Beginning the Linux Command Line

Sander van Vugt

Apress®

Beginning the Linux Command Line

Copyright © 2009 by Sander van Vugt

ISBN-13 (pbk): 978-1-4302-1889-0

ISBN-13 (electronic): 978-1-4302-1890-6

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Michelle Lowman
Technical Reviewer: Mary Ann C. Tan
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Beth Christmas
Copy Editor: Ami Knox
Associate Production Director: Kari Brooks-Copony
Production Editor: Elizabeth Berry
Compositor: Linda Weidemann, Wolf Creek Publishing Services
Proofreader: Nancy Sixsmith
Indexer: Brenda Miller, Odessa&Cie
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Contents at a Glance

Contents

About the Author

SANDER VAN VUGT is an independent trainer and consultant who lives in the Netherlands and works in the extended EMEA (Europe, Middle East, and Africa) area. Sander has been a speaker at major Linux conferences worldwide, such as LinuxWorld in San Francisco and Linux.conf.au in Australia. He specializes in Linux high availability, storage solutions, and performance problems, and has successfully implemented Linux clusters across the globe. Sander has written several books about Linux-related subjects, including *The Definitive Guide to SUSE Linux Enterprise Server* (Apress, 2006), *Beginning Ubuntu Server Administration* (Apress, 2008), and *Pro Ubuntu Server Administration* (Apress, 2008).

Sander's articles can be found on several international web sites and in magazines such as *SearchEnterpriseLinux.com*, *Linux Journal*, and *Linux Magazine*. He works as a volunteer for the Linux Professional Institute (LPI), contributing topics for different certification levels. Most important, Sander is the father of Alex and Franck, and is the loving husband of Florence. For more information, consult Sander's web site: `www.sandervanvugt.com`. Sander can be reached by e-mail at `mail@sandervanvugt.com`.

About the Technical Reviewer

MARY ANN C. TAN has experience in many fields, including slinging regular expressions, watching Linux servers, writing telecom billing systems, being an obsessive-compulsive spreadsheet user, and arguing about machine learning. She is learning Italian, has forgotten most of her Mandarin, trains cats using Cat-Kwan-Do, and sings videoke to survive the Manila night. She currently does GUI development for a telecom testing company as her day job.

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 OpenSUSE. I've chosen to cover installation of just one distribution, because if you don't have any Linux installed yet, you probably don't care what you install. If you do care what distribution to work with, you probably have it installed already.

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.

There are exercises available for this book as well, which you can download from `www.sandervanvugt.com/exercises`. These exercises provide an excellent solution for learning Linux in a classroom environment.

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. For instance, exactly what Linux are we talking about? 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.

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 decide not to do it alone. To find other people who wanted to work with him, he posted a message on

Usenet, a major 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.

Open Source

Right from the start, Torvalds released his software as open source software—that is, software whose computer code is freely available to anyone. 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.

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.

Linux Turning Mainstream

The years between 1993 and 1998 marked the rise of the Linux operating system. One of the most important reasons for this is that it provided a very affordable alternative for the expensive UNIX operating system that was used on many mission-critical server systems. 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 hundreds 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 important Linux distributor for many years; only recently have SUSE and Ubuntu posed a threat to the commercial success of Red Hat in the enterprise environment.

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. 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. 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, there are two directions in SUSE Linux. SUSE Linux Enterprise is the commercial software that offers support, and it exists in three different flavors: SUSE Linux Enterprise

Server, SUSE Linux Enterprise Desktop, and SUSE Linux Enterprise Real Time, a tuned version that allows financial companies to process real-time transactions. With SUSE Linux Enterprise Desktop, Novell 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 (with the exception of SUSE Linux Enterprise Real Time) are freely downloadable at `www.novell.com/downloads`.

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`.

Ubuntu

Ubuntu is a relatively new Linux distribution. It has become quite successful because its founder, the South African millionaire Mark Shuttleworth, made it an extremely user-friendly distribution and even gives away CDs with the Ubuntu Desktop for free. Anyone can order as many CDs of this distribution he or she likes from `www.ubuntu.com`.

Apart from the Ubuntu Desktop, Ubuntu has a server edition as well, which due to the success of the desktop version has become quite popular (though not yet as popular as the Red Hat and SUSE server versions). Both versions of Ubuntu Linux are available for free; customers who are interested in getting support can purchase it from Canonical, a separate company that specializes in Ubuntu support.

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 9.04 was released in April 2009. 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 18 months. The special thing about this version is the extended period of support that is offered. For desktops this is 5 years, and for servers its 7 years.

■ **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

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. To help you with 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 SUSE Linux.

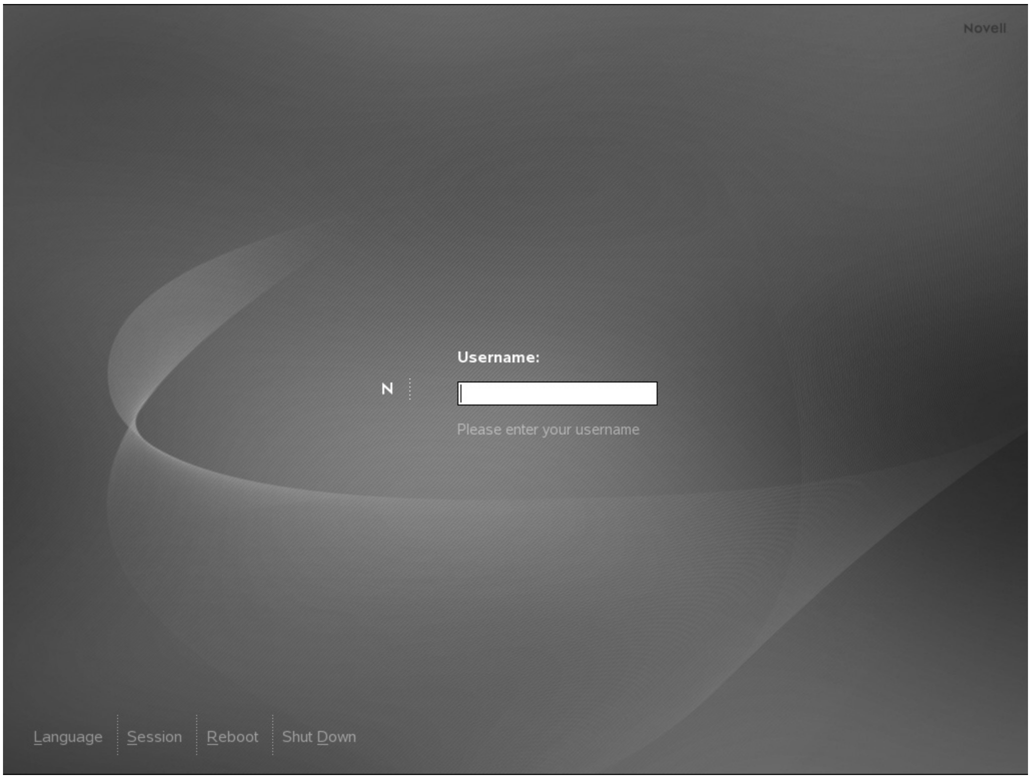
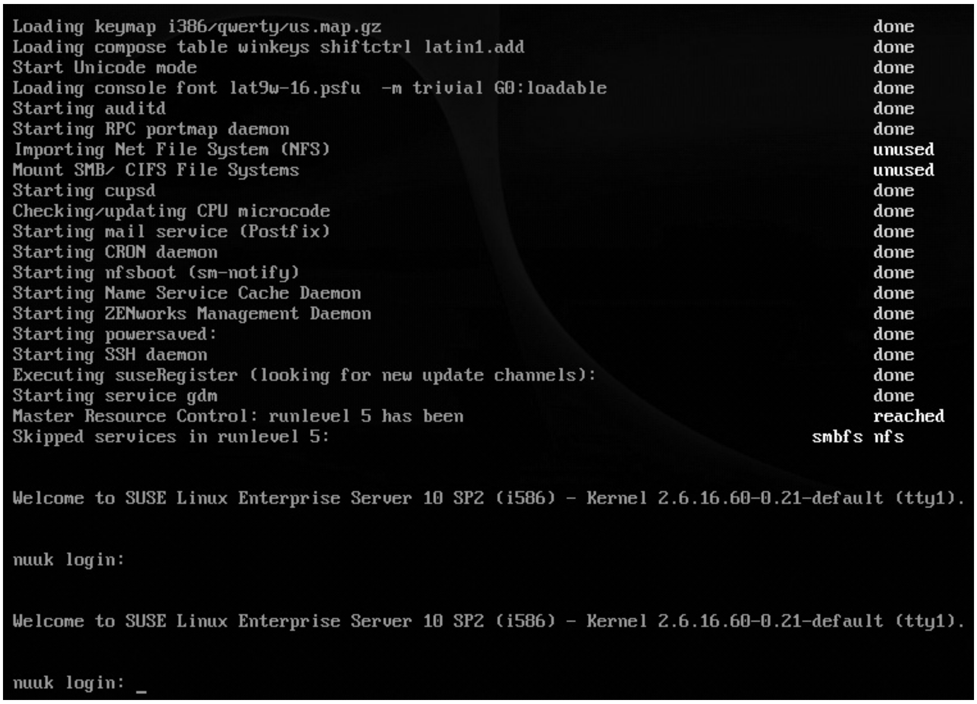


Figure 1-1. *The graphical login screen on SUSE*

When working with a graphical environment, it is the graphical environment that provides you with the login screen. More specifically, it is the `xdm` process that starts the graphical login screen. So what you see in Figure 1-1 is really the result of this `xdm` process.

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.

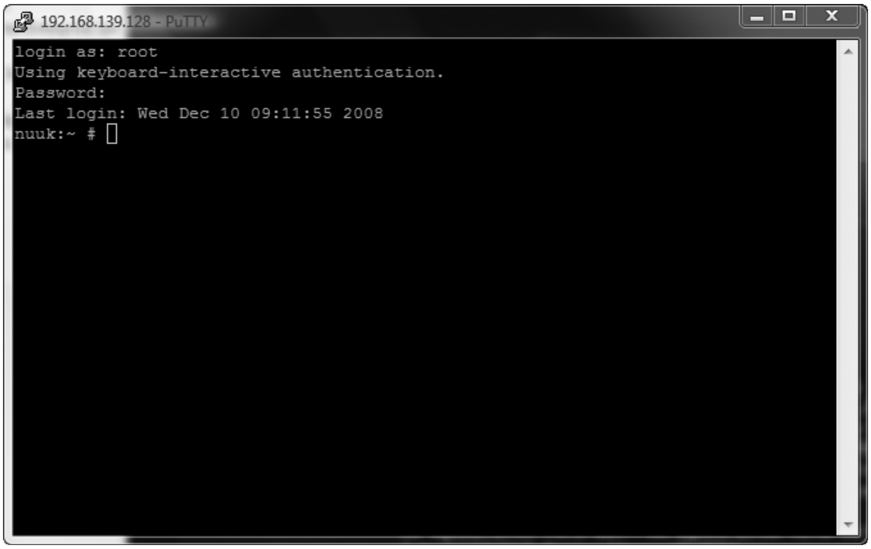


```
Loading keymap i386/qwerty/us.map.gz                                     done
Loading compose table winkeys shiftctrl latin1.add                       done
Start Unicode mode                                                       done
Loading console font lat9w-16.psfu  -m trivial G0:loadable               done
Starting auditd                                                          done
Starting RPC portmap daemon                                              done
Importing Net File System (NFS)                                          unused
Mount SMB/ CIFS File Systems                                             unused
Starting cupsd                                                           done
Checking/updating CPU microcode                                          done
Starting mail service (Postfix)                                          done
Starting CRON daemon                                                     done
Starting nfsboot (sm-notify)                                             done
Starting Name Service Cache Daemon                                       done
Starting ZENworks Management Daemon                                      done
Starting powersaved:                                                     done
Starting SSH daemon                                                      done
Executing suseRegister (looking for new update channels):                done
Starting service gdm                                                     done
Master Resource Control: runlevel 5 has been                             reached
Skipped services in runlevel 5:                                    smbfs nfs


Welcome to SUSE Linux Enterprise Server 10 SP2 (i586) - Kernel 2.6.16.60-0.21-default (tty1).


nuuk login:


Welcome to SUSE Linux Enterprise Server 10 SP2 (i586) - Kernel 2.6.16.60-0.21-default (tty1).


nuuk login: _
```

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 cold server room that you enter only if really necessary. Therefore, as a server administrator, you may use a remote access tool like PuTTY to get shell access to the server. In Figure 1-3, you can see what the PuTTY login screen looks like.

■ **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 SSH on your Linux computer as well. SSH is covered in detail in Chapter 11.



```
login as: root
Using keyboard-interactive authentication.
Password:
Last login: Wed Dec 10 09:11:55 2008
nuuk:~ # 
```

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

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

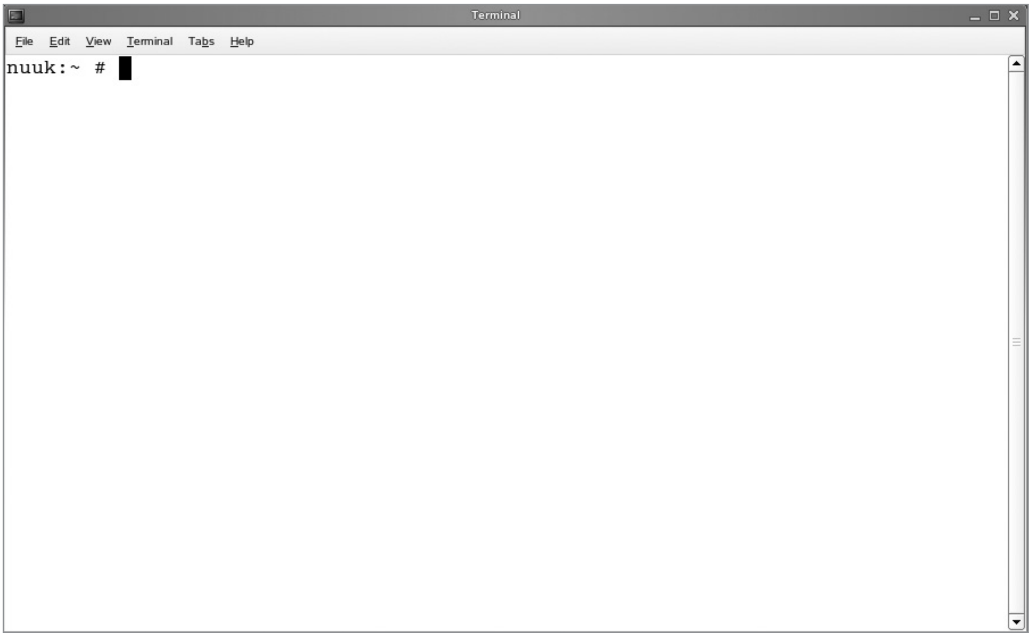
To log in to a Linux machine, you need a user account name and a password. You should already know what username to use if you installed the machine yourself. If someone else installed the machine for you, ask him or her what username you should use. This username will also have a password. At the login 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 is not very common, 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 a system administrator, 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. 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.



```
nuuk:~ #
```

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 `nuuk`. 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.

Note Since in open source there are no rules, 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."

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 the default shell 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. 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. 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 you 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. 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.

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`.)

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.

Listing 1-4. *You Can Use `less` As a Viewer to Read Text Files*

```
nuuk:/ # less /etc/hosts
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1       localhost

# special IPv6 addresses
::1             localhost ipv6-localhost ipv6-loopback

fe00::0         ipv6-localnet

ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes
ff02::2         ipv6-allrouters
ff02::3         ipv6-allhosts
127.0.0.2       nuuk.sander.gl nuuk
/etc/hosts lines 1-23/23 (END)
```

The `less` command can be very useful in a pipe as well. By using piping, you'll send the result of the first command to the second command. So if you use `ls -R / | less`, the `ls -R /` command executes and sends its result to the `less` command. `less` will function as a pager in this situation and show you the output of the first command screen by screen (see Listing 1-5). It will also show you the current position that you are at; this is indicated by `lines 1-23`, which you see at the end of the example file. Press the spacebar to proceed to the next screen of output.

Listing 1-5. *By Piping to `less`, You Can Display the Results of a Command That Gives a Large Amount of Output Screen by Screen*

```
nuuk:/ # ls -R / | less
/:
.rnd
bin
boot
dev
etc
home
lib
lost+found
media
mnt
opt
proc
root
sbin
somedir
srv
sys
tmp
usr
var

/bin:
lines 1-23
```

Redirection

Another operator that is very useful in the Linux command shell is the redirection operator, `>`. By default, a command will show its result on your computer monitor. In Linux slang, you can also say that the shell will send the result of a command to standard output, abbreviated to STDOUT, which is usually your computer monitor. Using redirection, you can send it somewhere else.

■ **Tip** If possible, try all commands described in this section immediately after reading about them. Without trying them yourself, it may be quite hard to understand what they are doing.

Let's use the command `ls -l` once more as an example. If you just type the command, you will see its result on STDOUT. However, if you type `ls -l > somewhere`, you'll tell the command to send its output somewhere else, in this case to a file that has the name `somewhere`. This file will be created in the current directory if it doesn't exist. If a file with this name already exists, you will overwrite it by using this command. In case you want to add to an existing file instead of overwriting it, use `ls -l >> somewhere`. The double redirector tells the command to append to the contents of the file instead of overwriting it. If the file doesn't already exist, the command will create it. So if you want to be sure never to overwrite an existing file by accident when using redirection, use `>>` at all times instead of `>`.

Some commands give you error messages apart from output. The good thing is that you can redirect these error messages also. To do this, use `2>` instead of `>`. So if `ls -l` gives you a lot of error messages as well (which isn't very likely, but you never know), you can send all of them to the file `errors`, which will be created in the current directory if you use `ls -l 2> errors`. And it is even possible to redirect the standard output of a command in one direction, while sending the error output somewhere else. For instance, the command `ls -l > output 2> errors` will create two files, the file `output` for the regular output and the file `errors` for the error output.

Instead of sending the results of a command to a file, you can redirect to some of the Linux special devices as well. Every piece of hardware in Linux can be addressed by using a device file. For instance, there is the device file `/dev/null`, which can be used as a digital waste bin. Everything that you send to `/dev/null` will immediately disappear into thin air. So if you just don't want to see any error messages at all, instead of saving them somewhere on your system, you can redirect the error messages to the `/dev/null` device. The following example shows how to do so:

```
ls -l 2> /dev/null
```

In this example, the regular output is still written to your current terminal, but you just won't see error messages anymore.

Apart from output, you can also use redirection on input for a command. This is used not as often, but can be useful for commands that open an interactive prompt where you are expected to provide input for the command. An example of this is the Linux `mail` command that you can use on the command line.

■ **Tip** You can use the `mail` command for some simple mail handling from a terminal screen, but if your server is configured properly, you can even use it to send mail to other users on the Internet. The only thing you need to do is set up DNS on your server.

Consider the command `mail -s hello root`. This command opens a command prompt that will allow you to compose a mail message to the user root (whose name is provided as the argument to the command). The option `-s hello` specifies the subject, in this case `hello`. In Listing 1-6, you can see the result of this command.

Listing 1-6. *Composing a Mail Message with* `mail`

```
nuuk:/ # mail -s hello root
Hi root, how are you.
.
EOT
```