THE EXPERT'S VOICE® IN LINUX

Beginning the Linux Command Line

Learn how to put the command line to work to manage files, administer users and groups, install new software, run network services, and create basic shell scripts.

—

Second Edition

—

Sander van Vugt

Apress®

Beginning the Linux Command Line

Second Edition

Sander van Vugt

Apress®

Beginning the Linux Command Line, Second edition

ISBN-13 (pbk): 978-1-4302-6830-7

ISBN-13 (electronic): 978-1-4302-6829-1

Managing Director: Welmoed Spahr
Lead Editor: Michelle Lowman
Technical Reviewer: Stewart Watkiss
Editorial Board: Steve Anglin, Louise Corrigan, Jim DeWolf, Jonathan Gennick, Robert Hutchinson, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Mark Powers
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this text is available to readers at www.apress.com/9781430268307. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

This book is dedicated to Florence. Your love is making me invincible.

Contents at a Glance

Contents

About the Author

Sander van Vugt is a Linux expert working from the Netherlands as an author, technical trainer, and consultant for clients around the world. Sander has published several books about different Linux distributions and is a regular contributor to major international Linux-related web sites. As a consultant, he is specialized in Linux high availability and performance optimization. As a technical trainer, Sander is an authorized trainer for SUSE Linux Enterprise Server and Red Hat Enterprise Linux. More information about the author can be found at his web site at `www.sandervanvugt.com`.

Introduction

This book is for anyone who wants to master Linux from the command line. When writing it, I had in mind system administrators, software developers, and enthusiastic users who want to get things going from the Linux command line. For beginning users, this may be a daunting task, as Linux commands often have many options documented only in man pages that are not that easy to understand.

This book is distribution agnostic. That is, while writing it, I've checked all items against Ubuntu, Red Hat, and SUSE. Since most distributions are quite similar to one of these three, this book should help you with other distributions as well. There is only one item in the book that is not distribution agnostic: the Appendix, which explains how to install either CentOS or Ubuntu.

The book begins with an introduction to exactly what I'm talking about when discussing Linux and its different appearances: the distributions. In Chapter 1, you'll also find essential information on how to log on to the computer and how to find out more about the way a command should be used. Chapter 2 follows with some essential Linux commands. After reading this chapter, you'll already start to feel at ease on the Linux command line; among other things, it teaches you how to work with files and directories and how to communicate with other users. Chapter 3 moves the focus to one of the most important tasks you'll perform when working with Linux: working with files. In this chapter, you'll learn not only how to copy files and make directories, but also how to mount devices to your Linux system.

Working with Linux from the command line means working with text files. In Chapter 4, you'll learn about the tools that are at your disposal to do this. You'll get familiar with some of the classic tools, such as find and grep, and also with some of the more advanced tools, such as awk and sed. Following that, in Chapter 5 you'll learn more about partitions, logical volumes, and other advanced file system management tasks. After reading this chapter, you'll start feeling at ease on the Linux command line. Chapters 6 and 7 move on to two other essential subjects: the management of users and permissions.

Chapter 8 covers a topic that seems to be handled differently by all the Linux distributions: software management. This chapter teaches you about generic ways to install and manage software packages, such as rpm and dpkg, and also about some of the distribution- specific ways to deal with these tasks, such as apt-get, rpm, and zypper. Chapters 9 and 10 cover tasks that are important for system administration. In these chapters, you'll learn how to manage processes and how to handle logging on your computer.

By the time you reach Chapters 11 and 12, you're ready to explore network-related tasks. In these chapters, you'll learn how to configure a network interface and how to set up the Samba and NFS file services. Chapters 13 and 14 cover two advanced but useful topics: kernel management and shell scripting. After you finish the last chapter, you'll have all the knowledge you need to work with Linux from the command line.

In Appendix B you'll find some additional exercises, which help making this book an excellent study guide that can be used in classroom environments.

I hope you enjoy reading this book and that it prepares you for getting things done from the Linux command line!

CHAPTER 1

Starting Linux Command-Line Administration

To unleash the full power of Linux, as a Linux administrator you will spend most of your time typing commands on the Linux command line, the so-called shell prompt. For someone who is new to the command line, the things that advanced users do there may look like magic. In this chapter, you'll learn about the following topics:

- History of the Linux operating system
- What is open source?
- What are distributions?
- Logging in to Linux
- Command basics: working with commands, options, and arguments
- Using piping and redirection
- Getting help with `--help` and `man`
- Working with the shell

Linux Distributions

For someone new to Linux, the operating system may appear a little bit strange. Due to its open source character, there are different versions (the so-called distributions) of Linux. After some Linux history, this chapter teaches you about the differences and similarities between these distributions so that you'll be able to pick the Linux distribution that fits your needs in the best possible way.

Linux History

Linux started around 1991 all because the Finnish student Linus Torvalds wasn't too happy with Minix, the educational version of the UNIX operating system that he had to work with at the University of Helsinki. In particular, the ability of the kernel (which is the heart of the operating system) of this Minix distribution didn't please him much. He decided to create a better kernel and gave it the name Linux.

Possibly the smartest thing that Torvalds did when starting his initiative was deciding not to do it alone. To find other people who wanted to work with him, he posted a message on Usenet, a major social media platform in those days that could be used to exchange information with other people and get help from other people.

The initiative by Torvalds didn't stand on its own. Many other software developers had already started initiatives to create free software for the UNIX operating system. The only thing that really was missing at that moment was a kernel that was stable enough to go into production and offer a complete and free alternative to the expensive UNIX operating system.

Open Source

Right from the start, Torvalds released his software as open source software—that is, software whose source code is freely available to anyone. This brings a major benefit: other programmers have access to the source code and thus it becomes much easier to fix issues in the code, with the result of getting code that is more stable and reliable. This open source initiative fitted well into many other open source programs that were a part of the GNU initiative. The acronym GNU stands for GNU is Not UNIX, which means that this is about software written for the UNIX platform but doesn't use UNIX licensing. This GNU initiative was a part of the Free Software Foundation (FSF), which wanted to create free software for a better operating system experience.

When it came to licensing, Torvalds released his software under the GPL. In those days, GPL stood for GNU Public License, but nowadays it means General Public License. The details of this license are quite complex, but in essence it means that software released under the GPL can be used and modified by anyone, as long as the person modifying this software makes sure that his or her modifications will be released under the GPL as well. In brief: once software has a GPL, it will always stay GPL software. This prevents companies from making small modifications and then taking the software out of GPL and selling it for a lot of money.

In current Linux versions GPL still is very important. Most distributions consider it an essential property of the Operating system and will refuse software that doesn't comply with the conditions in the GPL license. By being so strict about the licensing, Linux distributions ensure that the software that is released as Open Source software will always stay Open Source software. Some distrubutions do also allow non-open source softare to be included, such as binary code for specific firmware. You will notice a different philosophical approach in this between the different distributions.

The First Distributions

Apart from the Linux kernel, lots of other programs were available under the GPL as well. In the early days, people who wanted to start using Linux had to go on the Internet and download these software programs themselves. Often, after downloading them, they even had to compile them for themselves. This compilation process was necessary to convert the program files, which were published as source code files only, to executable programs that users could execute on their computer.

Software compilation is not very easy to do, and for that reason, different people started to create collections that consisted of the Linux kernel and some other useful programs. One of the first persons to do so was Patrick Volkerding, who started his Slackware distribution in 1993. In those days, this distribution consisted of different software categories, all put together on no fewer than 43 diskettes. Volkerding was perhaps the first who made a successful Linux distribution that started to get used on servers all around the world, and his Slackware distribution still exists, although new releases are not published very frequently.

Linux Turning Mainstream

The years between 1993 and 1998 marked the rise of the Linux operating system. One of the most important reasons that it took of so rapidly, is that it provided a very affordable alternative for the expensive UNIX operating system that was used on many mission-critical server systems. In the early days of Linux, no support was available, but that didn't prevent scientists at different institutes around the globe to start working with it. Linux also acquired huge popularity rapidly in educational environments. Due to this popularity, during this period the most important Linux distributions were created.

Whereas Slackware was just a collection of software programs with an installation program that made working with Linux easy, other Linux distributions soon started to add value to the open source software. Some did this by adding commercial support to their software collection, others by creating programs and adding that to their distribution, and some hired developers to optimize the open source programs. The result is that nowadays dozens of Linux distributions are available for new Linux users. Of all these Linux distributions, only some really matter. In this book, I've focused on the three most important distributions: Red Hat, SUSE, and Ubuntu. By focusing on these three only, I am not making a statement about the quality of the other distributions; however, it makes sense to focus on these three as they make up more than 90% of the Linux market. Following are short descriptions of these three distributions.

Red Hat

North Carolina–based Linux distribution Red Hat had a major role in bringing Linux to the data center of many companies. The reason for the success of Red Hat was that this distributor added support for Linux. At one level, this is support of different hardware and software programs, which means that users of the supported hardware and software programs were guaranteed that they would work on Linux. Red Hat also added help for Linux users, available as a commercial added value to Linux.

Because Red Hat offered Linux with support, companies started putting aside their old flavors of UNIX and replacing them with the much cheaper Linux. This made Red Hat the most successful Linux distribution on the planet. Even though Ubuntu is widely adopted in environments where people don't want to pay for a supported Linux distribution, Red Hat by far is the leader of the commercial Linux distributions.

Currently, there are three product lines related to Red Hat. The most important of these is Red Hat Enterprise Linux (RHEL), which consists of two server versions and a desktop version. RHEL is a commercial product, so it is not available as a free download. It is open source software, however, but the only reason you can't download it for free is because Red Hat has added the Red Hat logo to the RHEL software, and this is something that users have to pay for.

Red Hat also founded the Fedora open source project. Basically, you can see this as the development environment for RHEL. Most new software components are first used and tested in Fedora, and if they are successful there, they will make it into RHEL as well. Fedora Linux is available for free download at `www.redhat.com/fedora`.

Since the only thing that is not free in Red Hat Enterprise Linux is the Red Hat logo, the CentOS (Community ENTerprise Operating System) distribution offers Red Hat Enterprise Linux software from which the Red Hat logo has been removed. This sounds illegal, but it isn't, as Red Hat is completely open source software. In fact, Red Hat has integrated the CentOS community within the Red Hat business, so that they can provide a freely available Linux distribution with the quality of Red Hat Enterprise Linux to those people and companies who aren't ready to pay for their Linux support yet. So if you want the stability of Red Hat Enterprise Linux, but don't want to pay for it, CentOS provides a good alternative. You can download CentOS at `www.centos.org`.

SUSE

The SUSE Linux distribution was founded in Germany. It became popular quite fast because from the beginning SUSE Linux came with lots of software packages that were shipped on DVDs in a large box, that contained installation manuals and even a stciker. SUSE was one of the first distributions that only sold their distribution and just delivered a demo system as freely available software, thus trying to make money out of it.

In 2004, Utah-based network software company Novell purchased SUSE and developed it into an enterprise-ready Linux distribution that could compete with Red Hat, which in that period still dominated the market. Currently, SUSE is an independent brand again.

Currently, there are two directions in SUSE Linux. SUSE Linux Enterprise is the commercial software that offers support, and it exists in two different flavors: SUSE Linux Enterprise Server, SUSE Linux Enterprise Desktop. With SUSE Linux Enterprise Desktop, SUSE has some success in bringing Linux to the enterprise desktop, whereas Red Hat, for example, still focuses on server versions. Interestingly, the SUSE Linux Enterprise products are freely downloadable at downloads.suse.com Notice however that in order to download updates and patches, a subscription is required. So even if you can download and install it for free, you'll need a subscription if you want to continue using it.

Apart from the SUSE Linux Enterprise products, there is OpenSUSE, which is a fully open source product. This version also offers a stable Linux distribution, but at the same time is used as a development platform for new software. You can download OpenSUSE at `www.opensuse.org`. All versions of SUSE Linux Enterprise Server are based on OpenSUSE.

Ubuntu

Ubuntu has become quite successful because its founder, the South African millionaire Mark Shuttleworth, made it an extremely user-friendly distribution and in the early days of Ubuntu even gave away CDs with the Ubuntu Desktop for free.

Apart from the Ubuntu Desktop, Ubuntu has a server edition as well, which due to the success of the desktop version has become quite. Both versions of Ubuntu Linux are available for free; customers who are interested in getting support can purchase it from Canonical, the company that has been created by Ubuntu founder Mark Shuttleworth to provide professional support services on Ubuntu.

Remarkable about Ubuntu Linux is the fact that there is a new software release every 6 months. By looking at the name of the distribution, you can see when it was released; for instance, Ubuntu 15.04 was released in April 2015. As enterprise users normally don't like upgrading their operating system every 6 months, there is also a Long Term Support (LTS) version that currently is released every 2 years. The special thing about this version is the extended period of support that is offered. For desktops and servers this is 7 years. For setting up a server that is based on Ubuntu Linux, it is highly recommended to use the LTS version.

■ **Note** This book focuses on Red Hat, SUSE, and Ubuntu Linux. You will notice, however, that 98% of the commands and configuration files covered in this book are available on other Linux distributions as well. This means that no matter what Linux distribution you use, the information in this book will be useful for you.

Logging In and Out

Now that you know about Linux in general, and about the Linux distributions more specifically, it's time to move on and get familiar with the essential tasks while working with Linux. Before you can do anything on a Linux computer, you have to log in. In this section, you'll learn about usernames and different ways you can use to make yourself known to your Linux computer.

Different Login Interfaces

Before starting to work on your Linux computer, you need to tell it who you are. The person that installed your Linux version has created a user account, and you must use this account and the associated password to make yourself known. To do this, Linux offers you a login prompt. This can be either a graphical or a nongraphical prompt. If you are working on a Linux desktop, you are likely to see a graphical environment. If, however, it is a server you a working from, you'll just see a shell login prompt.

In Linux, there often is a choice between different solutions. This means there is not just one unified graphical login prompt, but many, depending on the distribution that you are using and on the graphical environment that you have installed. You will notice that the graphical login screen for that reason will be different between the distributions. In Figure 1-1, you can see what it looks like on CentOS Linux.

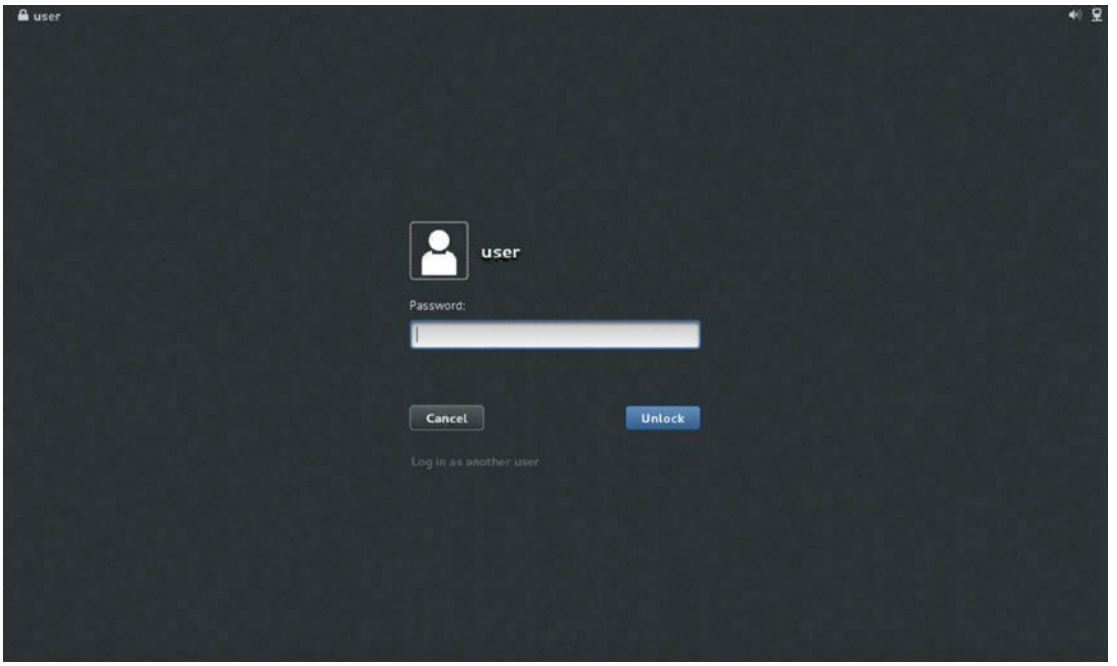

Figure 1-1. *The graphical login screen on CentOS*

When working with a graphical environment, it is the graphical environment that provides you with the login screen. More specifically, it is the `gdm` process that starts the graphical login screen. So what you see in Figure 1-1 is really the result of this `xdm` process. On specific graphical desktop environments, alternatives may be used, such as the gdm process which is the default on the GNOME graphical desktop.

Note: Historically, different graphical desktops have been offered on Linux. Currently, the GNOME desktop is used as the default on most Linux distributions. It runs on top of the X Windowing system, which is the generic system that provides a graphical screen, but with very basic functionality. GNOME adds functionality such as menu's to that screen. KDE is an alternative graphical desktop.

If you are working on a server, the graphical environment doesn't matter and is normally not started by default. That is because the graphical environment consumes resources, and these resources on server systems are better reserved for other purposes. Therefore, servers normally offer a text-based login prompt only. In Figure 1-2, you can see what this sort of login prompt looks like.

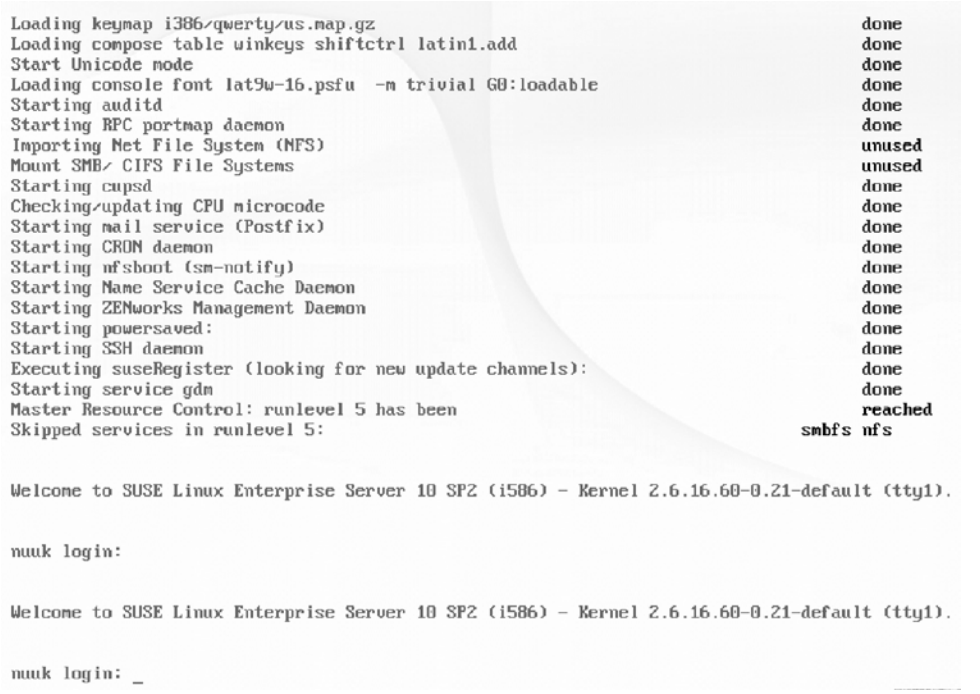

Figure 1-2. *The text-based login prompt*

There are other ways of connecting to a Linux machine as well. If you are a server administrator, your server will probably be installed in an air-conditioned and cold server room that you enter only if really necessary. Alternatively, your Linux server may be hosted somewhere remotely in the cloud, or as a Virtual Private Server (VPS) at somehosting provider. No matter how it is running, often your server isn't physically accessible. Therefore, as a server administrator, you may use a remote access tool like the Windows utility PuTTY to get shell access to the server. In Figure 1-3, you can see what the PuTTY login screen looks like.

Figure 1-3. *PuTTY is the de facto standard for accessing a Linux console remotely from a Windows workstation*

■ **Note** PuTTY is the de facto standard for accessing Linux machines from a Windows desktop. You can download PuTTY for free from www.putty.org. To use it, you need an SSH server running on your Linux computer as well. SSH is covered in detail in Chapter 11. On Linux workstations and Apple computers a native SSH client is integrated in the operating system, and you won't need PuTTY to establish a remote session with your Linux server.

As you can see, there are many ways to connect to a Linux machine. In all cases, you do need to provide user credentials. The next section gives more information about that.

Working with a User Account

To log in to a Linux machine, you need a user account name and a password. You should already know what username to use if you installed the machine yourself. If someone else installed the machine for you, ask him or her what username you should use. This username will also have a password. At the login prompt, you need to provide the username and password to make yourself known to the machine. This procedure is also known as *authentication*.

■ **Note** There are alternatives to passwords for authentication. For instance, you may use a smart card to authenticate on your machine. However, this requires additional hardware, and for this reason, in this book I will focus on password authentication.

When authenticating for the first time, you have to decide what user account to use. You can authenticate as a normal user, but you can authenticate with the account of the system administrator as well. The username for this account is root. On every Linux computer, there is a user with the name root, and this user account has no restrictions. The user root really is almighty. If you are connecting to Linux to perform system administration tasks, it makes sense to authenticate as root; after all, you need to do system administration, and for that purpose you need all the permissions there are. If, however, you are a normal user, you shouldn't make a habit of logging in as root by default. Just log in with your normal user account, and use `su` or `sudo` to become root when needed. In Chapter 2, you'll learn how to do this. At this time, just make sure that you are authenticated.

Command-Line Basics

The command line is important, because a system administrator can do anything from it. Even if your Linux installation is running a graphical interface as well, you'll need command line access to unleash the full power and potential of Linux. The only reason why many administrators are using a graphical interface on Linux, is because it allows them to run many terminal windows simultaneously, which makes it easy to read the documentation in one window, configure in another window and test the configuration in a third window for instance.

Linux has many, many commands, more than you will ever know, and new commands are added on a regular basis. All of these commands, though, share a common way of working. In this section, you'll learn about common elements that you will encounter in any Linux command. First, you'll learn about the common structure that every Linux command has. Next, we'll talk about characters that you can and can't use in Linux commands. Figure 1-4 shows what a command line looks like, when started as a terminal from a graphical environment.

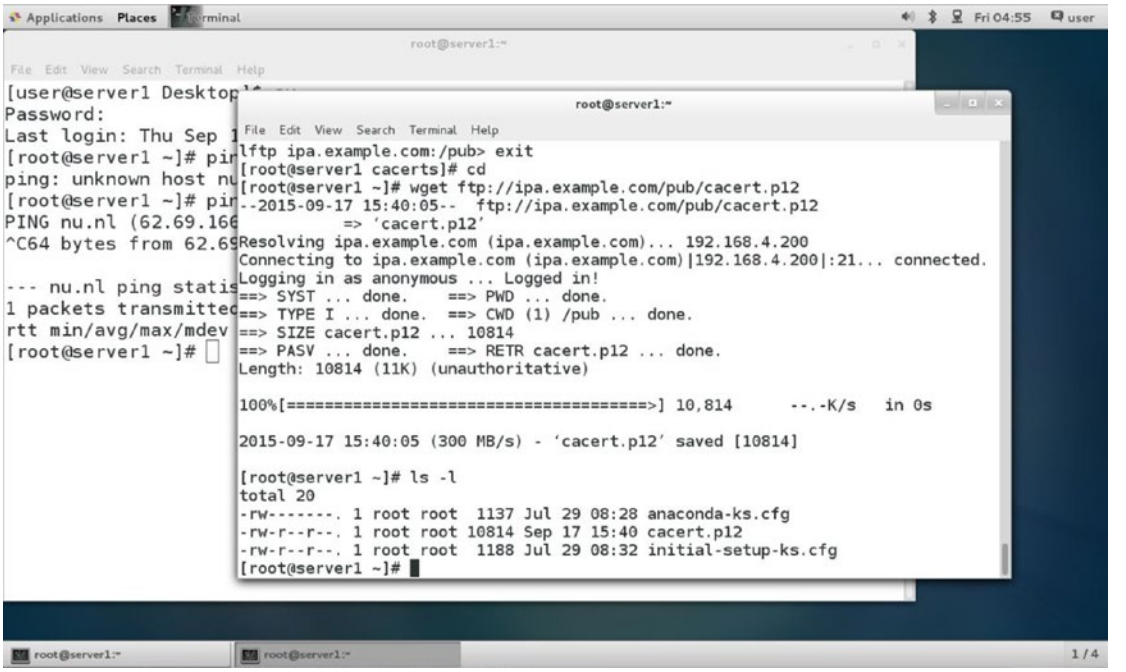

Figure 1-4. *The command line as seen from a terminal window*

The command line offers you a prompt that consists of different parts. The first part of the prompt, as you can see in Figure 1-4, is the name of the computer you are working on. In this case, the computer name is `server1`. Next, the prompt refers to the current directory in the file system where the user is at right now. In Figure 1-4, you can see a ~ sign instead of the name of a directory. This sign refers to the home directory, which is the folder in the file system where the user can store his or her personal files. Last, the # sign in this prompt indicates that the current user is *root*, the mighty system administrator. If you see anything other than #, the current user is not root, but a normal user with normal privileges. Be aware that if this is the case, some commands have limited use. For instance, on Linux a normal user cannot format hard disks, and many other tasks cannot be accomplished by non-root users.

■ **Note** Since in open source there are no rules that are strictly enforces to everyone, a developer can do as he or she likes. Therefore, words like "always" and "every" are not applicable in Linux, as there are often exceptions to the rules that are used in Linux. To keep this book readable, I will, however, use these words anyway. Just keep in mind that when you see the terms "every" and "always," it should read "almost every" and "almost always."

Bash: The Command Interpreter

When working on the command line, as an administrator you will be dealing with the shell. The shell is the command interpreter: it is responsible for making something out of the things that you type on the command line. How you work with commands is largely defined by the abilities of the shell. The shell itself is a program that your server starts automatically after you log in on your server, no matter if you've done so directly on the server console or via a remote session that you've started from PuTTY on your Windows workstation. Two shells are used quite often: Bash and Dash. Bash is the default shell on the current SUSE and Red Hat versions, and Dash is available on Ubuntu. The good news is that as a beginning command-line administrator, you don't really care which shell is used—both work in the same way. In the section "Working with the Shell" later in this chapter, you'll learn about some of its most important and most useful features.

Commands, Options, and Arguments

A Linux command normally consists of three parts: the command itself, the command options, and its arguments. For instance, the following example shows what a Linux command looks like:

```
useradd -m -G sales linda
```

This example consists of three parts, `useradd`, which is the command; `-m` and `-G` sales, which are both options; and `linda`, which is a generic argument. Notice that the word "sales" behind the -G is also a special item, it is an argument to the option -G. Further on in this section, I'll explain these components in more detail.

The command itself is the character string you type to activate a certain task. For instance, the command `ls` (see Listing 1-1) lists files. In Listing 1-1, you see the result of this command when used in the home directory of the user root (the Linux system administrator). Certain functionality is defined for this command. Linux has many commands, as mentioned previously; later in this chapter, in the section "Using `man` to Get Help," you'll learn how to get detailed usage information about them by using the `man` command.

Listing 1-1. Using the `ls` Command Without Options Shows Files in the Current Directory

```
nuuk:~ # ls
.ICEauthority  .exrc   .gnome2_                private         .metacity
.Xauthority    .fvwm   .gnupg                  .nautilus       .wapi
.bash_history  .gconf  .gstreamer-0.10         .qt             .xsession-errors
.config        .gconfd .gtkrc                  .recently-used  Desktop
.dmrc          .gnome  .gtkrc-1.2-gnome2       .skel           Documents
.esd_auth      .gnome2 .kbd                    .suse_          register.log bin
```

Options

Most commands have options as their second part. By using these options, you modify the behavior of the commands. Options are a part of the program code, they are fixed and the software programmer that created the program has provided the available options. For instance, the `ls` command just shows the names of files in the current directory, as you can see in Listing 1-1. If you want to see details, such as the file size, the permissions that are set for it, and information about the creation date, you can add the option `-l`. In Listing 1-2, you can see how this option modifies the behavior of the `ls` command.

Listing 1-2. By Adding an Option to a Command, You Modify Its Behavior

```
nuuk:~ # ls    -l
total 120
-rw-------    1    root    root    777    Dec    5    10:43    .ICEauthority
-rw-------    1    root    root    115    Dec    5    10:43    .Xauthority
-rw-------    1    root    root    2558   Nov    24   13:39    .bash_history
drwx------    3    root    root    4096   Nov    7    11:04    .config
-rw-------    1    root    root    24     Nov    7    11:03    .dmrc
-rw-------    1    root    root    16     Nov    7    11:03    .esd_auth
-rw-r--r--    1    root    root    1332   Nov    23    2005    .exrc
drwxr-xr-x    2    root    root    4096   Nov    7    10:47    .fvwm
drwx------    5    root    root    4096   Dec    5    10:43    .gconf
drwx------    2    root    root    4096   Dec    5    11:03    .gconfd
drwxr-xr-x    3    root    root    4096   Nov    7    11:04    .gnome
drwx------    6    root    root    4096   Nov    7    11:04    .gnome2
drwx------    2    root    root    4096   Nov    7    11:03    .gnome2_private
drwx------    3    root    root    4096   Nov    7    11:03    .gnupg
drwxr-xr-x    2    root    root    4096   Dec    5    10:43    .gstreamer-0.10
-rw-r--r--    1    root    root    123    Nov    7    11:03    .gtkrc
-rw-r--r--    1    root    root    134    Nov    7    11:03    .gtkrc-1.2-gnome2
drwxr-xr-x    2    root    root    4096   Nov    7    10:47    .kbd
drwx------    3    root    root    4096   Nov    7    11:03    .metacity
drwxr-xr-x    3    root    root    4096   Nov    7    11:04    .nautilus
drwxr-xr-x    2    root    root    4096   Nov    19   15:03    .qt
-rw-------    1    root    root    325    Dec    5    10:43    .recently-used
drwxr-xr-x    2    root    root    4096   Nov    7    11:03    .skel
-rw-r--r--    1    root    root    795    Dec    5    10:44    .suse_register.log
drwx------    3    root    root    4096   Nov    7    11:04    .thumbnails
drwxr-xr-x    2    root    root    4096   Dec    1    05:27    .wapi
-rw-r--r--    1    root    root    1238   Dec    5    10:43    .xsession-errors
drwxr-xr-x    2    root    root    4096   Nov    7    11:04     Desktop
drwx------    2    root    root    4096   Nov    7    11:04     Documents
drwxr-xr-x    2    root    root    4096   May    3    2007      bin
```

Options provide a method that is defined within the command code to modify the behavior of the command. This means that as a user or an administrator, you cannot add options yourself. The only way of doing this is to change the source code of the command and recomplie the command, which is not something that you want to be doing as a Linux user or administrator. Options are very specific to the command you use. Some commands don't have any options, and other commands can have more than 50. The `man` command normally gives you a complete list of all options that are available. Alternatively, many commands support the option --help, which will show a summary of all available options.

Many commands offer two different methods of working with options: the short option and the long option. For example, you can use the command `ls -lh`, which makes the `ls` command present its output in a human-readable way by showing kilobytes, megabytes, and gigabytes instead of just bytes. You can also use the short option `-h` in a long way, written as `--human-readable`. In Listing 1-3, you can see this option at work, combined with the option `-l`, which makes sure that the output of `ls` is given as a long listing. (Unfortunately, there is no long alternative for the short option `-l`.) Notice that if more than one option is used, all options can be specified together, without a - in front of each option. So the command **ls -lh** is valid, and so is the command **ls -l -h.**

Listing 1-3. Most Linux Commands Work with Short As Well As Long Options

```
nuuk:/somedir # ls -l -h
total 4.0M
-rwxr-xr-x 1 root root 1.5M Dec 5 11:32 vmlinux-2.6.16.60-0.21-default.gz
-rw-r--r-- 1 root root 1.3M Dec 5 11:32 vmlinuz
-rw-r--r-- 1 root root 1.3M Dec 5 11:32 vmlinuz-2.6.16.60-0.21-default
nuuk:/somedir # ls -l --human-readable
total 4.0M
-rwxr-xr-x 1 root root 1.5M Dec 5 11:32 vmlinux-2.6.16.60-0.21-default.gz
-rw-r--r-- 1 root root 1.3M Dec 5 11:32 vmlinuz
-rw-r--r-- 1 root root 1.3M Dec 5 11:32 vmlinuz-2.6.16.60-0.21-default
```

Short options are preceded by a sign, and you can add more than one short option after the sign. For instance, you can combine the options `-l` and `-h` from the example in Listing 1-4 as `ls -lh`. Long options are preceded by the `--` sign. For instance, `ls --human-readable` executes the `ls` command with just one option, which `Is --human-readable`. If by mistake you put just one in front of a long option, the long option is not interpreted as a long option, but as a collection of short options. This means that `ls -human-readable` would be interpreted as `ls -h -u -m -a -n -- -r -e -a -d -a -b -l -e`.

Arguments

Apart from options, many Linux commands have arguments. These are additional specifications that you can add to the command to tell it more precisely what to do, but the argument is typically not defined in the command code itself. For example, consider the command `ls -l /etc/hosts`:

```
nuuk:/somedir # ls -l /etc/hosts
-rw-r--r-- 1 root root 683 Nov 7 10:53 /etc/hosts
```

In this example, `/etc/hosts` is the argument. As you can imagine, you can use any other file name instead of `/etc/hosts`, and this is typical for arguments. They are not fixed, and you can use any argument you like as long as it is relevant in the context of the command. In this book, I'll make a very clear distinction between options and arguments.

You should be aware that not only commands have arguments, but also options have arguments as well. For example, consider the following command:

```
mail -s hello root
```

This command consists of four different parts:

- `mail`: The command itself
- `-s`: The option that tells the `mail` command what subject it should use
- `hello`: The argument of the option `-s`, which specifies what exactly you want to do with the option `-s`
- `root`: The argument of the command, which in this case makes clear to whom to send the mail message

As a rule of thumb, arguments at the end of the command are normally command arguments, and arguments for options are placed right next to the options. You may wonder now how to find out the differences between command arguments and arguments for options, but later in this chapter in the section "Getting Help," you'll see that it is fairly simple to differentiate the two argument types.

Piping and Redirection

To unleash the full power of Linux's many commands, you can use piping and redirection. By piping, you can send the result of a command to another command, and by using redirection, you can determine where the command should send its results.

Piping

Piping offers you great benefits in a Linux environment. By using piping, you can combine the abilities of two or more commands to create a kind of super command that offers even more capabilities. By creating the right pipes, you can really do amazing stuff. For an advanced Linux administrator, a command such as the following is pretty common (after reading all the chapters in this book, you should be able to understand what this command is doing):

```
kill `ps aux | grep y2 | grep -v grep | awk '{ print $2 }'`
```

As a Linux administrator, you absolutely need to know about piping, so let's start with an easy example. If you try a command like `ls -R /`, you will see that it gives a lot of output that scrolls over your screen without stopping. On Linux, there is a very useful command, named `less`, that you can use as a viewer for text files. For example, try `less /etc/hosts` (see Listing 1-4); this will open the `/etc/hosts` file in `less` to show the contents of the file (use `q` to quit `less`).

■ **Note** The `/etc/hosts` file contains a list of IP addresses and the matching host name. In a small network, you can use it as an alternative to using DNS for resolving host names.
