits the less program.

File and directory manipulation

This section shows how to create and manipulate files and directories in the filesystem.

Creating files

The most basic method for creating a file is using the **touch** command. This command creates an empty file using the filename and location specified. The following commands create a file named **sample01** in the home directory.

Code Listing 30

```
$ cd
$ touch sample01
```

The **cd** command allows the user to move into the user's home directory. Then, the **touch** command creates the file.

touch allows multiple files to be created at the same time. Absolute paths can be used as well. The following command creates two files, named **sample02** and **sample03**, in the home directory, assuming the user logged in is called **ubuntu-user**.

Code Listing 31

```
$ touch /home/ubuntu-user/sample02 /home/ubuntu-user/sample03
```

Also, **touch** is used to change the timestamps (i.e. dates and times of the most recent access and modification) on existing files and directories. There are several options for doing this. For example, the **-a** option changes the access time only, and the **-m** option changes the modification time only. The following command changes both access and modification time, to the current time for the file **sample03**.

Code Listing 32

```
$ touch –am /home/ubuntu-user/sample03
```

Another option regarding timestamps is **–r**, which is used with a pair of filenames following it. This option tells **touch** that the timestamps for the second file must be used for the first one, instead of the current time.

Code Listing 33

```
$ touch –r /home/ubuntu-user/sample03 /home/ubuntu-user/sample02
```

In the previous sample, the timestamps of **/home/ubuntu-user/sample03** are used for **/home/Ubuntu-user/sample02**.

Another nice option of **touch** is that, unlike the **cp** command (which is used to copy files and directories) and the **mv** command (which is used to move or rename files and directories), it doesn't automatically overwrite existing files with the same name and location. Instead, it changes the last access times for such files to the current time.

Also, **touch** allows you to modify timestamps by going back or forward a specified number of seconds. The **-B** option is used for going back, and the **-F** option is used for going forward.

Code Listing 34

```
$ touch –r /home/ubuntu-user/sample03 –B 40 /home/ubuntu-user/sample02
```

The previous sample makes **/home/ubuntu-user/sample02** 40 seconds older than **/home/ubuntu-user/sample03**.

The **-d** option is used to add a specific last-access time. A string in the *day, month, year, minute:second* format must follow the option.

Code Listing 35

```
$ touch –d '3 Feb 2016 12:15' /home/ubuntu-user/sample03
```

The previous sample sets the last access time of **/home/ubuntu-user/sample03** to February 3, 2016 at 12:15 P.M.

Partial date-time strings can be used. If the only date part is provided, the time part is automatically set to 0:00.

Code Listing 36

```
$ touch –d '3 Feb 2016' /home/ubuntu-user/sample03
```

Likewise, if the time part is provided with no date, it is automatically set to the current date.

Code Listing 37

```
$ touch –d '16:20' /home/ubuntu-user/sample03
```

Creating directories

In a similar way to the **touch** command, the **mkdir** command allows the creation of empty directories. As with the **touch** command, absolute or relative paths can be used. A directory named **test** is created within the home directory in the following sample.

Code Listing 38

```
$ cd
$ mkdir test
```

Again, **cd** moves the user to the home directory. Then, the **mkdir** command proceeds with the directory creation.

The following commands will cause an error:

Code Listing 39

```
$ cd
$ mkdir test/docs/pdfs
```

The operating system throws the following message:

Code Listing 40

```
$ mkdir: cannot create directory test/docs/pdfs: No such file or directory
```

The previous message is displayed because the directory **docs** doesn't exist at the time **mkdir** attempts to create the **pdfs** directory. To ensure that **mkdir** creates all directories necessary to build a given directory path, the **-p** flag must be used. Now, the commands look like the following.

Code Listing 41

```
$ cd
$ mkdir -p test/docs/pdfs
```

The **-p** flag tells **mkdir** that the **docs** directory must be created before the **pdfs** directory is created in case it doesn't exist.

Moving and renaming files and directories

The command **mv** is used to move a file from one location to another. The following sample moves the file **sample01**, located in the home directory, to the test directory, also located in the home directory.

Code Listing 42

```
$ cd
$ mv sample01 test
```

The first part of the command indicates all the items the user wishes to move and the destination directory is indicated at the end.

In case the user wants to move the file **sample01** back from the **test** directory, the special dot reference can be used to point to our current directory.

Code Listing 43

```
$ cd
$ mv test/sample01 .
```

The **mv** command is also used to rename files, which can seem unintuitive. In essence, moving and renaming are both just adjusting the location and name for an existing item.

So, to rename the **test** directory to **tested**, the following commands must be entered.

Code Listing 44

```
$ cd
$ mv test tested
```

In this case, the first part of the **mv** command indicates the item that will be renamed, and the new name is indicated at the end.

Note: It's important to realize that Ubuntu won't prevent the user from certain destructive actions. If the user is renaming a file and chooses a name that already exists, the previous file will be overwritten by the file that is being moved. There is no way to recover the previous file if it's accidentally overwritten.

Copying files and directories

To duplicate an existing item—that is, create a copy of it—the **cp** command is used. For instance, a copy of the **sample01** file can be created in a new file called **sample02**.

Code Listing 45

```
$ cd
$ cp sample01 sample02
```

Unlike with an **mv** operation, after which the **sample01** file no longer exists, now the **sample01** and **sample02** files are available.

Note: As with the `mv` command, it's possible to overwrite a file if the user isn't careful about the target filename used for the operation. For instance, if sample02 already existed in the previous example, its content would be completely replaced by the content of sample01.

Copying directories requires using the **-r** flag. This flag stands for "recursive," as it tells the **cp** command that it must copy the directory and all of its content. The flag is necessary even if the directory is empty.

The following sample duplicates the tested directory into a new directory called **production**, including its content.

Code Listing 46

```
$ cd
$ cp -r tested production
```

Unlike with files, in which an existing destination could result in an overwrite accident, if the target directory exists, the file or directory is copied into it.

Code Listing 47

```
$ cd
$ cp sample01 production
```

The previous sample creates a copy of **sample01** and places it into the production directory.

Removing files and directories

File deletion can be done with the **rm** command. The name of the file to be deleted must be passed to the **rm** command. The following sample deletes the **sample01** file located in the home directory.

Code Listing 48

```
$ cd
$ rm sample01
```

Likewise, to remove empty directories, the **rmdir** command must be used. The following commands remove the tested directory located in the home directory.

Code Listing 49

```
$ cd
$ rmdir tested
```

The previous sample will succeed if the tested directory has nothing within it. Otherwise, the operating system will throw the following message.

Code Listing 50

```
Rmdir: failed to remove tested: Directory not empty
```

To remove a non-empty directory, the command **`rm`** with the **`-r`** flag must be used. Again, the **`-r`** flag stands for "recursive." So, all the directory's content will be removed recursively, including the directory itself.

Code Listing 51

```
$ cd
$ rm -r tested
```

The previous commands remove the directory tested and everything that's stored within it.

Note: Be extremely careful when using any destructive command. There's no "Undo" command or a Recycle Bin (as in Windows) for these actions, so it is possible to accidentally destroy important files permanently.

Editing files

So far we've discussed file and directory manipulation, but this manipulation has been made at the object level. Now, we'll look at how to edit and add content to files.

A good starting point for file editing is the **`nano`** command. Like the **`less`** command previously explained, the **`nano`** command occupies the entire terminal until the user exits.

The nano editor can open existing files or create new ones. If the user wants to create a new file, the name of the file must be typed when the nano editor is called. The following commands allow you to edit the **`sample01`** file located in the home directory.

Code Listing 52

```
$ cd
$ nano sample01
```

Then, the nano interface will appear on the screen.

Figure 31: The nano Editor Interface

At the top of the screen, the name of the application and the name of the file are displayed. The content of the file (currently blank) is displayed at the middle, and a series of key combinations are displayed at the bottom of the screen. These combinations indicate some basic controls for the editor. The ^ character that appears in each combination means the Ctrl key.

Pressing Ctrl+G will display the help screen. Once the user finishes browsing the help screen, pressing Ctrl+X will bring the user back to the file for editing.

When the user is done editing the file, pressing Ctrl+O will save all the changes made to the file. Now, an entry appears at the bottom of the screen that allows the user to confirm the name for the file that will be saved.

Figure 32: Filename Confirmation Entry in nano Editor

To save the file, press Enter. Once the file is saved, the editor returns to its main screen to continue editing the file for further changes.

To quit the nano editor, the user can press Ctrl+X. If there are unsaved changes, the user will be asked about saving those changes.

Figure 33: Nano Editor Asking to Save Changes

The user can press Y to save the changes, N to discard them all and exit, or Ctrl+C to cancel the exit operation. If the user chooses to save the changes, the entry for the filename confirmation shown previously will appear. Again, to save the file, the user needs to press Enter. After the file is saved, the editor ends its execution.

Chapter summary

By now, the user has a basic understanding about Ubuntu Server filesystem navigation. This includes how to see the files and directories available with the **`ls`** command. The user should also know some basic file manipulation commands for viewing, copying, moving, or deleting files. Finally, some basic editing tasks were covered using the nano editor.

With these few skills, the user should be able to continue with the following chapters and learn how to get the most out of Ubuntu Server.

Chapter 5 Security

Overview

Security should always be considered when installing, deploying, and using any type of computer system. Although a fresh installation of Ubuntu is relatively safe for immediate use, it is important to have a balanced understanding of system security posture based on how it will be used after deployment.

This chapter provides a general view of security-related topics as they pertain to Ubuntu Server, and outlines simple measures the user may employ to protect the server and network from any number of potential security threats.

User management

This is a critical part of maintaining a secure system. Ineffective user and privilege management often lead many systems into being compromised. Therefore, it's important that the user understands how to protect the server through simple and effective user account management techniques.

Where is the root user?

Ubuntu developers made a deliberate decision to disable the administrative root account by default in all Ubuntu installations. This doesn't mean that the root account has been deleted or that it may not be accessed. It merely has been given a password that matches no possible encrypted value, and therefore may not log in directly by itself.

Instead, users are encouraged to make use of the **`sudo`** command discussed in the third chapter of this book to carry out system administrative duties. This simple yet effective methodology provides accountability for all user actions, and gives the administrator granular control over which actions a user can perform with said privileges.

By default, the initial user provided during the Ubuntu Server installation process is a member of the group **`sudo`**, which is added to the file **`/etc/sudoers`** (also discussed in the third chapter of this book) as an authorized sudo user. If the user wishes to give any other account full root access through **`sudo`**, this account simply needs to be added to the **`sudo`** group.

Managing users and groups

Adding and deleting users

The process for managing local users is handled by the packages named **`adduser`** and **`deluser`**. The packages **`useradd`** and **`userdel`** are also available for this process. The following command allows you to add a user named **`student01`** in the system.

Code Listing 53

```
$ sudo adduser student01 --quiet
```

When the previous command is used, Ubuntu asks for the password that will be assigned to the user. This password must be typed twice for Ubuntu Server to confirm it.

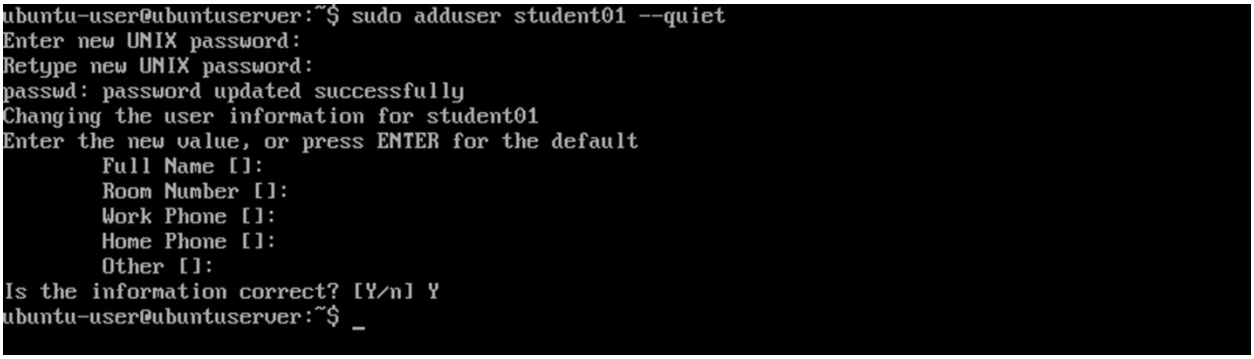

Figure 34: Adding the student01 User Account

The previous figure shows the output generated by the **adduser** command. After the user password is confirmed, Ubuntu asks for some values to update information for **student01**. For the purposes of this book, these data entries aren't important, so each item is intentionally left blank. At the end, **adduser** asks about the accuracy of the data previously entered. The user must press Y to save the user account information.

The **--quiet** flag included in the **adduser** command tells Ubuntu that no info about the account creation process must be displayed when the account is created. After the **student01** account is created, a directory named **student01** is created within the **/home** directory, and all access permissions for the **student01** user are automatically granted.

To delete a user account, the packages **deluser** or **userdel** can be used. The following command deletes the **student01** account created previously with **adduser**.

Code Listing 54

```
$ sudo deluser student01
```

Deleting an account in the previous way doesn't remove its respective home folder. In this case, the user must do it manually.

To erase a user account and its respective home folder at the same time, the **-remove-home** flag must be included in the **deluser** command.

Code Listing 55

```
$ sudo deluser --remove-home student01
```

The previous command deletes the **student01** account along with its respective home folder.

Adding and deleting groups

The **addgroup** and **delgroup** packages control the process of creating and deleting local groups. As for user management, there is also a set of packages named **groupadd** and **groupdel**. The user can type the following command to add a group named **syncfusion**.

Code Listing 56

```
$ sudo addgroup syncfusion
```

Likewise, the **delgroup** package can be used to remove a group. The following command removes the **syncfusion** group previously created.

Code Listing 57

```
$ sudo delgroup syncfusion
```

Adding a user to a group and removing a user from it

To assign a user to a local group, the **adduser** package can be used in the form **adduser <*username*> <*groupname*>**. The following example adds the user **student01** to the local group **syncfusion**.

Code Listing 58

```
$ sudo adduser student01 syncfusion
```

To remove a user from a local group, the **deluser** package can be used in the form **deluser <*username*> <*groupname*>**. The following command removes the user **student01** from the local group **syncfusion**.

Code Listing 59

```
$ sudo deluser student01 syncfusion
```

Displaying all user accounts

The command **compgen** with the **–u** flag is used to display all user accounts that exist in the server, like the following sample.

Code Listing 60

```
$ compgen -u
```

Displaying all local groups

To display all local groups in the server, the **-g** flag for the **compgen** command is used. The following sample lists all local groups belonging to the server.

Code Listing 61

```
$ compgen -g
```

Listing the users belonging to a group

The command **members** can be used to display a list of the users who belong to a given group. In order to use **members**, the user needs to install the members package by issuing the following command.

Code Listing 62

```
$ sudo apt-get install members
```

After the package is installed, the following command can be used to display a list with all the users who belong to the **syncfusion** group.

Code Listing 63

```
$ members syncfusion
```

User profile security

As mentioned in the previous section, when a new user is created, the **adduser** package creates a brand new home directory named **/home/*username***. ***Username*** corresponds to the name of the user that's just been created. The default profile is modeled after the contents found in the **/etc/skel** directory, which includes all profile basics.

If the server will be home to multiple users, the server administrator should pay close attention to the user home directory permissions to ensure confidentiality. By default, user home directories in Ubuntu are created with world read and execute permissions. This means that all users can browse and access the contents of the home directories belonging to other users. This may not be suitable for a secure environment.

The following command verifies the user's home directory permissions, assuming **ubuntu-user** is the user logged in.

Code Listing 64

```
$ ls –ld /home/ubuntu-user
```

The output generated by the command looks like the following sample.

Code Listing 65

```
drwxr-xr-x 2 ubuntu-user ubuntu-user 4096 2007-10-02 20:03 ubuntu-user
```

The first 10 characters of the output shown previously correspond to the permissions granted to the user's home directory. As mentioned in Chapter 4, the three characters at the end show the permissions for "other" users in the system, or in other words, the world permissions. In this case, the **r** and **x** characters mean that read (**r**) and run (**x**) permissions are given to the world. The following command must be used to remove these permissions.

Code Listing 66

```
$ sudo chmod 750 /home/ubuntu-user
```

Now, "other" users have no access to the user's home directory at all.

There's a much more efficient approach to instructing the **adduser** package to never give permissions to the world when creating user home folders. This can be accomplished by editing the **/etc/adduser.conf** file and modifying the **DIR_MODE** variable, assigning it a value of **750**. The nano editor can be used to modify this file.

Now, every time a user account is created, the home directory doesn't give access permissions to the world ("other" users).

Password policies

Setting a strong password policy is one of the most important aspects for a secure environment to avoid security breaches. If any form of remote access is going to be offered to the system, it must adequately address minimum password complexity requirements, maximum password lifetimes, and frequent audits of the authentication systems.

Minimum password length

By default, Ubuntu requires a minimum password length of six characters. This value is controlled in the file **/etc/pam.d/common-password**, which is outlined in the following snippet.

Code Listing 67

```
password [success=1 default=ignore] pam_unix.so obscure sha512
```

Assuming that an eight-character minimum length is required for all user passwords, the variable **minlen** must be changed. This change is shown in the following sample.

Code Listing 68

```
password [success=1 default=ignore] pam_unix.so obscure sha512 minlen=8
```

Note: Basic password checks and minimum length rules do not apply to the administrator using `sudo`-level commands to set up a new user.

Password expiration

It's very important to have a policy that gives a minimum and a maximum age to user passwords, forcing users to change them when they expire.

The following command shows the current status of a user account, where **`username`** corresponds to the name of the user account to be queried.

Code Listing 69

```
$ sudo chage -l username
```

The following snippet is used to view the current status for the **`ubuntu-user`** account.

Code Listing 70

```
$ sudo chage ubuntu-user
```

The output should look like the following:

Code Listing 71

```
Last password change : Feb 02, 2016
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
```

In this case, there are no policies applied to the **`ubuntu-user`** account. The following is an example for setting expiration date (**`-E`**) to 03/31/2016, a minimum password age (**`-m`**) of five days, a maximum password age (**`-M`**) of 90 days, and a warning time period (**`-W`**) of 14 days before password expiration.

Code Listing 72

```
$ sudo chage -E 03/31/2016 -m 5 -M 90 -W 14 ubuntu-user
```

By editing **`/etc/login.defs`**, the password expiration policy can be set for all users in the system.

Code Listing 73

```
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
```

Changing the **PASS_MAX_DAYS** value forces all users to change their password when the number of days specified has been reached.

Managing file and directory permissions

Changing ownership

The **chown** command allows the user to change ownership for a file or directory. The syntax for this command is:

Code Listing 74

```
$ sudo chown <owneruser>[:<ownergroup>] <filename/directoryname>

Where:
<owneruser> is the username which will take the ownership for the file or
folder

<ownergroup> corresponds to the group name which will take the ownership
(optional)

<filename/directoryname> corresponds to the name of the file or directory
which ownership is being changed
```

The following sample changes the ownership for the tested directory, assigning it to the **student01** user.

Code Listing 75

```
$ sudo chown student01 tested
```

To make sure that the **sudo** group and **ubuntu-user** account take ownership of the tested directory, the following command must be issued.

Code Listing 76

```
$ sudo chown ubuntu-user:sudo tested
```

To check ownership and permissions for files and directories, the **ls -l** command (previously explained in Chapter 4, "Managing files and directories") must be used.

Changing permissions

The **chmod** command is used to change file or folder permissions. The syntax for this command follows:

Code Listing 77

```
$ sudo chmod {options} <filename/directoryname>

Where:
{options} corresponds to the kind of permissions that will be granted to
the file or directory and to whom they will be granted.

<filename/directoryname> corresponds to the name of the file or directory
whose permissions are being changed
```

I described the kinds of permissions for a file or folder in Chapter 4. These kinds of permissions are read (**r**), write (**w**), and execute or run (**x**). I also explained that a dash (**-**) represents the absence of that kind of permission.

The previous way in which permissions are identified is known as *alphabetic notation*. This notation requires that all permissions be represented in that order, that is, read (**r**), write (**w**), and execute (**x**). If access for one or more of these permissions will be restricted, a dash (**-**) must be specified instead.

The permissions for a file or directory are given for the owner of the file or directory first, then the group owner, and finally other users (the world). This results in three groups of three values, as in the following example.

Code Listing 78

```
-rw-r--r-- 1 root root    2981 Apr 26  2012 adduser.conf
```

Disregarding the dash at the beginning of the entry in the previous example, the file **adduser.conf** has read (**r**) and write (**w**) permissions for the owner (**rw-**), read (**r**) permissions for the group owner (**r--**), and read (**r**) permissions for other users (**r--**). This entry was obtained using the **ls -l** command, which uses alphabetic notation.

There is another way to identify files or directories permissions, called *octal notation*. This method represents each permission category (owner, group owner, and others) by a number between 0 and 7. The following table matches the values for *alphabetic notation* with the octal notation.

Table 7: Alphabetic and Octal Notation Equivalences

Alphabetic Notation	Octal Notation Value
r (read)	4
w (write)	2
x (execute)	1
- (restricted)	0

In accordance with the previous table, 0 represents no permissions at all, and 7 (the sum of 4 + 2 + 1) represents full read, write, and execute permissions for a category. So, three digits will be used to represent file or directory permissions granted to the owner, the group owner, and other users, in that order.

For the **adduser.conf** file entry shown previously, its permissions representation in octal notation would be 644. This is, read and write permissions (**r** + **w** = 4 + 2 = 6) for the owner, read permissions (**r** + **-** + **-** = 4 + 0 + 0 = 4) for the owner group and read permissions (**r** + **-** + **-** = 4 + 0 + 0 = 4) for other users.

The **chmod** command uses octal notation to change permissions. If the user wants to change permissions for the **adduser.conf** file, it needs to represent these permissions in octal notation first. The following sample grants the **adduser.conf** file full permissions for the owner, read permissions for the group owner, and restricts the access for other users.

Code Listing 79

```
$ sudo chmod 740 adduser.conf
```

Tip: When changing permissions, it must be considered that certain areas of the filesystem and certain processes require specific permissions to run properly. Inadequate permissions can lead to non-functioning applications or errors. On the other hand, overly permissive settings can be a security risk.

The **chmod** command can also use symbolic mode to change permissions. In this case, a plus (+) sign is used to add a permission, and a minus (-) sign is used to remove a permission. Also, an equal (=) sign can be used to set an exact combination for permissions.

The following sample removes write and execution permissions for user, group, and others from the **adduser.conf** file.

Code Listing 80

```
$ sudo chmod =r adduser.conf
```

Permissions can be added or removed for a specific owner. In this case, the letters **u**, **g**, and **o** are used to specify user, group, and others, in that order. If the user wants to remove execution (**x**) permission from the file **adduser.conf** for others, the following sample can be used.

Code Listing 81

```
$ sudo chmod o-x adduser.conf
```

The following sample can be used to add write (**w**) permission to the file **adduser.conf** for the group owner.

Code Listing 82

```
$ sudo chmod g+w adduser.conf
```

Chapter summary

Security issues are very important when a computer system is being deployed. Ubuntu Server takes this into account and disables the root administrative account by default. This action helps users avoid accidents and possible system malfunction. Instead, Ubuntu Server encourages users to employ the **sudo** command, which grants elevated privileges to perform administrative duties, but at the same time provides accountability for all user actions.

Adding and deleting users or groups is another important task for keeping a system safe. To perform these actions, the commands **adduser**, **deluser**, **addgroup**, and **delgroup** are available. Along with these, it's highly recommended to protect each user's home directory from being accessed by other users. This can be accomplished using the **chmod** command to assign the value 750 to each directory's permissions. Also, changing the value of the **DIR_MODE** variable located in the **/etc/adduser.conf** file to 750 will protect the home directory for each new user added to the system from other users' access.

Password policies ensure that security breaches can be avoided the most when a system is deployed. Establishing a minimum password length can be controlled by the **/etc/pam.d/common-password** file by changing the value of the **minlen** variable. Also, the administrator can force each user to change passwords at regular intervals. This can be done by editing the **/etc/login.defs** file to change the value of the **PASS_MAX_DAYS** variable.

Finally, granting ownership and permissions to files and directories makes system security complete. The **chown** and **chmod** commands perform ownership and permissions-changing operations, in that order. Permissions can be identified with *alphabetic notation*, which uses the letters **r**, **w**, and **x** for read, write, and execute permissions. Also, octal notation can be used. This notation uses a series of values between 0 and 7 to represent the read (**r** = 4), write (**w** = 2), and execute (**x** = 1) permissions. A value of 0 means access is restricted.

Chapter 6 Networking

Networks consist of a set of devices (two or more) such as computers, printers, and other related equipment that are connected by either physical cable or wireless links. The purpose of this is to share and distribute information among the connected devices.

Network configuration

This section focuses on managing and configuring a network on the command line.

Ethernet interfaces

Ethernet networking interfaces refer to a circuit board or cards installed in a personal computer or workstation as a network client. These allow a computer or mobile device to connect to a local area network (LAN) using Ethernet protocol as the transmission mechanism.

Ethernet interfaces are identified by Ubuntu using the naming convention of **ethX**, where **X** represents a numeric value. The first Ethernet interface is typically identified as **eth0**, the second as **eth1**, and so on.

The following command can be used to quickly identify all available Ethernet interfaces.

Code Listing 83

```
$ ifconfig -a | grep -i eth
```

The output generated by the command should look like the following snippet.

Code Listing 84

```
eth0 Link encap:Ethernet HWaddr 00:15:c5:4a:16:5a
```

The previous snippet shows that only one Ethernet interface is available. In this case, **eth0** identifies the interface. The hexadecimal digits shown after the **HWaddr** abbreviation correspond to the MAC (Media Access Control) address. This address is a unique identifier assigned to the network interface when it is manufactured. The manufacturer “burns” this address into the interface circuit. A MAC address is formed by six sets of two digits each. This address never changes.

It’s also possible to obtain detailed info about Ethernet interfaces. For doing this, the **lshw** command must be used, like in the following snippet.

Code Listing 85

```
$ sudo lshw –-class network
```

The previous example shows the extended info for the **eth0** interface along with driver details, bus information, and all supported capabilities. The output for the command should look like the following snippet.

Code Listing 86

```
*-network
description: Ethernet interface
product: BCM4401-B0 100Base-TX
vendor: Broadcom Corporation
physical id: 0
bus info: pci@0000:03:00.0
logical name: eth0
version: 02
serial: 00:15:c5:4a:16:5a
size: 10MB/s
capacity: 100MB/s
width: 32 bits
clock: 33MHz
capabilities: pm pcix bus_master cap_list rom …
configuration: autonegotiation = on broadcast = yes …
resources: irq:17 memory:ef9fe000-ef9fffff
```

The user can view the configuration values for any network adapter using the **ethtool**. The following command shows the configuration information for the **eth0** adapter.

Code Listing 87

```
$ sudo ethtool eth0
```

The output should look like the following snippet.

Code Listing 88

```
Settings for eth0:
Supported ports: [ TP ]
Supported link modes: 10baseT/Half 10baseT/Full
100baseT/Half 100baseT/Full
1000baseT/Half 1000baseT/Full
Supports auto-negotiation: Yes
Advertised link modes: 10baseT/Half 10baseT/Full
100baseT/Half 100baseT/Full
1000baseT/Half 1000baseT/Full
Advertised auto-negotiation: Yes
```

```
Speed: 1000Mb/s
Duplex: Full
Port: Twisted Pair
PHYAD: 1
Transceiver: internal
Auto-negotiation: on
Supports Wake-on: g
Wake-on: d
Current message level: 0x000000ff (255)
Link detected: yes
```

It's also possible to gather statistics about network adapters. This can be useful for troubleshooting received (**rx**) and transmitted (**tx**) traffic issues. These statistics can be obtained by running the following command, in this case for the **eth0** adapter.

Code Listing 89

```
$ sudo ethtool –-S eth0
```

The command output should look like the following example.

Code Listing 90

```
NIC statistics:
     tx_packets: 148683
     rx_packets: 179489
     tx_errors: 0
     rx_errors: 0
     rx_missed: 0
     align_errors: 0
     tx_single_collisions: 0
     tx_multi_collisions: 0
     unicast: 116884
     broadcast: 25361
     multicast: 61674
     tx_aborted: 0
     tx_underrun: 0
```

From the previous output the user can see a list of transmitted (**tx**) packets, received (**rx**) packets, and packet errors.

Another useful task that you can do with the **ethtool** command is identify network cards in the system. This applies in cases where there is more than one adapter attached to the computer and the user wants to know which physical card is a recorder for a particular **ethX** logical device. The following example shows how to do this.

Code Listing 91

```
$ sudo ethhool -p eth0 5
```

In the previous snippet, the **-p** flag tells **ethtool** that it has to perform a physical identification for the **eth0** logical adapter. The number **5** at the end of the line corresponds to the number of seconds that the physical device LED will blink. Now, the user only has to look at the back of the computer to know which physical adapter is attached to the **eth0** logical device.

IP addressing

What is IP addressing?

IP addressing is the process of assigning an IP address to each device participating in a computer network that uses the Internet Protocol of communication.

An IP address is a unique numeric label that identifies a device on the Internet or a local network, allowing the device to be recognized by other connected systems.

IP addresses are defined as 32-bit numbers under the system known as Internet Protocol Version 4 (IPv4), which is the most widely used today. However, because of the growth of the Internet, a depletion of available addresses has been foreseen, and a new version of IP (IPv6) has emerged. This new version was developed in 1995 and uses 128 bits for the address. IP version 6 addresses have been deployed since the mid-2000s.

This section will focus on IPv4 addressing and will describe the appropriate commands for successfully doing it.

IPv4 addresses are represented in dot-decimal notation, which consists of four decimal numbers separated by dots, where each number ranges from 0 to 255. Each part represents a group of 8 bits (octet) of the address, for example, 172.16.254.1.

Since 1981, the Classful Network architecture is used for IP addressing under IPv4. This method divides the address space into five classes. Each class, based on the first four bits of the address, defines either a network size, i.e. number of hosts for unicast addresses (classes A,B,C), or a multicast network (class D). The fifth class is reserved for experimental purposes.

The following table summarizes the classes defined for universal unicast (sending messages to a unique destination) addressing.

Table 8: Classful Network Architecture

Class	**A**	**B**	**C**
Leading bits	0	10	110
Size of network number bit field	8	16	24

Class	A	B	C
Size of rest bit field	24	16	8
Number of networks	128 (2^7)	16,384 (2^{14})	2,097,152 (2^{21})
Addresses per network	16,777,216 (2^{24})	65,536 (2^{16})	256 (2^8)
Start address	0.0.0.0	128.0.0.0	192.0.0.0
End address	127.255.255.255	191.255.255.255	223.255.255.255

This design was intended to assign a unique IP address to a particular computer or device in a global end-to-end connectivity environment. However, this is not always necessary in the case of private networks because these networks are not always connected to the Internet. The computers or devices connected to these networks don't require global unique IP addresses, so three non-overlapping ranges of IPv4 addresses for private networks were reserved in RFC 1998 of the IETF (Internet Engineering Task Force). These addresses are not used on the Internet. The following table show these ranges.

Table 9: Private IPv4 Network Ranges

	Start	End	Number of addresses
24-bit block (/8 prefix, 1 x A)	10.0.0.0	10.255.255.255	16,777,216
20-bit block (/12 prefix, 16 x B)	172.16.0.0	172.31.255.255	1,048,576
16-bit block (/16 prefix, 256 x C)	192.168.0.0	192.168.255.255	65,536

These blocks are available for any user, but network administrators typically divide a block into subnets. For example, many small office or home office routers automatically use a default address range from 192.168.0.0 to 192.168.0.255.

IP addressing with Ubuntu Server

There are several commands in Ubuntu used to manage private IP addressing. One of them is **`ifconfig`**. This command allows the user to verify the IP address configuration for a particular network adapter.

Code Listing 92

```
$ ifconfig eth0
```

The previous example shows the address configuration for the **`eth0`** adapter. The following snippet shows the output display after execution of the command.

Code Listing 93

```
eth0 Link encap:Ethernet HWaddr 00:15:c5:4a:16:5a
inet addr:10.0.0.100 Bcast:10.0.0.255 Mask:255.255.255.0
inet6 addr: fe80::215:c5ff:fe4a:165a/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:466475604 errors:0 dropped:0 overruns:0 frame:0
TX packets:403172654 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:2574778386 (2.5 GB) TX bytes:1618367329 (1.6 GB)
Interrupt:16
```

A temporary IP address for a network adapter can be configured with the **ifconfig** command, like in the following example.

Code Listing 94

```
$ sudo ifconfig eth0 192.168.0.25 netmask 255.255.255.0
```

In this case, the IP address assigned will be lost after the computer is rebooted or powered off and powered on again. To configure the system to assign a specific IP address for the **eth0** network adapter, the file **/etc/network/interfaces** must be edited. The following example shows the lines that must be added to the file to assign a static IP address.

Code Listing 95

```
auto eth0
iface eth0 inet static
address 192.168.0.25
netmask 255.255.255.0
gateway 192.168.0.1
```

Then, this IP address will be always assigned every time the system is powered on.

On the other hand, if the user wants the system to assign IP addresses dynamically, the following lines must be added to the **/etc/network/interfaces** file.

Code Listing 96

```
auto eth0
iface eth0 inet dhcp
```

What is a netmask?

Looking at the lines of the **/etc/network/interfaces** file, when it is configured to assign a specific IP address for the network adapter, one of these lines refers to a parameter named netmask. A netmask is a 32-bit number used to divide an IP address into subnets and specify the network's available hosts. Two bits are always assigned automatically in a netmask. For

example, in 255.255.225.0, 0 is the assigned network address, and in 255.255.255.255, 255 is the assigned broadcast address. The 0 and 255 are always assigned and cannot be used.

The following table shows a netmask and its binary conversion.

Table 10: Netmask Example

Netmask	255.	255.	255.	255
Binary	11111111	11111111	11111111	11111111
Netmask length	8	16	24	32

The netmask length can be obtained by counting out the bits in the binary conversion. The previous example is a 32-bit address. However, this address is a broadcast address (used to distribute signals across a network) and does not allow any hosts (computers or other network devices) to be connected to it.

The commonly used netmask, which is 24-bit, is shown in the following table.

Table 11: Netmask Commonly Used

Netmask	255.	255.	255.	0
Binary	11111111	11111111	11111111	00000000
Netmask length	8	16	24	--

The previous netmask allows 2,097,150 networks or 254 different hosts with an IP range from 192.0.1.x to 223.255.254.x. These are plenty of addresses for one network.

The following table shows the commonly used network classes and the netmask that is associated to each one of them.

Table 12: Network Classes and Their Netmasks

Class	Netmask length	Number of networks	Number of hosts	Netmask
A	8	126	16,777,214	255.0.0.0
B	16	16,382	65,534	255.255.0.0
C	24	2,097,150	254	255.255.255.0

According to the previous table, all private networks created in most scenarios belong to Class C networks. So the netmask used for these networks is 255.255.255.0, which appears in the **`/etc/network/interfaces`** file.

What is a gateway?

One line of the **/etc/network/interfaces** makes a reference to a gateway. This is a device (node) in a computer network that acts as a key stopping point for data on its way to or from other networks. Thanks to gateways, the Internet is available to users of private networks.

In a company network, the gateway is usually a router, which takes traffic from any workstation to the outside network that is serving up webpages. In home networks, the gateway is the modem (or modem-router combo) that is provided by the Internet Service Provider (ISP).

For the example, in the **/etc/network/interfaces** file shown previously, the gateway device indicated for that function has an IP address of 192.168.0.1. Thus, if the IP address 192.168.0.3 has been assigned to the device that will serve as a gateway, then this IP address must be indicated beside the **gateway** parameter of the **/etc/network/interfaces** file.

Enabling and disabling network interfaces

If the user doesn't want to reboot the system to apply the **/etc/network/interfaces** file modifications, the user needs to manually disable the network interface first, and then manually enable it again to apply those changes. To manually disable the **eth0** network interface, the following command must be used.

Code Listing 97

```
$ sudo ifdown eth0
```

Then, the following command will enable the **eth0** interface.

Code Listing 98

```
$ sudo ifup eth0
```

The loopback interface

For IPv4, the loopback interface is a range of addresses starting from 127.0.0.1 to 127.255.255.254. All these addresses represent the local computer system. For most purposes, it's necessary to use only one IP address, and this is 127.0.0.1. This IP points to the hostname **localhost**. The localhost is commonly used when the user wants to point to the local computer through a network service, such as a web server.

By default, Ubuntu Server identifies the loopback interface with the logical name **lo**. Its configuration can be viewed using the **ifconfig** command, as in the following example.

Code Listing 99

```
$ ifconfig lo
```

The file **/etc/network/interfaces** has the lines responsible for automatically configuring the loopback interface. It's not recommended to change these lines. Here's an example:

Code Listing 100

```
auto lo
iface lo inet loopback
```

Configuring DHCP (Dynamic Host Configuration Protocol)

What is DHCP?

DHCP (Dynamic Host Configuration Protocol) is a network service that enables a device or computer to assign network settings automatically to other hosts attached to the network. This device is commonly known as a DHCP server.

The other hosts in the network are configured as DHCP clients and have no control over the settings they will receive from the DHCP server, and the configuration for those settings is transparent to the computer's user.

The most common settings provided by a DHCP server to the clients are:

- IP address and netmask
- IP address of the default gateway to use
- IP address of the DNS servers to use

The advantage of using DHCP servers is that changes to the network are made only in the DHCP server, and all network hosts will be reconfigured the next time they poll the DHCP server via their DHCP clients. Also, integrating new devices into the network is easier because the DHCP server will handle IP address availability, and conflicts with IP allocation will be reduced.

What is a DNS server?

In the previous section, I explained that one of the common settings provided by a DHCP server is the IP address of a DNS server.

A DNS (Domain Name Server) is a computer that maintains a directory of domain names and translates them to IP addresses. This is necessary because, although domain names are easy for people to remember, computer or network devices access websites based on IP addresses.

When a user types a web address (e.g., www.google.com), the Domain Name Server specified by the DHCP server looks for the web address entered and translates that name into a machine-friendly IP address (201.165.116.170), then directs the user's Internet connection to the correct website.

Installing DHCP

To install DHCP server in Ubuntu, the following command must be issued.

Code Listing 101

```
$ sudo apt-get install isc-dhcp-server
```

After the installation of the DHCP service, the file **/etc/default/isc-dhcp-server** must be edited to specify which network interfaces should listen to the service. The file should look like the following snippet.

Code Listing 102

```
# Defaults for isc-dhcp-server initscript
# sourced by /etc/init.d/isc-dhcp-server
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
# DHCPD_CONF=/etc/dhcp/dhcpd.conf

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g., "eth0 eth1"
INTERFACES="eth0"
```

In the previous example, the DHCP server is configured to use the **eth0** network interface to listen for all DHCP requests coming from all DHCP clients in the network.

Configuring DHCP

DCHP configuration can be done by editing the **/etc/dhcp/dhcpd.conf** file. The following sample shows a group of configuration settings.

Code Listing 103

```
# minimal sample /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;
subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.150 192.168.1.200;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "mydomain.example";
}
```

This configuration will make sure that the DHCP server gives clients an IP address from 192.168.1.150 to 192.168.1.250. It will lease an IP address for 600 seconds if the client doesn’t

ask for a specific timeframe. Otherwise, the maximum (allowed) lease will be 7,200 seconds. The server will also suggest the client to use 192.168.1.254 as a default gateway, and 192.168.1.1 and 192.168.1.254 as its DNS servers.

Time synchronization with NTP

What is NTP?

NTP (Network Time Protocol) is a protocol for synchronizing time over a network. The client requests the current time from a server and uses it to set its own clock.

Behind the previous description, there is a lot of complexity—there are tiers of NTP servers, with the tier one NTP servers connected to atomic clocks, and tier two and three servers spreading the load of actually handling requests across the Internet. Also, the client software is a lot more complex than you might think—it has to factor out communication delays, and adjust the time in a way that doesn't upset all the other processes that run on the server. Luckily, all that complexity is hidden from the user.

NTP in Ubuntu Server

Ubuntu Server comes with the **ntpdate** command as standard, and will run it once at boot time to set up the server time according to Ubuntu's NTP server, as shown in the following example.

Code Listing 104

```
$ ntpdate -s ntp.ubuntu.com
```

In case that **ntpdate** command wasn't installed during Ubuntu Server's installation process, the user can install it by typing the following.

Code Listing 105

```
$ sudo apt-get install ntpdate
```

As mentioned previously, **ntpdate** is run once at the server's boot time, which may lead to inconsistent logs or other kinds of time-related issues. Fortunately, Ubuntu Server has a daemon (service) which calculates the drift of the system clock and adjusts it continuously. This daemon is **ntpd**, and it can be installed issuing the following command.

Code Listing 106

```
$ sudo apt-get install ntp
```

NTP configuration

The **/etc/ntp.conf** file must be edited in order to change NTP configuration. The user can add or remove server lines. The following snippet shows the default NTP configuration file.

Code Listing 107

```
# Use servers from the NTP Pool Project. Approved by Ubuntu Technical Board
# on 2011-02-08 (LP: #104525). See http://www.pool.ntp.org/join.html for
# more information.
server 0.ubuntu.pool.ntp.org
server 1.ubuntu.pool.ntp.org
server 2.ubuntu.pool.ntp.org
server 3.ubuntu.pool.ntp.org
```

If any change is made to the **/etc/ntp.conf** file, the user has to reload the **ntpd** service.

Code Listing 108

```
$ sudo service ntp reload
```

Chapter summary

An understanding of networking is critical when installing and configuring a computer as a server. In this order of things, Ubuntu Server has a series of commands that allow you to configure the server in the network.

This chapter started with an overview of Ethernet interfaces and how they can be managed in the system. A list of all available interfaces can be obtained with the **ifconfig –a | grep –i eth** command. Also, detailed information for a specific network adapter can be shown with the **sudo lshw –class network** command. Finally, the **ethtool** command allows the user to view the adapter settings, gather statistics about an adapter, and identify one when there are several network adapters attached to the system.

Network connections can't be made without IP addressing; that is, without assigning an IP address to any device that is connected in the network. An IP address is a 32-bit numeric value defined under the IPv4 protocol, which identifies a device and an address for its location in the network. The IPv4 protocol divides the networks into five classes, labeled A, B, C, D, and E. For private networks (built at a home or office), the classes A, B, and C are used. The **ifconfig** command is used to manage IP addressing in Ubuntu Server, allowing the user to assign a specific IP address to the computer, a netmask, and a gateway in order to connect to external networks. This kind of IP addressing is known as static IP addressing. Also, Ubuntu Server allows dynamic (automatic) IP addressing. To establish dynamic or static addressing by default, the **/etc/network/interfaces** file must be edited.

Dynamic IP addressing is possible because of the DHCP protocol. This protocol allows a device in the network to take charge of assigning IP addresses to other devices connected in the

network. The device in which the DHCP protocol resides is known as a DHCP server. Ubuntu can be used as a DHCP server by installing the **`isc-dhcp-server`** service. Once the service is installed, it can be configured by editing the **`/etc/default/isc-dhcp-server`** and the **`/etc/dhcp/dhcpd.conf`** files.

Finally, it is possible to synchronize the system's clock with a remote server's time. To do this, the NTP (Network Time Protocol) must be used. Ubuntu comes with the **`ntpdate`** command and **`ntpd`** service to manage NTP. Configuration for NTP is done by editing the **`/etc/ntp.conf`** file.

Chapter 7 Sharing Network Resources with Windows

Introduction

In many cases, computer networks are comprised of diverse systems. While operating a network made up of Ubuntu Desktop and Ubuntu Server computers would be a good idea, some network environments consist of both Ubuntu and Microsoft Windows systems, and must work together in harmony. This chapter explains principles and tools used to configure Ubuntu Server to share network resources with Windows.

Successfully networking Ubuntu systems with Windows clients involves providing and integrating with services common to Windows environments. Such services assist the sharing of data and information about the computers and users involved in the network. One of the principle pieces of software that Ubuntu Server includes for Windows networking is Samba suite. The following sections will explain some of the common Samba use cases and how to install and configure the necessary software.

File server

One of the most common ways to network Ubuntu and Windows computers is to configure Samba as a file server. This section covers setting up a Samba server to share files with Windows clients.

Installation

To install Samba in the server, the following command must be entered.

Code Listing 109

```
$ sudo apt-get install samba
```

This command will download all necessary packages to install Samba, including software dependencies. When it is done, Samba will be ready to be configured.

Configuration

The main Samba configuration file is located in **`/etc/samba/smb.conf`**. The file has a significant number of comments to document various configuration directives. This file consists of sections and parameters. Each section is identified with a name enclosed in brackets. There are three special sections, *global* (which allows parameters for the whole Samba server operation to be defined), *homes*, and *printers*. Each subsequent section in the file describes a

shared resource, known as *share*. The section name corresponds to the name assigned to the shared resource, and the parameters within the section define the shared resource's attributes. A typical **/etc/samba/smb.conf** file should look like the following example.

Code Listing 110

```
[global]
workgroup = WORKGROUP
security = user
netbios name = ubuntuserver
server string = %h server (Samba, Ubuntu) %v

[sharedfolder]
comment = Ubuntu File Server Share
path = /srv/samba/sharedfolder
browsable = yes
guest ok = yes
read only = no
create mask = 0777
```

The meaning of the parameters included in the **global** section are explained in the following list:

- **workgroup**: This is the name of the Windows workgroup to which the server will be attached.
- **security**: This is the security level for all connections made from a Windows client.
- **netbios name**: This is the name that will be addressed by Windows to identify the server in the network.
- **server string**: This is a descriptive name that will appear in browse lists next to the machine name. **%h** will show the hostname, and **%v** will show the server version.

The section named **sharedfolder** refers to a shared folder that will be visible to Windows clients. The parameters for this section are explained in the following list:

- **comment**: A description that will appear in browsing lists, next to the resource name
- **path**: The absolute path to the folder that will be shared by the server. It's recommended to assign the ownership of this folder to the **nobody.nogroup** group.
- **browsable**: The folder can be browsed by Windows clients (yes).
- **guest ok**: The folder can be accessed by any user (yes).
- **read only**: The client user can add new files or folders (no).
- **create mask**: The default permissions that will be granted to every new file or folder created (according to the permissions rules for Ubuntu).

There must be a shared resource section for every folder that the user wants to be available for Windows clients. The same parameters apply for each subsequent section that is declared.

After the file **/etc/samba/smb.conf** is edited, all Samba services must be restarted to enable the new configuration. The following commands must be issued.

Code Listing 111

```
$ sudo smbd restart
$ sudo nmbd restart
```

Now, the user should be able to browse the folder located in the server by typing **\\ubuntuserver** (the NetBIOS name) or **\\<ipaddress>** (e.g., **\\192.168.0.23**) at the address bar of the Windows File Explorer.

Print server

Samba can be used to share printers installed on an Ubuntu server.

Installation

Before configuring Samba to share printers, a working CUPS (Common UNIX Printing System) installation must be in the server. To install CUPS, the user must type the following command.

Code Listing 112

```
$ sudo apt-get install cups
```

Once CUPS is installed, the service will be started automatically.

Behavior of CUPS may be configured by editing the **/etc/cups/cupsd.conf** file. This file contains the directives needed to establish the appropriate settings to fit the user's needs.

By default on Ubuntu Server, the CUPS server installation listens only in the loopback interface; that is, at IP address 127.0.0.1. It's necessary to instruct CUPS to listen on the actual network interface's IP address. For example, if the CUPS server resides in a computer whose network adapter has the IP address 192.168.0.35, the following example instructs CUPS server to listen on that address to make the server accessible to other computers on the network.

Code Listing 113

```
Listen 127.0.0.1:631           # existing loopback Listen
Listen /var/run/cups/cups.sock # existing socket Listen
Listen 192.168.10.250:631      # Listen on the LAN interface, Port 631
```

Then, the CUPS server needs to be restarted.

Code Listing 114

```
$ sudo systemctl restart cups
```

Installing a local printer

To install a local printer, the user needs to download a printer driver package (.ppd, PostScript® Printer Description) file from the manufacturer's website. Once the package is copied in a filesystem location, it can be installed using the CUPS **lpadmin** command, like in the following sample.

Code Listing 115

```
$ sudo /usr/sbin/lpadmin -p LaserJet -v parallel:/dev/lp1 -P /home/ubuntu-
user/laserket.ppd
```

The previous command will install a LaserJet printer (**-p**) in the parallel port LPT1 (**-v**), which is identified as **/dev/lp1** in Ubuntu. The driver package **laserjet.ppd** (**-P**) will be used. This package is stored in the user's home directory. The sample assumes that the user logged in is named **81buntu-user**.

Configuration

In order to share the printers in the server, the file **/etc/samba/smb.conf** must be edited to modify the **printers** section. The following sample configures Samba to allow any client on the local network to use the installed printers in the server, without prompting for a username and password.

Code Listing 116

```
[global]
workgroup = WORKGROUP
security = user
netbios name = ubuntuserver
server string = %h server (Samba, Ubuntu) %v

[printers]
browsable = yes
guest ok = yes
```

After the file **/etc/samba/smb.conf** is edited, all Samba services must be restarted to enable the new configuration.

Code Listing 117

```
$ sudo smbd restart
$ sudo nmbd restart
```

Securing file and print server

Security modes

According to the Common Internet Filesystem (CIFS) network protocol, two security levels are available: user-level and share-level. These modes are referenced in the Samba configuration file as follows:

- *security* = *user* (user-level): Requires clients to supply a username and password to connect to shares. To sync Samba user accounts with system accounts, the **libpam-smbpass** package must be installed.
- *security* = *domain*: Requires clients to supply a username and password to connect to shares. This is almost the same as security = user, with the exception that the username must exist in a Windows Primary or Backup Domain Controller located in the same network.
- *security* = *ads*: Active Directory support, which allows a computer running Samba to join as a member server for user authentication using LDAP/Kerberos.

Security = user

To implement user-level security mode, the **libpam-smbpass** package must be installed to sync the system users to the Samba user database.

Code Listing 118

```
$ sudo apt-get install libpam-smbpass
```

Then, the file **/etc/samba/smb.conf** must be edited and the parameter **guest ok** of every section needs to be changed. Also, the configuration must establish that Ubuntu passwords and Samba passwords must be synchronized so the users can access the shares in the server.

Code Listing 119

```
[global]
workgroup = WORKGROUP
security = user
netbios name = ubuntuserver
server string = %h server (Samba, Ubuntu) %v
obey pam restrictions = yes
pam password change = yes
passwd program = /usr/bin/passwd %u
passwd chat = *Enter\snew\s*\spassword:* %n\n *Retype\snew\s*\spassword:*
%n\n *password\supdated\ssuccessfully* .
unix password sync = yes

[sharedfolder]
comment = Ubuntu File Server Share
path = /srv/samba/sharedfolder
```

```
browsable = yes
guest ok = no
read only = no
create mask = 0777

[printers]
browsable = yes
guest ok = no
```

The previous example shows how the **/etc/samba/smb.conf** file should look in order to implement user-level security mode. The parameters added after **server string** ensure that the passwords for all server users match Samba user passwords. This must be configured so that connections are accepted when a Windows client attempts to connect to a share, and the password prompt dialog appears asking for credentials. Also, these parameters ensure that every time the user password is changed in Ubuntu Server, it is also changed in Samba.

Any time configuration changes, Samba services must be restarted.

Code Listing 120

```
$ sudo smbd restart
$ sudo nmbd restart
```

Now, every time a user tries to connect to the shared directories or printers, they will be prompted for a username and password and must supply its credentials to make the connection.

Security = domain

This security option allows the authentication of users against the Windows security infrastructure, so Samba reads the user's database information from a Windows domain controller.

The first step is to install Winbind, the part of Samba that is responsible for integrating Windows authentication and the user database into Ubuntu.

Code Listing 121

```
$ sudo apt-get install samba winbind
```

Now, the **/etc/samba/smb.conf** needs to be edited.

Code Listing 122

```
[global]
workgroup = WINDOWSDOMAIN
security = DOMAIN
netbios name = ubuntuserver
```

```
server string = %h server (Samba, Ubuntu) %v
idmap uid = 10000-20000
idmap gid = 10000-20000
windbind use default domain = Yes

[sharedfolder]
comment = Ubuntu File Server Share
path = /srv/samba/sharedfolder
browsable = yes
guest ok = no
read only = no
create mask = 0777

[printers]
browsable = yes
guest ok = no
```

The workgroup setting of the previous configuration file is set to indicate the Windows domain (**WINDOWSDOMAIN**) that will be used. The security option instructs Samba to use the domain for its user and group database.

The Windows users and groups need to be mapped to Ubuntu Server, so **winbind** needs to be instructed which IDs can be used. This is accomplished by specifying a range for both user and group IDs. The **idmap uid** and the **idmap gid** settings establish the ranges for users and groups. These ranges can't overlap those specified in the **/etc/passwd** and **/etc/group** files. In the previous configuration file, a range starting with 10,000 and ending with 20,000 is used for user and group IDs.

The **windbind use default domain** setting tells Samba that it won't need to prefix usernames with the domain (i.e. **WINDOWSDOMAIN\username**) from within Ubuntu, because the default domain will be assumed by default.

After saving the **smb.conf** file and before restarting Samba services, the user needs to join the server to the Windows domain. This can be done with the following command, where **password** corresponds to the Windows domain controller's administrator password.

Code Listing 123

```
$ sudo net rpc join -Uadministrator%'password'
```

Now, the Samba services must be restarted.

Code Listing 124

```
$ sudo smbd restart
$ sudo nmbd restart
```

In order to make Ubuntu use **winbind** in addition to the standard files (**/etc/password** and **/etc/group**) as a user and group database, the file **/etc/nsswitch.conf** must be edited, as in the following sample.

Code Listing 125

```
passwd: compat winbind
group: compat winbind
hosts: files dns winbind
```

After saving the file, **winbind** needs to know which user will initiate sessions in the domain controller. This can be set by issuing the following command, where **password** corresponds to the Windows domain controller's administrator password.

Code Listing 126

```
$ sudo net setauthuser -U Administrator%'password'
```

The following command will display a list with the users of the Windows domain.

Code Listing 127

```
$ sudo wbinfo -u
```

Security=ADS

This security type allows a computer with Ubuntu and Samba to participate in a Windows Active Directory domain as a member server, using Kerberos authentication.

First, the user needs to install Kerberos V5 in the system. The following commands accomplish this.

Code Listing 128

```
$ sudo apt-get install krb5-libs
$ sudo apt-get install krb5-workstation
$ sudo apt-get install krb5-server
$ sudo apt-get install krb5-user
```

Kerberos needs to be set up after installation. To do that, the **/etc/krb5.conf** file must be edited as in the following example (assuming the Active Directory domain name is **85buntu85ion.com** and Kerberos domain controller is installed in the computer with the IP 192.168.0.35).

Code Listing 129

```
[libdefaults]
  ticket_lifetime = 24h
  default_realm = SYNCFUSION.COM
  forwardable = true

[realms]
  SYNCFUSION.COM = {
    kdc = 192.168.0.35
    default_domain = SYNCFUSION.COM
  }

[domain_realm]
  .SYNCFUSION.COM = SYNCFUSION.COM
  SYNCFUSION.COM = SYNCFUSION.COM

[kdc]
  profile = /etc/krb5kdc/kdc.conf

[appdefaults]
  pam = {
    debug = false
    ticket_lifetime = 36000
    renew_lifetime = 36000
    forwardable = true
    krb4_convert = false
  }

[logging]
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmin.log
  default = FILE:/var/log/krb5lib.log
```

After that, the user needs to get a ticket for the Active Directory administrator user with the following command.

Code Listing 130

```
$ kinit Administrator
```

The system will ask for the administrator user password before executing this command. The following command can be used to check to see if a valid ticket was issued.

Code Listing 131

```
$ klist
```

The output for this command should look like the following.

Code Listing 132

```
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: Administrator@SYNCFUSION.COM

Valid starting    Expires           Service principal
27/03/2016 07:17  27/03/2016 17:17  krbtgt/SYNCFUSION.COM@SYNCFUSION.COM
      renew until 28/03/2016 07:17
```

The file **/etc/nsswitch.conf** must be edited to tell Ubuntu that Active Directory users are valid users, too. The **passwd**, **shadow**, and **group** lines should look like the following sample.

Code Listing 133

```
passwd: compat winbind
group: compat winbind
shadow: compat winbind
```

Now, the user needs to set up Samba to support the domain by editing the **/etc/samba/smb.conf** file. This file should look like the following sample.

Code Listing 134

```
[global]
    workgroup = SYNCFUSION
    # Active Directory System
    security = ads
    realm = SYNCFUSION.COM
    # Just a member server
    domain master = no
    local master = no
    preferred master = no
    # Disable printing error log messages when CUPS is not installed.
    Printcap name = /etc/printcap
    load printers = no
    idmap uid = 10000-99999
    idmap gid = 10000-99999
    idmap config SYNCFUSION:backend = rid
    idmap config SYNCFUSION:range = 10000-9999
    winbind enum users = yes
    winbind enum groups = yes
    # This way users log in with username instead of
```

```
username@syncfusion.com
    winbind use default domain = yes
    # Inherit groups in groups
    winbind nested groups = yes
    winbind refresh tickets = yes
    winbind offline logon = true
    # Becomes /home/syncfusion/username
    template homedir = /home/%D/%U
    # No shell access
    template shell = /bin/false
    client use spnego = yes
    client ntlmv2 auth = yes
    encrypt passwords = yes
    restrict anonymous = 2
    log file = /var/log/samba/samba.log
    log level = 2
```

Now, all services must be restarted after saving the file.

Code Listing 135

```
$ sudo winbind restart
$ sudo smbd restart
$ sudo nmbd restart
```

The server needs to join the domain. Having a valid Kerberos ticket, the following command should be executed.

Code Listing 136

```
$ sudo net ads joing -U Administrator
```

The output for the previous command should look like the following.

Code Listing 137

```
Enter Administrator's password:
Using short domain name – SYNCFUSION
Joined 'HOSTNAME' to realm 'Syncfusion.com'
```

All services need to be restarted again.

Code Listing 138

```
$ sudo winbind restart
$ sudo smbd restart
$ sudo nmbd restart
```

The following commands can be entered to list the domain users and groups.

Code Listing 139

```
$ sudo wbinfo -u # lists all the users in the domain
$ sudo wbinfo -g # lists all the groups in the domain
```

The following commands can be used to check if **winbind** is working correctly.

Code Listing 140

```
$ sudo getent passwd   # should return a list with all users on the local
                       # system and from the active directory

$ sudo getent group    # should return a list with all groups and their
                       # members, both from the local system and the
                       # active directory
```

Chapter summary

In most scenarios, computer networks work with both Ubuntu Server or desktop computers and Windows computers. It's necessary to make them work in harmony. One of the principal pieces of software that Ubuntu Server includes for Windows networking is Samba suite. One of the most common ways to network Ubuntu and Windows computers is to configure Samba as a file server. This can be accomplished by executing the **sudo apt-get install samba** command.

After installation, Samba can be configured by editing the **/etc/samba/smb.conf** file. This file consists of sections and parameters. Each section is identified with a name enclosed in brackets. There are three special sections named **global**, (which allows parameters to be defined for the whole Samba server operation), **homes**, and **printers**. Every additional section is known as a *share* and is intended to define a shared network resource, which will be accessed by Windows client computers. The name given to each of these additional sections corresponds to the name of the shared resource, and must be enclosed in brackets.

Each section in the **/etc/samba/smb.conf** file contains parameters. A parameter is a value that tells Samba how to behave according to the purpose of the section in which it is defined. An example of a parameter is **workgroup**, which indicates the name for the group of computers that belong to the network (e.g., softwareved), and it's declared in the **global** section of the file.

The **printers** section of the **/etc/samba/smb.conf** file contains the parameters needed to browse and access all the printers installed in Ubuntu Server from a Windows client. In order to work with printers, a working CUPS (Common UNIX Printing System) installation must be in the server.

Finally, secure access to files and printers can be configured in Samba. There are three security levels to do that: user-level, domain, and ADS. User-level security mode forces every user who wants to access a shared resource to supply credentials (username and password) in order to

be authenticated. Domain security level authenticates users against the Windows security infrastructure, so Samba reads users' database information from a Windows domain controller. To use this security level, **`winbind`** must be installed and configured to tell Ubuntu that domain controller's users are also valid users. Also, the **`/etc/samba/smb.conf`** needs to be edited to tell Samba that the domain security level will be used. The ADS security level allows a computer with Ubuntu and Samba to participate in a Windows Active Directory domain as a member server, using Kerberos authentication. Kerberos must be installed in the Ubuntu computer and configured by editing the **`/etc/krb5.conf`** file. Also, **`winbind`** and Samba must be configured to support the domain.

Chapter 8 Databases

Introduction

A computer running Ubuntu Server can host a DBMS (database management system) to store and process data over a network. Ubuntu Server provides two popular database systems: PostgreSQL and MySQL. This section will explain how to install and configure both.

Using PostgreSQL as a database system

PostgreSQL is an object-relational database system that has features of traditional commercial database systems with the enhancements found in next-generation DBMS. This section briefly explains how to install and configure a PostgreSQL server. If you wish to go any further with PostgreSQL, there's an excellent book in the *Succinctly* series called *Postgres Succinctly* written by Peter Shaw.

Installation

PostgreSQL can be installed using the following command.

Code Listing 141

```
$ sudo apt-get install postgresql
```

This command will download all packages necessary for a successful installation. Once the installation is complete, the user can configure the PostgreSQL server based on his or her own needs.

Configuring PostgreSQL

PostgreSQL allows connections at the localhost computer by default. This means that no other computer in the network can make a connection with the PostgreSQL server. Also, TCP/IP connections are disabled. To configure TCP/IP connections and allow other computers to connect to the PostgreSQL server, the configuration files of PostgreSQL must be edited. These configuration files are stored in the **/etc/postgresql/<version>/main** directory, where **<version>** refers to the version of PostgreSQL installed. Assuming that version 9.4 was installed in the computer, the configuration files can be found in the **/etc/postgresql/9.4/main** directory.

Configuring TCP/IP

To configure TCP/IP, the user needs to edit the **/etc/postgresql/9.4/main/postgresql.conf** file and locate the line **#listen_address =**

'localhost'. Then, the **#** sign must be removed and an asterisk (*****) must be placed instead of the localhost declaration, as in the following example.

Code Listing 142

```
listen_address = '*'
```

Configuring client connections

The default PostgreSQL installation allows connections made only by the localhost. The file **/etc/postgresql/9.4/main/pg_hba.conf** must be edited to allow other computers to make a connection to the PostgreSQL server. The following sample shows the section of the file that must be changed.

Code Listing 143

```
# TYPE   DATABASE   USER  ADDRESS         METHOD
# Ipv4 local connections:
host     all        all   192.168.0.0/24    md5
```

The **host** declaration at the beginning of the line in the previous example indicates the type of connection that PostgreSQL can allow. In this case, **host** means a computer connected into the network. Then, the **all** declaration next to **host** indicates that connections will be allowed for all databases that reside in the server. The second **all** declaration means that any PostgreSQL user can establish a connection. The **192.168.0.0/24** declaration, written in an IP address format, indicates that any computer in the local network with an IP address in the segment **192.168.0.x**, where **x** is any number between 1 and 254, will be allowed to make a connection with the server. Finally, the **md5** declaration indicates that the **md5** hashing method will be used for client authentication.

To apply all changes made to the files, the PostgreSQL server must be restarted by executing the following command.

Code Listing 144

```
$ sudo systemctl restart postgresql
```

Assigning a password to the postgres user

The **postgres** user is the one that holds all the major rights in PostgreSQL server. After installation and configuration, a password must be supplied to the **postgres** user in order to connect to the server. First, a connection to the PostgreSQL database named **template1** needs to be made.

Code Listing 145

```
$ sudo -u postgres psql template1
```

The previous example makes the connection between the **template1** database and the user **postgres**. Since **template1** is a default database with no storage purpose, a password is not necessary to make the connection. Now, the PostgreSQL prompt will appear on the screen.

Code Listing 146

```
Psql (9.4)
Type "help" for help.

Template1=#
```

The **postgres** user password can be changed by running the following command at the PostgreSQL prompt.

Code Listing 147

```
ALTER USER postgres WITH ENCRYPTED PASSWORD '<user_password>';
```

In the command previously shown, **<user_password>** must be replaced with the desired password for the **postgres** user.

To exit PostgreSQL, the user must type **\q** and press Enter.

Using MySQL as a database system

MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL). The system was originally conceived by the Swedish company MySQL AB, which was acquired by Oracle in 2008. Developers can still use MySQL under the GNU General Public License, but anyone who uses MySQL for non-open-source commercial projects must buy a commercial license from Oracle.

This section explains briefly how to install and configure MySQL. If you want to acquire further information about MySQL, see the official website for documentation.

Installation

The following command installs MySQL in the server.

Code Listing 148

```
$ sudo apt-get install mysql-server
```

The installation process will prompt for a password from the MySQL root user. The MySQL root user, like the Ubuntu Server root user in the operating system, is the one that holds the highest level of rights for the database server.

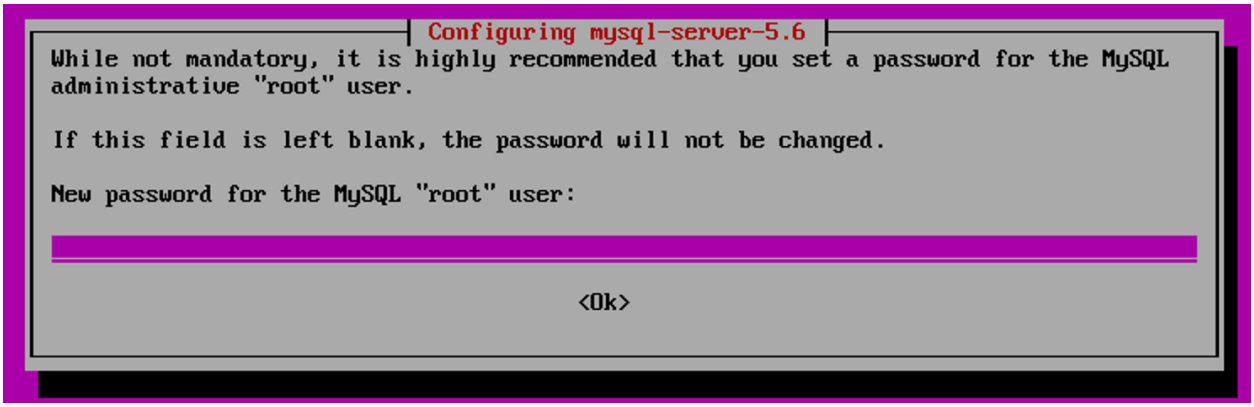

Figure 35: Dialog Box for Root User Password Assignment

The user can run the following command to verify that MySQL server is running.

Code Listing 149

```
$ sudo netstat -tap | grep mysql
```

The response of the previous command should look like the following example.

Code Listing 150

```
tcp 0 0 localhost:mysql *:* LISTEN 2556/mysqld
```

If MySQL server is not running after installation, the following command will start it.

Code Listing 151

```
$ sudo systemctl restart mysql
```

Configuring MySQL

The file **/etc/mysql/my.cnf** must be edited to configure MySQL settings. For example, to configure MySQL to listen for connections from network hosts, the **bind-address** directive of the **[mysqld]** section must be set to the server's IP address, like in the following example.

Code Listing 152

```
[mysqld]
bind-address = 192.168.0.5
```

It's assumed in the previous example that the IP address for the server is 192.168.0.5.

MySQL database engines

While the default configuration of MySQL provided by the Ubuntu packages is perfectly functional and performs well, there are things the user may wish to consider before using it.

MySQL is designed to allow data to be stored in different ways. These methods are referred to as either database or storage engines. There are two main engines that are interesting for the user: InnoDB and MyISAM. Storage engines are transparent to the end user. MySQL will handle things differently under the surface, but regardless of which storage engine is in use, the user will interact with the database in the same way.

Each engine has its own advantages and disadvantages. While it is possible (and may be advantageous) to mix and match database engines on a table level, doing so reduces the effectiveness of performance tuning the user can do, as it'll be splitting the resources between two engines instead of dedicating them to one. Here are descriptions for the engines:

- **MyISAM** is the older of the two engines. It can be faster than InnoDB under certain circumstances, and favors a read-only workload. Some web applications have been tuned around MyISAM (though that's not to imply that they will be slower under InnoDB). MyISAM also supports the **`FULLTEXT`** data type, which allows very fast searches of large quantities of text data. However, MyISAM is only capable of locking an entire table for writing. This means only one process can update a table at a time. As such, any application that uses the table scales this way might prove to be a hindrance. It also lacks journaling, which makes it harder for data to be recovered after a crash.
- **InnoDB** is a more modern database engine designed to be ACID-compliant, which guarantees database transactions are processed reliably. Write locking can occur on a row-level basis within a table. That means multiple updates can occur on a single table simultaneously. Data caching is also handled in memory within the database engine, allowing caching on a more efficient row-level basis rather than file block. To meet ACID compliance, all transactions are journaled independently of the main tables. This allows for much more reliable data recovery, as data consistency can be checked.

In MySQL 5.5 InnoDB is the default engine, and is highly recommended over MyISAM unless the user has a specific need for features unique to the engine. Again, the official website holds plenty of documentation on that matter.

Chapter summary

A computer running Ubuntu Server can be a host for a DBMS (database management system) to store and process data over a network. Ubuntu Server provides two popular database systems: PostgreSQL and MySQL.

Both PostgreSQL and MySQL can be installed by using the **`sudo apt-get install`** command. The difference is the name of the package to be installed. For PostgreSQL, the package is named **`postgresql`**, and for MySQL, the name of the package is **`mysql-server`**.

In the MySQL installation process, a dialog box will be shown to provide a password for the root user. The root user is the one with all administrative privileges for managing the database server.

PostgreSQL configuration is made by editing the **/etc/postgresql/<version>/main/postgresql.conf** file for main (global) configuration parameters, and the **/etc/postgresql/<version>/main/pg_hba.conf** file for client connection parameters. In both paths, **<version>** refers to the PostgreSQL version. If the user installs the 9.4 version, the paths for the configuration files would be **/etc/postgresql/9.4/main/postgresql.conf** for the main configuration file and **/etc/postgreslq/9.4/main/pg_hba.conf** for the client configuration file.

PostgreSQL has an administrative user called **postgres**. To assign a password for that user, a connection to the **template1** database needs to be made. This connection can be done with the **sudo –u postgres psql template1** command. This command will show the PostgreSQL command prompt. Then, using the **ALTER USER postgres WITH ENCRYPTED PASSWORD '<user_password>'** command, a new password for user **postgres** will be assigned.

MySQL can be configured by editing the file **/etc/mysql/my.cnf**. For example, to configure MySQL to listen for connections from network hosts, the **bind-address** directive of the **[mysqld]** section must be set to the server's IP address.

MySQL has two ways to store data in the system, known as database engines, and they are InnoDB and MyISAM. MyISAM is the older of the two engines. It can be faster than InnoDB under certain circumstances, and favors a read-only workload. Some web applications have been tuned around MyISAM. Besides, MyISAM supports the **FULLTEXT** data type, which allows very fast searches of large quantities of text data. However, MyISAM lacks journaling capabilities, which makes it harder to recover after a system crash. On the other hand, InnoDB is designed to be ACID-compliant, which guarantees database transactions are processed reliably. Also, it handles data caching in memory within the database engine, providing greater efficiency through a row-level basis, and journals all transactions independently of the main tables. This allows for much more reliable data recovery.

Chapter 9 Desktop Experience in Ubuntu Server

Overview

So far, all aspects related to the configuration and operation of Ubuntu Server have been made from the command line. For those users coming from Windows this might seem a little bit weird, since Windows Server offers a friendly GUI for performing both configuration and operational tasks. However, Ubuntu Server can also have a desktop interface like Windows, and bringing that interface to the server is just a few package installations away.

Installing the desktop environment

The following commands will install the desktop environment in Ubuntu Server.

Code Listing 153

```
$ sudo apt-get update
$ sudo apt-get install 97buntu-desktop
```

All necessary packages for installing a desktop environment will be downloaded. These packages include some applications like the Firefox web browser, the LibreOffice suite, and others. Because of this, the process could take several minutes, depending on the Internet connection bandwidth available.

After the download is complete, Ubuntu will take care of the installation of every package downloaded. This process can also consume several minutes, depending on the computer's resources available. Unfortunately, because the process is executed from the command line, there's no progress bar indicator to tell how far along the installation is.

When the installation ends, the command prompt will appear. To run Ubuntu's desktop environment, it's necessary to reboot the system with the following command.

Code Listing 154

```
$ sudo reboot
```

Now, Ubuntu Server will take the user into the desktop environment to work.

The login screen

Like Windows, Ubuntu Server starts by showing the login screen to ask for the user's credentials. These credentials correspond to the username and password for the administrative user created during the installation process, or to the username and password of any other user added after that process.

For the purposes of this section, the login screen has been divided into four sections. These sections are shown in Figure 36.

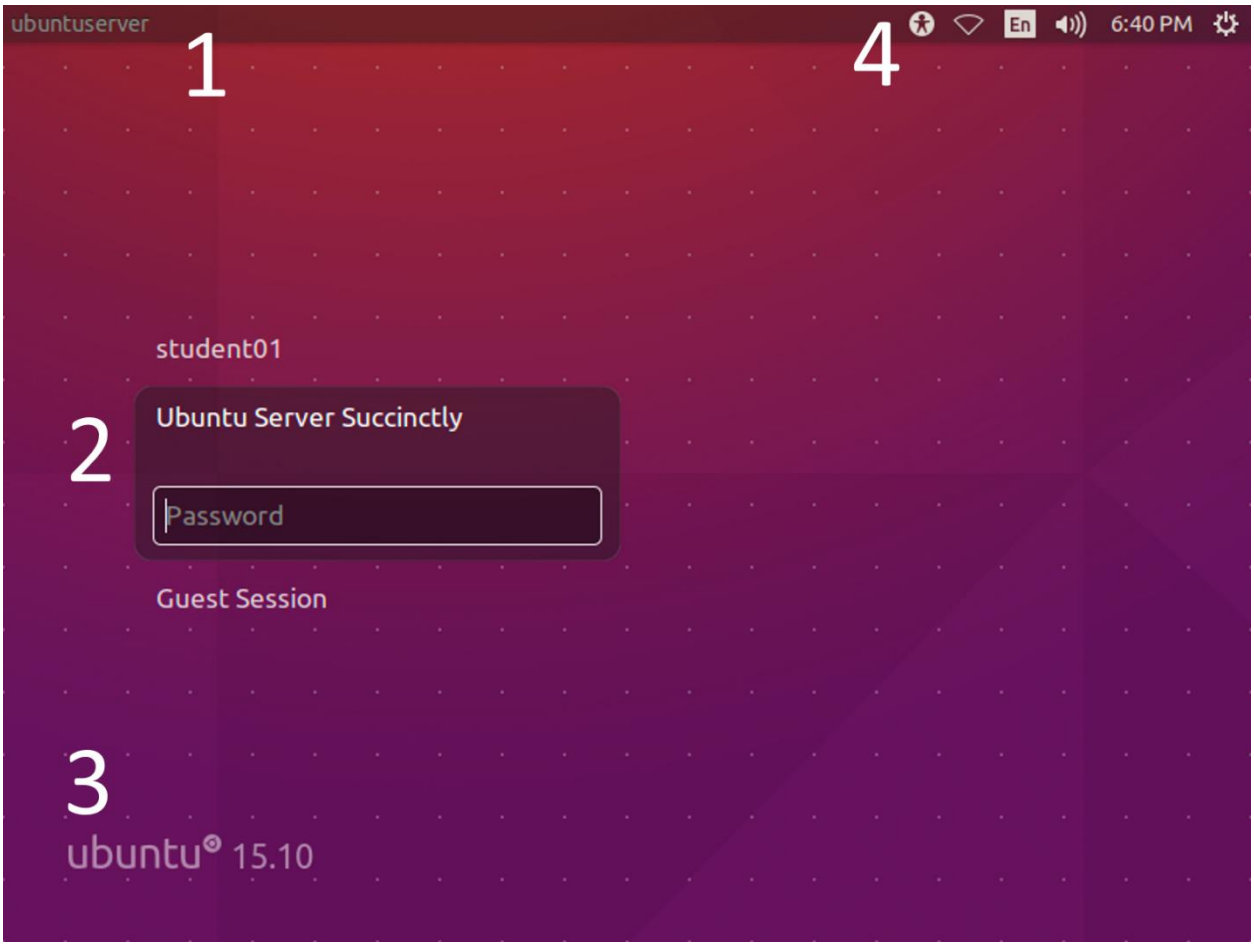

Figure 36: Login screen divided in four sections

The following table explains the sections shown in the previous figure.

Table 13: Purposes of Every Section in the Login Screen

Section	Description
1	Panel. This section shows the name assigned to the computer during the installation process.

Section	Description
2	Users list and password entry area. This area displays the name of all users registered in the system. By default, the password for the administrative user is required. Above the text entry that asks for the password, the user's full name is displayed, unless there is no full name defined. In that case, the username will be displayed. To log into the system, the password must be entered and Enter must be pressed.
3	This section displays the Ubuntu software's version.
4	Panel indicators. This area shows a set of six graphical element, intended to perform a series of common tasks that have no administrative clearance.

The panel indicators

These indicators are shown at the screen's upper-right hand corner. They are six graphical elements or icons that each have a specific function. They are shown in the following figure.

Figure 37: The Panel Indicators

The function for each icon is explained in the following sections.

Accessibility options menu

This menu is shown by clicking on the first icon in the common tasks bar, and contains the available options to provide handicapped-user-friendly computing work. The options appear in the following figure.

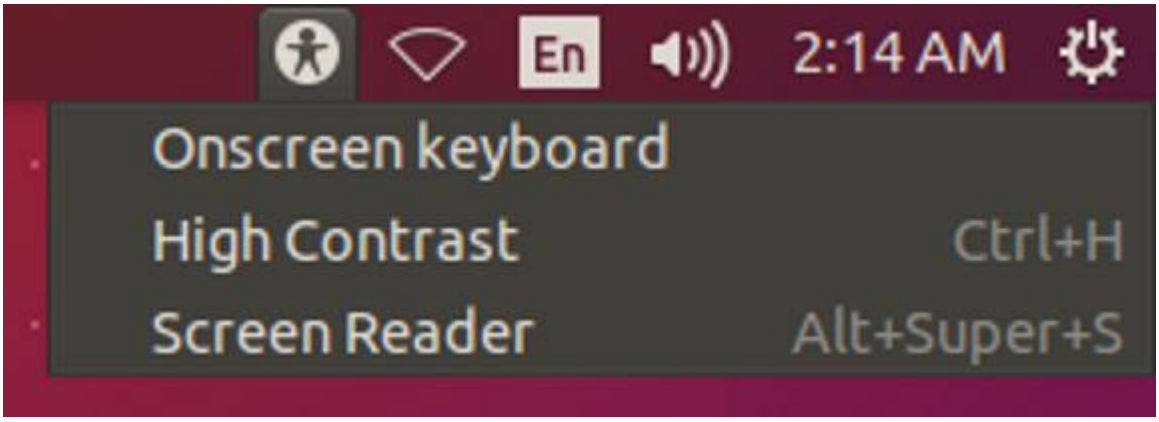

Figure 38: Accessibility Options in the Login Screen

The name of each option in the menu gives a good idea of its purpose. *Onscreen keyboard* will bring up a graphic keyboard on the screen, generally intended to be used in a touch screen device. The *High Contrast* option enables the desktop interface to be used easily for visually impaired people. The *Screen Reader* enables voice-guided help for vision-impaired and blind users. When one of these options is enabled, a check mark will appear to the left of its description. If no check mark is displayed, then the option is disabled.

Some options display a description for a combination of keys to the right of the menu. In this case, if the user presses Ctrl+H, the High Contrast option will be enabled or disabled. For the Screen Reader option, pressing Alt, the Super (Windows logo) key, S will enable or disable it.

Network connections menu

The second icon in the common tasks bar shows the network adapters available for making connections. At this point, the user only can connect to the network using any of the adapters shown in the menu, or get information about connections.

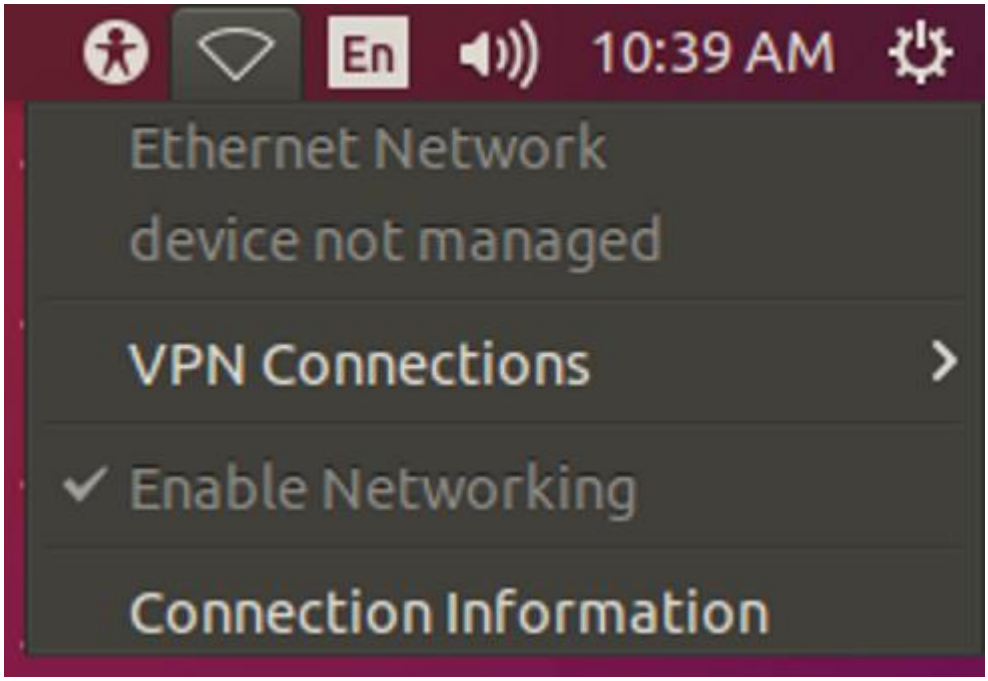

Figure 39: The Network Connections Menu

Language options menu

This menu appears when the user clicks the icon with the language initials (En for English), and shows the languages available to establish the keyboard layout to be used by the system. A keyboard layout can be selected by clicking on its name. A white dot appears to the left of the name corresponding to the keyboard layout currently used.

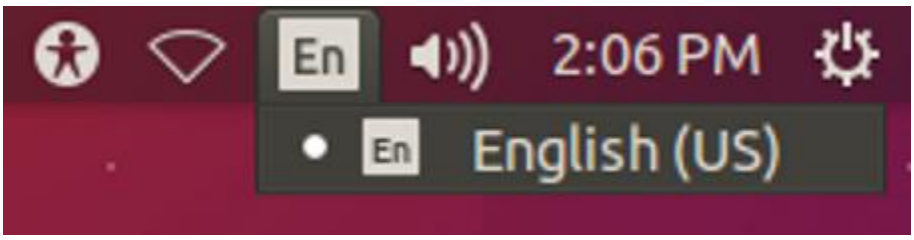

Figure 40: Keyboard Layouts Available

Sound configuration control

This menu is shown when the user clicks on the speaker icon. It is used to adjust or mute the machine's speakers.

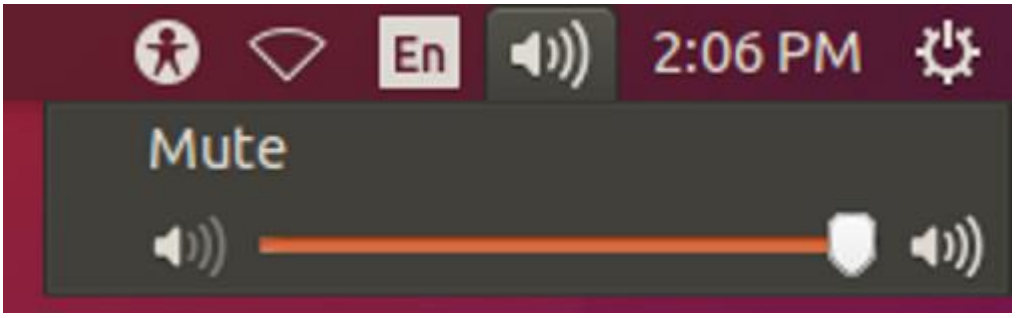

Figure 41: Sound Configuration Control

Interactive calendar

This calendar appears when the user clicks the clock in the panel indicators bar.

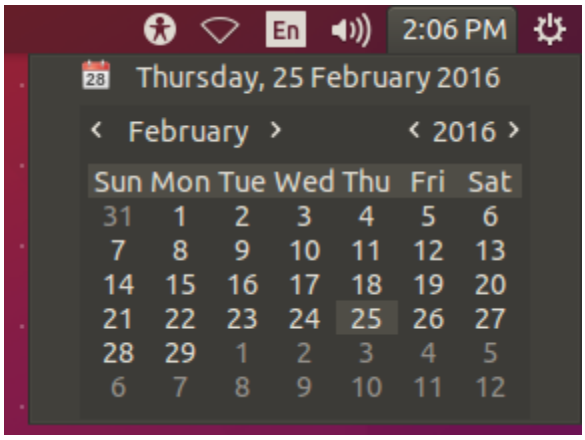

Figure 42: Interactive Calendar

The previous figure shows the interactive calendar. By default, the current date appears at the top. The day corresponding to this date is highlighted. The user can navigate through months and years by clicking the left and right arrows placed next to the month name and the year number.

Settings menu

This menu appears by clicking the gear placed at the end of the panel indicators bar.

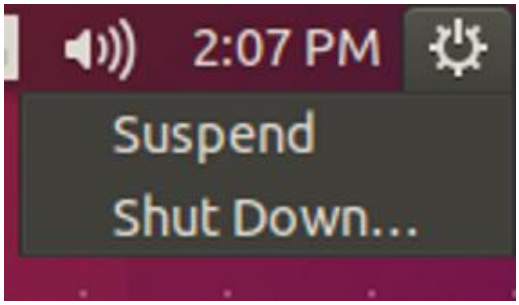

Figure 43: Settings Menu

Since the menu is called from the Login screen, the user is allowed to suspend, reboot, or shut down the computer.

The Shut Down dialog

This dialog appears when the user selects the **Shut Down** option from the **Settings** menu.

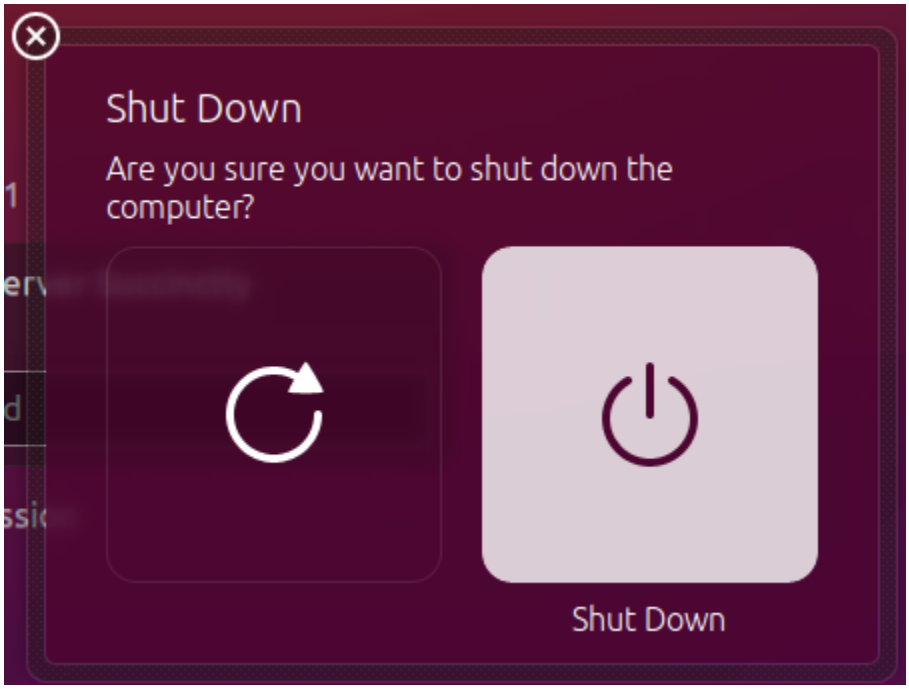

Figure 44: The Shut Down dialog

There are two huge icons in the **Shut Down** dialog. The icon on the right side of the dialog allows the user to power off (shut down) the computer by clicking on it. The icon on the left is used to reboot the computer.

The desktop interface

Once the user is logged in, the system launches the Ubuntu Desktop. This is the starting point for using the desktop interface.

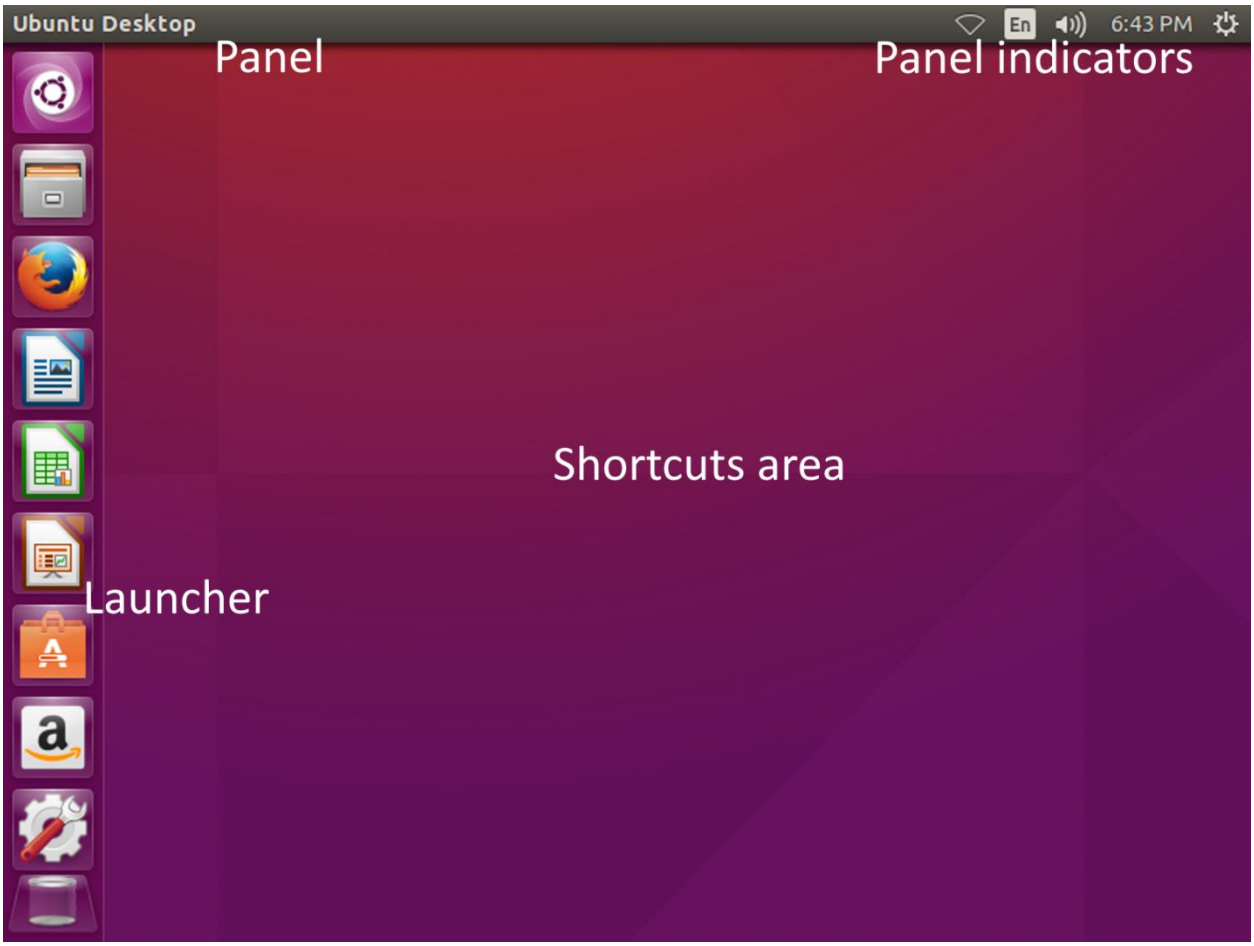

Figure 45: The Ubuntu Desktop Interface

Figure 45 shows the Ubuntu Desktop and labels its main features. At the top, the panel shows the menus of every active application. It also contains a set of iconic indicators which can be used as shortcuts for certain tasks, such as configuring date and time, configuring sound, connecting to a network, or changing global system settings. At the left, a task bar, known as the Launcher, holds icons that work as shortcuts to execute certain applications, such as the Firefox web browser, the file explorer, the LibreOffice suite, and the Home button. Finally, the shortcuts area allows the user to place shortcuts to files or applications in an iconic form for quick access.

Dialogs (or windows): The main element in the desktop environment

In the desktop environment, there are no commands to work with the system; graphical elements replace the commands. To show these elements to the user, the desktop environment needs a place for doing such things. This place is known as a dialog (or window).

A dialog has a series of elements where each one has a particular function. These elements can be used by clicking on them with the pointer. The following figure shows an example of a dialog.

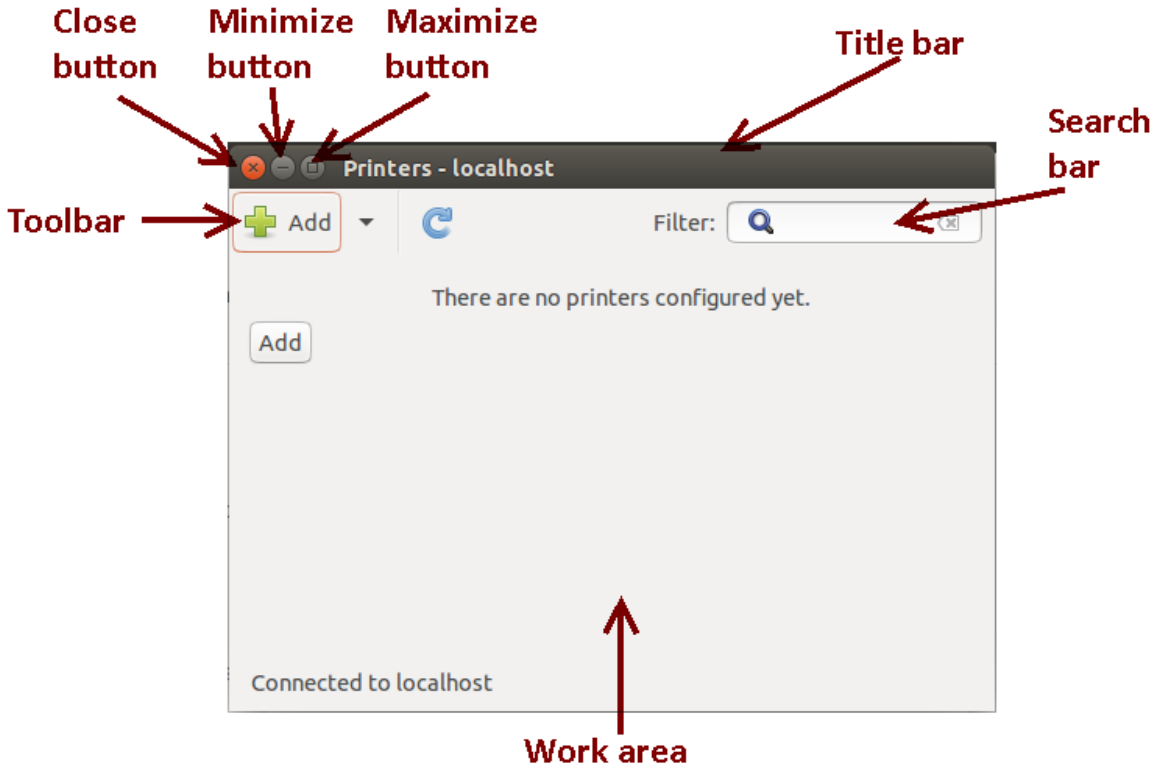

Figure 46: Dialog (Window) Structure

The function for each element shown in the following figure is detailed in the following list.

- **Close button**: Closes the dialog.
- **Minimize button**: Hides the dialog and places it as an icon in the desktop's Launcher.
- **Maximize button**: Expands the dialog to fit the entire desktop, excluding the areas covered by the panel and the Launcher.
- **Title bar**: Shows the name of the application, command, or location that is using the dialog.
- **Search bar**: Allows the user to search objects (files, directories, devices, etc.) by name.
- **Toolbar**: Serves as a container of certain commonly used commands, displayed as buttons.
- **Work area**: Serves as a container of other graphic elements (buttons, text entries, labels, etc.) used to work with the system.

Matching text commands with the desktop environment

Until now, the Ubuntu Server command line was used in all the examples in this book that involved the execution of any class of command. Now, in the following sections, some of those

commands that were explained will be matched with their corresponding actions in the desktop environment. Also, some important system settings will be explained.

Showing the contents of the user's home directory

The command **`ls /home/<username>`** was previously used to show the contents of the user's home directory. Doing this in the desktop environment is easier—it just takes a click on the **Files** button, located in the **Launcher**.

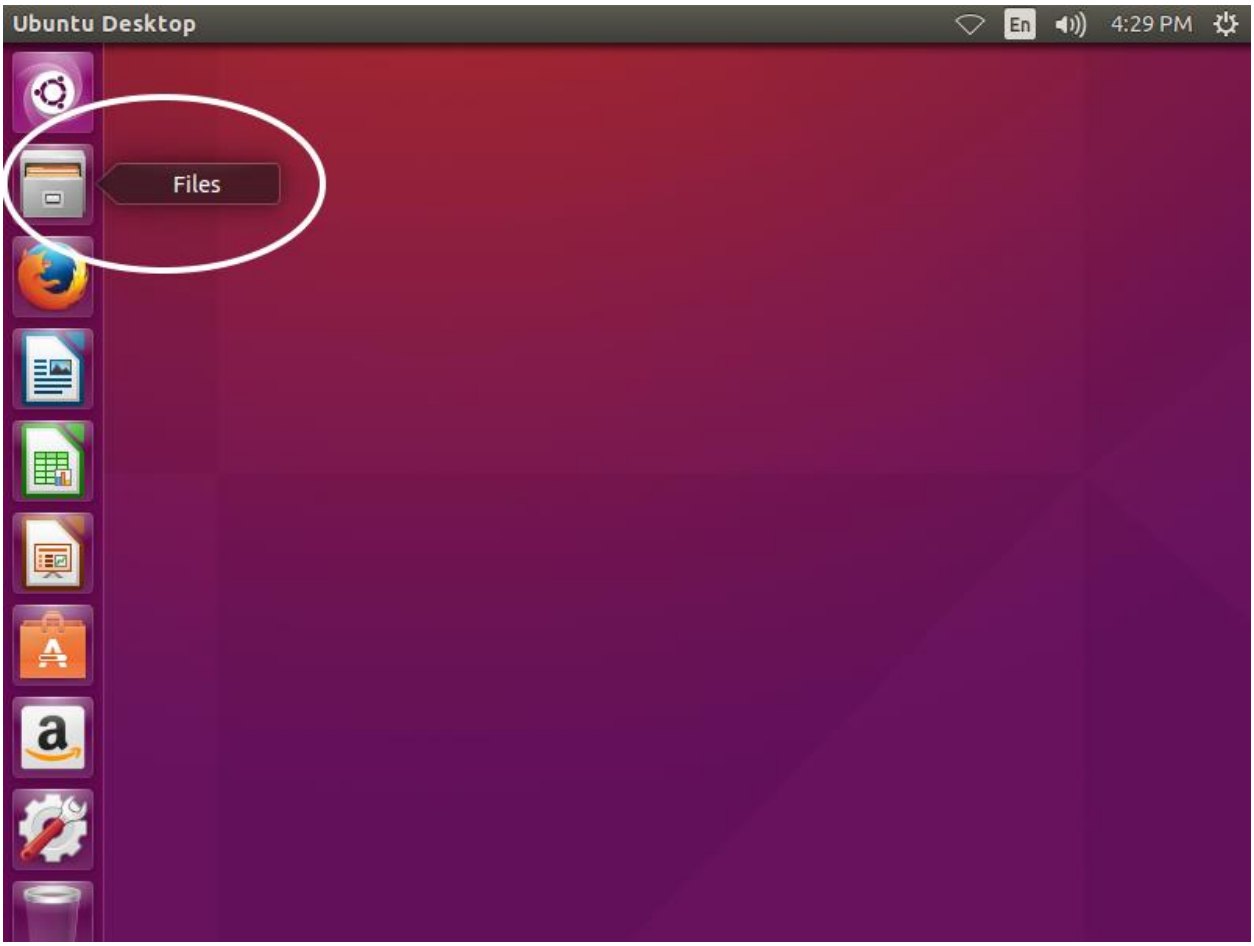

Figure 47: The File Explorer Button in the Launcher

When the user clicks on that button, the file explorer dialog will be shown, and the contents of the user's home directory will be displayed.

The file explorer dialog

The file explorer dialog is like a typical Windows dialog; it has a title bar and a content display area. It also has the commonly used **Close**, **Minimize**, and **Maximize** buttons.

The following figure shows the structure of the file explorer dialog with its elements fully identified.

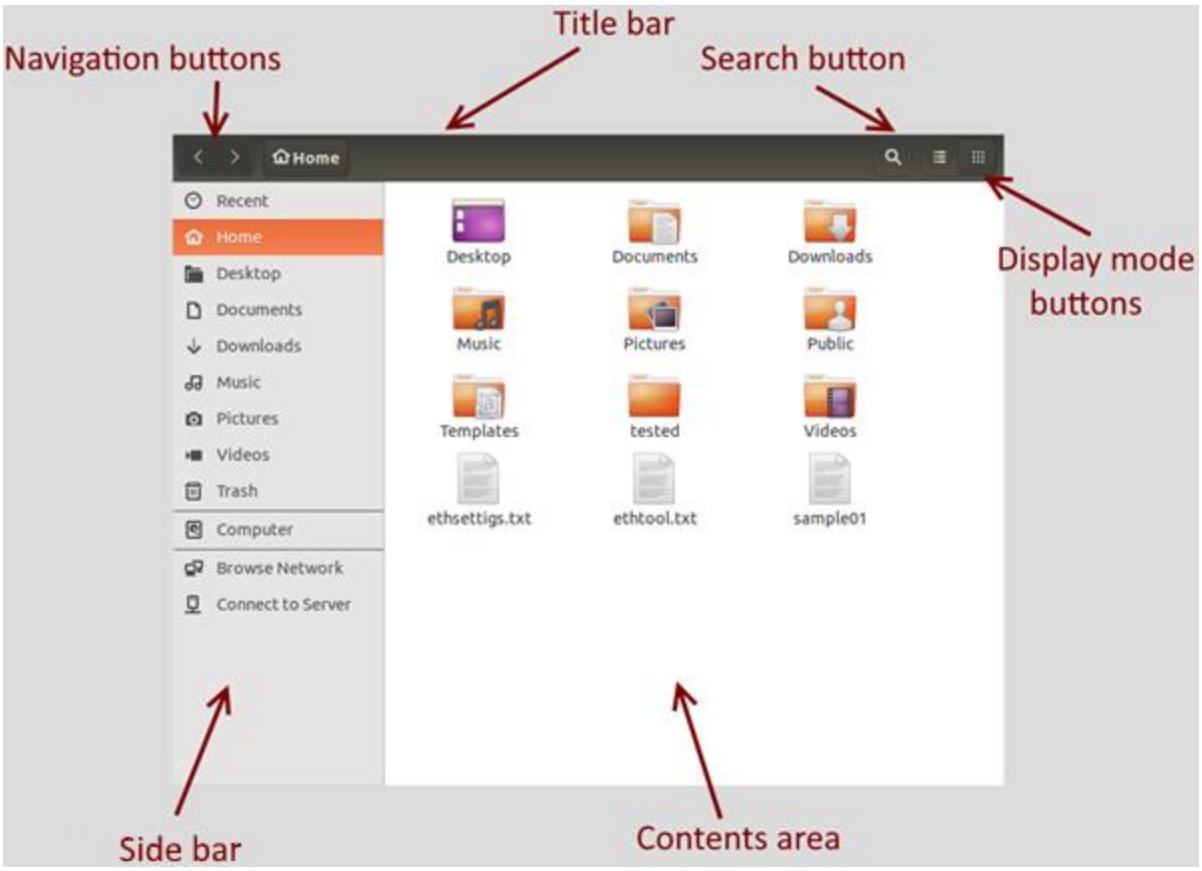

Figure 48: The File Explorer Dialog

- **Title bar**: Shows the folder's name in which the user is located. Home is displayed for the user's home directory.
- **Navigation buttons**: Allows the user to navigate back and forward in a nested directory.
- **Side bar**: Allows the user to go directly to one of the bookmarked filesystem locations, such as the Documents directory or the Downloads directory (used by the web browser to store files downloaded from the Internet).
- **Search button**: Activates the search bar to look for files or folders in the current directory.
- **Display mode buttons**: Shows the contents of the directory either as an iconic grid (the default) or as a list.
- **Contents area**: Shows the contents of the directory.

The search bar

This feature allows you to search files or directories by name within the current directory. Also, a series of criteria based on the file type can be added to refine the search.

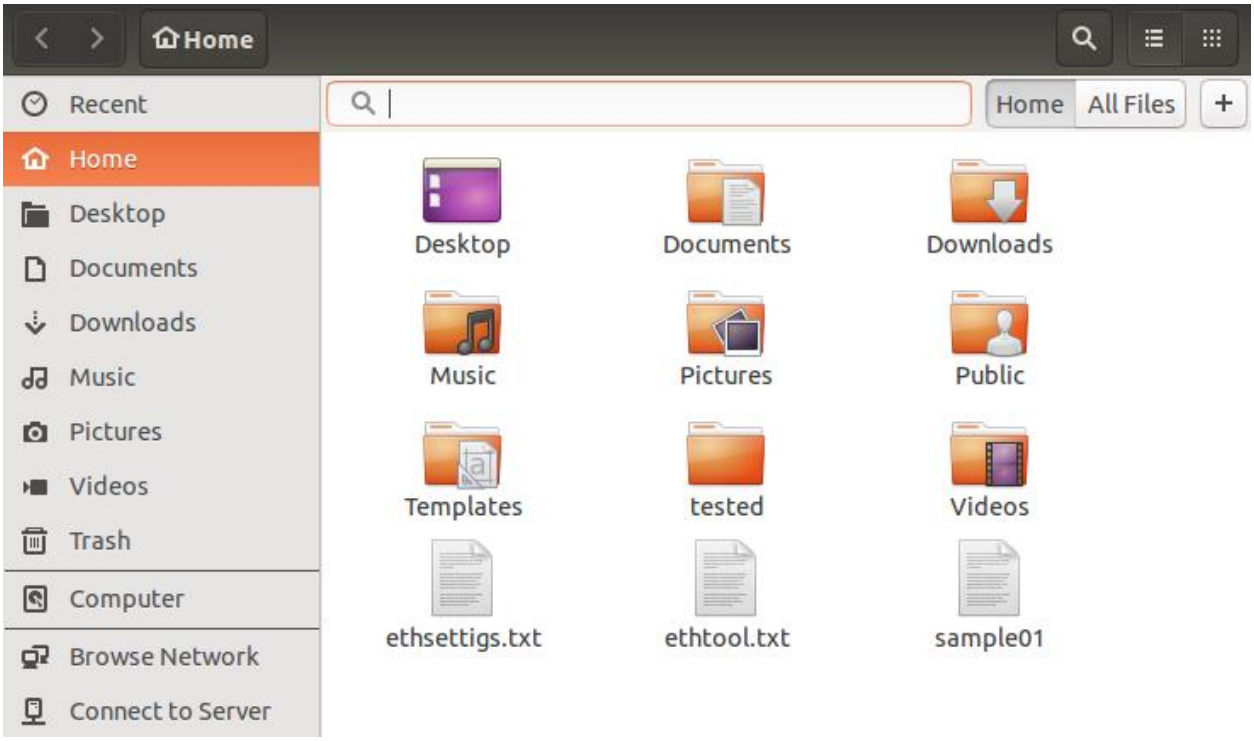

Figure 49: The File Explorer Dialog with Search Bar Activated

The file explorer dialog with an active search bar is shown in the previous figure. In order to search files, the user types the name to search for, and the file explorer will start the search as the filename changes in the text entry. The plus sign (+) at the end of the bar allows you to add a search criterion, based on the file type.

The default file type for searching is called *Any*, which implies that all files will be considered for searching, no matter what type of data is stored in them. Then, a series of common types are suggested to refine the search operation. These types are Documents, Music, Video, Picture, Illustration, Spreadsheet, Presentation, Pdf/Postscript, and Text File. The type names suggest the kind of data that must be stored for each file type to be considered for searching.

File type criteria are not limited to the types displayed. At the end of the file type list, the **Other Type** option is shown. When the user selects this option, a dialog box appears showing a list of different file types, based on the applications that are installed in the desktop environment. The user can select one of these types to include it in the search criteria.

When more than one criterion is specified, the search operation applies an **or** logical operation. For example, if Documents and Video are considered, searching is done by including all files whose names correspond with the text entry supplied, and for one or all of the file types indicated in the criteria.

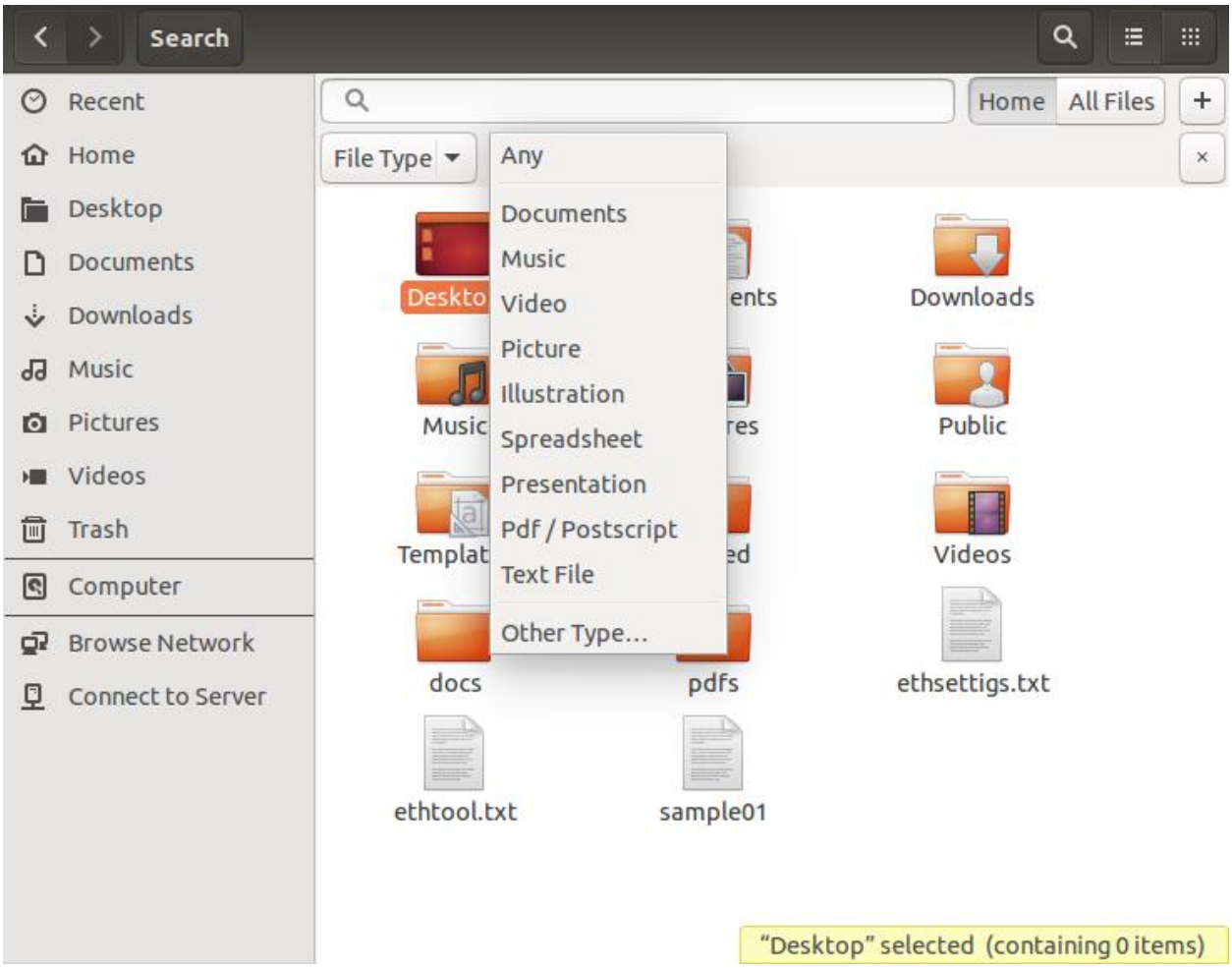

Figure 50: The Search Bar Showing the File Types List

If the user wanted to search for files that contain *eth* in their name, considering only text files, this search would look like the following figure.

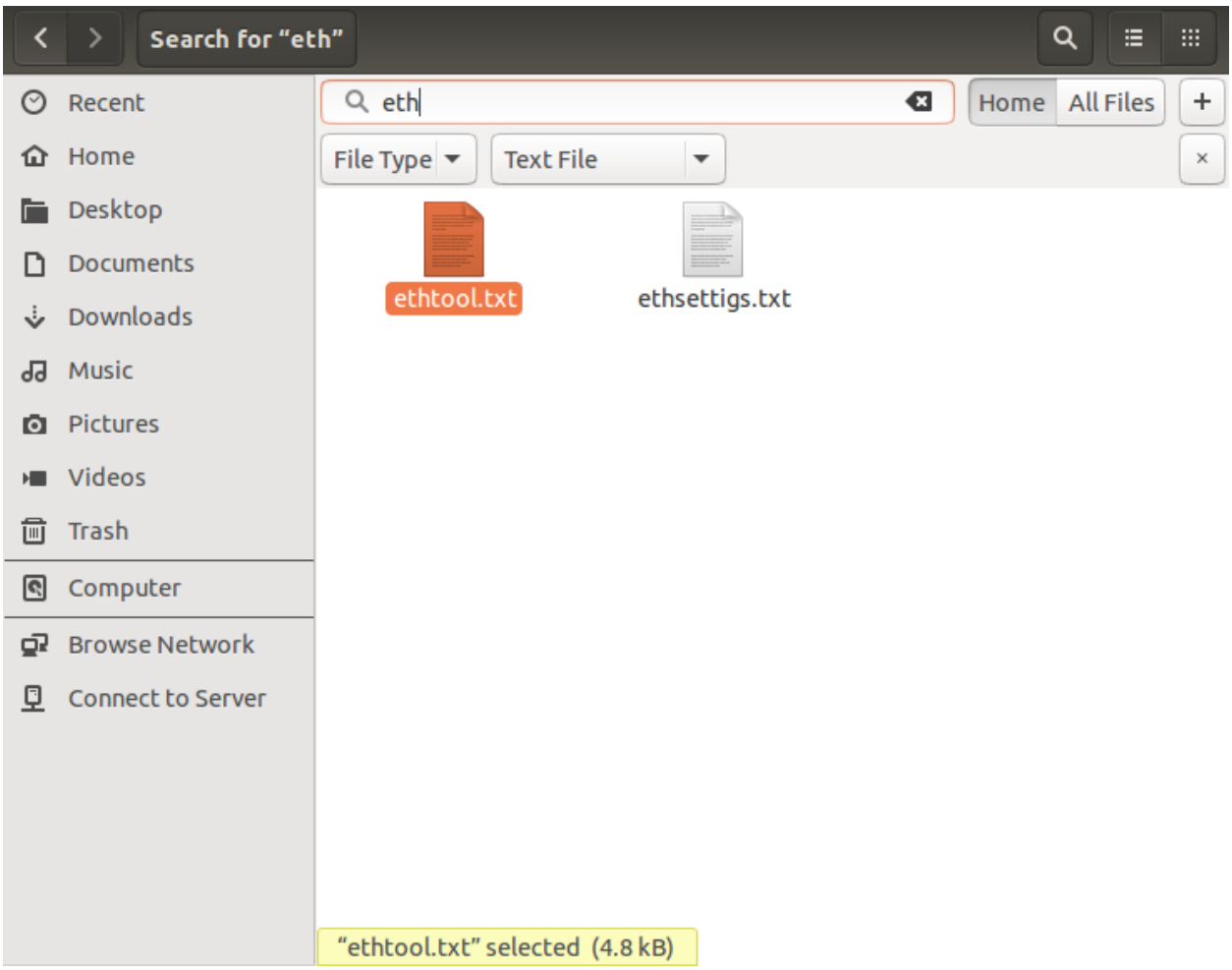

Figure 51: The File Explorer Dialog with Search Results

The menu bar for the file explorer dialog

The file explorer has several commands grouped into sections called menus. These menus are placed in a graphical element called the menu bar. This bar can be displayed by placing the pointer over the desktop’s panel. The following figure shows the File Explorer’s menu bar.

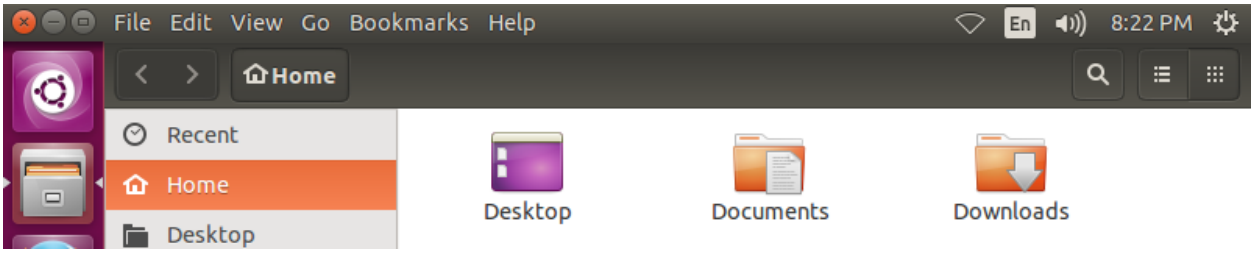

Figure 52: The File Explorer Menu Bar

There are six groups of commands or menus, named **File**, **Edit**, **View**, **Go**, **Bookmarks**, and **Help**. The classic **Close**, **Minimize**, and **Maximize** buttons also appear at the beginning of the

bar. To execute any of these commands, the user must click on the desired menu and then click on the command to execute it.

The File menu

The commands located in this menu correspond to certain tasks about file management or navigation. The menu can grow and shrink, depending on the file type of the currently selected items, or if there is no item selected in the content area. The commands associated with the File menu are as follows:

- **New Window**: Opens a new file explorer dialog. This command can also be executed by pressing Ctrl+N.
- **New Tab**: Creates a new tab in the content area. The user can navigate in both tabs to different locations in the filesystem. This command can also be executed by pressing Ctrl+T.
- **New Folder**: Creates a new directory in the current location. This command can also be executed by pressing Shift+Ctrl+N.
- **New Document:** Creates an empty document file in the current location.
- **Open**: Shows the content of the currently selected directory in the content area. This command can also be executed by pressing Ctrl+O.
- **Open in New Tab:** Shows the content of the currently selected directory, creating a new tab in the content area. This command can also be executed by pressing Shift+Ctrl+T.
- **Open in New Window**: Shows the content of the currently selected directory by opening a new file explorer dialog box. This command can also be executed by pressing Shift+Ctrl+O.
- **Open With**: Shows a set of commands that correspond to all possible applications capable of opening the directory or file selected.
- **Connect to Server**: Allows the user to connect to another server in the network.
- **Properties**: Shows file or folder permissions and other properties related to the current selected items.
- **Close**: Closes the file explorer dialog box.
- **Close All Windows**: Closes all file explorer dialogs opened.

As previously mentioned, the commands shown in the File menu depend on the selection of an item or items in the content area. If no item is selected, the Open commands disappear, except for the Open With command.

The following figures show the File menu in three different scenarios: When no item is selected, when a directory is selected, and when a file is selected. Notice how the menu changes according to the situation.

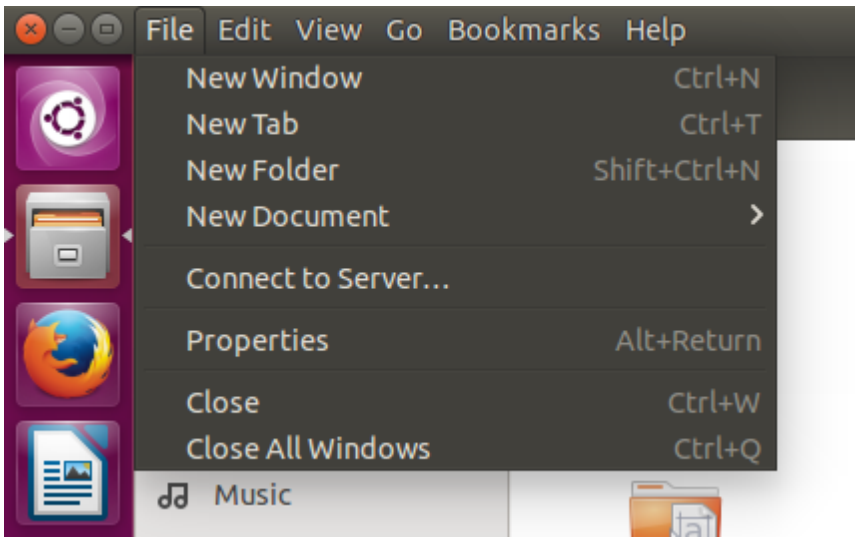

Figure 53: File Menu Commands When No Item is Selected

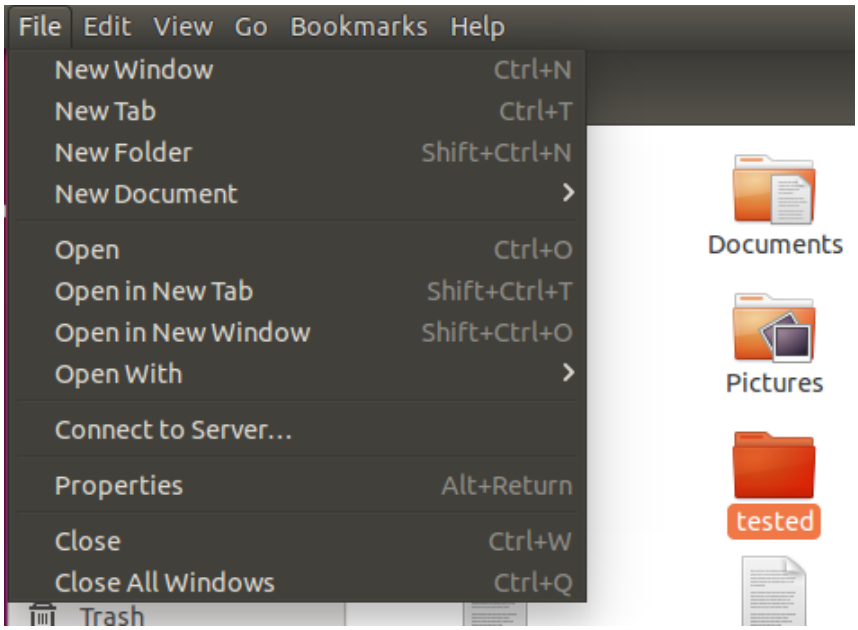

Figure 54: File Menu Commands When a Folder is Selected

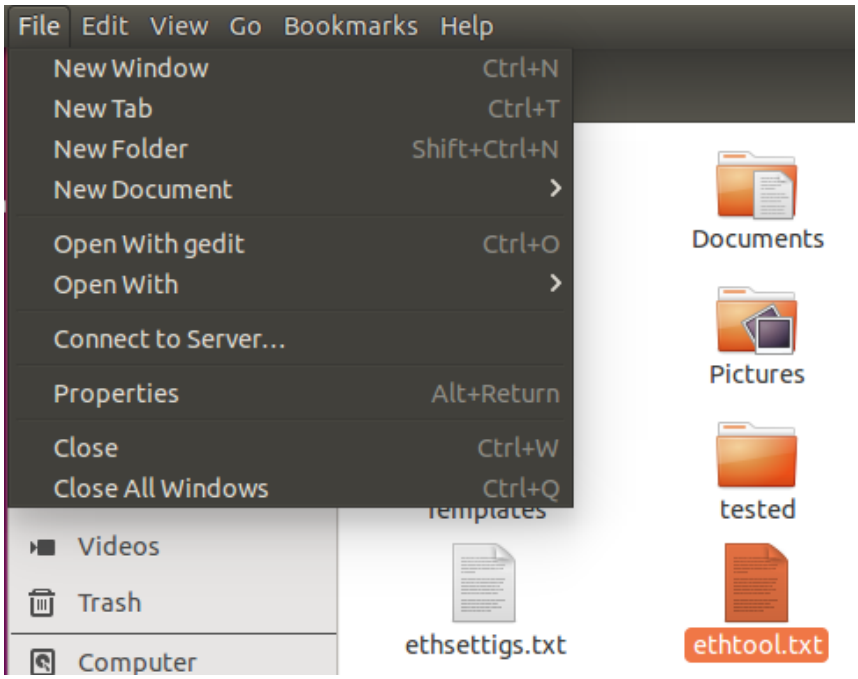

Figure 55: File Menu Commands When a File is Selected

The Edit menu

The Edit menu groups all commands related to selecting files or folders, moving files, deleting files, renaming files, and creating shortcuts. Also, the behavior of the file explorer dialog can be changed. The commands associated with the Edit menu are:

- **Undo**: Reverts the actions performed by the last command executed. Another way to execute the command is by pressing Ctrl+Z.
- **Redo**: Repeats execution of the last command. Another way to execute the command is by pressing Shift+Ctrl+Z.
- **Cut**: Marks the file or folder selected to be moved to another location. Another way to execute the command is by pressing Ctrl+X.
- **Copy**: Sends a copy of the file or folder selected to the clipboard. Another way to execute the command is by pressing Ctrl+C.
- **Paste**: Places the copy of the file or folder that's on the clipboard to the current location. Another way to execute the command is by pressing Ctrl+V.
- **Select All**: Selects all items in the current location. Another way to execute the command is by pressing Ctrl+A.
- **Select Items Matching**: Selects all items that comply with a criterion. Another way to execute the command is by pressing Ctrl+S.
- **Invert Selection**: Selects all items opposite the current selection. That is, if no items are selected in the current location, all items will be selected by this command. Another way to execute the command is by pressing Shift+Ctrl+I.
- **Make Links**: Creates shortcuts to access the current file or directory selected. Another way to execute the command is by pressing Ctrl+M.
- **Rename**: Allows the current item selected to be renamed. Another way to execute the command is by pressing the F2 key.

- **Move to Trash**: Moves the item currently selected to the trash. This item will no longer be displayed. Another way to execute the command is by pressing the Delete key.
- **Preferences**: Allows the behavior of the file explorer dialog to be changed.

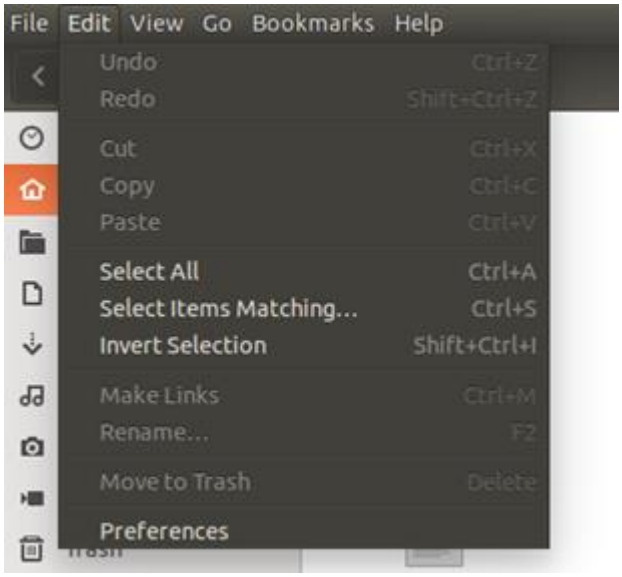

Figure 56: Edit Menu Commands

The View menu

This group of commands allows you to change the way items are displayed in the file explorer dialog box. The commands associated with the View menu are:

- **Reload**: Refreshes the content of the current location. Pressing Ctrl+R executes the command, too.
- **List**: Shows the content of the current location as a list, which includes details for each item. Pressing Ctrl+1 executes the command, too.
- **Icons**: Shows the content of the current location as an iconic view (the default). Pressing Ctrl+2 executes the command, too.
- **By Name**: Sorts the content of the current location by file or directory name.
- **By Size**: Sorts the content of the current location by file or directory size.
- **By Type**: Sorts the content of the current location by file type.
- **By Modification Date**: Sorts the content of the current location by the date of last modification.
- **Reversed Order**: In an iconic view, places the icons in a reversed order according to the default view, which places directories before files.
- **Reset View to Defaults**: Restores all view parameters to their default values.
- **Show Hidden Files**: Shows all files stamped with the hidden attribute. Pressing Ctrl+H executes the command, too.
- **Show Sidebar**: Shows and hides the side bar (where the bookmarks are displayed). Pressing F9 executes the command, too.
- **Zoom In**: Magnifies the content of the file explorer. Pressing Ctrl+Plus Sign executes the command, too.

- **Zoom Out**: Reduces the content of the File Explorer. Pressing Ctrl+Hyphen executes the command, too.
- **Normal Size**: Shows the content of the File Explorer at its normal scale (100%). Pressing Ctrl+0 executes the command, too.

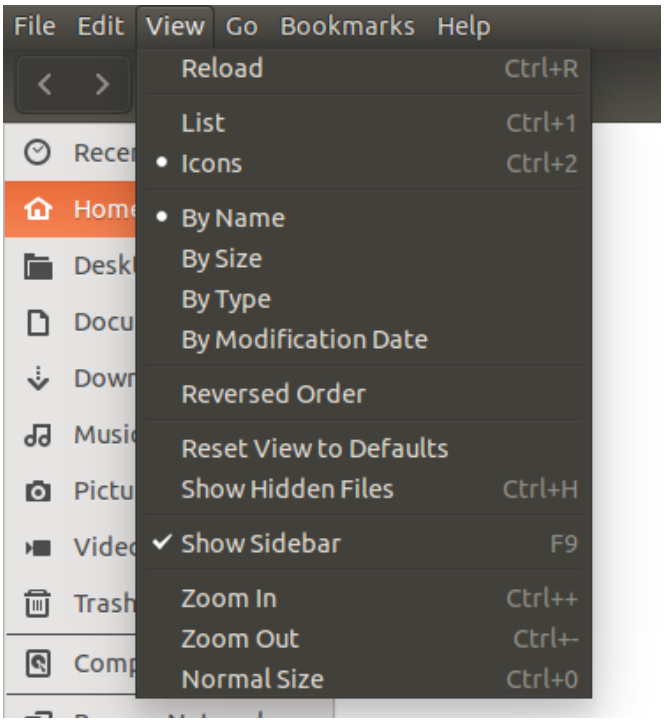

Figure 57: View Menu Commands

The Go menu

The Go menu commands allow operations that involve moving to another location in the filesystem. The commands associated with the Go menu are:

- **Open Parent**: Opens the directory that is an ancestor of the current directory, according to the filesystem hierarchy. Pressing Ctrl+Up Arrow executes the command, too.
- **Back**: Goes to the directory the File Explorer was in before viewing the current directory. Pressing Ctrl+Left Arrow executes the command, too.
- **Forward**: Goes to the directory that was been viewed, prior to the execution of a Back command. Pressing Ctrl+Right Arrow executes the command, too.
- **Enter Location**: Shows a pop-up dialog box to specify a location (absolute or relative) to go to directly.
- **Search for Files**: Activates the search bar of the file explorer dialog.

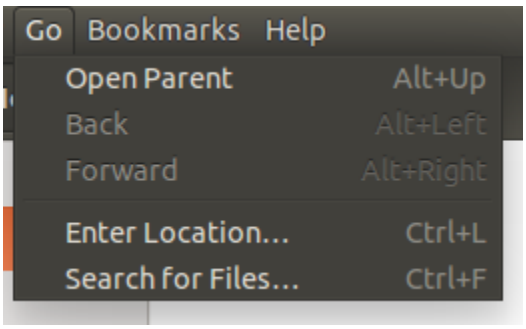

Figure 58: Go Menu Commands

The Bookmarks menu

The Bookmarks menu commands allow you to create shortcuts that point to specific locations in the filesystem in order to make them available quicker. The commands associated with the Bookmarks menu are:

- **Bookmark this Location**: Creates a shortcut for the current directory. If the current directory already has a shortcut defined, this command is disabled. Pressing Ctrl+D executes the command, too.
- **Bookmarks**: Shows the list of bookmarks available. Pressing Ctrl+B executes the command, too.

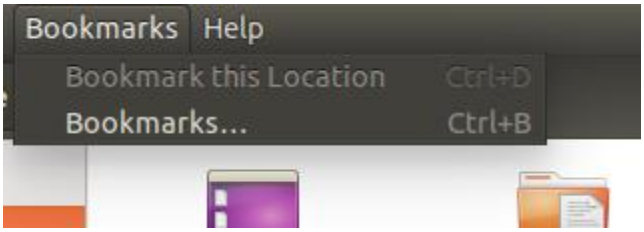

Figure 59: Bookmarks Menu Commands

Note: Bookmarks appear in the sidebar of the file explorer dialog.

The Help menu

This group of commands allows you to browse help documentation related to the file explorer. The commands for this menu are:

- **All Topics**: Shows all the topics covered by the file explorer help documentation. Pressing the F1 key executes the command, too.
- **Search for files**: Shows documentation about searching files and directories.
- **Sort files and folders**: Shows documentation about sorting the content of the file explorer dialog.
- **Find a lost file**: Shows documentation about the way a user can find a file in the filesystem.

- **Share and transfer files**: Shows documentation about file sharing and file transferring in the network.
- **About**: Show credits about the Files (file explorer) application.

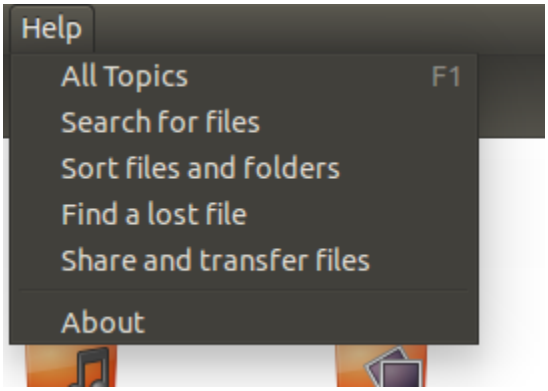

Figure 60: Help Menu Commands

Executing the shutdown and reboot commands

The **`shutdown`** and **`reboot`** commands explained previously can be executed in the desktop environment by selecting the **Shut Down** option located in the **Settings** menu that is shown by clicking the gear icon located at the end of the desktop panel.

The Shut Down command shows the dialog explained in *The shut down dialog* section earlier in this chapter.

The Settings menu in the desktop interface

Unlike in the login screen, the **Settings** menu shown in the desktop interface has more options (commands) available. Some of these options perform actions only available for system administrators. So, when the user attempts to execute one of these options, a dialog will appear asking for a password to check the authentication level for that user.

The options available in the **Settings** menu are:

- **About This Computer**: Shows detailed information about the computer on which Ubuntu Server is installed.
- **Ubuntu Help**: Shows a dialog with general help about the desktop environment.
- **System Settings**: Shows a dialog for customizing the system.
- **Lock/Switch Account**: Locks the computer and shows the lock screen.
- **Log Out**: Closes the current user's session and returns to the login screen.
- **Suspend**: Puts the computer into a hibernate state. When the user "wakes up" the computer, the Lock screen is displayed, asking for a password.
- **Shut Down**: As explained in the previous section, this allows the user to execute the shut down or reboot commands.

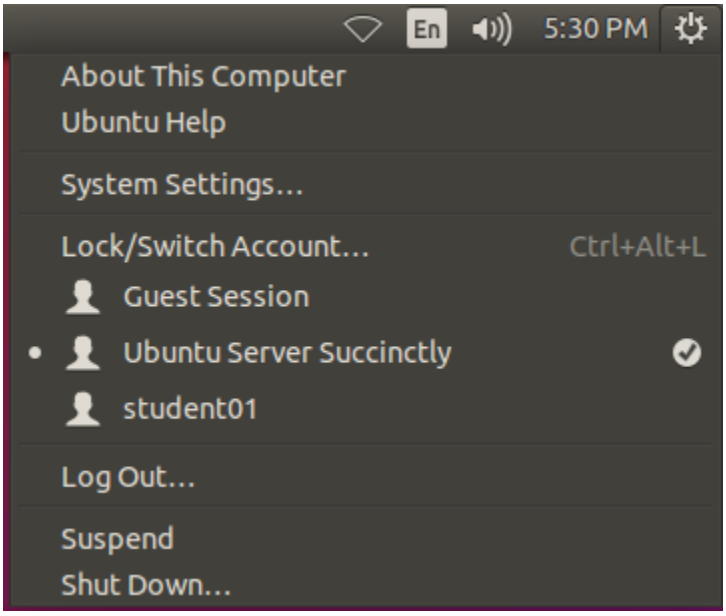

Figure 61: The Settings Menu

Besides the commands previously mentioned, the **Settings** menu shows a section with all registered user accounts. A checkmark is displayed to the right of the name of the current logged-in user.

The lock screen

Locking the computer when it is unattended is an important security issue, and the desktop environment of Ubuntu takes this into account. The lock screen is in charge of this task. The user only needs to press Ctrl+Alt+L to bring up this screen. Another way to lock the screen is by selecting the **Lock/Switch Account** item in the **Settings** menu, as explained in the previous section.

The lock screen looks almost like the login screen, except that only the name of the user logged in is displayed. The user must type their password to get back into the desktop interface.

The following figure shows the lock screen.

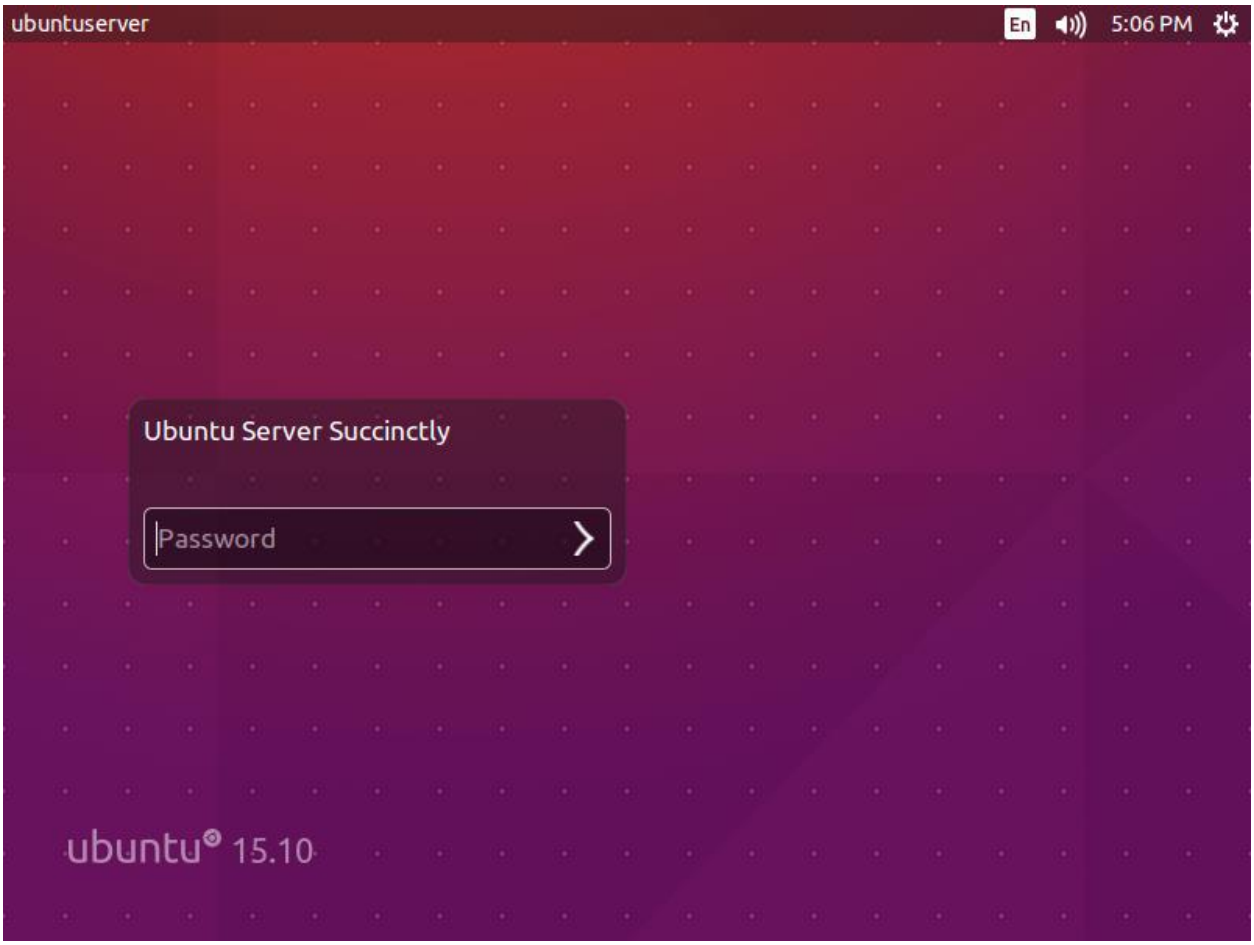

Figure 62: The Lock Screen

Customizing the system

The user can customize the system in many ways by selecting the **System Settings** option in the **Settings** menu. This customization includes the appearance of the interface, adding or removing hardware, and system-level changes like user account management or updating the software.

Many tasks explained in previous chapters, such as networking and security, can be managed by accessing the **System Settings** dialog. The dialog is divided into three sections:

- **Personal**: Allows you to change settings related to the interface appearance, text entry device usage, and privacy parameters.
- **Hardware**: Allows you to configure currently installed hardware and install new devices in order to attach them to the computer.
- **System**: Allows you to configure global system settings like software updates, time and date, and user accounts.

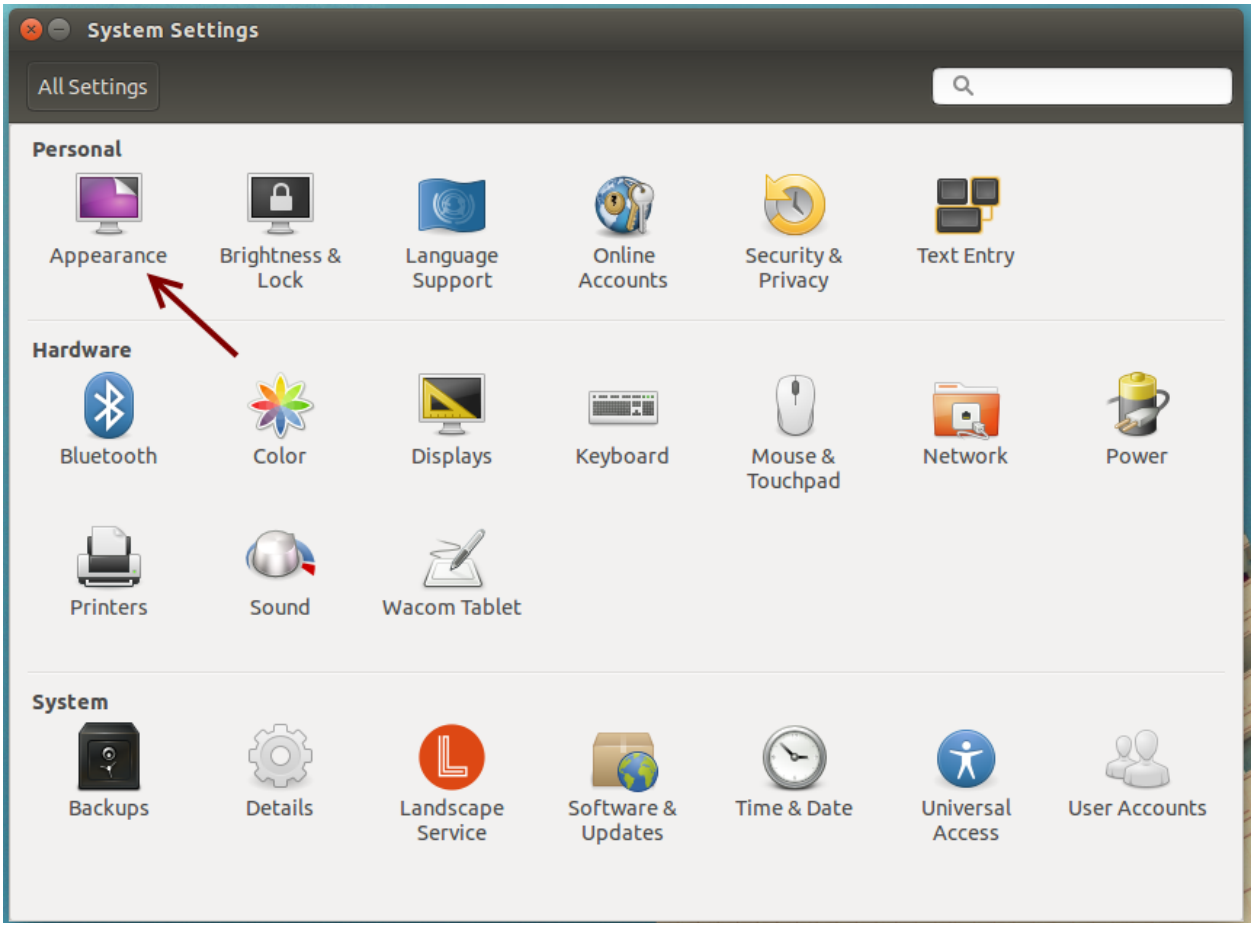

Figure 63: The System Settings Dialog with the Appearance Icon Highlighted

The Appearance dialog

The **Appearance** dialog can be shown by clicking on the **Appearance** icon, located in the **Personal** section of the **System Settings** dialog.

The **Appearance** dialog contains two sections:

- **Look**: Allows you to change the desktop's background and the size of the Launcher's icons.
- **Behavior**: Allows you to modify the behavior of the Launcher and the menus for dialogs (windows).

Changing the desktop background

The desktop background can be modified to show any desired image in the **Look** section of the **Appearance** dialog.

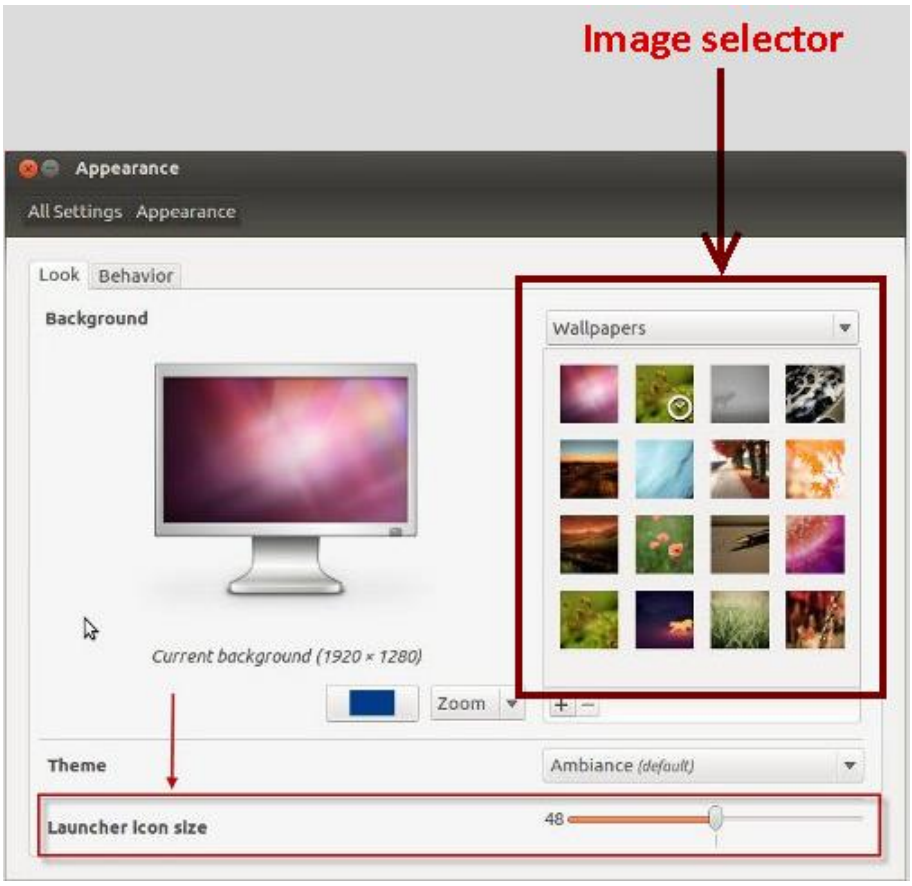

Figure 64: The Appearance Dialog Showing the Look Section

To change the background image for the desktop, the user needs to select a picture from the image selector shown in the previous figure. Ubuntu immediately replaces the previous image with the one selected. Also, the size of the icons for the Launcher can be changed with the slider control located at the bottom of the dialog. The size of icons is measured in pixels. The minimum size accepted is 16 pixels and the maximum size for the Launcher icons is 64 pixels. The default size is 48 pixels. Icon size changes take place immediately.

Changing the behavior of the Launcher and where menus are shown

In some cases, working with applications may be more comfortable if the screen area covered by the Launcher is available. A good idea for dealing with this issue is to tell the desktop environment that the Launcher must be hidden every time a dialog becomes active. The Behavior section of the Appearance dialog is used to do this.

Regarding application menus, the *File explorer dialog* section of this chapter explained that menus are displayed by placing the pointer over the desktop panel. This is true not only for the file explorer application, but for all applications executed in the desktop environment. This can also be changed in the Behavior section of the Appearance dialog.

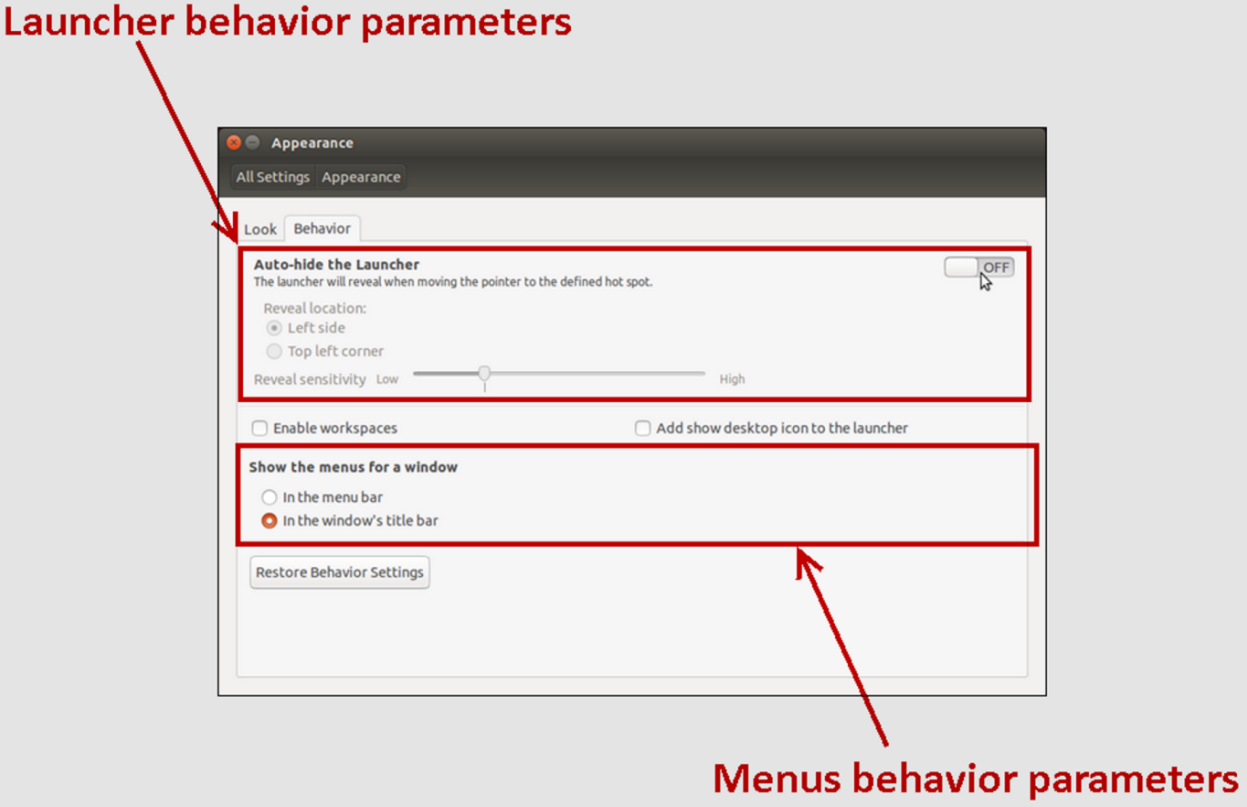

Figure 65: The Behavior Section of the Appearance Dialog

The previous figure highlights the areas for both the Launcher parameters and the menu parameters.

Dragging the button over the **OFF** label located at the right of the Launcher parameters area makes the desktop environment hide the Launcher immediately. The user can indicate where on the screen the pointer must be placed to reveal the Launcher when it's needed. There are two places available: the left side, or the top-left corner of the screen. The sensitivity for the reveal process can also be adjusted. The lower the sensitivity adjustment, the more difficult (time consuming) the reveal process will be.

The menu behavior can be changed by selecting one of two options available in the dialog. The first one tells the desktop environment that all menus will be placed in the panel, which is the default behavior. The second one forces all menus to be placed in their own application dialog.

Controlling screen brightness and automatic computer lock

Nowadays, taking care of the natural environment has become one of the more important issues in many industries. Computers and software are not exceptions. So, to save power (and make the battery life cycle longer), the desktop environment allows you to adjust the brightness for the screen if a notebook computer is being used, and can turn the screen off for any kind of computer when there's no activity in the system. Also, the lock screen can be configured to run automatically, either when the screen turns off, or when a certain amount of time has elapsed. The **Brightness & Lock** item in the **System Settings** dialog provides these options.

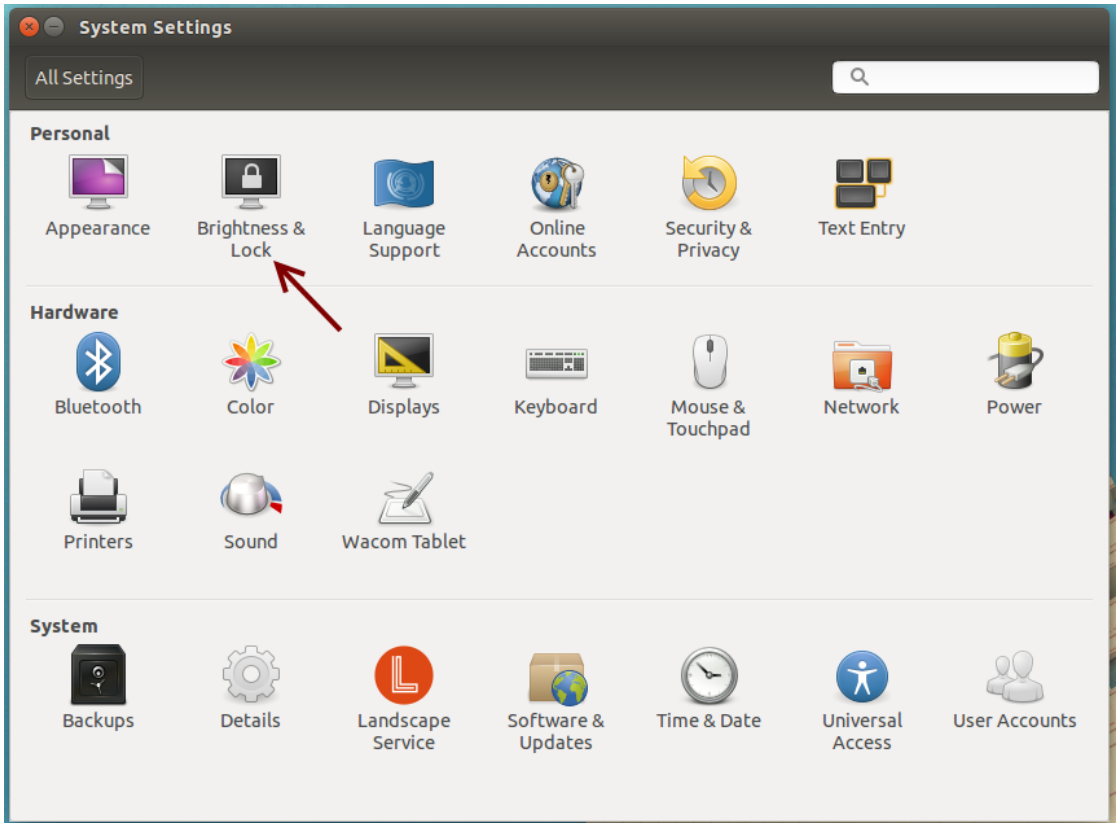

Figure 66: The Brightness & Lock Icon

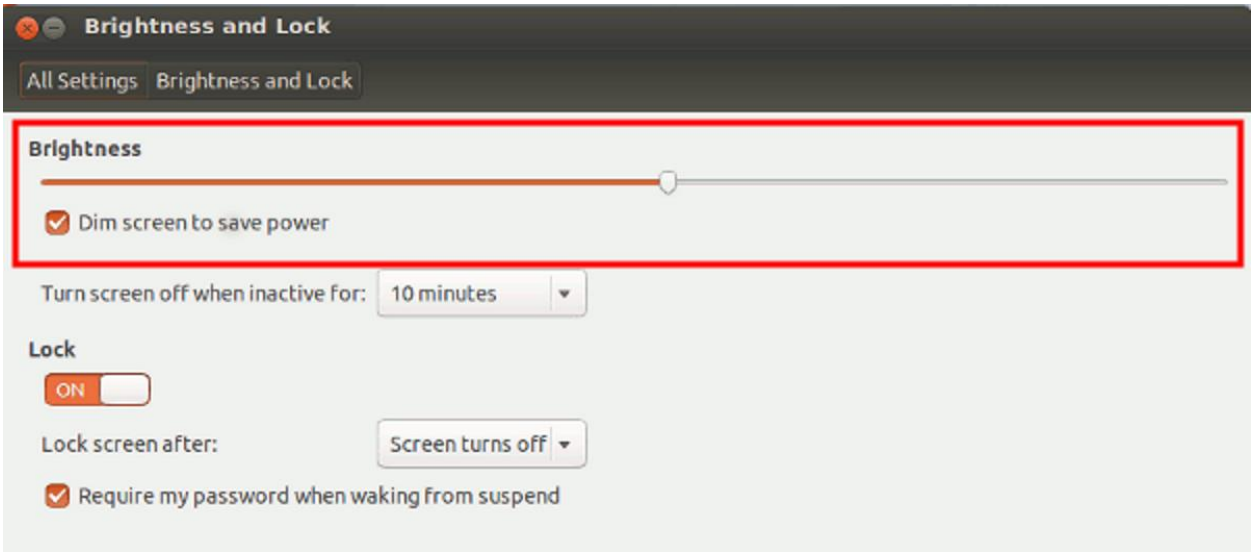

Figure 67: Brightness and Lock Parameters Dialog

To adjust brightness, the user can drag the slider shown in Figure 67. Dragging the slider to the left will reduce the screen brightness. The opposite will occur if the slider is dragged to the right. The **Dim screen to save power** option will reduce screen brightness when the system is idle (no keyboard or mouse movement). Screen brightness reduction can be avoided by clearing this option.

The **Turn screen off when inactive for** drop-down under the **Brightness** slider allows you to specify how many seconds must elapse after the system is idle to turn the screen off. This can be avoided by selecting **Never** from the list.

The lock screen can be disabled by dragging the **Lock** button shown in the dialog to the left. Dragging it to the right will enable the Lock screen. The **Turn screen off when inactive for** drop-down located after the **Lock** button allows you to specify how many seconds must elapse after the system turns idle to lock the computer automatically, and as a result, show the lock screen. If the user selects the **Screen turns off** option, the computer will be locked every time the screen is turned off. Then, when the user turns on the screen, the lock screen will be shown and the system will ask for the user password.

Managing user accounts

The **User Accounts** dialog is the desktop equivalent to all text commands about user accounts explained in the Chapter 5, "Security." This dialog can be shown by clicking the **User Accounts** icon in the **System Settings** dialog.

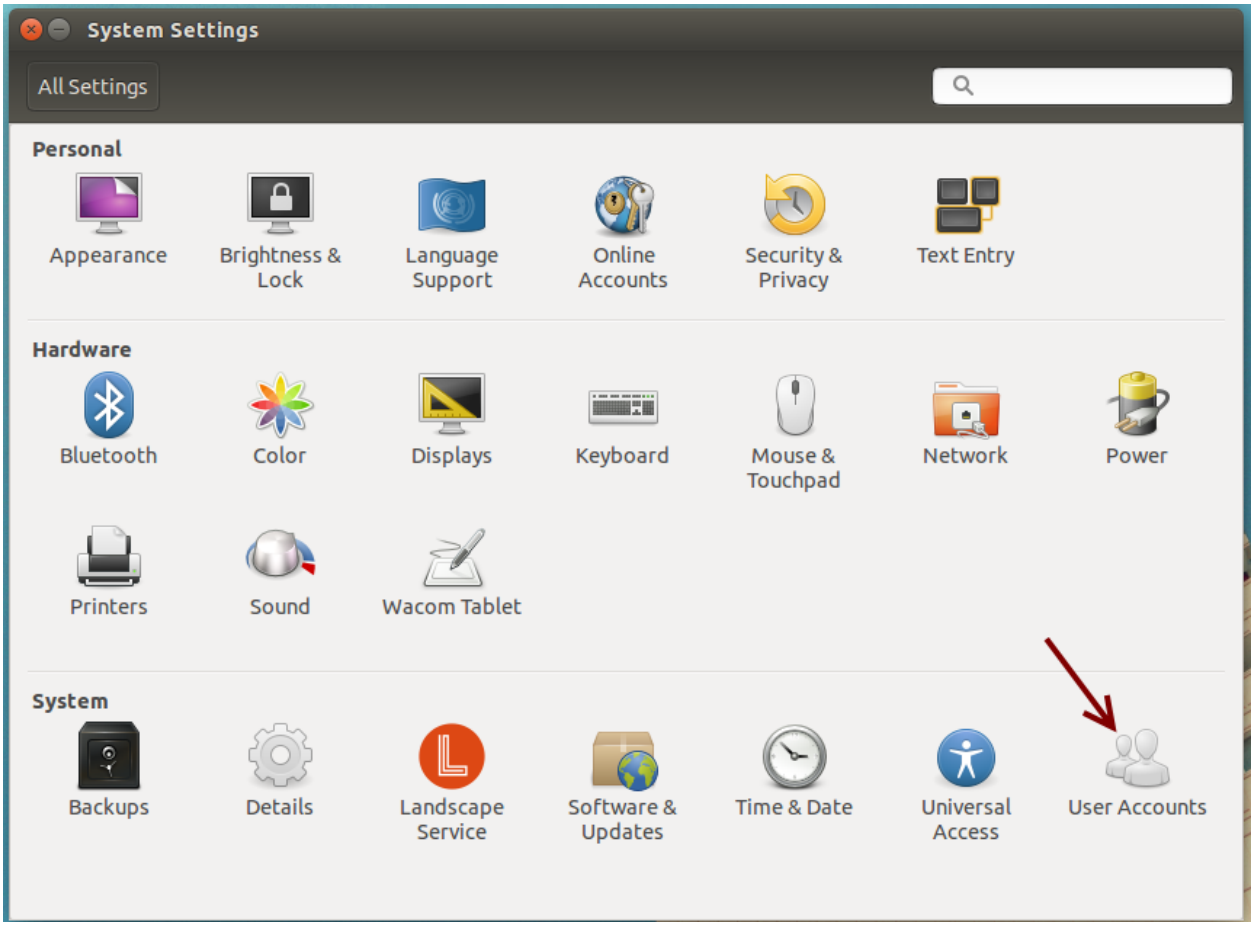

Figure 68: User Accounts Icon

Then, the **User Accounts** dialog will be shown.

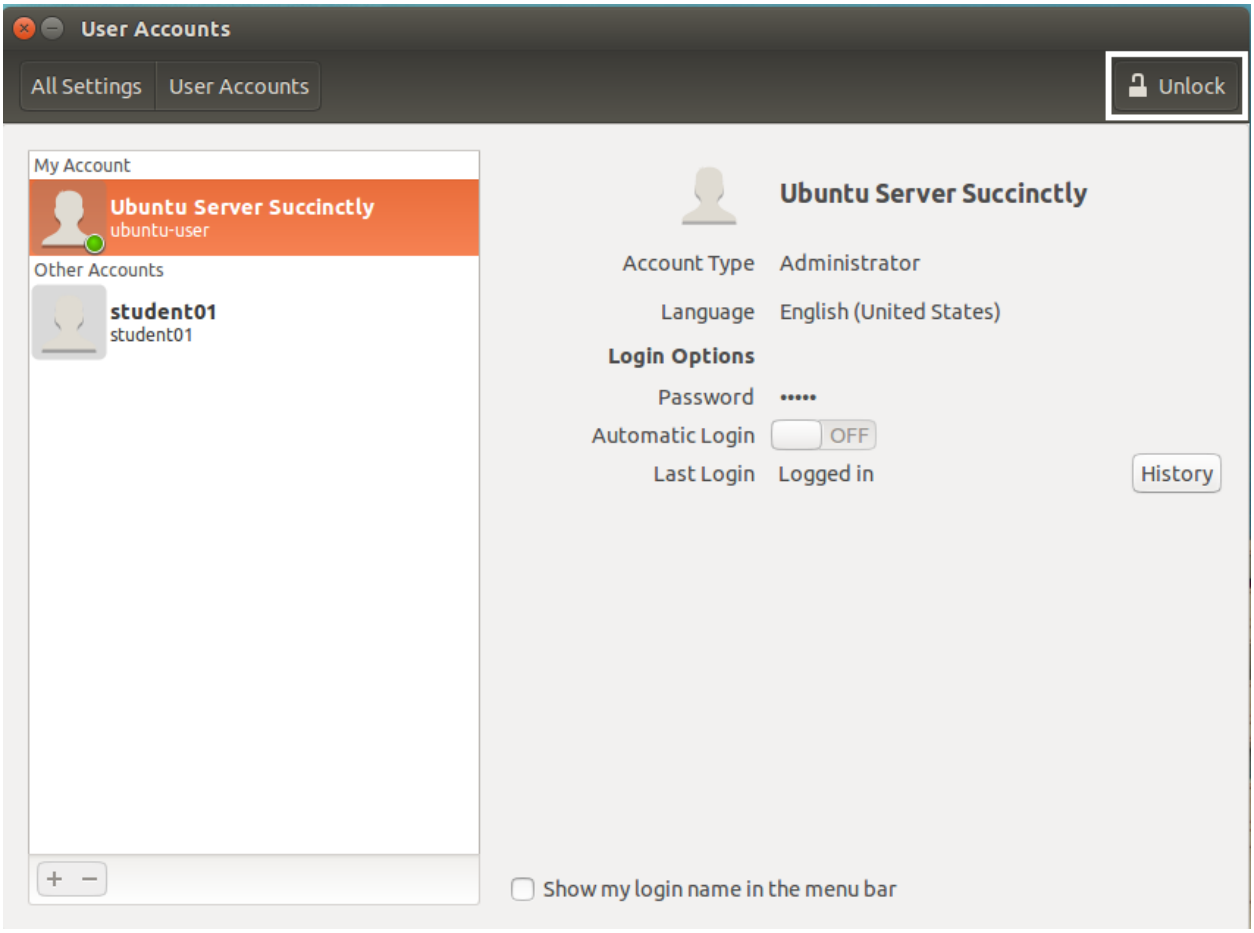

Figure 69: User Accounts Dialog

In the previous figure, the **Unlock** button is highlighted to indicate that any action with user accounts is prohibited. That is because user account management is an administrative task. It's necessary to unlock the dialog before managing users. This is the same as executing the **sudo** command explained at the beginning of this book. The user needs to click the **Unlock** button, and then the system will ask for the user password in order to continue.

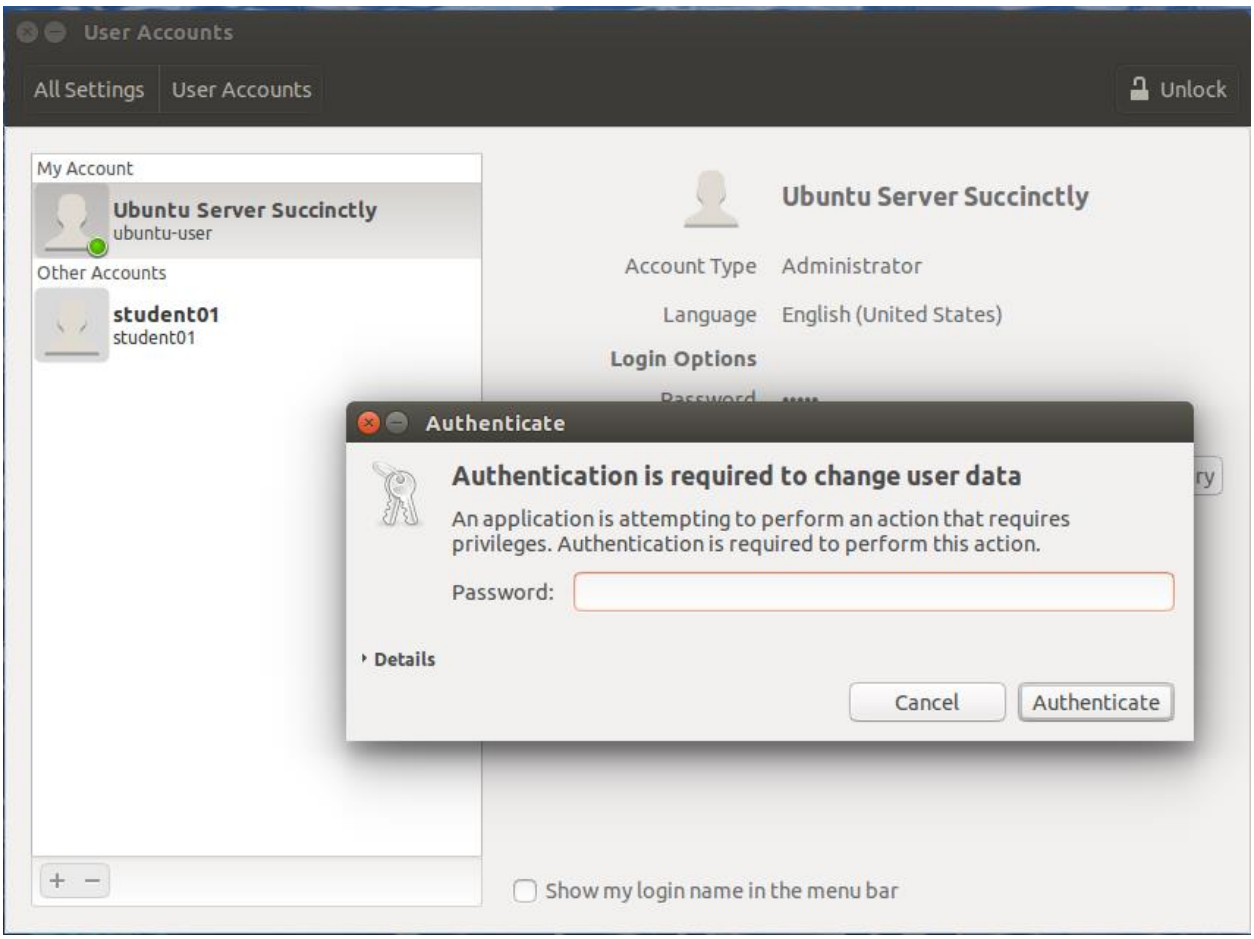

Figure 70: Authentication Dialog

The dialog will be unlocked after the user enters the correct password, and user account management will be allowed.

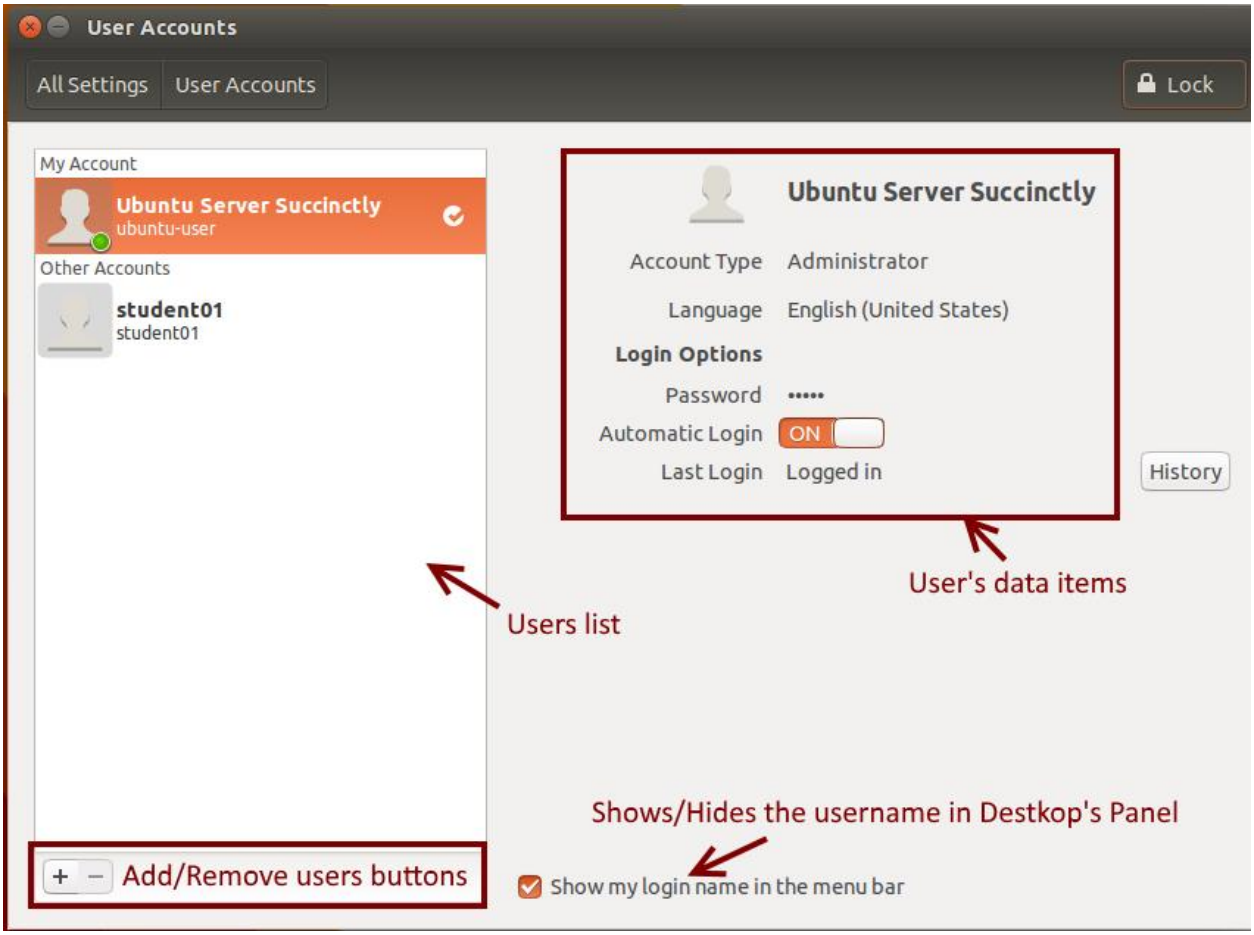

Figure 71: User Accounts Dialog Enabled

The buttons with the plus (+) and minus (-) characters, both located at the bottom of the dialog, can be used to add or remove user accounts. The minus button will be disabled automatically when the account for the currently logged-in user is selected.

When a user account is selected, the data associated with this account is displayed at the right side of the dialog, allowing the currently logged-in user to edit all items. Each one can be edited by clicking over it, except for the account type when the account selected belongs to the currently logged-in user. The following table explains user account data items.

Table 14: User Account Data Items

Item	Description
Account Type	Either Administrator or Standard. A Standard account can't make changes to the system.
Language Options	Language or language variation that is used by the user.
Password	User password to log in to the system.

Item	Description
Automatic Login	Allows the user to skip the password request when the computer is powered on, providing an automatic login. This can be accomplished by placing the Automatic Login button to the right, which will display an ON signal with an orange background color.

The **Show my login name in the menu bar** check box located at the bottom of the dialog allows you to display the user's full name at the right side of the desktop's panel.

Adding accounts

A user account can be added by clicking the **+** (plus) button, located at the lower-left side of the dialog. Then, the following dialog will be shown.

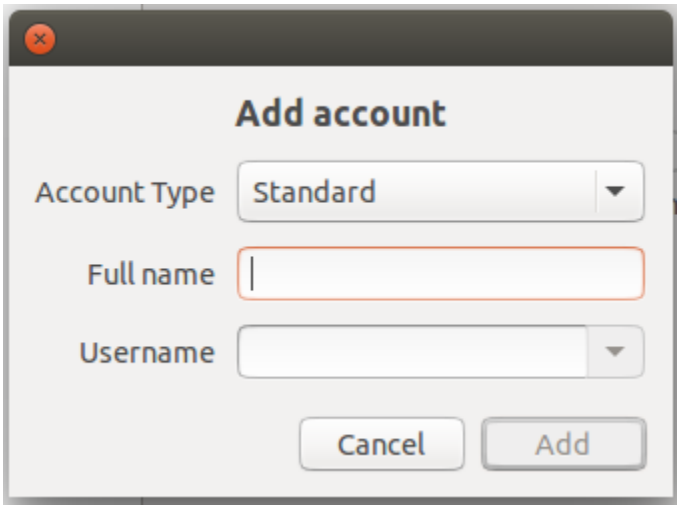

Figure 72: Add Account Dialog

The **Add account** dialog contains three items that must be filled to create the user account. These items are.

- **Account Type**: Either Standard or Administrator. Standard users can't make changes to the system.
- **Full Name**: User's full name (e.g., John Summers) to make the account name human-friendly.
- **Username**: User account name for system purposes (e.g., johnsummers).

The account will be added after clicking the **Add** button.

Deleting accounts

A user account can be deleted by clicking the **–** (minus) button located at the left bottom of the dialog. This action will show the following dialog.

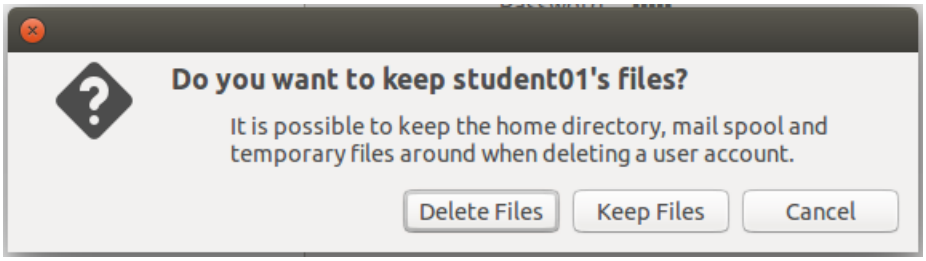

Figure 73: Delete Account Dialog

The options displayed in the dialog box correspond to those supplied by the **`deluser`** command, which was explained in the "Security" chapter of this book. The **Cancel** option is used to dismiss the delete account action with no consequences. The other buttons perform the following actions:

- **Delete Files**: The equivalent of the **`deluser --remove-home`** command, which deletes the user account and its home directory.
- **Keep Files**: The equivalent of the **`deluser`** command, which deletes the user account only, keeping the home directory on the disk.

Installing a printer

To install a printer device, the user needs to click on the **Printers** icon of the **System Settings** dialog.

Figure 74: The Printers Icon in the System Settings Dialog

Then, the printers dialog will appear on the screen.

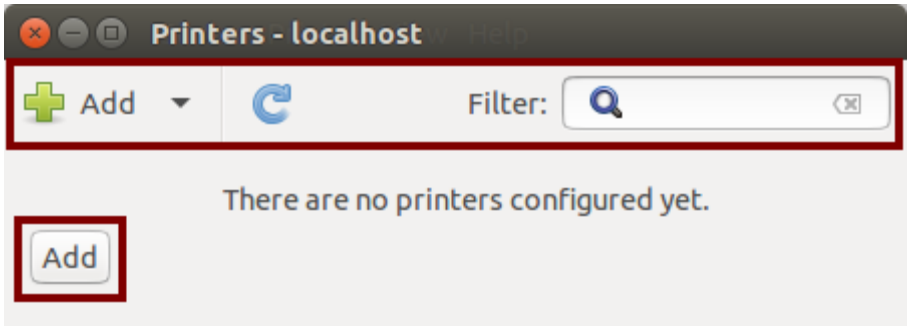

Figure 75: Printers Dialog for localhost

A new printer can be added by clicking the **Add** button located in the toolbar at the top, or by clicking the **Add** button that appears in the dialog box. The **New Printer** dialog will appear.

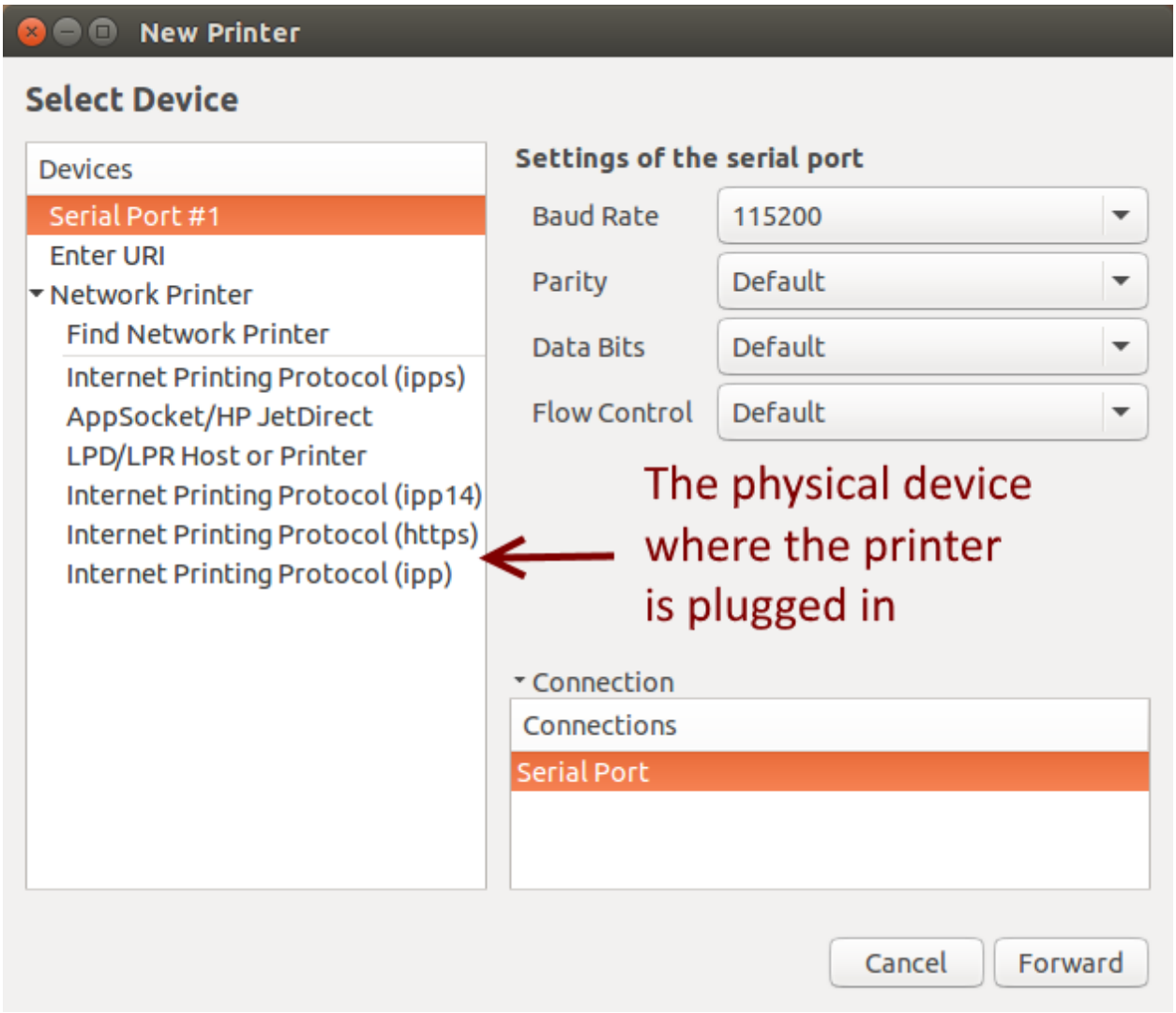

Figure 76: New Printer Dialog Asking for a Device

At this point, the user needs to select the physical device where the printer to add is plugged in, and then click the **Forward** button. In the previous figure, it's assumed that the device where the printer is plugged in is a Serial (RS232) port.

Now, the system will allow the driver selection for the printer model that is being installed. There are three choices for selecting a driver. The first one is selecting it from the internal driver database; the second allows you to choose a previously downloaded PPD file located in the filesystem; the third one allows you to search the Ubuntu website for a specific manufacturer and model in order to download the appropriate driver.

The installation process will continue after you click the **Forward** button.

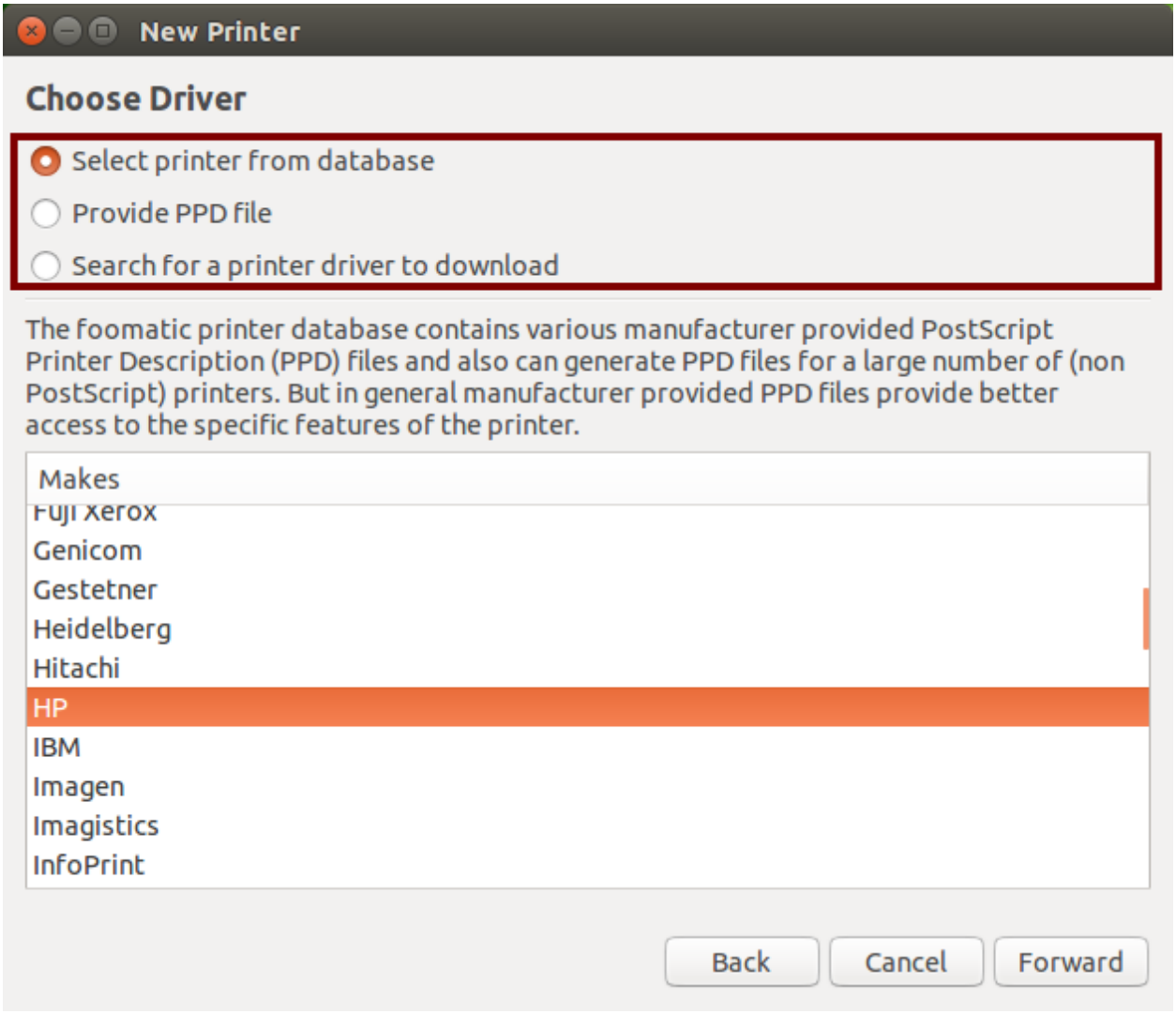

Figure 77: Driver Selection Dialog

When the user selects a manufacturer from the list, Ubuntu will ask for a specific model that must be selected from a list. Then, the user is asked for a description of the printer to assign a human-readable description and, optionally, a human-readable physical location (for example, "Sales Department") for that printer.

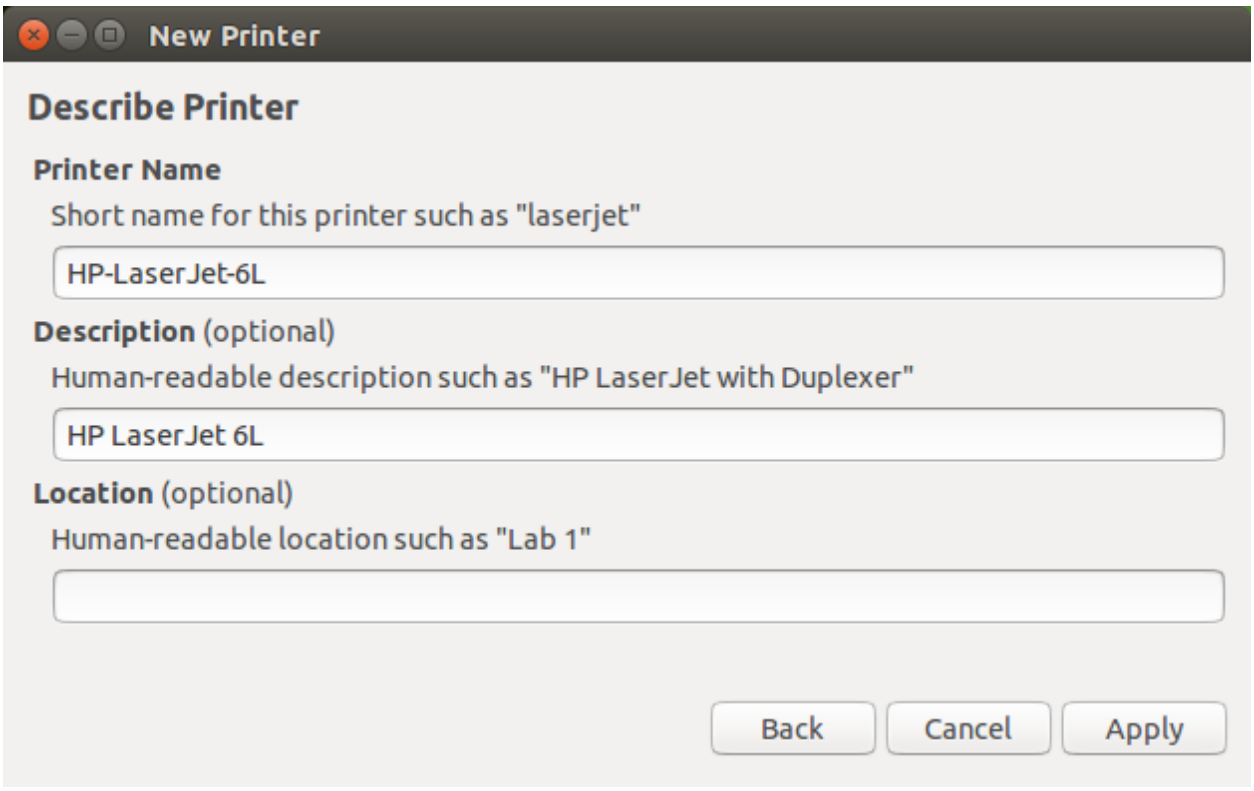

Figure 78: Printer Description Dialog

The installation process finishes when the user clicks the **Apply** button. Then, the system will ask to print a test page.

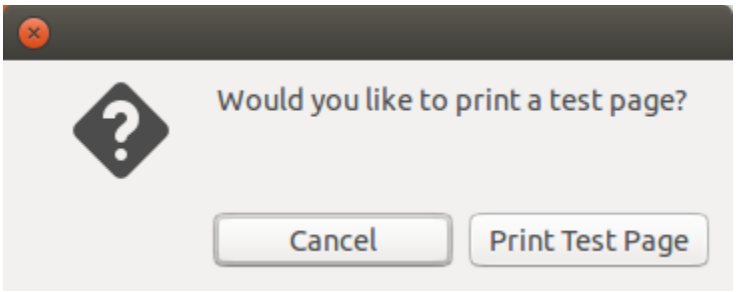

Figure 79: Test Page Dialog

The user can choose either to send a test page to the printer or cancel the request. In both cases, the installation process ends.

The Ubuntu Software Center

Ubuntu Software Center is a utility for installing, purchasing, and removing software in Ubuntu, a major part of Ubuntu's overall software handling.

Ubuntu Software Center lets the user browse and install thousands of free and paid applications available for Ubuntu. The user can view available software by category, or search quickly by name or description. You can also examine the software already installed, and remove items that are no longer needed.

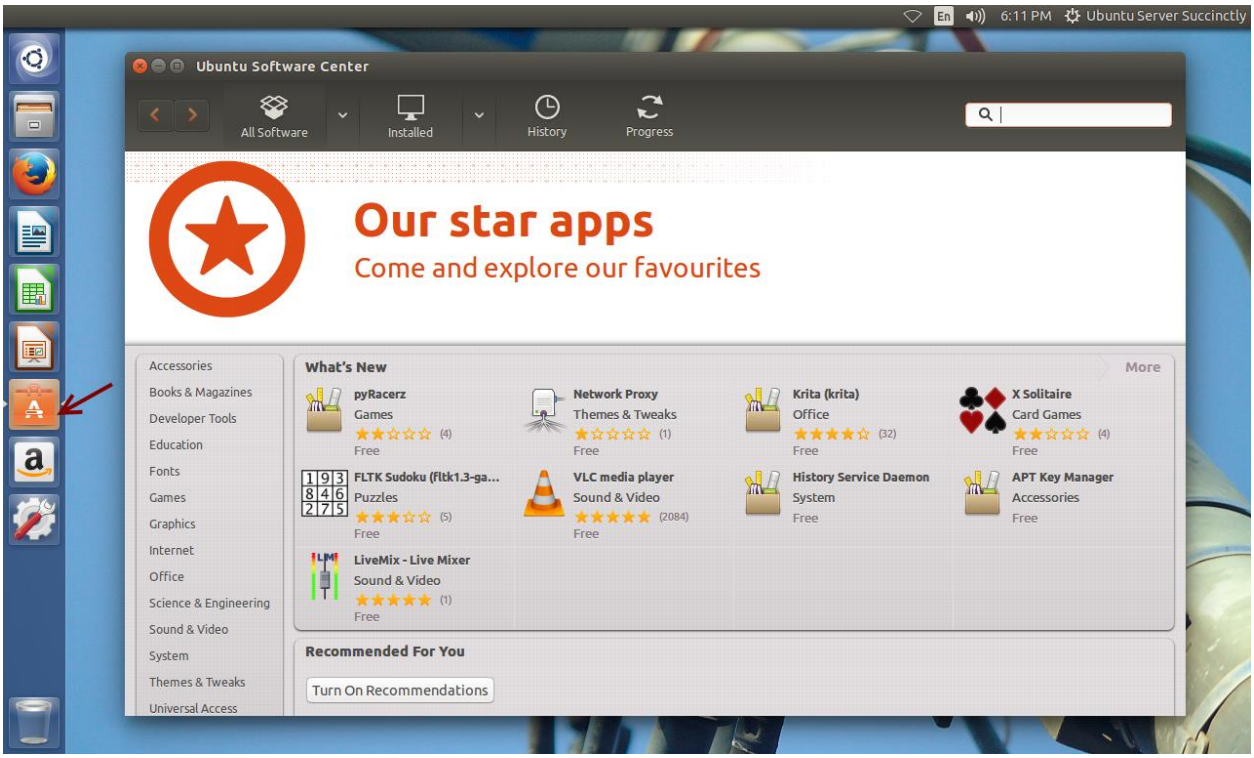

Figure 80: Ubuntu Software Center Main Screen

Ubuntu Software Center can be executed from the Launcher, as shown in the previous figure.

The Software Center toolbar

This toolbar contains four buttons for browsing all software available.

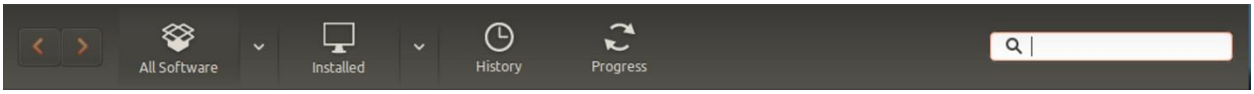

Figure 81: Software Center Toolbar

The **All Software** button shows all software available in a list, either provided by Ubuntu or from Canonical partners. When a software listed is already installed, a green checkmark is placed on its icon.

The **Installed** button filters the software list to show all software that is installed in the system only. Also, this software can be provided by Ubuntu or from Canonical partners.

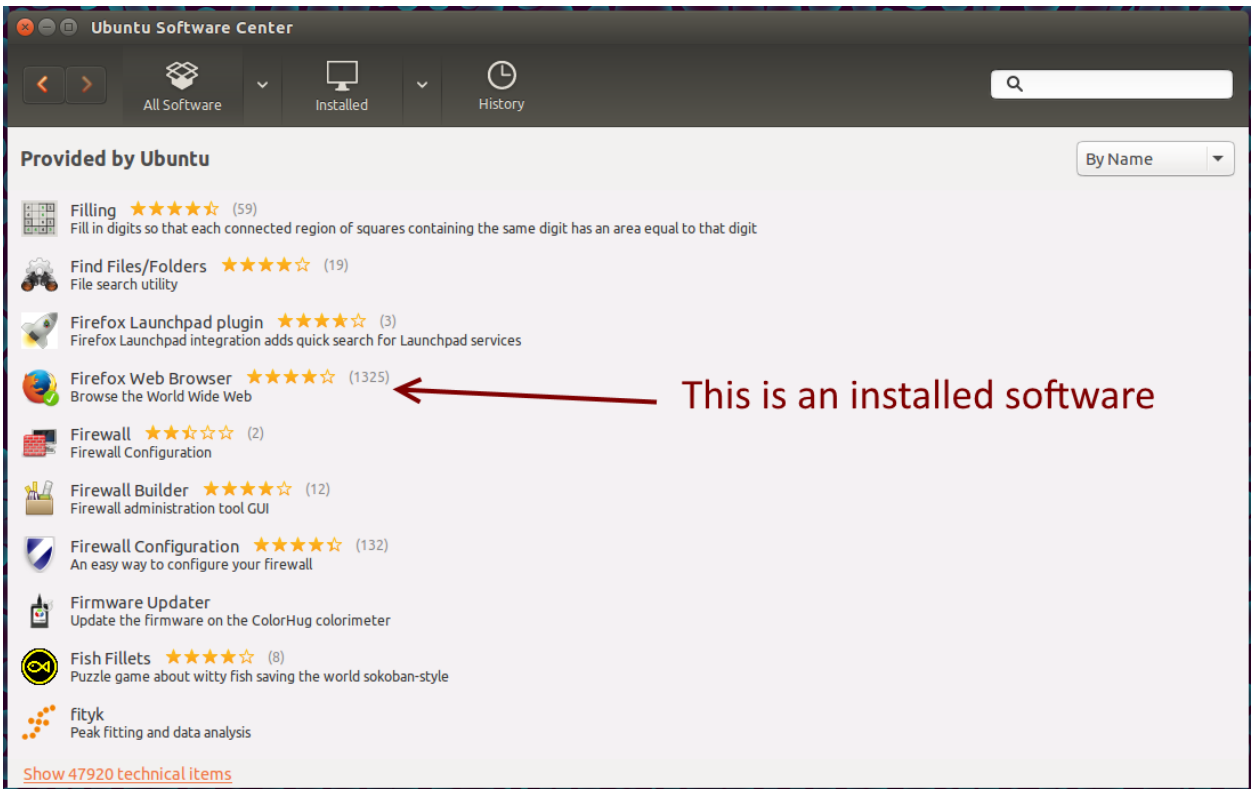

Figure 82: The All Software List

To remove software that is already installed, the user must select the software to uninstall from the list, and then click the **Remove** button.

Figure 83: Selecting Software to be Uninstalled

In the opposite way, new software can be installed by selecting the desired one from the list, and then clicking the **Install** button.

Figure 84: Selecting Software to be Installed

The **History** button allows the user to track every installation, update, or software removal by date. The list is presented from the most recent date of the operation. There are four groups in this list:

- **All changes**: Displays all changes made to the system. That includes installations, updates, and software removals.
- **Installations**: Displays only software installation operations.

- **Updates**: Displays only software update operations.
- **Removals**: Displays only those operations related to software uninstall operations.

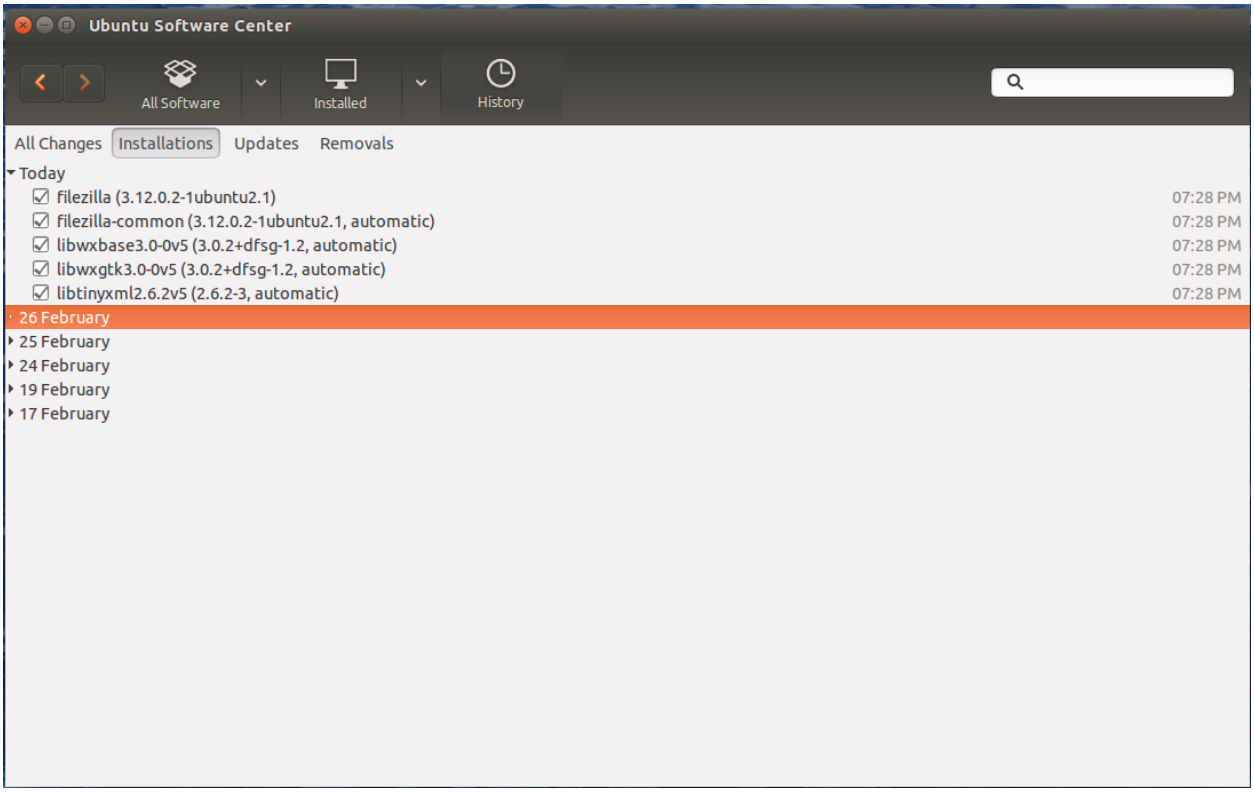

Figure 85: The Software History List

Finally, the **Progress** button is displayed only when a software install or software uninstall operation is in progress. When the operation finishes, the button disappears.

Searching for applications and other stuff

There is an interface in Ubuntu that allows searching for applications, files and folders, and even the web, from a text entry. This interface is called Dash, and can be activated by pressing the Super (Windows logo) key or by clicking the **Dash** button in the **Launcher**.

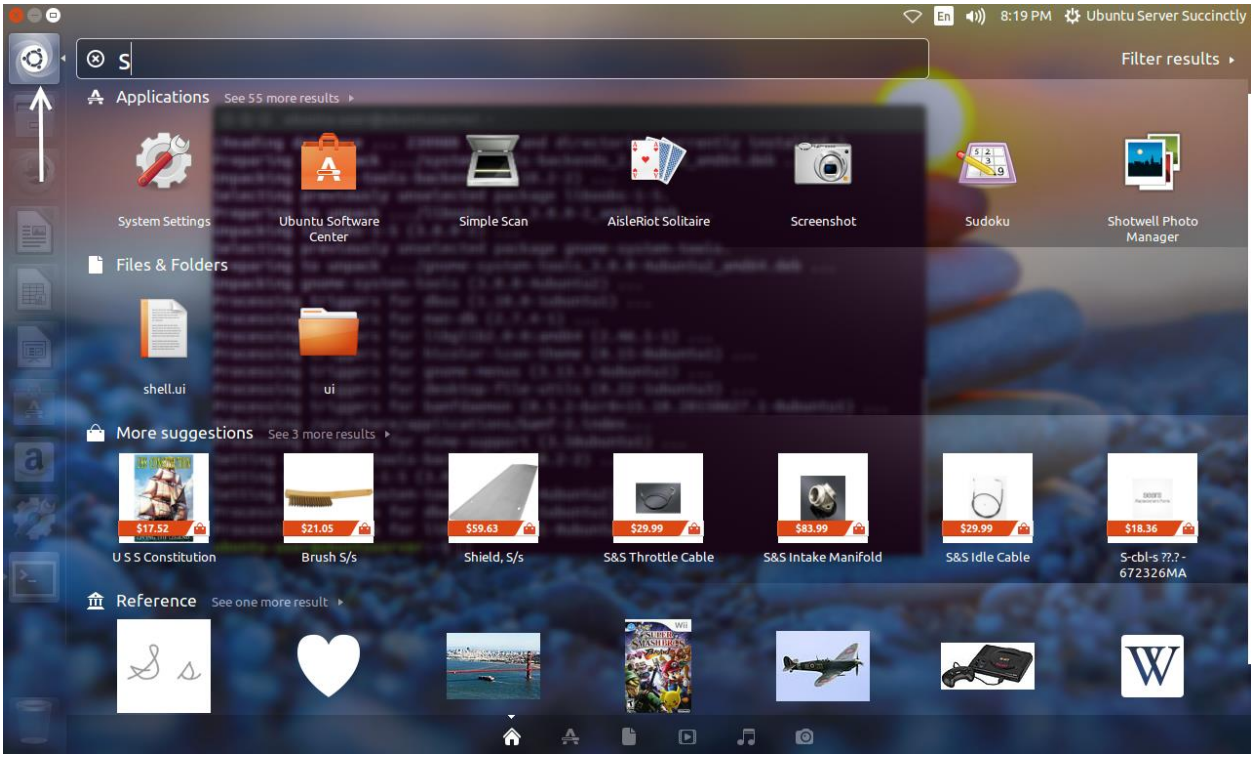

Figure 86: The Dash Activated in the Desktop

Every time a character is entered in the search bar, the Dash searches the system and then the web to find elements that match the text typed. This search is also performed when the user erases a character from the text entry in the search bar. The Dash searches for files, folders, applications, music, videos, and photos.

When a search is made, the user can click on the desired element's icon and the object will be opened by the proper application.

To close the Dash, the user clicks the **Dash** button located in the **Launcher**, or presses the Esc key.

The Terminal

The desktop environment offers an interface that allows you to type and execute text-based commands, in the same way as if these commands were typed in the command line. This interface is called Terminal, and can be shown by pressing Ctrl+Alt+T.

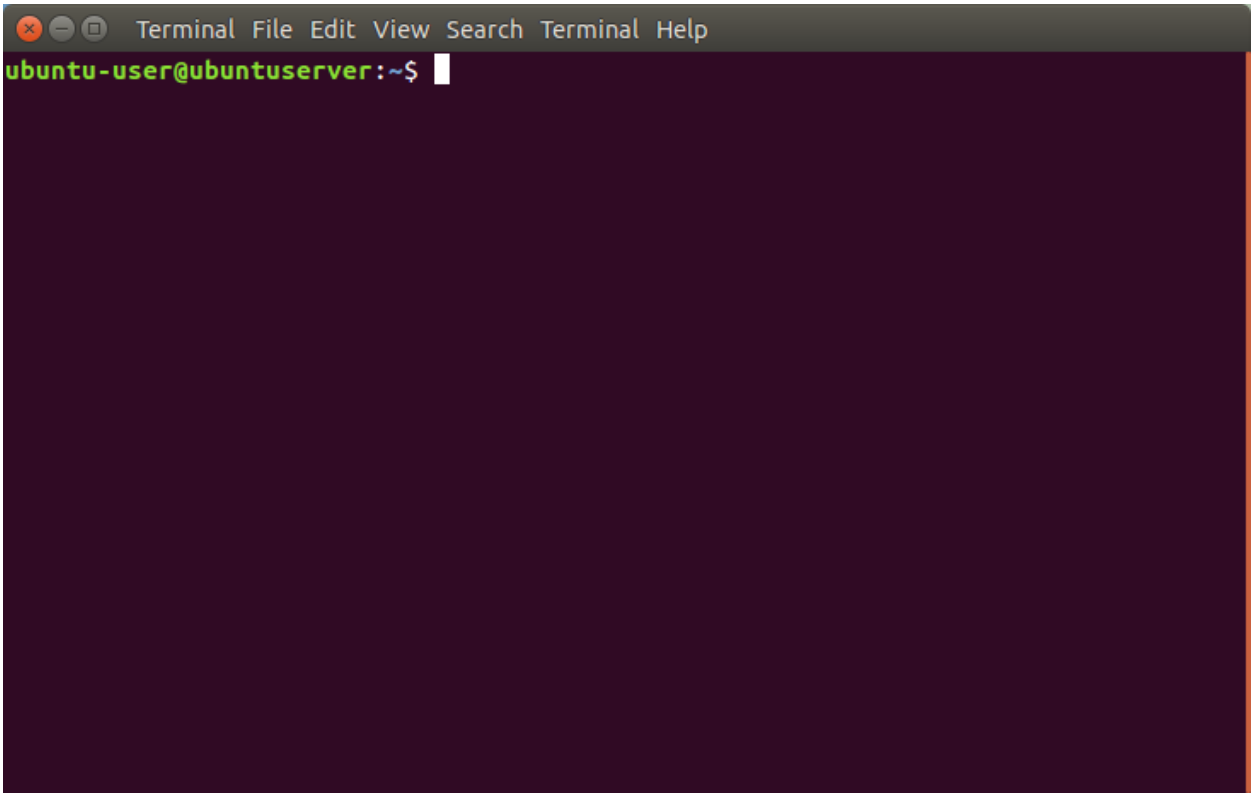

Figure 87: The Terminal Interface

The purpose of using this interface is to complete some tasks that can be done faster by typing text commands than using graphical applications. To close this interface, the user must type **exit** at the command prompt.

A common Terminal task performed by users is software installation. In some cases, installing an application can be more easily done by executing a single command, compared to navigating through the Software Center.

For example, if the name for the User and Groups utility software is previously known (gnome-system-tools), it's easier to install the package by using the **`sudo apt-get install`** command. The following figure shows the Terminal interface while executing this command.

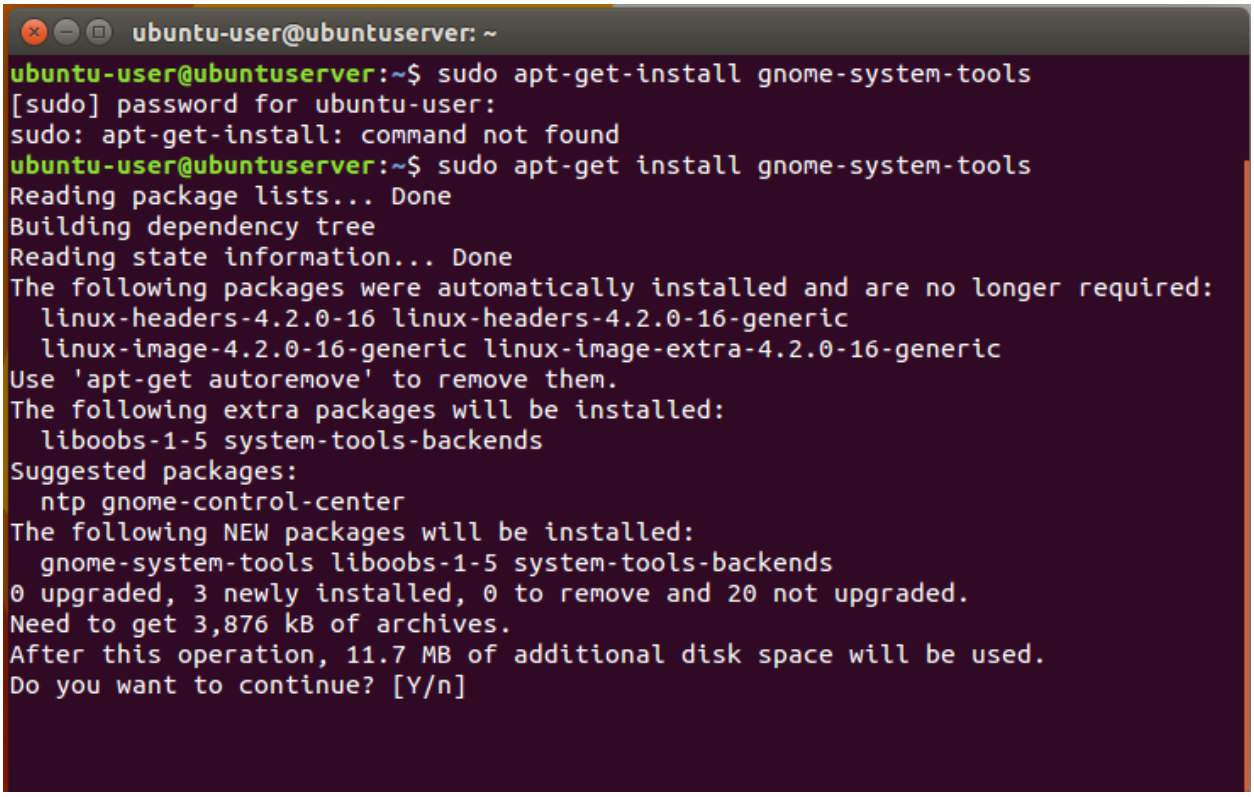

Figure 88: Installing the User and Groups Utility

Also, removing the User and Groups utility is easy by using the following command.

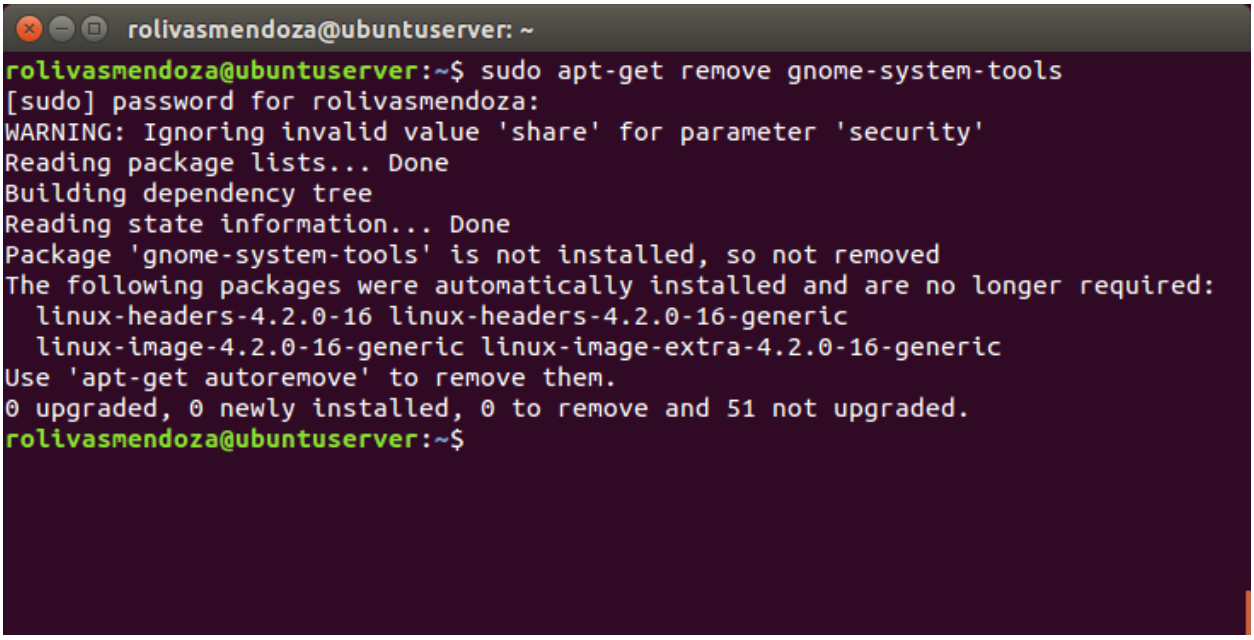

Figure 89: Removing the User and Groups Utility

Writing shell scripts in Ubuntu

In Ubuntu, a series of commands can be executed at once by writing them into a text file. This file is known as a shell script, and can be created with a text editor.

There is a Text Editor application for the desktop interface. To execute it, the user needs to call the Dash and write **text** in the Search bar.

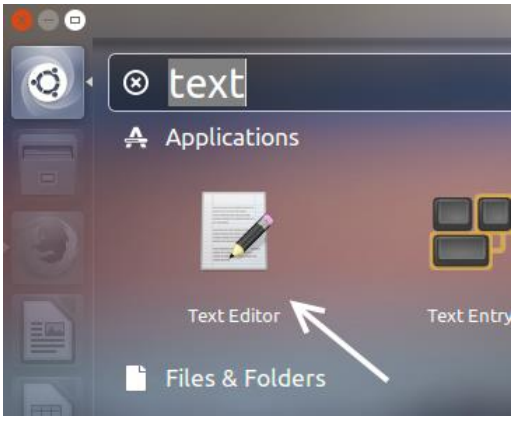

Figure 90: The Text Editor for the Desktop

The following sample shows a simple script.

Code Listing 155

```
#!/bin/bash
#First script
echo "Hello World"
```

The first line of the script is important. This is a special clue, called a shebang, given to the shell indicating which program is used to interpret the script. In this case, it is **`/bin/bash`**.

The second line is a comment. Everything that appears after a **`#`** symbol is ignored by bash. As shell scripts become bigger and more complicated, comments become vital. They are used by programmers to explain what is going on so that other programmers can figure it out. The last line is the **`echo`** command. This command simply displays its arguments on the screen.

Once the script is saved in disk, the next thing to do is give it permissions to be executed by the shell. The following figure shows how to give execution permissions for the script and the result of running it, assuming that this script will be saved in a file named **`hello_world`** in the **`/home/Ubuntu-user`** directory.

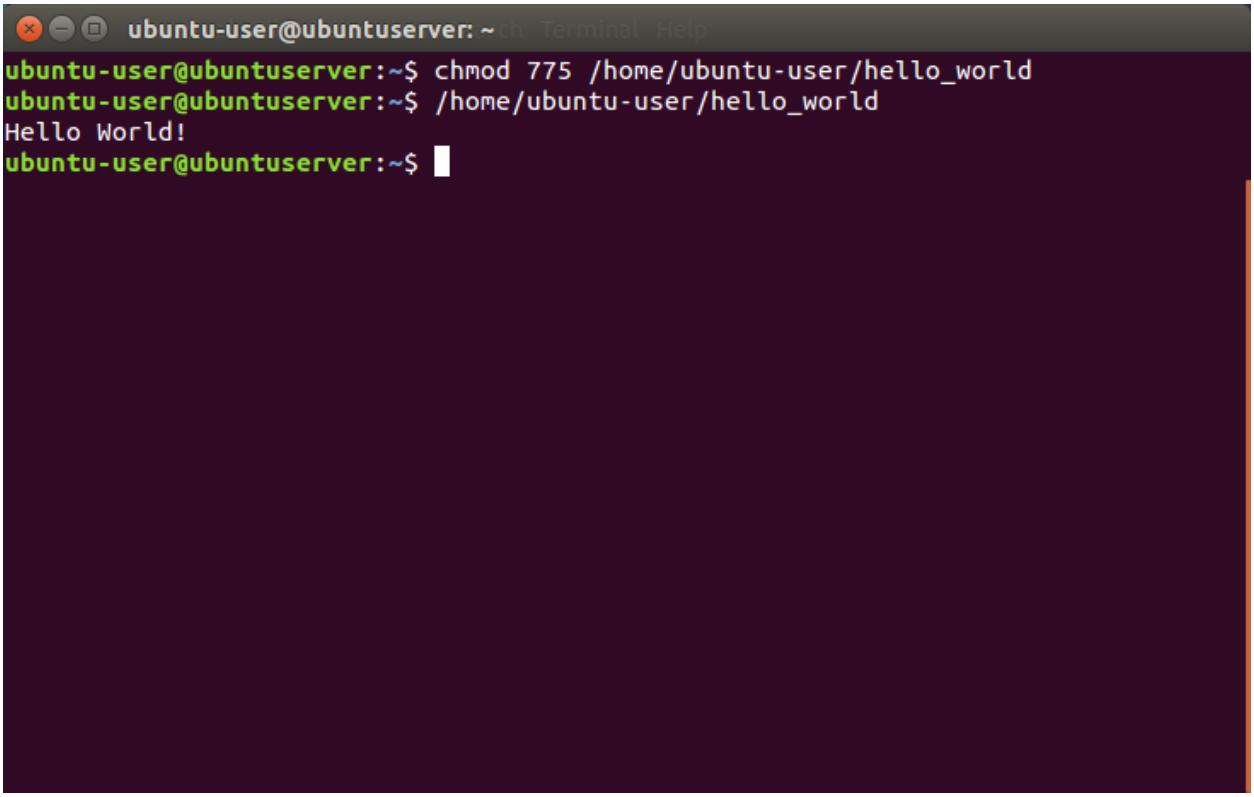

Figure 91: Giving Execution Permissions and Executing the Script

Going a little further, the following script sample makes a copy of the user's home directory content to a backup folder, located in the **/home** directory.

Code Listing 156

```
#!/bin/bash
#Backup Script
echo "Backing Up Home Directory"
cp -r /home/139buntu-user /home/backups/Ubuntu-user
```

For the most part, commands that can be entered on the command line are suitable for a shell script. A couple things might be considered regarding shell scripts:

- There are different command interpreters, called shells. The default is bash, but there are others, such as zsh, ksh, dash, perl, python, etc.
- To run a shell script, the user needs to make the file executable, which can be done with the **chmod** command.
- An extension can be added to the file name, usually .sh, but this is not necessary.
- **#!/bin/bash** must be written on the very first line of the file to tell Ubuntu what program to use to run the script.

Chapter summary

Ubuntu Server provides a GUI (Graphical User Interface) to replace the command line that is installed by default. This interface is called the desktop environment, and can be deployed by using the **`sudo apt-get update`** and the **`sudo apt-get install 140buntu-desktop`** commands.

Once installed, the login screen replaces the text login prompt. The user needs to supply a username and password in order to access the system.

The main program of the desktop interface is Ubuntu Desktop, which provides a point of entry for all applications installed in the system, and for all system management programs available.

The main element for user interaction in the desktop environment is called a dialog. A dialog has a series of elements, where each one has a particular function. These elements can be used when you click on them with the pointer. Also, some of these elements allow the user to enter text when needed.

The desktop environment provides a set of graphical applications to mimic the behavior of text commands. One of these applications is the **Files** dialog (file explorer), which is the equivalent of the **`ls`** command.

The **`shutdown`** and **`reboot`** commands have a graphical equivalent, called the **Shut Down** dialog. This dialog can be launched by clicking the **Shut Down** option in the **Settings** menu. This menu is accessed by clicking the gear icon, located in the upper-right corner of the Ubuntu Desktop screen.

A user's session can be locked by pressing the Super (Windows logo) key+L, bringing up the Lock screen. To get back into the working session, the user must enter a password.

The **System Settings** dialog is used to customize the system. This dialog has three sections: Personal, which allows you to customize the interface appearance, text entry devices, and user privacy settings; Hardware, which allows you to install new hardware devices or configure installed ones; and System, which allows you to configure global settings like date and time, software updates, and user accounts.

The Ubuntu Software Center is a tool provided to manage installed applications, and to install new ones. The tool can filter software into three categories: All software available (installed in the system or located on the Ubuntu website), installed software in the system, and the history of software installation operations.

The Dash is an interface used for searching any content in the system or in the web. This interface can be executed by pressing the Super (Windows) key. It performs a text search to locate what the user is looking for. The kind of content searched can be files, directories, applications, music, images, or videos. The search operation is performed in the local system first, and then on the Internet.

Sometimes, typing a text command is a better choice instead of using the graphical interface. In this case, Ubuntu provides an interface called Terminal. This interface can be shown by pressing Ctrl+Alt+T. Any of the text commands explained in this book can be used in the Terminal interface.

A series of commands can be executed at once by placing the commands in a text file. This file is called a shell script, and can created using a text editor. Ubuntu Desktop has a Text Editor application available. This application can be executed by searching for it using Dash, and then clicking on the application's icon.

For the most part, commands that can be entered on the command line are suitable for a shell script. A couple of things might be considered regarding shell scripts:

- There are different command interpreters, called shells. The default is bash, but there are others, such as zsh, ksh, dash, perl, python, etc.
- To run a shell script, the user needs to make the file executable, which can be done with the **`chmod`** command.
- An extension can be added to the file name, usually .sh, but this is not necessary.
- **`#!/bin/bash`** must be written on the very first line of the file to tell Ubuntu which program to use to run the script.

Chapter 10 Summary

Ubuntu Server is an operating system created from the vision and the idea of Mark Shuttleworth, who gathered a bunch of people in April 2014 to ask them for a better operating system. These people named themselves the Warthogs and gave themselves a six-month deadline to build a proof-of-concept OS, and named that first release the Warty Warthog.

The Warty Warthog exceeded the group's expectations and the most optimistic predictions, and became Ubuntu. The name "Ubuntu" was taken from a South African term that can be translated as "humanity toward others."

In order to sustain the Ubuntu vision, and going beyond that vision, Mark Shuttleworth founded a company named Canonical Ltd. The primary goal of this company was the development and support of the Ubuntu distribution. Canonical ensures that Ubuntu's bottom-line commitments are kept.

Ubuntu Server is available for the three major architectures: Intel X86, AMD64, and ARM. A computer with 512 MB of RAM and a 1 GHz processor can be used to install Ubuntu. The installation program can be downloaded from the Ubuntu website as an ISO image file.

To start the installation process, the ISO image file must be burned to a CD/DVD disc. Then, the CD/DVD disc must be placed in a CD/DVD drive in order to make the computer boot with the installation program.

The installation process takes a series of steps, starting with the language selection for the process and the user interface. During the installation, the user can configure the keyboard layout, the network parameters for the server, and the system clock.

To make the computer boot with Ubuntu, a hard disk must be used to store the operating system. A process called hard disk partitioning must be performed first. Disk partitioning is the creation of one or more regions on a hard disk or other kind of storage, so that an operating system can manage data in each region separately. Once the partitions are created, the disk stores the information about their locations and sizes in a special area known as the partition table.

To create partitions, Ubuntu needs to identify the hard disks attached to the computer. The naming convention for a hard drive starts with a slash and the **`dev`** abbreviation. This abbreviation stands for "device." Then, **`dev`** is followed by another slash and the **`sd`** abbreviation, which stands for SCSI (Small Computer System Interface) device. An example for a hard disk identification would be **`/dev/sd`** and a letter at the end, starting with **`a`**. So, the first drive found in the computer would be **`/dev/sda`**, the second disk would be **`/dev/sdb`**, and so on.

Once a disk partition is created, it needs to be formatted before it can be used. This process includes stamping the partition with a file system, which is a way to name and place files logically. There are several file systems available for Ubuntu, such as Ext4, Ext3, and Ext2. Besides these, there's another type called swap. Swap is a small section of unformatted hard disk that Ubuntu uses as virtual memory.

Ubuntu needs to create mount points in order to work properly. A mount point is a directory (typically empty) in the file system that is in the hard drive. The common mount points in Ubuntu are **/boot**, which stores the boot loader files; **/** (root), which is the root directory; and **/home**, which stores all users' files and directories.

The user must enter a username and password, which will be used to work with the system before the installation process starts copying files to the hard disk. The installation program will ask for a confirmation to ensure that the password was entered correctly.

After the installation process ends, the user must provide a username and password to log in to the system. If this succeeds, the command prompt will be shown on the screen. The command prompt is the place where the user must type text commands in order to work with the server.

A command named **sudo** is the most important in Ubuntu, and is used to grant superuser and other privileges to any command that is executed along with it. This is the best option for doing administrative tasks, because every time **sudo** ends its execution, it revokes the elevated privileges and brings the system to a normal user state. In that way, fatal accidents can be avoided, the because **sudo** command always asks for the active user password. A file named **sudoers** can modify this behavior. The **sudoers** file tells the **sudo** command which users can gain superuser privileges, and in some cases, how certain commands must be executed, and by whom.

Every file or directory stored in the filesystem must have a name. This name must follow certain rules to be used, such as the name for a file or directory must be unique within the location where it is placed. Also, all names are case sensitive. This means that a file named Place is different from another named place.

There are commands available to navigate within the Ubuntu Server filesystem. The user can see the files and directories in the system with the **ls** command, and can move around using the **cd** command. Also, file viewing is allowed by using the **ls** command, which displays the content of a file on the screen. The **mkdir** command is used to create a new directory and the **touch** command creates a new empty file.

Ubuntu Server takes into account security issues as well. For this reason, it disables the root administrative account by default to prevent accidents and possible system malfunction. Adding and deleting users or groups is another important way to keep the system safe. To perform these actions, the commands **adduser**, **useradd**, **deluser**, **userdel**, **addgroup**, **groupadd**, **delgroup**, and **gropupdel** are available. It's highly recommended to protect each user's home directory using the **chmod** command to assign the value 750 to the directory's permissions. Changing the value of the **DIR_MODE** variable located in the **/etc/adduser.conf** file to 750 will protect the home directory for each new user added to the system from other users' access.

Password policies ensure that security breaches can be avoided. Establishing a minimum password length in the **/etc/pam.d/common-password** file and forcing users to change their passwords periodically by editing the **/etc/login.defs** file are good security practices.

Granting ownership and permissions to files and directories makes security complete. The **chown** and **chmod** commands are used to do this. Permissions can be identified with *alphabetic notation*, which uses the letters **r**, **w**, and **x** for read, write, and execute permissions. *Octal*

notation also exists, and uses a series of values between 0 and 7 to represent the read (**r** = 4), write (**w** = 2), and execute (**x** = 1) permissions. A value of 0 signifies access restriction.

Networking is a substantial theme when installing and configuring a computer as a server. Ubuntu Server has a series of commands that allow you to configure the server in the network. A list of all available interfaces can be obtained with the **ifconfig –a | grep –I eth** command, and detailed information about a network adapter can be displayed with the **sudo lshw –class network** command. The **ethtool** command allows the user to view the adapter settings and gather statistics about it.

IP addressing is another important issue in networking. This means assigning an IP address to any device in the network. An IP address is a 32-bit numeric value defined under the Ipv4 protocol to identify and address a location for a device in the network. The Ipv4 protocol divides networks into five classes, A, B, C, D, and E. For private networks (built at a home or office), the classes A, B, and C are used. The **ifconfig** command allows you to manage IP addressing in Ubuntu Server. This kind of IP addressing is known as static IP addressing. Ubuntu Server also allows dynamic (automatic) IP addressing. In order to establish dynamic or static addressing by default, the **/etc/network/interfaces** file must be edited.

Dynamic IP addressing is possible because of the DHCP protocol. This protocol allows a device in the network to be in charge of assigning IP addresses to other connected devices. This device is known as a DHCP server. Ubuntu can be used as a DHCP server by installing the **isc-dhcp-server** service.

The networking process allows you to synchronize the system's clock with a remote server's time. To do this, the NTP (Network Time Protocol) must be used. Ubuntu manages NTP with the **ntpdate** command and the **ntpd** service.

In most scenarios, computer networks work with both Ubuntu Server or Ubuntu Desktop computers and Windows computers. It's necessary to make them work in harmony. Ubuntu Server includes the Samba suite for Windows networking. It is necessary to configure Samba as a file server for sharing resources with Windows computers by using the **sudo apt-get install samba** command.

Samba can be used for customization by editing the **/etc/samba/smb.conf** file. This file consists of sections and parameters. Each section is identified with a name enclosed in brackets, and there are three of them that are special: **global** (which allows you to define the parameters for the whole Samba server operation), **homes**, and **printers**. Every additional section defines a **share**, which is a shared network resource. The file can have as many of those shares as needed.

Each section in the **/etc/samba/smb.conf** file contains parameters. A parameter is a value that tells Samba how to behave. An example of a parameter is **workgroup**, which indicates the name for the group of computers in the network, and it's declared in the **global** section of the file.

The **printers** section of the **/etc/samba/smb.conf** file contains the parameters needed to browse and access all the printers installed in Ubuntu Server. The server must have a working CUPS (Common UNIX Printing System) installation to share printers.

Secure access to files and printers can be configured in Samba. There are two security levels: user-level and share-level. User-level security mode enforces every user to supply credentials (username and password). Share-level security mode allows you to define which users or groups of users are allowed to access shared resources.

A computer running Ubuntu Server can be a host for a DBMS (database management system) in order to store and process data over a network. Ubuntu Server provides two popular database systems: PostgreSQL and MySQL.

Both PostgreSQL and MySQL can be installed by using the **`sudo apt-get install`** command. The difference is the name of the package to be installed. For PostgreSQL, the package is named **`postgresql`**, and for MySQL, the name of the package is **`mysql-server`**.

In the MySQL installation process, a dialog box will be shown to provide a password for the root user. The root user is the one with all administrative privileges for managing the database server.

In PostgreSQL, the administrative user is called **`postgres`**. To assign a password for that user, a connection to the **`template1`** database needs to be made. This connection can be done with the **`sudo –u postgres psql template1`** command. This command will show the PostgreSQL command prompt. Then, using **`the ALTER USER postgres WITH ENCRYPTED PASSWORD '<user_password>'`** command, a new password for the **`postgres`** user will be assigned.

Ubuntu Server provides a GUI (graphical user interface) to replace the command line that is installed by default. This interface is called the desktop environment and can be deployed by using the **`sudo apt-get update`** and **`sudo apt-get install145buntuu-desktop`** commands.

Once installed, the Login screen replaces the text login prompt. The user needs to supply a username and password to access the system.

The main program of the desktop interface is Ubuntu Desktop, which provides a point of entry for all applications installed in the system, and for all system management programs available.

The main element for user interaction in the desktop environment is called a dialog. A dialog has a series of elements, each with a particular function. A dialog's elements can be used by clicking on them using the pointer. Some of these elements allow the user to enter text when it is needed.

The desktop environment provides a set of graphical applications to mimic the behavior of text commands. One of these applications is the file explorer dialog (Files), which is the equivalent of the **`ls`** command.

The **`shutdown`** and **`reboot`** commands have a graphical equivalent, called the **Shut Down** dialog. This dialog can be launched by clicking the **Shut Down** option in the **Settings** menu. This menu is accessed by clicking the gear icon, located in the upper-right corner of the Ubuntu Desktop screen.

A user's session can be locked by pressing the Super (Windows logo) key+L, bringing up the lock screen. To get back into the working session, the user must enter a password.

There are several tools in the desktop environment available for managing the system. The **System Settings** dialog is used to customize the interface, the hardware, and the global system settings. The Ubuntu Software Center is a tool provided to manage installed applications and to install new ones. The Dash is an interface used for searching for any content in the system or on the web. The type of content searched for can be files, directories, applications, music, images, or videos. Sometimes, typing a text command is a better choice instead of using the graphical interface. For doing this, Ubuntu provides an interface called Terminal. Any of the text commands explained in this book can be used in the Terminal interface.

Finally, the user can execute a series of commands at once by placing them in a text file called a shell script. This file can be created using a text editor. The interpreter for this script is called bash. There are others available as well, like dash or python. For the most part, commands that can be entered on the command line are suitable for a shell script.

Conclusion

With many Linux distributions offered, Ubuntu (Server or Desktop) is one of the most reliable and easy-to-implement distributions in the market.

Conceived at the beginning as a “better operating system,” Ubuntu has evolved to become a mature operating system that is supported by a huge worldwide community, and is perhaps the only Linux distribution that is sponsored by a private company, with no separate commercial and open-source versions. This fact is very important, if we consider that having a paid team in charge of an open-source development ensures that promises made about Ubuntu’s enhancements and delivery times will be kept.

On the other hand, the desktop environment makes the transition from Windows to Linux very easy. An IT company that needs to deliver a Linux-based solution in a short period of time can take advantage of Ubuntu’s features to build a consultant team—with no previous Linux experience—faster than with other Linux distributions.

Nowadays, Ubuntu is taking the next step in computing by reaching smartphone software and hardware niches, tablets, cloud services, and IoT. Canonical, the company in charge of Ubuntu development, has a group of important partners, such as IBM, Cisco, AMD, and Hewlett-Packard.

For all the previous details mentioned, Ubuntu is probably the best option available in the IT industry for Linux computing.