- Table of Contents
- Index
- Reviews
- Reader Reviews
- Errata

Linux in a Nutshell, 4th Edition
By

Stephen Figgins, Ellen Siever, Aaron Weber

Publisher : O'Reilly
Pub Date : June 2003
ISBN : 0-596-00482-6

Pages : 944

Slots : 1

Comprehensive but concise, *Linux in a*

Nutshell is an essential desktop reference for

the commands that users of Linux utilize every day. It

covers all substantial user, programming, administration,

and networking commands for the most common Linux

distributions. It's several quick references rolled into

one: sed, gawk, RCS, CVS, vi, Emacs, bash, tcsh, regular

expressions, package management, bootloaders, and desktop

environments are all covered in this clear, to-the-point

volume, along with core command-line utilities.

- Table of Contents
- Index
- Reviews
- Reader Reviews
- Errata

Linux in a Nutshell, 4th Edition

By

Stephen Figgins, Ellen Siever, Aaron Weber

Publisher : O'Reilly
Pub Date : June 2003
ISBN : 0-596-00482-6
Pages : 944

Slots : 1

Copyright
Preface
Other Resources
Conventions

We'd Like to Hear from You

Acknowledgments

Chapter 1.

Introduction

Section 1.1.

The Excitement of Linux

Section 1.2.

Distribution and Support

Section 1.3.

Commands on Linux

Section 1.4.

What This Book Offers

Section 1.5.

Sources and Licenses

Section 1.6.

Beginner's Guide

Chapter 2.

System and Network Administration Overview

Section 2.1.

Common Commands

Section 2.2.

Overview of Networking

Section 2.3.

Overview of TCP/IP

Section 2.4.

Overview of Firewalls and Masquerading

Section 2.5.

Overview of NFS

Section 2.6.

Overview of NIS

Section 2.7.

Administering NIS

Section 2.8.

RPC and XDR

Chapter 3.

Linux Commands

Section 3.1.

Alphabetical Summary of Commands

Chapter 4.

Boot Methods

Section 4.1.

The Boot Process

Section 4.2.

LILO: The Linux Loader

Section 4.3.

GRUB: The Grand Unified Bootloader

Section 4.4.

GRUB Commands

Section 4.5.

Loadlin: Booting from MS-DOS

Section 4.6.

Dual-Booting Linux and Windows NT/2000/XP

Section 4.7.

Boot-Time Kernel Options

Section 4.8.

initrd: Using a RAM Disk

Chapter 5.

Red Hat and Debian Package Managers

Section 5.1.

The Red Hat Package Manager

Section 5.2.

The Debian Package Manager

Chapter 6.

The Linux Shells: An Overview

Section 6.1.

Purpose of the Shell

Section 6.2.

Shell Flavors

Section 6.3.

Common Features

Section 6.4.

Differing Features

Chapter 7.

bash: The Bourne-Again Shell

Section 7.1.

Invoking the Shell

Section 7.2.

Syntax

Section 7.3.

Variables

Section 7.4.

Arithmetic Expressions

Section 7.5.

Command History

Section 7.6.

Job Control

Section 7.7.

Built-in Commands

Chapter 8.

tcsh: An Extended C Shell

Section 8.1.

Overview of Features

Section 8.2.

Invoking the Shell

Section 8.3.

Syntax

Section 8.4.

Variables

Section 8.5.

Expressions

Section 8.6.

Command History

Section 8.7.

Command-Line Manipulation

Section 8.8.

Job Control

Section 8.9.

Built-in Commands

Chapter 9.

Pattern Matching

Section 9.1.

Filenames Versus Patterns

Section 9.2.

Metacharacters, Listed by Linux Program

Section 9.3.

Metacharacters

Section 9.4.

Examples of Searching

Chapter 10.

The Emacs Editor

Section 10.1.

Emacs Concepts

Section 10.2.

Typical Problems

Section 10.3.

Notes on the Tables

Section 10.4.

Summary of Commands by Group

Section 10.5.

Summary of Commands by Key

Section 10.6.

Summary of Commands by Name

Chapter 11.

The vi Editor

Section 11.1.

Review of vi Operations

Section 11.2.

vi Command-Line Options

Section 11.3.

ex Command-Line Options

Section 11.4.

Movement Commands

Section 11.5.

Edit Commands

Section 11.6.

Saving and Exiting

Section 11.7.

Accessing Multiple Files

Section 11.8.

Window Commands

Section 11.9.

Interacting with the Shell

Section 11.10.

Macros

Section 11.11.

Miscellaneous Commands

Section 11.12.

Alphabetical List of Keys in Command Mode

Section 11.13.

Syntax of ex Commands

Section 11.14.

Alphabetical Summary of ex Commands

Section 11.15.

vi Configuration

Chapter 12.

The sed Editor

Section 12.1.

Command-Line Syntax

Section 12.2.

Syntax of sed Commands

Section 12.3.

Group Summary of sed Commands

Section 12.4.

Alphabetical Summary of sed Commands

Chapter 13.

The gawk Scripting Language

Section 13.1.

Command-Line Syntax

Section 13.2.

Patterns and Procedures

Section 13.3.

gawk System Variables

Section 13.4.

PROCINFO Array

Section 13.5.

Operators

Section 13.6.

Variable and Array Assignments

Section 13.7.

Group Listing of gawk Commands

Section 13.8.

Alphabetical Summary of Commands

Chapter 14.

RCS

Section 14.1.

Overview of RCS Commands

Section 14.2.

Basic RCS Operations

Section 14.3.

General RCS Specifications

Section 14.4.

Alphabetical Summary of RCS Commands

Chapter 15.

CVS

Section 15.1.

Basic Concepts

Section 15.2.

CVS Command Format

Section 15.3.

Common Global Options

Section 15.4.

Gotchas

Section 15.5.

CVS Administrator Reference

Section 15.6.

CVS User Reference

Chapter 16.

Graphical Desktop Overview

Section 16.1.

Desktop Environments and Window Managers

Section 16.2.

Desktop Differences: Development

Chapter 17.

GNOME

Section 17.1.

Desktop Overview

Section 17.2.

The Panel

Section 17.3.

The GNOME Menu and the Menu Panel Menus

Section 17.4.

The GNOME Control Center

Section 17.5.

History and Changes in GNOME 2

Chapter 18.

KDE

Section 18.1.

Desktop Overview

Section 18.2.

The Panel

Section 18.3.

The KDE Control Center

Chapter 19.

An Alternative Window Manager: fvwm2

Section 19.1.

Running fvwm2

Section 19.2.

Implementing Window Manager Customizations

Section 19.3.

Adding Keyboard Shortcuts

Section 19.4.

Customizing Menus

Section 19.5.

The WinList: Switching the Focus

Colophon
Index

Copyright © 2003, 2000, 1999, 1997 O'Reilly & Associates, Inc.

Printed in the United States of America.

Published by O'Reilly & Associates, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly & Associates books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://safari.oreilly.com). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly & Associates, Inc. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly & Associates, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps. The association between the image of an Arabian horse and the topic of Linux is a trademark of O'Reilly & Associates, Inc.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Preface

This is a book about

Linux, a freely available clone of the Unix operating system whose uses range from embedded systems and personal data assistants (PDAs) to corporate servers, web servers, and massive clusters that perform some of the world's most difficult computations.

Whether you are using Linux for personal software projects, for a small office or home office (the so-called SOHO environment), to provide services to a small group of colleagues, or to administer a site responsible for millions of email and web connections each day, you need quick access to information on a wide range of tools. This book covers all aspects of administering and making effective use of Linux systems. Among its topics are booting, package management, and the configuration of the GNOME and KDE desktops. But foremost in *Linux in a Nutshell* are the immeasurable utilities and commands that make Linux one of the most powerful and flexible systems available.

In addition to the tools and features written specifically for it, Linux has inherited many from the Free Software Foundation's GNU project, the Berkeley Software Distribution (BSD), the X Window System (XFree86), and contributions from major corporations as well as the companies that created the major Linux distributions. More recent projects extend Linux in exciting ways, some through changes to the kernel and some through libraries and applications that radically change the user's experience; the GNOME and KDE desktops are the most prominent examples.

This book is a quick reference for the basic commands and features of the Linux operating system. As with other books in O'Reilly's "In a

Nutshell" series, this book is geared toward users who know what they want to do and have some idea how to do it, but can't always remember the correct command or option.

Other Resources

This book doesn't tell you how to install and come up to speed on a Linux system. For that, you will probably want O'Reilly's *Learning Red*

Hat Linux, which contains a Linux distribution on CD-ROM

and provides help with installation and configuration. Alternatively, *Running Linux* is an in-depth guide suitable for all major distributions. For networking information, check out the *Linux Network Administrator's*

Guide. In addition to these and other Linux titles,

O'Reilly's wide range of Unix, X,

Perl, and Java titles may also be of interest to the Linux user.

Online Documentation

The Internet is full of information about Linux. One of the best resources is the Linux Documentation Project at http://www.tldp.org (or one of the dozens of mirror sites around the world), which has numerous short guides called HOWTOs, along with some full manuals. For online information about the GNU utilities covered in this book, consult http://www.gnu.org (also widely mirrored).

The Free Software Foundation, which is in charge of the GNU project, publishes its documentation in a number of hard-copy and online books about various tools.

Each distribution maintains its own web site, and contains documentation for the software it provides as well as guides to maintaining your system under that distribution.

Web Sites

As befits a hot phenomenon, Linux is the central subject of several web sites and a frequent topic of

discussion on others. Some sites offer original content; others just have links to articles posted elsewhere and threaded discussions (which can be a useful service). Among the sites frequented by Linux users are:

http://www.lwn.net

Linux Weekly News, a site with weekly in-depth articles and frequent news updates

http://www.linuxgazette.com

Linux Gazette, a site published monthly by *Linux Journal* with articles and tips in many languages

http://linuxsecurity.com

Linux Security, a collection of security-related news

http://slashdot.org

Slashdot, a famous discussion list

http://linuxinsider.com

Linux Insider, a news feed

http://linuxtoday.com

Linux Today, another news feed

http://newsforge.com

NewsForge, a more general computing-related news feed

Linux Journal and Linux Magazine

Linux

Journal

and *Linux Magazine* are monthly magazines for the Linux community, written and published by a number of Linux

activists. With both print editions and web sites, they offer

articles ranging from novice questions and answers to kernel

programming internals. *Linux Journal*, at http://www.linuxjournal.com, is the older magazine and is published by S.S.C. Incorporated,

http://www.ssc.com.

Linux Magazine is at http://www.linuxmagazine.com.

Usenet Newsgroups

Most people can receive Usenet news at work or through their ISPs. While this communications technology has lost ground in the past several years to web-based threaded discussions, it is still a valuable source of help and community connections on many topics. The following Linux-related

newsgroups are popular:

comp.os.linux.announce

> A moderated newsgroup containing announcements of new software, distributions, bug reports, and goings-on in the Linux community. All Linux users should read this group. Submissions may be mailed to linux-announce@news.ornl.gov.

comp.os.linux.help

> General questions and answers about installing or using Linux.

comp.os.linux.admin

> Discussions relating to systems administration under Linux.

comp.os.linux.development

> Discussions about developing the Linux kernel and the system itself.

comp.os.linux.networking

> Discussions relating to networking with Linux.

comp.os.linux.security

> Help with firewalls, securing servers, and other security issues.

comp.os.linux.x

> Help on getting the X graphical window system to work. This list used to see some of the highest traffic of any Linux group back when distributions had more trouble setting up graphics automatically.
>
> This is no longer the case, thanks to the increasing sophistication of autodetection and configuration software.

There are also several newsgroups devoted to Linux in languages other than English, such as *fr.comp.os.linux* in French and *de.comp.os.linux* in German.

Online Linux Support

There are many ways of obtaining help online, where volunteers from around the world offer expertise and services to assist users with questions and problems.

The freenode IRC service is an Internet relay chat network devoted to so-called

"peer-directed" projects,

particularly those involving free software. Some of its channels are designed to provide online Linux support services.

Internet relay chat is a network service that allows you to talk interactively on the Internet to other users.

IRC networks support multiple channels where different groups of people type their thoughts. Whatever you type in a channel is seen by all other users of that channel.

There are a number of active channels on the

freenode IRC network where you will find users 24 hours a day, 7 days a week who are willing and able to help you solve any Linux problems you may have, or just chat. You can use this service by installing an IRC client (some distributions install them by default), connecting to server name

irc.freenode.org:6667, and joining a channel focusing on Linux, such as:

`#linpeople`	General help and discussion.
`#debian`	Help for Debian distribution.

`#gentoo`	Help for Gentoo distribution.
`#redhat`	Help for Red Hat distribution.
`#suse`	Help for SuSE distribution.

And so on. Please be sure to read up on the rules of chat etiquette before chatting. In particular, the participants in these groups tend to expect people to read documentation and do some experimentation before asking for help with a problem.

Linux User Groups

Many Linux User Groups around the world offer direct support to users. Typically, Linux User Groups engage in such activities as installation days, talks and seminars,

demonstration nights, and purely social events. Linux User Groups are a great way of meeting other Linux users in your area. There are a number of published lists of Linux User Groups. Some of the

better-known ones are:

- Groups of Linux Users Everywhere: http://www.ssc.com/glue/groups

- LUGregistry: http://www.linux.org/users

Constant Width Bold

is used in examples to show commands or other text that should be typed literally by the user.

Constant Width Italic

is used in examples to show text that should be replaced with user-supplied values.

%, $

are used in some examples as the tcsh shell prompt (%) and as the Bourne or bash shell prompt ($).

[ ]

surround optional elements in a description of syntax. (The brackets themselves should never be typed.) Note that many commands show the argument [files]. If a filename is omitted, standard input (e.g., the keyboard) is assumed. End with an end-of-file character.

EOF

indicates the end-of-file character (normally Ctrl-D).

|

is used in syntax descriptions to separate items for which only one alternative may be chosen at a time.

This icon indicates a note, which is an important aside to its nearby text.

This icon indicates a warning.

A final word about syntax. In many cases, the space between an option and its argument can be omitted. In other cases, the spacing (or lack of spacing) must be followed strictly. For example, -wn (no intervening space) might be interpreted differently from -w n. It's important to notice the spacing used in option syntax.

We'd Like to Hear from You

We have tested and verified all of the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing:

O'Reilly & Associates, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (in the United States or Canada)

(707) 829-0515 (international or local)

(707) 829-0104 (fax)

There is a web page for this book, which lists errata, examples, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/linuxnut4/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about books, conferences, Resource Centers, and the O'Reilly Network, see the O'Reilly web site at:

http://www.oreilly.com

Acknowledgments

This fourth edition of *Linux in a Nutshell* is the result of the cooperative efforts of many people. Thanks to Andy Oram for his editorial skills, as well as for pitching in to check existing chapters and update and write new material as needed.

For technical review, thanks go to Matt Welsh of *Running Linux* and *Installation and Getting Started Guide* fame, Michael K. Johnson of Red Hat Software, Robert J. Chassell, Phil Hughes of *Linux Journal*, Laurie Lynne Tucker, Arnold Robbins, Julian T. J. Midgley, Matthias Kalle Dalheimer, Terry Dawson, Doug Moreen, Ron Passerini, and Mark Stone.

Chapter 1. Introduction

It is hard to chart the rise of Linux over its twelve years of existence without risking the appearance of exaggeration and hyperbole. During the past five years alone, Linux has grown from a student/hacker playground to an upstart challenger in the server market to a well-respected system taking its rightful place in educational and corporate networks. Many serious analysts claim that its trajectory has just begun, and that it will eventually become the world's most widespread operating system.

Linux was first developed by Linus Torvalds at the University of Helsinki in Finland. From his current location in Silicon Valley, Linus continues to centrally coordinate improvements. The Linux kernel continues to develop under the dedicated cultivation of a host of other programmers and hackers all over the world, joined by members of programming teams at major computer companies, all connected through the Internet.

By

"kernel,"

we mean the core of the operating system itself, not the applications (such as the compiler, shells, and so forth) that run on it. Today, the term

"Linux"

is often used to mean a software environment with a Linux kernel along with a large set of applications and other software components.

In this larger meaning, many people prefer the term GNU/Linux, which acknowledges the central role played by tools from the Free Software Foundation's GNU project in the development of the kernel.

Linux systems cannot be technically referred to as a "version of Unix," as they have not undergone the required tests and licensing. [1] However, Linux offers all the common programming interfaces of standard Unix systems, and as you can see from this book, all the common Unix utilities have been reimplemented on Linux. It is a powerful, robust, fully usable system for those who like Unix.

[1] Before an operating system can be called "Unix," it must be branded

by The Open Group.

The historical impact of Linux goes beyond its role as a challenge to all versions of Unix as well as Microsoft Windows, particularly on servers. Linux's success has also inspired countless other free software or open source (defined at http://opensource.org) projects, including Samba, GNOME, and a mind-boggling collection of innovative projects that you can browse at numerous sites like SourceForge (http://sourceforge.net). As both a platform for other developers and a development model, Linux gave a tremendous boost to the GNU project, and has also become a popular platform for Java development. In short, Linux is a central participant in the most exciting and productive free software movement ever seen.

If you haven't obtained Linux yet, or have it but don't know exactly how to get started using it, see Other Resources in the preface.

1.1 The Excitement of Linux

Linux is, first of all, free software: anyone can download the source from the Internet or buy it on a low-cost CD-ROM. But Linux is becoming well known because it's more than free

softwareit's unusually good software. You can get more from your hardware with Linux and be assured of fewer crashes; even its security is better than many commercial alternatives.

Linux first appeared in organizations as ad hoc installations by hackers running modest web servers or development systems at universities and research institutions, but now extends deeply into corporations around the world. People deploying Linux for mission-critical systems tend to talk about its ample practical advantages, such as the ability to deliver a lot of bang for the buck and the ease of deploying other powerful tools on Linux such as Apache, Samba, and Java environments. They also cite Linux's ability to grow and sprout new features of interest to large numbers of users. But these advantages can be traced back to the concept of software freedom, which is the root of the broad wave of innovation driving Linux.

As free software, Linux revives the grand creativity and the community of sharing that Unix was long known for.

The unprecedented flexibility and openness of Unixwhich newcomers usually found confusing and frustrating, but eventually found they couldn't live withoutcontinually

inspired extensions, new tools like Perl, and experiments in computer science that sometimes ended up in mainstream commercial computer systems.

Many programmers fondly remember the days when AT&T provided universities with Unix source code at no charge, and the University of Berkeley started distributing its version in any manner that allowed people to get it. For these older hackers, Linux brings back the spirit of working togetherall the more so because the Internet is now so widespread. And for the many who are too young to remember the first round of open systems or whose prior experience has been constricted by trying to explore and adapt proprietary operating systems, now is the time to discover the wonders of freely distributable source code and infinitely adaptable interfaces.

The economic power behind Linux's popularity is its support for an enormous range of hardware. People who are accustomed to MS-DOS and Microsoft Windows are often amazed at how much faster their hardware appears to work with Linuxit makes efficient use of its resources.

For the first several years, users were attracted to Linux for a variety of financial and political reasons, but soon they discovered an unexpected benefit: Linux works better than many commercial systems. With the Samba file and print server, for instance, Linux serves a large number of end-user PCs without crashing. With the Apache web server, it provides more of the useful features web administrators want than competing products do. Embedded versions of the Linux kernel are in growing use because, although they are larger than the most stripped-down operating systems, they deliver a range of powerful features within a remarkably small footprint.

Opinions still differ on how suitable Linux is as a general-purpose desktop system. But the tremendous advances in usability and stability of the desktop software and its applications are undisputed. Soon (if not today), one will find Linux in many offices and other end-user environments. Meanwhile, the strides made by Linux in everyday computing tasks are reflected in the new audio and CD-related commands found in this edition.

1.2 Distribution and Support

While it is convenient to download one or two new programs over the Internet and fairly feasible to download something as large as the Linux kernel, getting an entire working system over the Internet is difficult without a high-speed Internet connection. Over the years, therefore, commercial and noncommercial packages called

distributions

have emerged. The first distribution consisted of approximately 50

diskettes, at least one of which would usually turn out to be bad and have to be replaced. When CD-ROM drives became widespread, Linux really took off.

After getting Linux, the average user is concerned next with support.

While Usenet newsgroups offer very quick responses and meet the needs of many intrepid users, you can also buy support from the vendors of the major distributions and a number of independent experts. Linux is supported at least as well as commercial software. When you

buy a distribution from a vendor, you typically are entitled to a period of free support as well.

Intel's x86 family and other compatible chips are still by far the most common hardware running Linux, but Linux is also now commercially available on a number of other hardware systems, notably the PowerPC, the 64-bit Intel Itanium processor, Sun Microsystems' SPARC, and the Alpha (created by Digital Equipment Corporation).

1.3 Commands on Linux

Linux commands are not the same as standard Unix ones.

They're better! This is because most of them are

provided by the GNU project run by the Free Software Foundation (FSF). GNU means "GNU's Not

Unix"the first word of the phrase is expanded

with infinite recursion.

Benefiting from years of experience with standard Unix utilities and advances in computer science, programmers on the GNU project have managed to create versions of standard tools that have more features, run faster and more efficiently, and lack the bugs and inconsistencies that persist in the original standard versions.

While GNU provided the programming utilities and standard commands like **grep**, many of the system and network administration tools on Linux came from the Berkeley Software Distribution (BSD). In addition, some people wrote tools that specifically allow Linux to deal with special issues such as filesystems. This book documents all the standard Unix commands that are commonly available on most Linux distributions.

The third type of software most commonly run on Linux is the X Window System, ported by the XFree86 project to standard Intel chips. While this book cannot cover the wide range of utilities that run on X, we briefly cover some of the useful customizations you can make to your KDE, GNOME, or **fvwm2** desktop.

1.4 What This Book Offers

Originally based on the classic O'Reilly & Associates quick reference, *Unix in a Nutshell*, this book has been expanded to include much information that is specific to Linux. These enhancements include chapters on:

- Package managers (which make it easy to install, update, and remove related software files)
- The KDE and GNOME desktops and the **fvwm2** window manager
- Boot parameters
- The CVS version control system

The book also contains dozens of Linux-specific commands, along with tried-and-true Unix commands that have been supporting users for decades (though they continue to sprout new options).

This book does not cover the graphical tools contained in most distributions of Linux. Many of these, to be sure, are quite useful and can form the basis of everyday work. Examples of these tools include OpenOffice (the open source version of the StarOffice suite distributed by Sun Microsystems), Evolution (a mail, calendar, and office productivity tool from Ximian), Mozilla (the open source cousin of the Netscape web browser), and the GIMP (a graphic image

manipulation program and the inspiration for the GNOME project). But they are not Linux-specific, and their graphical models do not fit well into the format of this book.

While you can do a lot of valuable work with the graphical applications, the core of Linux use is the text manipulation and administration done from the command line, within scripts, or using text editors such as **vi** and Emacs.

Linux is still mostly a command-driven system, and this book continues to focus on this level of usage. In your day-to-day work, you'll likely find yourself moving back and forth

between graphical programs and the commands listed in this book.

Every distribution of Linux is slightly different, but you'll find that the commands we document are the

ones you use most of the time, and that they work the same on all distributions. Basic commands, programming utilities, system administration, and network administration are all covered. However, some areas were so big that we had to leave them out. The many applications that depend on the X Window System

didn't make the cut. Nor did the many useful

programming languages like Java, Perl, and Python with which users can vastly expand the capabilities of their systems. XML

isn't covered here either. These subjects would

stretch the book out of its binding.

Linux in a Nutshell doesn't teach you Linuxit is, after all, a quick referencebut

novices as well as highly experienced users will find it of great value. When you have some idea of what command you want but aren't sure just how it works or what combinations

of options give you the exact output required, this book is the place to turn. It can also be an eye-opener, making you aware of options that you never knew about before.

Once you're over the hurdle of installing Linux, the first thing you need to do is get to know the common utilities run from the shell prompt. If you know absolutely nothing about Unix, we recommend you read a basic guide (introductory chapters in the O'Reilly books *Learning Red Hat Linux* and *Running Linux* can get you started.) This chapter and Chapter 2 offer a context for understanding different kinds of commands (including commands for programming, system administration, and network administration). Chapter 3 is the central focus of the book, containing about one half its bulk.

The small chapters immediately following Chapter 3

help you get your system set up. Since most users do not want to completely abandon other operating systems (whether a Microsoft Windows system, OS/2, or some Unix flavor), Linux often resides on the same computer as other systems. Users can then boot the system they need for a particular job. Chapter 5

describes the commonly used booting options on Intel systems, including LILO (Linux Loader), GRUB (the GRand Unified Bootloader), and Loadlin. Chapter 5 covers the Red Hat package manager (**rpm**)which is

supported by many distributions, including Red Hat, SuSE, Mandrake, and Calderaand the Debian package manager. Package managers are useful for installing and updating software; they make sure you have all the files you need in the proper versions.

All commands are interpreted by the *shell*. The shell is simply a program that accepts commands from the user and executes them. Different shells sometimes use slightly different syntax to mean the same thing. Under Linux, two popular shells are **bash** and **tcsh** (which on Linux has supplanted the older **csh**), and they differ in

subtle ways. (One of the nice things about Linux and other Unix systems is that you have a variety of shells to choose from, each with strengths and weaknesses.) We offer an introduction to shells in Chapter 6, thorough coverage of **bash** in Chapter 7, and a guide to **tcsh** in Chapter 8. You may decide to read these after you've used Linux for a while, because they mostly

cover powerful, advanced features that you'll want

when you're a steady user.

To get any real work done, you'll have to learn some big, comprehensive utilities, notably an editor and some scripting tools. Two major editors are used on Linux: **vi** and Emacs. Emacs is covered in Chapter 10 and **vi** in Chapter 11. Chapter 12

and Chapter 13 cover two classic Unix tools for manipulating text files on a line-by-line basis: **sed** and **gawk** (the GNU version of the traditional **awk**). O'Reilly offers separate books about these topics that you may find valuable, as they are not completely intuitive upon first use. (Emacs does have an excellent built-in tutorial, though; to invoke it, press **Ctrl-H** followed by **t** for "tutorial.")

CVS (Concurrent Versions System) and RCS (Revision Control System) manage files so you can retrieve old versions and maintain different versions simultaneously. Originally used by programmers who have complicated requirements for building and maintaining

applications, these tools have turned out to be valuable for anyone who maintains files of any type, particularly when coordinating a team of people.

CVS has become a distribution channel for thousands of free software projects. Chapter 14 presents RCS commands, and Chapter 15 presents CVS commands.

Graphical desktops are covered in four chapters. Chapter 16 is a brief overview to the major options on Linux systems. It is followed by Chapter 17 on the GNOME desktop, Chapter 18 on the KDE desktop, and Chapter 19 on the **fvwm2** window manager.

Our goal in producing this book is to provide convenience, and that means keeping the book (relatively) small. It certainly

doesn't have everything the manual pages have; but

you'll find that it has what you need 95% of the

time.

1.5 Sources and Licenses

Some distributions contain the source code for Linux; it is also easily available for download at http://www.kernel.org and elsewhere. Source code is similarly available for all the utilities on Linux (unless your vendor offers a commercial application or library as a special enhancement). You may never bother looking at the source code, but it's key to Linux's strength. Under

the Linux license, the source code has to be provided by the vendor, and it permits those who are competent at such things to fix bugs, provide advice about the system's functioning, and submit improvements that benefit everyone. The license is the GNU

project's well-known General Public License, also known as the

GPL or "copyleft," invented and

popularized by the Free Software Foundation.

The FSF, founded by Richard Stallman, is a phenomenon that many people might believe to be impossible if it did not exist. (The same goes for Linux, in fact15 years ago, who would have imagined a robust operating system developed by collaborators over the Internet and made freely redistributable?) One of the most popular

editors on Unix, GNU Emacs, comes from the FSF. So do **gcc** and **g++**

(C and C++ compilers), which for a while set the standard in the industry for optimization and fast code. One of the largest projects within GNU is the GNOME desktop, which encompasses several useful general-purpose libraries and applications that use these libraries to provide consistent behavior and interoperability.

Dedicated to the sharing of software, the FSF provides all its code and documentation on the Internet and allows anyone with a whim for enhancements to alter the source code. One of its projects is the Debian distribution of Linux.

To prevent hoarding, the FSF requires that the source code for all enhancements be distributed under the same GPL that it uses. This encourages individuals or companies to make improvements and share them with others. The only thing someone cannot do is add enhancements and then try to sell the product as commercial softwarethat is, to withhold the source code. That would be taking advantage of the FSF and the users. You can find the GPL in any software covered by that license, or online at http://www.gnu.org/copyleft/gpl.html.

As we said earlier, many Linux tools come from BSD instead of GNU.

BSD is also free software. The license is significantly different, but that probably don't concern you as a user. The effect of the

difference is that companies are permitted to incorporate the software into their proprietary products, a practice that is severely limited by the GNU license.

1.6 Beginner's Guide

If you're just beginning to work on a Linux system, the abundance of commands might prove daunting. To help orient you, the following lists present a sampling of commands on various topics.

1.6.1 Communication

ftp	File Transfer Protocol.
login	Sign on.
rlogin	Sign on to remote system.
rsh	Run shell or single command on remote system (not particularly secure).
ssh	Run shell or single command on remote system (secure).
talk	Exchange messages interactively with other terminals.
telnet	Connect to another system.
tftp	Trivial File Transfer Protocol.

uudecode	Decode file prepared for mailing by **uuencode**.
uuencode	Encode file containing binary characters for mailing.
vacation	Respond to mail automatically.

1.6.2 Comparisons

cmp	Compare two files, byte by byte.
comm	Compare items in two sorted files.
diff	Compare two files, line by line.
diff3	Compare three files.

1.6.3 File Management

cat	Concatenate files or display them.
chfn	Change user information for finger, email, etc.

cksum	Compute checksum.
chmod	Change access modes on files.
chsh	Change login shell.
cp	Copy files.
csplit	Split a file into pieces with a specific size or at specific locations.
dd	Copy files in raw disk form.
file	Determine a file's type.
head	Show the first few lines of a file.
less	Display files by screenful.
ln	Create filename aliases.
ls	List files or directories.
merge	Merge changes from different files.

mkdir	Create a directory.
more	Display files by screenful.
mv	Move or rename files or directories.
newgrp	Change current group.
pwd	Print working directory.
rcp	Copy files to remote system.
rm	Remove files.
rmdir	Remove directories.
split	Split files evenly.
tail	Show the last few lines of a file.
wc	Count lines, words, and characters.

1.6.4 Media

cdda2wav	Rip a CD to create a computer-friendly WAV format.
cdparanoia	Rip a CD while providing extra features.
cdrdao	Copy a CD.
cdrecord	Record to a CD.
eject	Eject a removable disk or tape.
mpg321	Play an MP3 file.
volname	Provide the volume name of a CD-ROM.

1.6.5 Printing

lpq	Show status of print jobs.
lpr	Send to the printer.
lprm	Remove print job.
lpstat	Get printer status.

pr	Format and paginate for printing.

1.6.6 Programming

ar	Create and update library files.
as	Generate object file.
bison	Generate parsing tables.
cpp	Preprocess C code.
flex	Lexical analyzer.
g++	GNU C++ compiler.
gcc	GNU C compiler.
ld	Link editor.
m4	Macro processor.
make	Create programs.

ranlib	Regenerate archive symbol table.
rpcgen	Translate RPC to C code.
yacc	Generate parsing tables.

1.6.7 Program Maintenance

cvs	Manage different versions (revisions) of source files.
etags	Generate symbol list for use with the Emacs editor.
gdb	GNU debugger.
gprof	Display object file's profile data.
imake	Generate makefiles for use with **make**.
make	Maintain, update, and regenerate related programs and files.
nm	Display object file's symbol table.
patch	Apply patches to source code.

rcs	Manage different versions (revisions) of source files.
size	Print the size of an object file in bytes.
strace	Trace system calls and signals.
strip	Strip symbols from an object file.

1.6.8 Searching

apropos	Search manpages for topic.
egrep	Extended version of **grep**.
fgrep	Search files for literal words.
find	Search the system for files by name and take a range of possible actions.
grep	Search files for text patterns.
locate	Search a preexisting database to show where files are on the system.

strings	Search binary files for text patterns.
whereis	Find command.

1.6.9 Shell Programming

echo	Repeat command-line arguments on the output.
expr	Perform arithmetic and comparisons.
printf	Format and print command-line arguments.
sleep	Pause during processing.
test	Test a condition.

1.6.10 Storage

bzip2	Compress files to free up space.
cpio	Create and unpack file archives.

gunzip	Expand compressed (*.gz* and *.Z*) files.
gzip	Compress files to free up space.
shar	Create shell archive.
tar	Copy files to or restore files from an archive medium.
zcat	Display contents of compressed files.

1.6.11 System Status

at	Execute commands later.
atq	Show jobs queued by **at**.
atrm	Remove job queued by **at**.
chgrp	Change file group.
chown	Change file owner.
crontab	Automate commands.

date	Display or set date.
df	Show free disk space.
du	Show disk usage.
env	Show environment variables.
finger	Display information about users.
kill	Terminate a running command.
printenv	Show environment variables.
ps	Show processes.
stty	Set or display terminal settings.
who	Show who is logged in.

1.6.12 Text Processing

col	Process control characters.

cut	Select columns for display.
emacs	Work environment with powerful text editing capabilities.
ex	Line editor underlying **vi**.
expand	Convert tabs to spaces.
fmt	Produce roughly uniform line lengths.
fold	Break lines.
gawk	Process lines or records one by one.
ghostscript	Display PostScript or PDF file.
groff	Format **troff** input.
ispell	Interactively check spelling.
join	Merge different columns into a database.
paste	Merge columns or switch order.

rev	Print lines in reverse.
sed	Noninteractive text editor.
sort	Sort or merge files.
tac	Print lines in reverse.
tr	Translate (redefine) characters.
uniq	Find repeated or unique lines in a file.
vi	Visual text editor.
xargs	Process many arguments in manageable portions.

1.6.13 Miscellaneous

banner	Make posters from words.
bc	Arbitrary precision calculator.
cal	Display calendar.

clear	Clear the screen.
man	Get information on a command.
nice	Reduce a job's priority.
nohup	Preserve a running job after logging out.
passwd	Set your login password.
script	Produce a transcript of your login session.
su	Become a superuser.
tee	Simultaneously store output in file and send to screen.
which	Print pathname of a command.

Chapter 2. System and Network Administration Overview

Section 2.1. Common Commands

Section 2.2. Overview of Networking

Section 2.3. Overview of TCP/IP

Section 2.4. Overview of Firewalls and Masquerading

Section 2.5. Overview of NFS

Section 2.6. Overview of NIS

Section 2.7. Administering NIS

Section 2.8. RPC and XDR

2.1 Common Commands

Following are lists of commonly used system administration commands.

2.1.1 Clocks

hwclock	Manage hardware clock.
rdate	Get time from network time server.
zdump	Print list of time zones.
zic	Create time conversion information files.

2.1.2 Daemons

apmd	Advanced Power Management daemon.
atd	Queue commands for later execution.
bootpd	Internet Boot Protocol daemon.

fingerd	Finger daemon.
ftpd	File Transfer Protocol daemon.
identd	Identify user running TCP/IP process.
imapd	IMAP mailbox server daemon.
inetd	Internet services daemon.
klogd	Manage **syslogd**.
lpd	Printer daemon.
mountd	NFS mount request server.
named	Internet domain name server.
nfsd	NFS daemon.
pppd	Maintain Point-to-Point Protocol (PPP) network connections.
rdistd	Remote file distribution server.

rexecd	Remote execution server.
rlogind	**rlogin** server.
routed	Routing daemon.
rpc.rusersd	Remote users server.
rpc.statd	NFS status daemon.
rshd	Remote shell server.
rwhod	Remote who server.
sshd	Secure shell daemon.
syslogd	System logging daemon.
talkd	Talk daemon.
tcpd	TCP network daemon.
tftpd	Trivial File Transfer Protocol daemon.

update	Buffer flush daemon.
ypbind	NIS binder process.
yppasswdd	NIS password modification server.
ypserv	NIS server process.

2.1.3 Hardware

agetty	Start user session at terminal.
arp	Manage the ARP cache.
cardctl	Control PCMCIA cards.
cardmgr	PCMCIA card manager daemon.
cfdisk	Maintain disk partitions (graphical interface).
fdisk	Maintain disk partitions.
hdparm	Get and set hard drive parameters.

kbdrate	Manage the keyboard's repeat rate.
ramsize	Print information about RAM disk.
setkeycodes	Change keyboard scancode-to-keycode mappings.
setserial	Set serial port information.
slattach	Attach serial lines as network interfaces.

2.1.4 Host Information

arch	Print machine architecture.
dig	Query Internet domain name servers. Replaces **nslookup**.
dnsdomainname	Print DNS domain name.
domainname	Print NIS domain name.
free	Print memory usage.

host	Print host and zone information.
hostname	Print or set hostname.
nslookup	Query Internet domain name servers. (Deprecated)
uname	Print host information.

2.1.5 Installation

cpio	Copy file archives.
install	Copy files into locations providing user access and set permissions.
rdist	Distribute files to remote systems.
tar	Copy files to or restore files from an archive medium.

2.1.6 Mail

fetchmail	Retrieve mail from remote servers.
formail	Convert input to mail format.
mailq	Print a summary of the mail queue.
makemap	Update **sendmail**'s database maps.
rmail	Handle **uucp** mail.
sendmail	Send and receive mail.

2.1.7 Managing Filesystems

To Unix systems, a *filesystem* is a device (such as a partition) that is formatted to store files. Filesystems can be found on hard drives, floppies, CD-ROMs, or other storage media that permit random access.

The exact format and means by which the files are stored are not important; the system provides a common interface for all *filesystem types* that it recognizes. Under Linux, filesystem types include the Second Extended (ext2) Filesystem, which you probably use to store Linux files. This filesystem was developed primarily for Linux and supports 256-character filenames and 4-terabyte maximum filesystem size. (It is "second" because it is the successor to the extended filesystem type.) Other common filesystem types include the MS-DOS filesystem, which allows files on MS-DOS partitions

and floppies to be accessed under Linux, and the ISO 9660 filesystem used by CD-ROMs.

The 2.4 kernel adds optional support for an enhanced version of the ext2 filesystem, the Third Extended (ext3) Filesystem. Many Linux distributions ship kernels with this support preconfigured. The ext3 filesystem is essentially an ext2 filesystem with an added journal. Since it is in all other ways identical to the ext2 system, it is both forward and backward compatible with ext2all ext2 utilities work with ext3 filesystems. When the kernel interacts with an ext3 filesystem, writes to disk are first written to a log or journal before they are written to disk. This slows down writes to the filesystem, but reduces the risk of data corruption in the event of a power outage. It also speeds up reboots after a system unexpectedly loses power. When rebooting with an ext2 filesystem, the **fsck** utility scans the entire disk to ensure that all data blocks are listed as either used or free, that each data block is claimed by only one file or directory, and that all files are available in the directory tree. When rebooting an ext3 filesystem using a supported kernel, it need only check the journal and complete any tasks that were left uncompleted.

To change an ext2 filesystem into an ext3 filesystem, simply add a journal using the **tune2fs** utility with the **-j** option on the unmounted device. If the filesystem is listed in */etc/fstab*, change its specified filesystem from ext2 to ext3, or **auto** (**mountd** will automatically detect the right system).

You can specify the level of journaling the kernel should use in the mount options field (or using **mount -o**). There are three data options:

data=journal

> Log all filesystem data and metadata changes. All changes to the filesystem are written twice, once to the journal and once to the filesystem itself. This is the slowest but safest mode.

data=ordered

Log filesystem metadata. Flush data updates to disk before changing metadata. This is the default mode. This slows writes a small amount, but ensures the filesystem is always in sync with changes to its metadata.

data=writeback

Log filesystem metadata. Use the ext2 write process to write data changes. This is the fastest journaling mode. While it maintains filestructure integrity, the contents of files may contain old, stale data when the filesystem is restarted from an unexpected shutdown.

The last field in the */etc/fstab* entry specifies at what point in the boot process the filesystem should be verified with **fsck**. It should never need to be checked because journaling ensures filesystem integrity. You can set this field to **0**. You can also turn off the automatic check of the filesystem that normally occurs every 20th time the system is mounted using **tune2fs -i 0 -c 0**.

Although not covered in this edition of Linux in a Nutshell, Linux supports three other open source journaling filesystems: IBM's Journaled Filesystem (JFS), SGI's Extensible Filesystem (XFS), and the Naming System Venture's Reiser Filesystem (ReiserFS). In some situations these can be faster than ext3. Some Linux distributions use these alternative filesystems by default.

debugfs	Debug ext2 filesystem.
dosfsck	Check and repair a DOS or VFAT filesystem.
dumpe2fs	Print information about superblock and blocks group.

e2fsck	Check and repair an ext2 filesystem.
e2image	Store disaster recovery data for an ext2 filesystem.
fdformat	Format floppy disk.
fsck	Another name for **e2fsck**.
fsck.ext2	Check and repair an ext2 filesystem.
fsck.minix	Check and repair a MINIX filesystem.
fuser	List processes using a filesystem.
mke2fs	Make a new ext2 filesystem.
mkfs	Make new filesystem.
mkfs.ext2	Another name for **mke2fs**.
mkfs.ext3	Yet another name for **mke2fs**.
mkfs.minix	Make new MINIX filesystem.

mklost+found	Make *lost+found* directory.
mkraid	Set up a RAID device.
mkswap	Designate swap space.
mount	Mount a filesystem.
raidstart	Activate a RAID device.
raidstop	Turn off a RAID device.
rdev	Describe or change values for root filesystem.
resize2fs	Enlarge or shrink an ext2 filesystem.
rootflags	List or set flags to use in mounting root filesystem.
showmount	List exported directories.
swapoff	Cease using device for swapping.
swapon	Begin using device for swapping.

sync	Write filesystem buffers to disk.
tune2fs	Manage an ext2 filesystem.
umount	Unmount a filesystem.

2.1.8 Managing the Kernel

depmod	Create module dependency listing.
insmod	Install new kernel module.
lsmod	List kernel modules.
modinfo	Print kernel module information.
modprobe	Load new module and its dependent modules.
rmmod	Remove module.
sysctl	Examine or modify kernel parameters at runtime.

2.1.9 Networking

chat	Establish dial-up IP connections.
dip	Establish dial-up IP connections.
gdc	Administer **gated** routing daemon.
ifconfig	Manage network interfaces.
ipchains	Administer firewall facilities (2.2 kernel).
iptables	Administer firewall facilities (2.4 kernel).
named	Translate between domain names and IP addresses.
nameif	Assign names to network devices.
netstat	Print network status.
nfsstat	Print statistics for NFS and RPC.
nsupdate	Submit dynamic DNS update requests.
portmap	Map daemons to ports.

rarp	Manage RARP table.
route	Manage routing tables.
routed	Dynamically keep routing tables up to date.
rpcinfo	Report RPC information.
ruptime	Check how long remote system has been up.
rwho	Show who is logged into remote system.
traceroute	Trace network route to remote host.

2.1.10 Printing

checkpc	Examine and repair printer settings.
lpc	Control line printer.
tunelp	Tune the printer parameters.

2.1.11 Security and System Integrity

badblocks	Search for bad blocks.
chroot	Change root directory.

2.1.12 Starting and Stopping the System

bootpd	Internet Boot Protocol daemon.
bootpgw	Internet Boot Protocol gateway.
bootptest	Test **bootpd**.
halt	Stop or shut down system.
init	Change runlevel.
reboot	Shut down, then reboot system.
runlevel	Print system runlevel.
shutdown	Shut down system.
telinit	Change the current runlevel.

uptime	Display uptimes of local machines.

2.1.13 System Activity and Process Management

A number of additional commands in Chapter 3 are particularly useful in controlling processes, including **kill**, **killall**, **killall5**, **pidof**, **ps**, and **who**.

fuser	Identify processes using file or filesystem.
ipcrm	Remove interprocess communication message queue, semaphore array, or shared memory segment.
ipcs	Print interprocess communication information.
renice	Change the priority of running processes.
top	Show most CPU-intensive processes.
vmstat	Print virtual memory statistics and process statistics.

2.1.14 Users

chpasswd	Change multiple passwords.
groupadd	Add a new group.
groupdel	Delete a group
groupmod	Modify groups.
grpck	Check the integrity of group system files.
grpconv	Convert group file to shadow group file.
lastlog	Generate report of last user login times.
newusers	Add new users in a batch.
pwck	Check the integrity of password system files.
pwconv	Convert password file to shadow passwords.
rusers	Print **who**-style information on remote machines.
rwall	Print a message to remote users.

useradd	Add a new user.
userdel	Delete a user and that user's home directory.
usermod	Modify a user's information.
w	List logged-in users.
wall	Write to all users.
whoami	Show how you are currently logged in.

2.1.15 Miscellaneous

anacron	Schedule commands for periodic execution.
atrun	Schedule commands for later execution.
cron	Schedule commands for specific times.
dmesg	Print bootup messages after the system is up.
ldconfig	Update library links and do caching.

logger	Send messages to the system logger.
logrotate	Compress and rotate system logs.
run-parts	Run all scripts in a directory.

2.2 Overview of Networking

Networks connect computers so that the different systems can share information. For users and system administrators, Unix systems have traditionally provided a set of simple but valuable network services that let you check whether systems are running, refer to files residing on remote systems, communicate via electronic mail, and so on.

For most commands to work over a network, each system must be continuously running a server process in the background, silently waiting to handle the user's request. This kind of process is called a *daemon*. Common examples, on which you rely for the most basic functions of your Linux system, are **named** (which translates between numeric IP addresses and more human-readable alphanumeric names), **lpd** (which sends documents to a printer, possibly over a network), and **ftpd** (which allows you to connect to another machine via **ftp**).

Most Unix networking commands are based on Internet protocols. These are standardized ways of communicating across a network on hierarchical layers. The protocols range from addressing and packet routing at a relatively low layer to finding users and executing user commands at a higher layer.

The basic user commands that most systems support over Internet protocols are generally called TCP/IP commands, named after the two most common protocols. You can use all of these commands to communicate with other Unix systems besides Linux systems. Many can also be used to communicate with non-Unix systems, as a wide variety of systems support TCP/IP.

This section also covers NFS and NISwhich allow for transparent file and information sharing across networksand **sendmail**.

2.2.1 TCP/IP Administration

ftpd	Server for file transfers.
gated	Manage routing tables between networks.
host	Print host and zone information.
ifconfig	Configure network interface parameters.
named	Translate between domain names and IP addresses.
netstat	Print network status.
nslookup	Query domain name servers.
ping	Check that a remote host is online and responding.
pppd	Create PPP serial connection.
rdate	Notify time server that date has changed.
route	Manage routing tables.

routed	Dynamically keep routing tables up to date.
slattach	Attach serial lines as network interfaces.
sshd	Server for secure shell connections.
telnetd	Server for Telnet sessions from remote hosts.
tftpd	Server for restricted set of file transfers.

2.2.2 NFS and NIS Administration

domainname	Set or display name of current NIS domain.
makedbm	Rebuild NIS databases.
portmap	DARPA port to RPC program number mapper.
rpcinfo	Report RPC information.
ypbind	Connect to NIS server.
ypcat	Print values in NIS database.

ypinit	Build new NIS databases.
ypmatch	Print value of one or more NIS keys.
yppasswd	Change user password in NIS database.
yppasswdd	Update NIS database in response to **yppasswd**.
yppoll	Determine version of NIS map at NIS server.
yppush	Propagate NIS map.
ypserv	NIS server daemon.
ypset	Point **ypbind** at a specific server.
yptest	Check NIS configuration.
ypwhich	Display name of NIS server or map master.
ypxfr	Transfer NIS database from server to local host.

FE80:0000:0000:0000:0202:B3FF:FE1E:8329

FE80::202:B3FF:FE1E:8329

When IPv4 and IPv6 networks are mixed, the IPv4 address can be packed into the lower four bytes, yielding an address like 0:0:0:0:0:0:192.168.1.2, or ::192.168.1.2, or even ::C0A8:102.

Because improvements in IPv4, including CIDR, have relieved much of the pressure to migrate to IPv6, organizations have been slow to adopt IPv6. Some use it experimentally, but communication between organizations using IPv6 internally are still usually encapsulated inside IPv4 datagrams, and it will be a while before IPv6 becomes common.

If you wish to connect to the Internet, contact an Internet Service Provider (ISP) and have them assign you a network address or range of addresses. If you are not connecting to an outside network, you can choose your own network address as long as it conforms to the IP address syntax. You should use the special reserved addresses provided in RFC 1597, which lists IP network numbers for private networks

that don't have to be registered with the IANA (Internet Assigned Numbers Authority). An IP address is different from an Ethernet address, which is assigned by the manufacturer of the physical Ethernet card.

2.3.2 Gateways and Routing

Gateways are hosts responsible for exchanging routing information and forwarding data from one network to another. Each portion of a network that is under a separate local administration is called an *autonomous system* (AS). Autonomous systems connect to each other via exterior gateways. An AS also may contain its own system of networks, linked via interior gateways.

2.3.2.1 Gateway protocols

Gateway protocols include:

EGP (Exterior Gateway Protocol)
BGP (Border Gateway Protocol)

> Protocols for exterior gateways to exchange information

RIP (Routing Information Protocol)

Interior gateway protocol; most popular for LANs

Hello Protocol
OSPF (Open Shortest Path First)

Interior gateway protocols

2.3.2.2 Routing daemons

While most networks will use a dedicated router as a gateway, GNU Zebra and routed, the routing daemons, can be run on a host to make it function as a gateway. Only one of them can run on a host at any given time. Zebra is the gateway routing daemon that replaces the older gated routing daemon. It allows a host to function as both an exterior and interior gateway, and simplifies the routing configuration by combining the protocols RIP, Hello, BGP, EGP, and OSPF into a single package. We do not cover GNU Zebra in this book.

routed, a network routing daemon that uses RIP, allows a host to function as an interior gateway only, and manages the Internet routing tables. For more details on routed, see Chapter 3.

2.3.2.3 Routing tables

Routing tables provide information needed to route packets to their destinations. This information

includes destination network, gateway to use, route status, and number of packets transmitted. Routing tables can be displayed with the netstat command.

2.3.3 Name Service

Each host on a network has a name that points to information about that host. Hostnames can be assigned to any device that has an IP address. Name service translates the hostnames (which are easy for people to remember) to IP addresses (the numbers the computer deals with).

2.3.3.1 DNS and BIND

The Domain Name System (DNS) is a distributed database of information about hosts on a network. Its structure is similar to that of the Unix filesysteman inverted tree, with the root at the top. The branches of the tree are called domains (or subdomains) and correspond to IP addresses. The most popular implementation of DNS is the BIND (Berkeley Internet Name Domain) software.

DNS works as a client/server application. The resolver is the client, the software that asks questions about host information. The name server is the process that answers the questions. The server side of BIND is the named daemon. You can interactively query name servers for host information with the dig and host commands. See Chapter 3 for more details on named, dig, and host.

The name server of a domain is responsible for keeping (and providing on request) the names of the machines in its domain. Other name servers on the network forward requests for these machines to the nameserver.

2.3.3.2 Domain names

The full domain name is the sequence of names from the current domain back to the root, with a period separating the names. For instance, oreilly.com indicates the domain oreilly (for O'Reilly & Associates), which is under the domain com (for commercial). One machine under this domain is www.oreilly.com. Top-level domains include:

aero

Air-transport industry

biz

Commercial organizations

com

Commercial organizations

coop

Cooperatives

edu

United States educational organizations

gov

United States government organizations

info

Informative sites

int

International organizations

mil

United States military departments

museum

Museums

name

Names of individuals

net

Commercial Internet organizations, usually Internet service providers

org

Miscellaneous organizations

pro

Professionals, including accountants, lawyers, and physicians

Some domains (e.g., edu, gov, and mil) are sponsored by organizations that restrict their use; others (e.g., com, info, net, and org) are unrestricted. Countries also have their own two-letter top-level domains based on two-letter country codes. One special domain, arpa, is used for technical infrastructure purposes. The Internet Corporation for Assigned Names and Numbers (ICANN) oversees top-level domains and provides contact information for sponsored domains.

2.3.4 Configuring TCP/IP

Certain commands are normally run in the system's startup files to enable a system to connect to a network. These commands can also be run interactively.

2.3.4.1 ifconfig

The network interface represents the way that the networking software uses the hardwarethe driver, the IP address, and so forth. To configure a network interface, use the ifconfig command. With ifconfig, you can assign an address to a network interface, setting the netmask, broadcast address, and IP address at boot time. You can also set network interface parameters, including the use of ARP, the use of driver-dependent debugging code, the use of one-packet mode, and the address of the correspondent on the other end of a point-to-point link. For more information on ifconfig, see Chapter 3.

2.3.4.2 Serial-line communication

There are two protocols for serial-line communication: Serial Line IP (SLIP) and Point-to-Point Protocol (PPP). These protocols let computers transfer information using the serial port instead of a network card and a serial cable instead of an Ethernet cable. SLIP is rarely used anymore, having been replaced by PPP.

PPP was intended to remedy some of SLIP's failingsit can hold packets from non-Internet protocols, it implements client authorization and error detection/correction, and it dynamically configures each network protocol that passes through it. Under Linux, PPP exists as a driver in the kernel and as the daemon pppd. For more information on pppd, see Chapter 3.

2.3.5 Troubleshooting TCP/IP

The following commands can be used to troubleshoot TCP/IP. For more details on these commands, see Chapter 3.

ifconfig

: Provide information about the basic configuration of the network interface.

netstat

: Display network status.

ping

Indicate whether a remote host can be reached.

nslookup

Query the DNS name service.

traceroute

Trace route taken by packets to reach network host.

2.4 Overview of Firewalls and Masquerading

A firewall computer is a secure system that sits between an internal network and an external network (i.e., the Internet). It is configured with a set of rules that it uses to determine what traffic is allowed to pass and what traffic is barred. While a firewall is generally intended to protect the network from malicious or even accidentally harmful traffic from the outside, it can also be configured to monitor traffic leaving the network. As the sole entry point into the system, the firewall makes it easier to construct defenses and monitor activity.

The firewall can also be set up to present a single IP address to the outside world, even though multiple IP addresses may be used internally. This is known as

masquerading.

Masquerading can act as additional protection, hiding the very existence of a network. It also saves the trouble and expense of obtaining multiple IP addresses.

IP firewalling and masquerading are implemented in Linux Version 2.4

with

netfilter, also known as **iptables**.

In Linux Version 2.2, they are implemented with the **ipchains**

facility. The 2.0 kernels used a command called **ipfwadm**, which will not be covered here. The two newer commands are very similar, but differ in some of the organization of their rules. The firewalling facilities built into the 2.4 kernel are also designed to be extensibleif there is some function missing from the implementation, you can add it.

Most distributions come with firewall support already built into the kernel. If this is not the case in your distribution, you need to compile firewall support into the kernel by running **make config** with the 2.4

kernel and selecting the following network options:

- Network packet filtering (replaces **ipchains**)
- TCP/IP networking

Then select the following options from the IP: Netfilter Configuration menu:

- IP tables support (required for filtering/masq/NAT)

- Packet filtering

There are several extended target and matching rule modules you may wish to compile as well. The behavior of those extension modules is described under the **iptables** command. If you have an existing firewall designed for the 2.2 kernel or the 2.0 kernel, you can compile support for these older-style commands and use them with your new kernel instead of the newer **iptables** style of netfiltering.

For **ipchains** support in the 2.2 kernel, select these options:

- Network firewalls

- TCP/IP networking

- IP: firewalling

If you want to support a transparent proxy service on your firewall, select the following option:

- IP: transparent proxy support

If you want your firewall to support masquerading, select the following options as well:

- IP: masquerading
- IP: ICMP masquerading

The packet filtering facilities provide built-in rule sets. Each network packet is checked against each rule in the rule set until the packet either matches a rule or is not matched by any rule. These sets of rules are called

chains.

In the 2.4 kernel, these chains are organized into tables that separate filtering functions from masquerading and packet mangling functions. In either the 2.4 or 2.2 kernel, if a match is found, the counters on that rule are incremented and any target for that rule is applied. A target might accept, reject, or masquerade a packet, or even pass it along to another chain for processing. Details on the chains provided in both **iptables** and **ipchains** can be found under the description of the appropriate command.

In addition to these chains, you can create your own user-defined chains. You might want a special chain for your PPP interfaces or for packets from a particular site. To call a user-defined chain, you just make it the target for a match.

It is possible to make it through a chain without matching any rules that have a target. If no rule matches the packet in a user-defined chain, control returns to the chain from which it was called, and the next rule in that chain is checked. If no rule matches the packet in a built-in chain, a default policy for that chain is used. The default policy can be any of the special targets that determine what is done with a packet. The valid targets for each command are detailed in Chapter 3.

In the 2.4 kernel, you use the **iptables** command to define the rules. Once you have the rules defined you can use **iptables-save** to create a file with all the rule definitions, and **iptables-restore** to restore those definitions when you reboot. The equivalent 2.2 kernel commands are **ipchains**, **ipchains-save**, and **ipchains-restore**.

For more information on the kinds of decisions you need to make and the considerations that go into defining the rules, see a general book on firewalls such as *Building Internet Firewalls* (O'Reilly). For more details on **ipchains** or **iptables**, consult the *Linux Network Administrator's Guide*

(O'Reilly), or one of the relevant HOWTOs, such as the "Linux 2.4 Packet Filtering

HOWTO." These HOWTOs and a number of tutorials are available on the Netfilter web site at http://www.netfilter.org/.

/projects hostname1(rw) hostname2(ro)

mount servername:/projects /mnt/nfs/projects

A mount request calls the server's mountd daemon, which checks the access permissions of the client and returns a pointer to a filesystem. Once a directory is mounted, it remains attached to the local filesystem until it is unmounted with the umount command or until the local system is rebooted.

Usually, only a privileged user can mount filesystems with NFS. However, you can enable users to mount and unmount selected filesystems using the mount and umount commands if the user option is set in /etc/fstab. This can reduce traffic by having filesystems mounted only when needed. To enable user mounting, create an entry in /etc/fstab for each filesystem to be mounted.

2.6 Overview of NIS

The Network Information System (NIS) refers to the service formerly known as Sun Yellow Pages (YP). It is used to make configuration information consistent on all machines in a network. It does this by designating a single host as the master of all the system administration files and databases and distributing this information to all other hosts on the network. The information is compiled into databases called maps. NIS is built on the RPC protocol. There are currently two NIS servers freely available for Linux, **yps** and **ypserv**. We describe **ypserv** in this book.

2.6.1 Servers

In NIS, there are two types of serversmaster servers and slave servers. Master servers are responsible for maintaining the maps and distributing them to the slave servers. The files are then available locally to requesting processes.

2.6.2 Domains

An NIS domain is a group of hosts that use the same set of maps. The maps are contained in a subdirectory of */var/yp* having the same name as the domain. The machines in a domain share

password, host, and group file information. NIS domain names are set with the **domainname** command.

2.6.3 NIS Maps

NIS stores information in database files called

maps. Each map consists of a pair of **dbm** database files, one containing a directory of keys (a bitmap of indices) and the other containing data values.

The non-ASCII structure of **dbm** files necessitates using NIS tools such as **yppush** to move maps between machines.

The file */var/yp/YP_MAP_X_LATE* contains a complete listing of active NIS maps as well as NIS aliases for NIS

maps. All maps must be listed in this file in order for NIS to serve them.

2.6.4 Map Manipulation Utilities

The following utilities are used to administer NIS maps:

makedbm

Make **dbm** files. Modify only **ypserv**'s map and

any nondefault maps.

ypinit

Build and install NIS databases. Manipulate maps when NIS is being initialized. Should not be used when NIS is already running.

yppush

Transfer updated maps from the master server.

2.7 Administering NIS

NIS

is enabled by setting up NIS servers and NIS clients. The

descriptions given here describe NIS setup using **ypserv**, which does not support a master/slave

server configuration. All NIS commands depend on the RPC **portmap** program, so make sure it is installed

and running before setting up NIS.

2.7.1 Setting Up an NIS Server

Setting up an NIS server involves the

following steps:

1. Set a domain name for NIS using **domainname**
2. Edit the *ypMakefile*, which identifies which

   databases to build and what sources to use in building them
3. Copy the *ypMakefile* to

   /var/yp/Makefile
4. Run **make** from the

   /var/yp directory, which builds the databases

   and initializes the server
5. Start **ypserv**, the NIS server daemon

2.7.2 Setting Up an NIS Client

Setting up an NIS client involves only the

following steps:

1. Set the domain name for NIS using **domainname**, which should be the same name used

   by the NIS server

2. Run **ypbind**

2.7.3 NIS User Accounts

NIS

networks have two kinds of user accounts: distributed and local.

Distributed accounts must be administered from the master machine;

they provide information that is uniform on each machine in an NIS

domain. Changes made to distributed accounts are distributed via NIS

maps. Local accounts are administered from the local computer; they

provide account information unique to a specific machine. They are

not affected by NIS maps, and changes made to local accounts do not

affect NIS. When NIS is installed, preexisting accounts default to

local accounts.

2.8 RPC and XDR

RPC (Remote Procedure

Call) is the session protocol used by both NFS and NIS. It allows a

host to make a procedure call that appears to be local but is really

executed remotely on another machine on the network. RPC is

implemented as a library of procedures, plus a network standard for

ordering bytes and data structures called

XDR (eXternal Data

Representation).

Chapter 3. Linux Commands

This chapter presents the Linux user, programmer, and system administration commands. These are entered into a shell at the console or on a virtual terminal on a graphical desktop.

Each entry is labeled with the command name on the outer edge of the page. The syntax line is followed by a brief description and a list of available options. Many commands come with examples at the end of the entry. If you need only a quick reminder or suggestion about a command, you can skip directly to the examples.

Typographic conventions for describing command syntax are listed in the Preface. For help in locating commands, see the index at the back of this book.

We've tried to be as thorough as possible in listing options. The basic command information and most options should be correct; however, there are many Linux distributions and many versions of commands. New options are added and sometimes old options are dropped. You may, therefore, find some differences between the options you find described here and the ones on your system. When there seems to be a discrepancy, check the manpage. For most commands you can also use the option **help** to get a brief usage message.

(Even when it isn't a valid option, it will usually result in an "invalid option" error

along with the usage message.)

Traditionally, commands take single-letter options preceded by a single hyphen, like **-d**. A more recent convention allows long options

preceded by two hyphens, like **debug**. Often, a feature can be invoked through either the old style or the new style of options.

aclocal [<tt>options</tt>]

addr2line [<tt>options</tt>] [<tt>addresses</tt>]

agetty [<tt>options</tt>] <tt>port</tt> <tt>baudrate</tt> [<tt>term</tt>]

anacron [<tt>options</tt>] [<tt>job</tt>]

apmd [<tt>options</tt>]

apropos <tt>string</tt> <tt>..</tt>.

apt

ar <tt>key</tt>[<tt>args</tt>] [<tt>posname</tt>] [<tt>count</tt>] <tt>archive</tt> [<tt>files</tt>]

ar r mylib.a `ls *.o`

arch

arp [<tt>options</tt>]

arp -a eris

arp -s illuminati 00:05:23:73:e6:cf

arp -Ds fnord eth0 pub

arp -i eth0 -d fnord pub

as [<tt>options</tt>] <tt>files</tt>

at [<tt>options</tt>] <tt>time</tt>

$ at 1:00 am tomorrow

at> ./total_up > output

at> mail joe < output

at> <EOT>   Entered by pressing Ctrl-D

job 1 at 2003-03-19 01:00

$ at 1:00 am tomorrow < scriptfile

$ at 1945 December 9

$ at 7:45pm Dec 9

$ at 3 am Saturday

$ at now + 5 hours

$ at noon next day

atd <tt>
options</tt>

atq [<tt>
options</tt>]

atrm [<tt>
options</tt>] <tt>job</tt> [<tt>
job..</tt>.]

audiosend
[<tt>email</tt><span
class="docEmphBold">@<tt>
address</tt>]

autoconf [<tt>
options</tt>] [<tt>template_file</tt>]

autoheader
[<tt>options</tt>] [<tt>template_file
</tt>]

automake [<tt>options</tt>] [<tt>template_file</tt>]

autoreconf [<tt>options</tt>]

autoscan [<tt>options</tt>] [<tt>directory</tt>]

autoupdate [<tt>options</tt>] [<tt>file</tt>]

badblocks [<tt>options</tt>] <tt>device</tt> <tt>block-count</tt>

banner [<tt>option</tt>] [<tt>characters</tt>]

/usr/games/banner -w50 Happy Birthday! |lpr

basename <tt>name</tt> [<tt>suffix</tt>]

basename <tt>option</tt>

basename /usr/lib/libm.a

libm.a

% basename /usr/lib/libm.a .a

libm

bash [<tt>
options</tt>] [<tt>file</tt> [<tt>
arguments</tt>]]

sh [<tt>
options</tt>] [<tt>file</tt> [<tt>
arguments</tt>]]

batch [<tt>
options</tt>] [<tt>time</tt>]

bc [<tt>
options</tt>] [<tt>files</tt>]

if (x= =y) {i = i + 1} else {i = i - 1}

while (i>0) {p = p*n; q = a/b; i = i-1}

for (i=1; i<=10; i++) i*5

ibase = 8   Octal input

20   Evaluate this octal number

16 Terminal displays decimal value

obase = 2   Display output in base 2 instead of base 10

20   Octal input

10000 Terminal now displays binary value

ibase = A   Restore base-10 input

scale = 3   Truncate results to 3 decimal places

8/7   Evaluate a division

1.001001000 Oops! Forgot to reset output base to 10

obase=10   Input is decimal now, so A isn't needed

8/7

1.142 Terminal displays result (truncated)

define p(r,n){   Function p uses two arguments

auto v   v is a local variable

v = r^n   r raised to the n power

return(v)}   Value returned

scale=5

x=p(2.5,2)   x = 2.5 ^ 2

x   Print value of x

6.25

length(x)   Number of digits

3

scale(x)   Number of places right of decimal point

2

biff [<tt>arguments</tt>]

bison [<tt>options</tt>] <tt>file</tt>

bootpd [<tt>options</tt>] [<tt>configfile</tt> [<tt>dumpfile</tt>]]

bootps dgram udp wait root /etc/bootpd bootpd

<tt>hostname</tt>:<tt>tg</tt>=<tt>value</tt>:<tt>tg</tt>=<tt>value</tt>:<tt>tg</tt>=<tt>value</tt>

bootpgw [<tt>options</tt>] <tt>server</tt>

bootps dgram udp wait root /etc/bootpgw bootpgw

bootptest [<tt>options</tt>] <tt>server</tt> [<tt>template</tt>]

bzcmp [<tt>options</tt>] <tt>file1 file2</tt>

bzdiff [<tt>options</tt>] <tt>file1 file2</tt>

bzgrep [<tt>options</tt>] <tt>pattern</tt> [<tt>file..</tt>.]

bzip2 [<tt>options</tt>] <tt>filenames</tt>

bunzip2 [<tt>options</tt>] <tt>filenames</tt>

bzcat [<tt>option</tt>] <tt>filenames</tt>

bzip2recover <tt>filenames</tt>

bzless [<tt>options</tt>] <tt>file</tt>

bzmore [<tt>options</tt>] <tt>file</tt>

c++ [<tt>options</tt>] <tt>files</tt>

c++filt [<tt>options</tt>] [<tt>symbol</tt>]

cal [options] [[<tt>month</tt>] <tt>year</tt>]

cal 12 1995

cal 1994 > year_file

cardctl [<tt>options</tt>] <tt>command</tt>

cardmgr [<tt>options</tt>]

cat [<tt>options</tt>] [<tt>files</tt>]

cat ch1   Display a file

cat ch1 ch2 ch3 > all   Combine files

cat note5 >> notes   Append to a file

cat > temp1   Create file at terminal; end with EOF

cat > temp2 << STOP   Create file at terminal; end with STOP

cc [<tt>options</tt>] <tt>files</tt>

cdda2wav [<tt>options</tt>] [<tt>output.wav</tt>]

cdda2wav

cdda2wav -B

cdparanoia [<tt>options</tt>] <tt>span</tt> [<tt>outfile</tt>]

cdrdao <tt>command</tt> [<tt>options</tt>] <tt>toc-file</tt>

cdrecord [<tt>general-options</tt>] dev=<tt>device</tt> [<tt>track-options</tt>]

<tt>track1,track2</tt>...

cfdisk [<tt>options</tt>] [<tt>device</tt>]

chage [<tt>options</tt>] <tt>user</tt>

chat [<tt>options</tt>] [<tt>script</tt>]

ogin:-BREAK-ogin:

chattr [<tt>options</tt>] <tt>mode</tt> <tt>files</tt>

chattr +a myfile   As superuser

checkpc [<tt>options</tt>]

chfn [<tt>options</tt>] [<tt>username</tt>]

chfn -f "Ellen Siever" ellen

chgrp [<tt>options</tt>] <tt>newgroup</tt> <tt>files</tt>

chgrp [<tt>options</tt>]

chmod [<tt>options</tt>] <tt>mode</tt> <tt>

files</tt>

chmod [<tt>
options</tt>] --
reference=<tt>filename</tt> <tt>
files</tt>

chmod u+x file

chmod 751 file

chmod u=rwx,g=rx,o=x file

chmod =r file

chmod 444 file

chmod a-wx,a+r file

chmod 4755 file

chown [<tt>
options</tt>] <tt>newowner</tt> <tt>
files</tt>

chown [<tt>
options</tt>] --
reference=<tt>filename</tt> <tt>
files</tt>

chpasswd
[<tt>option</tt>]

chroot <tt>
newroot</tt> [<tt>command</tt>]

chsh [<tt>
options</tt>] [<tt>username</tt>]

chsh -s /bin/tcsh

chvt <tt>
N</tt>

cksum [<tt>files</tt>]

clear

cmp [<tt>options</tt>] <tt>file1</tt> <tt>file2</tt> [<tt>skip1</tt> [<tt>skip2</tt>]]

cmp -s old new && echo 'no changes'

col [<tt>options</tt>]

tbl myfile | nroff | col | more

man ls | col -b > out.print

colcrt [<tt>options</tt>] [<tt>files</tt>]

colrm [<tt>start</tt> [<tt>stop</tt>]]

colrm 3 5 < test1 > test2

column [<tt>options</tt>] [<tt>files</tt>]

comm [<tt>options</tt>] <tt>file1</tt> <tt>file2</tt>

comm -12 siskel_top10 ebert_top10

compress [<tt>options</tt>] <tt>files</tt>

cp [<tt>options</tt>] <tt>file1</tt> <tt>file2</tt>

cp [<tt>options</tt>] <tt>files directory</tt>

cp --parents jphekman/book/ch1 newdir

cd /archives && cp -av /home/guest guest

cpio <tt>flags</tt> [<tt>options</tt>]

i: bcdf mnrtsuv B SVCEHMR IF

o: 0a c vABL VC HM O F

p: 0a d lm uv L V R

find . -name "*.old" -print | cpio -ocBv > /dev/rst8

cpio -icdv "*save*" < /dev/rst8

find . -depth -print | cpio -padm /mydir

cpp [<tt>options</tt>] [<tt>ifile</tt> [<tt>ofile</tt>]]

cron

crontab [<tt>options</tt>] [<tt>file</tt>]

Minute 0-59

Hour 0-23

Day of month 1-31

Month 1-12

Jan, Feb, Mar, ...

Day of week 0-6, with 0 = Sunday

Sun, Mon, Tue, ...

59 3 * * 5 find / -print | <tt>backup_program
</tt>

0 0 1,15 * * echo "Timesheets due" | mail <tt>
user</tt>

csh [<tt>
options</tt>] [<tt>file</tt> [<tt>
arguments</tt>]]

csplit [<tt>
options</tt>] <tt>file</tt> <tt>
arguments</tt>

csplit -k -f chap. novel '/CHAPTER/' '{20}'

csplit -k address_list 4 {99}

ctags [<tt>
options</tt>] <tt>files</tt>

<tt>name file context</tt>

:set tags= tagsfile

:tag name

cut <tt>
options</tt> [<tt>files</tt>]

cut -d: -f1,5 /etc/passwd

who | cut -d" " -f1

cut -c4 file | paste - file

cvs [<tt>
options</tt>] <tt>cvs-command</tt>
[<tt>command-options</tt>] [<tt>
command-args</tt>]

date [<tt>
options</tt>] [<span
class="docEmphBold">+<tt>format
</tt>] [<tt>date</tt>]

$ date +"%A %j %n%k %p"

Tuesday 248

15 PM

date 0701040095

date +"Hello%t Date is %D %n%t Time is %T"

Hello Date is 05/09/93

Time is 17:53:39

dd <tt> options</tt>

dd if=caps_file of=small_file conv=lcase

data_retrieval_cmd | dd of=out conv=sync,block

deallocvt <tt> N</tt>

debugfs [[<tt>option</tt>] <tt>device</tt>]

depmod [<tt>options</tt>] <tt>modules</tt>

devdump <tt>isoimage</tt>

df [<tt>options</tt>] [<tt>name</tt>]

diff [<tt>options</tt>] [<tt>diroptions</tt>] <tt>file1</tt> <tt>file2</tt>

diff3 [<tt>options</tt>] <tt>file1</tt> <tt>file2</tt> <tt>file3</tt>

dig [@<tt>server</tt>] [<tt>options</tt>] [<tt>name</tt>] [<tt>type</tt>] [<tt>class</tt>] [<tt>query-options</tt>]

dig @<tt>
server</tt> <tt>name</tt> <tt>
type</tt>

dig <tt>-
h</tt>

dip [<tt>
options</tt>] [<tt>chat</tt> <tt>
scriptfile</tt>]

dir [<tt>
options</tt>] [<tt>file</tt>]

dircolors [<tt>
options</tt>] [<tt>file</tt>]

dirname <tt>
pathname</tt>

dmesg [<tt>
options</tt>]

<span
class="docEmphBold">dnsdomainname

dnssec-keygen [<tt>options</tt>] <tt>domain-name</tt>

dnssec-makekeyset [<tt>options</tt>] <tt>key-identifiers</tt>

dnssec-signkey [<tt>options</tt>] <tt>keyset</tt> <tt>key-identifiers</tt>

dnssec-signzone [<tt>options</tt>] <tt>zonefile</tt> [<tt>key-identifiers</tt>]

domainname [<tt>name</tt>]

dosfsck [<tt>options</tt>] <tt>device</tt>

fsck.msdos [<tt>options</tt>] <tt>device</tt>

du [<tt>options</tt>] [<tt>directories</tt>]

dumpe2fs <tt>device</tt>

dumpkeys [<tt>options</tt>]

e2fsck [<tt>options</tt>] <tt>device</tt>

fsck.ext2 [<tt>options</tt>] <tt>device</tt>

e2image [<tt>option</tt>] <tt>device</tt> <tt>file</tt>

e2label <tt>device</tt> [<tt>label</tt>]

echo [options] [<tt>string</tt>]

/bin/echo "testing printer" | lp

/bin/echo "TITLE\nTITLE" > file ; cat doc1 doc2 >> file

/bin/echo "Warning: ringing bell \a"

egrep [<tt>options</tt>] [<tt>regexp</tt>] [<tt>files</tt>]

egrep 'Victor(ia)*' file

egrep '(Victor|Victoria)' file

egrep -n '(old|new)\.doc?' files

eject [<tt>options</tt>] [<tt>device</tt>]

elvtune [<tt>options</tt>] <tt>devices</tt>

emacs [<tt>options</tt>] [<tt>files</tt>]

env [<tt>option</tt>] [<tt>variable</tt>=<tt>value ..</tt>.] [<tt>command</tt>]

etags [<tt>options</tt>] <tt>files</tt>

<tt>name file context</tt>

M-x visit-tags-table

M-x find-tag

ex [<tt>options</tt>] <tt>file</tt>

expand [<tt>options</tt>] [<tt>files</tt>]

expr <tt>
arg1</tt> <tt>operator</tt> <tt>
arg2</tt> [<tt>operator</tt> <tt>
arg3</tt> <tt>..</tt>.]

expr 5 + 10 / 2

expr \(5 + 10 \) / 2

i=`expr $i + 1`

expr $a = hello

expr $b + 5 \>= 10

expr substr character 5 3

expr $p : '.*'   Result is 11

expr $p : '\(.*\)'   Result is "version.100"

expr $p : '[a-z]*'   Result is 7

expr $p : '\([a-z]*\)'   Result is "version"

expr $x : '\(.....\)' \| $x

mv $x `expr $x : '\(.....\)' \| $x</b>`

false

fdformat [<tt>option</tt>] <tt>device</tt>

fdisk [<tt>options</tt>] [<tt>device</tt>]

fetchmail [<tt>options</tt>] [<tt>servers..</tt>.]

fgconsole

fgrep [<tt>options</tt>] <tt>pattern</tt> [<tt>files</tt>]

fgrep -v ' ' file

fgrep -f spell_list file

file [<tt>options</tt>] <tt>files</tt>

ascii text

c program text

c-shell commands

data

empty

iAPX 386 executable

directory

[nt]roff, tbl, or eqn input text

shell commands

symbolic link to ../usr/etc/arp

file * | grep roff

find [<tt>pathnames</tt>] [<tt>conditions</tt>]

find $HOME -print

find /work -name chapter1

find /work -name 'memo*' -user ann -print

find / -type d -name 'man*' -print

find . \! -name '[A-Z]*' -exec lpr { }\;

gzip `find . \! -name '*.gz' -print`

find / -size 0 -ok rm { } \;

find / -mtime -2 -print

find /book -print | xargs grep '[Nn]utshell'

$ find . -name 'kt[0-9]'

./kt1

./kt2

$ find . -path './kt[0-9]'

./kt1

./kt2

$ find . -regex './kt[0-9]'

./kt1

./kt2

finger [<tt>options</tt>] <tt>users</tt>

in.fingerd [<tt>options</tt>]

flex [<tt>options</tt>] [<tt>file</tt>]

fmt [<tt>options</tt>] [<tt>files</tt>]

fold [<tt>option</tt>] [<tt>files</tt>]

formail [<tt>options</tt>]

free [<tt>options</tt>]

fsck [<tt>options</tt>] [<tt>filesystem</tt>]

<tt>..</tt>.

fsck.minix [<tt>options</tt>] <tt>device</tt>

ftp [<tt>options</tt>] [<tt>hostname</tt>]

in.ftpd [<tt>options</tt>]

fuser [<tt>options</tt>] [<tt>files</tt> | <tt>filesystems</tt>]

g++ [<tt>options</tt>] <tt>files</tt>

gawk [<tt>options</tt>] '<tt>script</tt>' [<tt>var</tt>=<tt>value..</tt>.] [<tt>files</tt>]

gawk [<tt>
options</tt>] -
f <tt>scriptfile</tt> [<tt>var
</tt>=<tt>
value..</tt>.] [<tt>files</tt>]

gcc [<tt>
options</tt>] <tt>files</tt>

gdb [<tt>
options</tt>] [<tt>program</tt> [<tt>
core</tt>|<tt>pid</tt>]]

getkeycodes

gpm [<tt>
options</tt>]

gprof [<tt>
options</tt>] [<tt>object_file</tt>]

grep [<tt>
options</tt>] <tt>pattern</tt> [<tt>
files</tt>]

grep -c /bin/tcsh /etc/passwd

grep -l '^#include' /usr/include/*

grep -c pattern files | grep :0

groff [<tt>
options</tt>] [<tt>files</tt>]

troff [<tt>
options</tt>] [<tt>files</tt>]

groffer [<tt>
viewing_options</tt>] [<tt>
man_options</tt>] [<tt>groff_options
</tt>]

[<tt>file-spec..</tt>.]

groffer <tt>
filespec</tt>

groupadd [<tt>options</tt>] <tt>group</tt>

groupdel <tt>group</tt>

groupmod [<tt>options</tt>] <tt>group</tt>

groups [<tt>options</tt>] [<tt>users</tt>]

grpck [<tt>option</tt>] [<tt>files</tt>]

grpconv

grpunconv

gs [<tt>options</tt>] [<tt>files</tt>]

gunzip [<tt>options</tt>] [<tt>files</tt>]

gzexe [<tt>option</tt>] [<tt>files</tt>]

gzip [<tt>options</tt>] [<tt>files</tt>]

gunzip [<tt>options</tt>] [<tt>files</tt>]

zcat [<tt>options</tt>] [<tt>files</tt>]

halt [<tt>options</tt>]

hdparm [<tt>options</tt>] [<tt>device</tt>]

head [<tt>options</tt>] [<tt>files</tt>]

head -20 phone_list

grep '(202)' phone_list | head

hexdump [<tt>options</tt>] <tt>file</tt>

host [<tt>options</tt>] <tt>name</tt> [<tt>server</tt>]

hostid

hostname [<tt>option</tt>] [<tt>nameofhost</tt>]

htdigest [-c] <tt>filename realm username</tt>

hwclock [<tt>option</tt>]

iconv [<tt>options</tt>] <tt>files</tt>

id [<tt>options</tt>] [<tt>username</tt>]

in.identd [<tt>options</tt>] [<tt>kernelfile</tt> [<tt>kmemfile</tt>]]

ifconfig [<tt>interface</tt>]

ifconfig [<tt>interface</tt> <tt>address_family</tt> <tt>parameters</tt> <tt>addresses</tt>]

imake <tt>options</tt>

/**/#

% xmkmf

% make Makefiles

% make includes

% make depend

imapd

inetd [<tt>options</tt>] [<tt>configuration_file</tt>]

`info [<tt>options</tt>] [<tt>topics</tt>]

init [<tt>bootflags</tt>] [<tt>runlevel</tt>]

insmod [<tt>options</tt>] <tt>file</tt> [<tt>symbol=value</tt> <tt>..</tt>.]

install [<tt>options</tt>] [<tt>source</tt>] <tt>destination</tt>

ipchains <tt>command</tt> [<tt>options</tt>]

ipchains-restore [<tt>options</tt>]

ipchains-save [<tt>chain</tt>] [<tt>option</tt>]

ipcrm [<tt>options</tt>]

ipcs [<tt>options</tt>]

iptables <tt>command</tt> [<tt>options</tt>]

iptables-restore [<tt>options</tt>]

iptables-save [<tt>options</tt>]

isodump <tt>isoimage</tt>

isoinfo [<tt>options</tt>]

isosize [<tt>option</tt>] <tt>iso9660-img-file</tt>

isovfy <tt>isoimage</tt>

ispell [<tt>options</tt>] [<tt>files</tt>]

join [<tt>options</tt>] <tt>file1</tt> <tt>file2</tt>

kbd_mode [<tt>option</tt>]

kbdrate [<tt>options</tt>]

kernelversion

kill [<tt>options</tt>] [<tt>pids</tt> | <tt>

commands</tt>]

killall [<tt>
options</tt>] <tt>names</tt>

killall5

klogd [<tt>
options</tt>]

ksyms [<tt>
options</tt>]

last [<tt>
options</tt>] [<tt>username</tt>]
[<tt>ttynumber</tt>]

lastb [<tt>
options</tt>] [<tt>username</tt>]
[<tt>ttynumber</tt>]

lastlog [<tt>
options</tt>]

ld [<tt>
options</tt>] <tt>objfiles</tt>

ldconfig [<tt>options</tt>] <tt>directories</tt>

ldd [<tt>options</tt>] <tt>programs</tt>

less [<tt>options</tt>] [<tt>filename</tt>]

lesskey [-o <tt>output-file</tt> | --output=<tt>output-file</tt>] [<tt>input-file</tt>]

lftp [<tt>options</tt>] [<tt>url</tt>]

lftpget [<tt>options</tt>] <tt>url</tt>

link <tt>file1 file2</tt>

ln [<tt>options</tt>] <tt>sourcename</tt> [<tt>destname</tt>]

ln [<tt>
options</tt>] <tt>sourcenames
destdirectory</tt>

loadkeys [<tt>
options</tt>] [<tt>filename</tt>]

locale [<tt>
options</tt>] [<tt>name</tt>]

locale -ck LC_TIME

locale day mon

locate [<tt>
options</tt>] <tt>pattern</tt>

lockfile [<tt>
options</tt>] <tt>filenames</tt>

logger [<tt>
options</tt>] [<tt>message..</tt>.]

login [<tt>
name</tt> | <tt>option</tt>]

logname [<tt>option</tt>]

logrotate [<tt>options</tt>] <tt>config_files</tt>

look [<tt>options</tt>] <tt>string</tt> [<tt>file</tt>]

losetup [<tt>options</tt>] <tt>loopdevice</tt> [<tt>file</tt>]

lpc [<tt>options</tt>] [<tt>command</tt>]

lpd [<tt>options</tt>]

lpq [<tt>options</tt>] [<tt>jobid</tt>]

lpr [<tt>options</tt>] [<tt>files</tt>]

lprm [<tt>options</tt>] [<tt>jobid</tt>]

lpstat [<tt>options</tt>] [<tt>queues</tt>]

ls [<tt>options</tt>] [<tt>names</tt>]

lsattr [<tt>options</tt>] [<tt>files</tt>]

lsmod

m4 [<tt>options</tt>] [<tt>macros</tt>] [<tt>files</tt>]

mail [<tt>options</tt>] [<tt>users</tt>]

mail

mail user

mailq [<tt>options</tt>]

mailstats [<tt>options</tt>]

mailto [<tt>options</tt>] <tt>recipients</tt>

make [<tt>options</tt>] [<tt>targets</tt>] [<tt>macro</tt> <tt>definitions</tt>]

<tt>name</tt>=<tt>string</tt>

<tt>targets</tt> : <tt>dependencies</tt>

<tt>targets</tt> : <tt>dependencies</tt> ; <tt>command</tt>

<tt>conditional</tt>

<tt>Text if true</tt>

else

<tt>Text if false</tt>

endif

.<tt>suffix</tt>.<tt>suffix</tt>:

.<tt>suffix</tt>:

<tt>name</tt> = <tt>string</tt>

define <tt>name</tt>

<tt>string</tt>

endef

include <tt>files</tt>

$(cc) -o $@ $<

%.o : %.c

$(cc) -o $@ $<

$(*D) $$(@D) $(?D) $(<D)

$(%D) $(@D) $(^D)

$(*F) $$(@F) $(?F) $(<F)

$(%F) $(@F) $(^F)

makedbm [<tt>options</tt>] <tt>infile</tt> <tt>outfile</tt>

makedbm [<tt>option</tt>]

BEGIN { FS =":";OFS = "\t";}

{ print $1, $0}

makemap [<tt>options</tt>] <tt>type</tt> <tt>name</tt>

<tt>key value</tt>

man [<tt>options</tt>] [<tt>section</tt>] [<tt>title</tt>]

manpath [<tt>options</tt>]

mattrib [<tt>options</tt>] <tt>filenames</tt>

mbadblocks drive

mcat [<tt>option</tt>] <tt>drive</tt>

mcd <tt>drive</tt>

mcopy [<tt>options</tt>] <tt>sourcefile</tt> <tt>target</tt>

mdel [<tt>option</tt>] <tt>file</tt>mdeltree [<tt>option</tt>] <tt>tree</tt>

mdir [<tt>options</tt>] <tt>dir</tt>

mdu [<tt>option</tt>] <tt>dir</tt>

merge [<tt>options</tt>] <tt>file1</tt> <tt>file2 file3</tt>

<<<<<<< file1

relevant lines from file1

= == == ==

relevant lines from file3

>>>>>>> file3

merge -L <tt>x</tt> -L <tt>y</tt> -L <tt>z</tt> <tt>file_a</tt> <tt>file_b</tt> <tt>file_c</tt>

mesg [<tt>option</tt>]

metamail [<tt>options</tt>] [<tt>filename</tt>]

metasend [<tt>options</tt>] [<tt>filename</tt>]

mformat [<tt>options</tt>] drive

mimencode [<tt>options</tt>] [<tt>filename</tt>] [-o <tt>output_file</tt>]

minfo [<tt>options</tt>] <tt>drive</tt>

mkdir [<tt>options</tt>] <tt>directories</tt>

mkdir -m 444 personal

mkdir work; cd work

mkdir junk; cd junk

mkdir questions; cd ../..

mkdir -p work/junk/questions

mkdosfs [<tt>
options</tt>] <tt>device</tt> [<tt>
blocks</tt>]<span
class="docEmphBold">mkfs.msdos [<tt>
options</tt>] <tt>device</tt> [<tt>
blocks</tt>]

mke2fs [<tt>
options</tt>] <tt>device</tt> [<tt>
blocks</tt>]

mkfs.ext2
[<tt>options</tt>] <tt>device</tt>
[<tt>blocks</tt>]

mkfifo [<tt>
option</tt>] <tt>names</tt>

mkfs [<tt>
options</tt>] [<tt>fs-options</tt>]
<tt>filesys</tt> [<tt>blocks</tt>]

mkfs.ext3 [<tt>options</tt>] <tt>device</tt> <tt>size</tt>

mkfs.minix [<tt>options</tt>] <tt>device</tt> <tt>size</tt>

mkisofs [<tt>options</tt>] -o <tt>file</tt> <tt>pathspecs</tt>

cdrecord -msinfo

-M <tt>file</tt> -C 0,0 -check-oldnames

mklost+found

mknod [<tt>options</tt>] <tt>name type</tt> [<tt>major minor</tt>]

mkpasswd [<tt>options</tt>] [<tt>user</tt>]

mkraid [<tt>options</tt>] <tt>devices</tt>

mkswap [<tt>option</tt>] <tt>device</tt> [<tt>size</tt>]

mktemp [<tt>options</tt>] <tt>template</tt>

mlabel [<tt>options</tt>] <tt>drive</tt>[<tt>label</tt>]

mmd [<tt>option</tt>] <tt>dirname</tt>

mmount <tt>drive </tt>[<tt>mount-arguments</tt>]

mmove [<tt>options</tt>] <tt>sourcefile</tt> <tt>targetfile</tt>

mmove [<tt>options</tt>] <tt>sourcefiles</tt> <tt>targetdir</tt>

modinfo [<tt>options</tt>] <tt>object-file</tt>

modprobe [<tt>options</tt>] [<tt>modules</tt>]

more [<tt>options</tt>] [<tt>files</tt>]

more -cd file

nroff doc | more -u

man more|more +/scroll

mount [<tt>options</tt>] [[<tt>device</tt>] <tt>directory</tt>]

rpc.mountd [<tt>options</tt>]

mpartition [<tt>options</tt>] <tt>drive</tt>

mpg123 [<tt>options</tt>] <tt>file</tt>

mpg321 [<tt>options</tt>] <tt>file</tt>

mpg123 [<tt>options</tt>] <tt>file</tt>

mrd [<tt>option</tt>] <tt>directory</tt>

mren [<tt>options</tt>] <tt>file</tt> <tt>name</tt>

mshowfat <tt>file</tt>

mt [<tt>option</tt>] <tt>operation</tt> [<tt>count</tt>]

<tt>command</tt> [<tt>options</tt>] [<tt>arguments</tt>]

mtoolstest

mtype [<tt>options</tt>] <tt>files</tt>

mv [<tt>option</tt>] <tt>sources</tt> <tt>target</tt>

mzip [<tt>options</tt>] [<tt>drive</tt>:]

named [<tt>options</tt>]

namei [<tt>options</tt>] <tt>pathname</tt> [<tt>pathname</tt> . . .]

nameif [<tt>options</tt>] [<tt>name</tt> <tt>macaddress</tt>]

netstat [<tt>options</tt>]

newaliases

newgrp [<tt>group</tt>]

newusers <tt>file</tt>

rpc.nfsd [<tt>option</tt>] <tt>n</tt>

nfsstat [<tt>options</tt>]

nice [<tt>option</tt>] [<tt>command</tt> [<tt>arguments</tt>]]

nm [<tt>options</tt>] [<tt>objfiles</tt>]

nohup <tt>command</tt> [<tt>arguments</tt>]

nohup <tt>
option</tt>

nslookup
[-option...]
[<tt>host_to_find</tt> | <span
class="docEmphBold">-[<tt>server
</tt>]]

nsupdate [<tt>
options</tt>] [<tt>filename</tt>]

objcopy [<tt>
options</tt>] <tt>infile</tt> [<tt>
outfile</tt>]

objdump [<tt>
options</tt>] <tt>objfiles</tt>

od [<tt>
options</tt>] [<tt>files</tt>]

od --traditional [<tt>file</tt>] [[+]<tt>offset</tt> [[+]<tt>label</tt>]]

openvt [<tt>options</tt>] [--] [<tt>command</tt>] [<tt>arguments</tt>]

passwd [<tt>user</tt>]

paste [<tt>options</tt>] <tt>files</tt>

paste x y z > file

who | paste - -

paste -s -d"\t\n" list

patch [<tt>options</tt>] [<tt>original</tt> [<tt>patchfile</tt>]]

#ifdef

string

#endif

pathchk [<tt>option</tt>] <tt>filenames</tt>

perl

pidof [<tt>options</tt>] <tt>programs</tt>

ping [<tt>options</tt>] <tt>host</tt>

rpc.portmap [<tt>options</tt>]

poweroff [<tt>options</tt>]

pppd [<tt>tty</tt>] [<tt>speed</tt>] [<tt>

options</tt>]

pr [<tt>
options</tt>] [<tt>files</tt>]

praliases [<tt>
options</tt>]

printenv [<tt>
option</tt>] [<tt>variables</tt>]
printenv <tt>
option</tt>

printf <tt>
formats</tt> [<tt>strings</tt>]<span
class="docEmphBold">printf <tt>
option</tt>

printf '%s %s\n' "My files are in" $HOME

printf '%-25.15s %s\n' "My files are in"
$HOME

ps [<tt>
options</tt>]

pwck [<tt>option</tt>] [<tt>files</tt>]

pwconvpwunconv

pwd

python

quota [<tt>options</tt>] [<tt>user</tt>|<tt>group</tt>]

raidstart [<tt>options</tt>] [<tt>devices</tt>]

raidstop [<tt>options</tt>] [<tt>devices</tt>]

ramsize [<tt>option</tt>] [<tt>image</tt> [<tt>size</tt> [<tt>offset</tt>]]]

ranlib <tt>filename</tt>

ranlib <tt>option</tt>

rarpd [<tt>options</tt>] [<tt>interface</tt>]

raw [<tt>options</tt>] [<tt>rawnode</tt> [<tt>blockdevice</tt>]]

rcp [<tt>options</tt>] <tt>file1</tt> <tt>file2</tt>

rcp [<tt>options</tt>] <tt>file ... directory</tt>

rcs [<tt>options</tt>] <tt>files</tt>

rdate [<tt>options</tt>] [<tt>host..</tt>.]

rdev [<tt>options</tt>] [<tt>image</tt> [<tt>value</tt> [<tt>offset</tt>]]]

rdist [<tt>options</tt>] [<tt>names</tt>]

rdistd <tt>options</tt>

readelf <tt>option</tt>[...] <tt>elffiles</tt>

readlink <tt>file</tt>

readlink <tt>option</tt>

reboot [<tt>options</tt>]

rename <tt>from</tt> <tt>to</tt> <tt>files</tt>

% rename test mytest test*

renice [<tt>priority</tt>] [<tt>options</tt>] [<tt>target</tt>]

reset [<tt>options</tt>] [<tt>terminal</tt>]

Ctrl-JresetCtrl-J

resize2fs [<tt>options</tt>] <tt>device</tt> [<tt>size</tt>]

rev [<tt>file</tt>]

rexecd <tt>command-line</tt>

richtext [<tt>options</tt>] [<tt>file</tt>]

rlogin [<tt>options</tt>] <tt>rhost</tt>

in.rlogind [<tt>options</tt>]

rm [<tt>options</tt>] <tt>files</tt>

rmail [<tt>options</tt>] <tt>users</tt>

rmdir [<tt>options</tt>] <tt>directories</tt>

rmmod [<tt>options</tt>] <tt>modules</tt>

rootflags [<tt>option</tt>] <tt>image</tt> [<tt>flags</tt> [<tt>offset</tt>]]

route [<tt>options</tt>] [<tt>command</tt>]

add [-net | -host] <span

class="docEmphasis">address [modifiers]

del [-net | -host] address [modifiers]

routed [<tt>options</tt>] [<tt>logfile</tt>]

rpcgen [<tt>options</tt>] <tt>file</tt>

rpcinfo [<tt>options</tt>] [<tt>host</tt>] [<tt>program</tt>] [<tt>version</tt>]

$ rpcinfo -p

$ rpcinfo -p klaxon

$ rpcinfo -b ypserv version | uniq

rpm [<tt>options</tt>]

rsh [<tt>options</tt>] <tt>host</tt> [<tt>command</tt>]

rshd [<tt>options</tt>]

runlevel [<tt>utmp</tt>]

run-parts [<tt>options</tt>] [<tt>directory</tt>]

rup [<tt>options</tt>] [<tt>hosts</tt>]

ruptime [<tt>options</tt>]

rusers [<tt>options</tt>] [<tt>host</tt>]

rpc.rusersd

rwall <tt>host</tt> [<tt>file</tt>]

rwho [<tt>option</tt>]

rwhod [<tt>options</tt>]

scp [<tt>options</tt>] <tt>file1</tt>[...] <tt>file2</tt>

<tt>file</tt>

<tt>host:file</tt>

<tt>user@host:file</tt>

screen [<tt>options</tt>] [<tt>command</tt> [<tt>args</tt>]]

script [<tt>option</tt>] [<tt>file</tt>]

sdiff -o <tt>outfile</tt> [<tt>options</tt>] <tt>from to</tt>

sed [<tt>options</tt>] [<tt>command</tt>] [<tt>files</tt>]

sendmail [<tt>flags</tt>] [<tt>address..</tt>.]

seq [<tt>options</tt>] [<tt>first</tt> [<tt>increment</tt>]] <tt>last</tt>

setfdprm [<tt>options</tt>] <tt>device</tt> [<tt>name</tt>]

setkeycodes <tt>scancode</tt> <tt>keycode</tt>

setleds [<tt>options</tt>]

setmetamode [<tt>options</tt>]

setsid <tt>command</tt> [<tt>arguments</tt>]

setterm [<tt>options</tt>]

sftp [<tt>options</tt>] <tt>host</tt>

<tt>host</tt>

<tt>[user</tt>@]<tt>host</tt>[:<tt>file</tt> [<tt>file</tt>] ...]

<tt>[user</tt>@]<tt>host</tt>[:<tt>dir</tt>[/]]

sh [<tt>options</tt>] [<tt>file</tt> [<tt>arguments</tt>]]

shar [<tt>options</tt>] <tt>files</tt>

shar -S [<tt>options</tt>]

showkey [<tt>options</tt>]

showmount [<tt>options</tt>] [<tt>host</tt>]

shred [<tt>options</tt>] <tt>files</tt>

shutdown [<tt>options</tt>] <tt>when</tt> [<tt>message</tt>]

size [<tt>options</tt>] [<tt>objfile..</tt>.]

skill [<tt>signal</tt>] [<tt>options</tt>] <tt>processes</tt>

snice [<tt>priority</tt>] [<tt>options</tt>] <tt>processes</tt>

slattach [<tt>options</tt>] [<tt>tty</tt>]

sleep <tt>amount</tt>[<tt>units</tt>]

sleep <tt>option</tt>

snice [<tt>priority</tt>] [<tt>options</tt>] <tt>processes</tt>

sort [<tt>options</tt>] [<tt>files</tt>]

wc -l * | sort -r

sort -fd wordlist | uniq -c

sort -nk3,4 -t: /etc/passwd

split [<tt>options</tt>] [<tt>infile</tt> [<tt>prefix</tt>]]

split bigfile

cat list[1-4] | split -10 - new.

ssh [<tt>options</tt>] hostname [<tt>command</tt>]

ssh-add [<tt>options</tt>] [<tt>files</tt>]

ssh-add -e|-s <tt>reader</tt>

ssh-agent [<tt>options</tt>] [<tt>command</tt> [<tt>arguments</tt>]]

ssh-keygen [<tt>options</tt>]

ssh-keyscan [<tt>options</tt>]

sshd [<tt>options</tt>]

stat [options] <tt>files</tt>

stat /

File: "/"

Size: 4096 Blocks: 8 IO Block: 4096 Directory

Device: 303h/771d Inode: 2 Links: 19

Access: (0755/drwxr-xr-x) Uid: (0/ root) Gid: (0/ root)

Access: Thu Jan 2 04:02:40 2003

Modify: Wed Jan 1 23:03:20 2003

Change: Wed Jan 1 23:03:20 2003

stat -f /

File: "/"

ID: 0 0 Namelen: 255 Type: ext2/ext3

Blocks: Total: 2612475 Free: 1869472 Available: 1736735 Size: 4096

Inodes: Total: 1329696 Free: 1150253

rpc.statd [<tt>options</tt>]

strace [<tt>options</tt>] <tt>command</tt> [<tt>arguments</tt>]

strfile [<tt>options</tt>] <tt>input_file</tt> [<tt>output_file</tt>]

unstr [-c <tt>delimiter</tt>] <tt>input_file</tt>[.<tt>ext</tt>] [<tt>output_file</tt>]

strings [<tt>options</tt>] <tt>files</tt>

strip [<tt>
options</tt>] <tt>files</tt>

stty [<tt>
options</tt>] [<tt>modes</tt>]

su [<tt>
option</tt>] [<tt>user</tt>] [<tt>
shell_args</tt>]

su -c 'find / -name *.c -print' nobody

sudo [<tt>
options</tt>] [<tt>command</tt>]

sum [<tt>
options</tt>] <tt>files</tt>

swapoff [<tt>
options</tt>] [<tt>devicelist</tt>]

swapon [<tt>
options</tt>] <tt>devices</tt>

sync

sysctl [<tt>options</tt>] [<tt>key</tt>]

syslogd [<tt>options</tt>]

syslogd

tac [<tt>options</tt>] [<tt>file</tt>]

tail [<tt>options</tt>] [<tt>files</tt>]

= =><tt>filename</tt><= =

grep '\.Ah ' file | tail -20

echo "$name" | tail -c

tail -2b bigfile

talk <tt>person</tt> [<tt>ttyname</tt>]

talkd [<tt>options</tt>]

tar [<tt>options</tt>] [<tt>tarfile</tt>] [<tt>other-files</tt>]

tar --<tt>long-option</tt> -<tt>function-options files</tt>

tar --modification-time -xvf tarfile.tar

tar cvf /dev/rmt0 /bin /usr/bin

tar tvf /dev/rmt0

tar xvf /dev/rmt0 /bin

tar cvf - `find . -print` > backup.tar

tcpd

tcsh [<tt>options</tt>] [<tt>file</tt> [<tt>arguments</tt>]]

tee [<tt>options</tt>] <tt>files</tt>

s -l | tee savefile View listing and save for later

telinit [<tt>option</tt>] [<tt>runlevel</tt>]

telnet [<tt>options</tt>] [<tt>host</tt> [<tt>port</tt>]]

telnetd [<tt>options</tt>]

test <tt>expression</tt>

[<tt>expression</tt>]

tftp [<tt>options</tt>] [<tt>host</tt> [<tt>port</tt>]]

tftp>

in.tftpd [<tt>
options</tt>] [<tt>directories</tt>]

time [<tt>
options</tt>] <tt>command</tt> [<tt>
arguments</tt>]

/usr/bin/time -f "\t%U user,\t%S system,\t%x
status" ls -Fs

tload [<tt>
options</tt>] [<tt>tty</tt>]

top [<tt>
options</tt>]

touch [<tt>
options</tt>] <tt>files</tt>

tr [<tt>
options</tt>] [<tt>string1</tt> [<tt>
string2</tt>]]

cat file | tr 'A-Z' 'a-z'

tr ' ' '

' < file

cat file | tr -s "" "

" > new.file

tr -d : < file > new.file

traceroute [<tt>options</tt>] <tt>host</tt> [<tt>packetsize</tt>]

troff

true

tset [<tt>options</tt>] [<tt>terminal</tt>]

reset [<tt>options</tt>] [<tt>terminal</tt>]

[<tt>port type</tt>][<tt>operator</tt>][<tt>baud rate</tt>][:]<tt>terminal type</tt>

tty [<tt>options</tt>]

tune2fs [<tt>options</tt>] <tt>device</tt>

mke2fs -O <tt>jrnl-dev</tt> <tt>ext-jrnl</tt>

tunelp <tt>device</tt> [<tt>options</tt>]

ul [<tt>options</tt>] [<tt>names</tt>]

umount [<tt>options</tt>] [<tt>directory</tt>|<tt>special-device</tt>]

uname [<tt>options</tt>]

uncompress [<tt>options</tt>] <tt>files</tt>

unexpand [<tt>options</tt>] [<tt>files</tt>]

unicode_start [<tt>font</tt> [<tt>umap</tt>]]

unicode_stop

uniq [<tt>options</tt>] [<tt>file1</tt> [<tt>file2</tt>]]

uniq list list.new

sort names | uniq -d

unshar [<tt>options</tt>] [<tt>files</tt>]

update [<tt>options</tt>]

uptime [<tt>option</tt>]

useradd [<tt>options</tt>] [<tt>user</tt>]

userdel [<tt>option</tt>] <tt>user</tt>

usermod [<tt>options</tt>] <tt>user</tt>

users [<tt>file</tt>]users <tt>option</tt>

usleep [<tt>microseconds</tt>]usleep [<tt>options</tt>]

uudecode [-o <tt>outfile</tt>] [<tt>file</tt>]

uuencode [-m] [<tt>file</tt>] <tt>name</tt>

% uuencode flower12.jpg rose.jpg > rose.uue

% uuencode flower12.jpg flower12.jpg | mail el@oreilly.com

uuidgen [<tt>option</tt>]

vacationvacation [<tt>options</tt>] [<tt>user</tt>]

\<tt>user</tt>, "|/usr/bin/vacation <tt>user</tt>"

$ cd

$ vacation -I

$ cat .forward

\jp, "|/usr/bin/vacation -r3w jp"

$ cat .vacation.msg

From: jp@wizard-corp.com (J. Programmer, via the vacation program)

Subject: I'm out of the office ...

Hi. I'm off on a well-deserved vacation after finishing

up whizprog 1.0. I will read and reply to your mail

regarding "$SUBJECT" when I return.

Have a nice day.

vdir [<tt>options</tt>] [<tt>files</tt>]

vi [<tt>options</tt>] [<tt>files</tt>]

vidmode [<tt>option</tt>] <tt>image</tt> [<tt>mode</tt> [<tt>offset</tt>]]

vim

vmstat [<tt>options</tt>] [<tt>interval</tt> [<tt>count</tt>]]

volname [<tt>devfile</tt>]

w [<tt>options</tt>] [<tt>user</tt>]

wall [<tt>file</tt>]

wall [<tt>message</tt>]

watch [<tt>options</tt>] <tt>command</tt> [<tt>cmd_options</tt>]

wc [<tt>options</tt>] [<tt>files</tt>]

who | wc -l

wc -w essay.[123]

wc -l < $file

whatis <tt>keywords</tt>

whereis [<tt>options</tt>] <tt>files</tt>

% cd /usr/bin

% whereis -u -M /usr/share/man/man1 -S /usr/src -f *

which [<tt>options</tt>] [--] [<tt>commands</tt>]

which cc ls

/usr/bin/cc

ls: aliased to ls -sFC

who [<tt>options</tt>] [<tt>file</tt>]

who am i

$ who -uH

NAME LINE TIME IDLE PID COMMENTS

Earvin ttyp3 Apr 16 08:14 16:25 2240

Larry ttyp0 Apr 17 07:33 . 15182

whoami

whois[<tt>
options</tt>] <tt>query</tt>[@<tt>
server</tt>[:<tt>port</tt>]]

fwhois [<tt>
options</tt>] <tt>query</tt>[@<tt>
server</tt>[:<tt>port</tt>]]

write <tt>user</tt> [<tt>tty</tt>]

<tt>message</tt>

xargs [<tt>options</tt>] [<tt>command</tt>]

find / -print | xargs grep pattern > out &

echo $* | xargs -n2 diff

cat file | xargs -n1

ls olddir | xargs -i -t mv olddir/{ } newdir/{ }

xinetd [<tt>options</tt>]

service <tt>servicename</tt>

{

<tt>attribute1 = valueset1</tt>

<tt>attribute2 = valueset2</tt>

}

yacc [<tt>
options</tt>] <tt>file</tt>

yes [<tt>
strings</tt>]

yes [<tt>
option</tt>]

ypbind [<tt>
options</tt>]

ypcat [<tt>options</tt>] <tt>map</tt>

ypchfn [<tt>option</tt>] [<tt>user</tt>]

ypinit [<tt>options</tt>]

ypmatch [<tt>options</tt>] <tt>key...mname</tt>

yppasswd [<tt>options</tt>] [<tt>name</tt>]

rpc.yppasswdd [<tt>options</tt>]

<tt>username</tt> o:<tt>oldpassword</tt> p:<tt>password</tt> s:<tt>shell</tt> g:<tt>gcos</tt>

yppoll [<tt>options</tt>] <tt>map</tt>

yppush [<tt>options</tt>] <tt>mapnames</tt>

ypserv [<tt>options</tt>]

ypset [<tt>options</tt>] <tt>server</tt>

yptest [<tt>options</tt>]

ypwhich [<tt>options</tt>] [<tt>host</tt>]

ypxfr [<tt>options</tt>] <tt>mapname</tt>

zcat [<tt>options</tt>] [<tt>files</tt>]

zcmp [<tt>options</tt>] <tt>files</tt>

zdiff [<tt>options</tt>] <tt>files</tt>

zdump [<tt>options</tt>] [<tt>zones</tt>]

zforce [<tt>names</tt>]

zgrep [<tt>options</tt>] [<tt>files</tt>]

zic [<tt>options</tt>] [<tt>files</tt>]

Rule <tt>NAME FROM TO TYPE IN ON AT SAVE LETTERS</tt>

Zone <tt>NAME GMTOFF RULES/SAVE FORMAT</tt> [<tt>UNTIL</tt>]

Link <tt>LINK-FROM LINK-TO</tt>

Leap <tt>YEAR MONTH DAY HH:MM:SS CORR R</tt>|<tt>S</tt>

zless <tt>files</tt>

zmore [<tt>files</tt>]

znew [<tt>options</tt>] [<tt>files</tt>]

Uncompress .Z files and recompress them in .gz format.

Options

-9

Optimal (and slowest) compression method.

-f

Recompress even if filename.gz already exists.

-t

Test new .gz files before removing .Z files.

-v

Verbose mode.

-K

If the original .Z file is smaller than the .gz file, keep it.

-P

Pipe data to conversion program. This saves disk space.

Chapter 4. Boot Methods

This chapter describes some techniques for booting your Linux system. Depending on your hardware and whether you want to run any other operating systems, you can configure the system to boot Linux automatically or to provide a choice between several operating systems. Choosing between operating systems is generally referred to as *dual booting*, but you can actually boot more than two.

Once your Linux system is installed, rebooting the system is generally pretty straightforward. But with the wide variety of hardware and software in use, there are many possibilities for configuring your boot process. The most common choices are:

- Boot Linux from a floppy or bootable CD, leaving any other operating system to boot from the hard drive.

- Use the Linux Loader, LILO.[1] This is the traditional method of booting and lets you boot both Linux and other operating systems.

  [1] LILO is a boot program for i386-architecture machines. On the Alpha, the equivalent boot program is called MILO (Mini Loader), and on the SPARC, it is SILO.

- Use the Grand Unified Bootloader (GRUB), the GNU graphical boot loader and command shell. Like LILO, GRUB lets you boot both Linux and other operating systems. For now, GRUB runs only on i386-based systems.

- Run Loadlin, which is an MS-DOS program that boots Linux from within DOS.

Other boot managers that can load Linux are available, but we don't discuss them here. We also

won't talk further about booting from a floppy or CD, except to say that whatever method you choose for booting, be sure to have a working boot disk available for emergency use. In particular, don't experiment with the files and options in this chapter unless you have a boot disk, because any error could leave you unable to boot from the hard disk. Note, though, that one of the advantages of using GRUB is that if there is a problem booting from the menu, it drops you down to the command-line interface so you can enter commands directly and try to recover. Also, see Section 4.3.1.1 for information on making a GRUB boot floppy.[2]

[2] Unfortunately, there is no standard set of instructions we can provide for making a

bootable CD. Your best bet is to use a bootable installation CD for your distribution. Also, instructions and utilities are available online for making bootable CDs.

% /sbin/lilo -u

C:> fdisk /mbr

•

For Windows 2000 and Windows XP, which do not have an fdisk command, boot your computer from the Windows CD (or the Windows boot floppies if you can't boot from your CD drive). When you see "Welcome to Setup," press R (for repair) and, in Windows 2000, you then press C. Select your Windows installation from the numbered list that is displayed (there may be only one entry) and enter the administrator password at the prompt. Enter the command fixmbr at the command-line prompt and confirm it with y. After the MBR has been restored, type exit to reboot.

The common element in all three methods is that they replace the boot loader on the MBR with the original Microsoft boot loader. The boot loader on the MBR is the one that will be used to boot the system. This means that if you want to switch from LILO to

GRUB, say, or from GRUB to LILO, you don't need to uninstall the old loader; simply install the new one.

The rest of this chapter describes the various techniques for booting Linux and the options that you can specify to configure both the boot loader and the Linux kernel. Whether you use LILO, GRUB, or Loadlin, you can pass options to the loader and specify options for the kernel.

boot=/dev/hda # The boot device is /dev/hda
map=/boot/map # Save the map file as /boot/map
install=/boot/boot.b # The file to install as the new boot sector prompt # Always display the boot prompt
timeout=30 # Set a 3-second (30 tenths of a second) timeout

image=/boot/vmlinuz # Linux image file label=linux # Label that appears at the boot prompt
root=/dev/hda2 # Location of the root filesystem
vga=ask # Always prompt the user for VGA mode
read-only # Mount read-only to run fsck for a filesystem check

other=/dev/hda1 # Location of the partition
label=win98

table=/dev/hda # Location of the partition table

Global section

boot=/dev/hda2

map=/boot/map

delay=30

timeout=50

prompt

vga=ask

Image section: For regular Linux
image=/boot/vmlinuz

label=linux root=/dev/hda2

install=/boot/boot.b map=/boot/map read-only ## Image section: For testing a new Linux kernel
image=/testvmlinuz

label=testlinux root=/dev/hda2

install=/boot/boot.b map=/boot/map read-only optional # Omit image if not available when map is built ## Image section: For booting DOS

other=/dev/hda1

label=dos loader=/boot/chain.b table=/dev/hda # The current partition table ## Image section: For booting Windows 98

other=/dev/hda1

label=win98

loader=/boot/chain.b table=/dev/hda

disk=/dev/sda

bios=0x80 # First disk is usually 0x80, second is usually 0x81

sectors=...

heads=...

partition=/dev/sda1

start=2048

<tt>port</tt>[,<tt>bps</tt>[<tt>parity</tt>[<tt>bits</tt>]]]

serial=0,2400n8

image=/dev/fd0

range=1+512 # take 512 sectors, starting with sector 1

append="mem=128M"

lilo [<tt>options</tt>]

% lilo -q -v

% lilo -I linux /boot/vmlinuz-2.0.34-0.6

% lilo -q

linux *

test

% mount /dev/hda2 /mnt % lilo -r /mnt

-R command-line

Set the default command for the boot loader the next time it executes. The command executes once and then is removed by the boot loader. This option typically is used in reboot scripts, just before calling shutdown -r.

-t

Indicate that this is a testdo not really write a new boot sector or map file. Can be used with -v to find out what lilo would do during a normal run.

-u device-name

Uninstall lilo by restoring the saved boot sector from /boot/boot.nnnn, after validating it against a timestamp. device-name is the name of the device on which LILO is installed, such as /dev/hda2.

-U device-name

Like -u, but do not check the timestamp.

-V

Print the lilo version number.

(hd0,0)/grub/grub.conf

[<tt>device</tt>][<tt>offset</tt>]+<tt>length</tt>[,<tt>offset</tt>]+<tt>length</tt>...

rootnoverify (hd0,0)

makeactive

chainloader +1

(fd0) /dev/fd0

(hd0) /dev/hda

% dd if=stage1 of=/dev/fd0 bs=512 count=1

% dd if=stage2 of=/dev/fd0 bs=512 seek=1

% dd if=stage1 of=/dev/fd0 bs=512 count=1

1+0 records in

1+0 records out

% dd if=stage2 of=/dev/fd0 bs=512 seek=1

254+1 records in

254+1 records out

$ mke2fs /dev/fd0

% mount /mnt

% mkdir /mnt/boot

% mkdir /mnt/boot/grub

$ /sbin/grub --batch <<EOT

root (fd0)

setup (fd0)

quit

EOT

grub-install <tt>options</tt> <tt>install-device</tt>

grub-install /dev/hda

% grub-install --grub-shell="grub --read-only" /dev/fd0

grub> find /boot/grub/stage1

(hd0,0)

grub> root (hd0,0)

grub> setup (hd0)

grub> setup (hd0,0)

default=0 # default to the first entry

timeout=20 # set the timeout to 20 seconds

splashimage=(hd0,0)/grub/splash.xpm.gz # the splash image displayed with

the menu

title Red Hat Linux (2.4.18-14)

root (hd0,1)

kernel /vmlinuz-2.4.18-14 ro root=LABEL=/ hdc=ide-scsi

initrd /initrd-2.4.18-14.img

Red Hat Linux (2.4.18-14)

grub [<tt>options</tt>]

% grub --no-floppy

GRUB version 0.92 (640K lower / 3072K upper memory)

[Minimal BASH-like line editing is supported. For the first word, TAB

lists possible command completions. Anywhere else TAB lists the possible

completions of a device/filename.]

grub>

grub>

Possible commands are: blocklist boot cat chainloader cmp color configfile

debug device displayapm displaymem dump embed find fstest geometry halt help

hide impsprobe initrd install ioprobe kernel lock makeactive map md5crypt

module modulenounzip pager partnew parttype password pause quit read reboot

root rootnoverify savedefault serial setkey setup terminal testload testvbe

unhide uppermem vbeprobe

grub> help

blocklist FILE boot

cat FILE chainloader [--force] FILE

color NORMAL [HIGHLIGHT] configfile FILE

device DRIVE DEVICE displayapm

displaymem find FILENAME

geometry DRIVE [CYLINDER HEAD SECTOR [
halt [--no-apm]

help [--all] [PATTERN ...] hide PARTITION

initrd FILE [ARG ...] kernel [--no-mem-option] [--type=TYPE]

makeactive map TO_DRIVE FROM_DRIVE

md5crypt module FILE [ARG ...]

modulenounzip FILE [ARG ...] pager [FLAG]

partnew PART TYPE START LEN parttype PART TYPE

quit reboot

root [DEVICE [HDBIAS]] rootnoverify [DEVICE [HDBIAS]]

serial [--unit=UNIT] [--port=PORT] [-- setkey [TO_KEY FROM_KEY]

setup [--prefix=DIR] [--stage2=STAGE2_ terminal [--dumb] [--timeout=SECS] [--

testvbe MODE unhide PARTITION

uppermem KBYTES vbeprobe [MODE]

You can add the --all option to see all the commands.

To get help for a specific command, add the command name (e.g., help read). help treats the text you enter as a pattern; therefore, if you enter help find, you'll get help for the find command, but if you

enter help module, you'll get help for both module and modulenounzip.

grub> blocklist (hd0,1)/grub/[Tab]

Possible files are: grub.conf splash.xpm.gz menu.lst device.map stage1

stage2 e2fs_stage1_5 fat_stage1_5 ffs_stage1_5 jfs_stage1_5 minix_stage1_5

reiserfs_stage1_5 vstafs_stage1_5 xfs_stage1_5

grub> blocklist (hd0,1)/grub/stage2

(hd0,1)33306+24,33332+231

bootp [--with-configfile]

color <tt>
normal</tt> [<tt>highlight</tt>]

color light-gray/blue cyan/black

device <tt>
drive</tt> <tt>file</tt>

grub> device (fd0) /floppy-image

grub> device (hd0) /dev/sd0

dhcp [<span
class="docEmphBold">--with-configfile]

hide <tt>
partition</tt>

ifconfig
[--server=<tt>
server</tt>] [--
gateway=<tt>gateway</tt>] [<span
class="docEmphBold">--mask=<tt>
mask</tt>]

[--address= <tt>address</tt>]

pager [<tt> flag</tt>]

partnew <tt> part type from to</tt>

parttype <tt> part type</tt>

password [--md5] <tt> passwd</tt> [<tt>file</tt>]

rarp

serial [<tt> options</tt>]

setkey [<tt> to-key from-key</tt>]

&

@

\

{

[

^

’

"

!

<

#

)

.

?

;

_

splashimage <tt>file</tt>

splashimage=(hd0,0)/grub/splash.xpm.gz

terminal [<tt>options</tt>] [console] [serial]

tftpserver <tt>ipaddress</tt>

unhide <tt>partition</tt>

blocklist <tt>file</tt>

grub> blocklist (hd0,1)/grub/grub.conf

(hd0,1)33746+2

boot

cat <tt>file</tt>

chainloader [--force] <tt>file</tt>

cmp <tt>file1 file2</tt>

configfile <tt>file</tt>

debug

displayapm

displaymem

grub> displaymem

EISA Memory BIOS Interface is present

Address Map BIOS Interface is present

Lower memory: 640K, Upper memory (to first chipset hole): 3072K

[Address Range Descriptor entries immediately follow (values are 64-bit)]

Usable RAM: Base Address: 0x0 X 4GB + 0x0,

Length: 0x0 X 4GB + 0xa0000 bytes

Reserved: Base Address: 0x0 X 4GB + 0xa0000,

Length: 0x0 X 4GB + 0x60000 bytes

Usable RAM: Base Address: 0x0 X 4GB + 0x100000,

Length: 0x0 X 4GB + 0x300000 bytes

dump <tt>from to</tt>

embed <tt>stage1.5 device</tt>

find <tt>file</tt>

fstest

<partition-offset-sector, byte-offset, byte-length>

[disk-offset-sector]

geometry <tt>drive</tt> [<tt>cylinder head sector</tt> [<tt>total_sector</tt>]]

halt [--no-apm]

help [--all] [<tt>patterns</tt>]

impsprobe

initrd <tt>file</tt> [<tt>args</tt>]

install [<tt>options</tt>] <tt>stage1_file</tt> [d] <tt>dest_dev stage2_file</tt> [<tt>addr</tt>] [p]

[<tt>config_file</tt>] [<tt>real_config_file</tt>]

ioprobe <tt>drive</tt>

kernel [--non-mem-option] <tt>file</tt> [...]

lock

makeactive

map <tt>to from</tt>

grub> map (hd0) (hd1)

grub> map (hd1) (hd0)

md5crypt

module <tt>file</tt> [...]

modulenounzip <tt>files</tt>

pause <tt>messages</tt>

quit

read <tt>addr</tt>

reboot

root <tt>device</tt> [<tt>hdbias</tt>]

rootnoverify <tt>device</tt> [<tt>hdbias</tt>]

savedefault

setup [<tt>options</tt>] <tt>install_device</tt> [<tt>image_device</tt>]

testload <tt>file</tt>

testvbe <tt>mode</tt>

uppermem <tt>kbytes</tt>

vbeprobe [<tt>mode</tt>]

For a VBE BIOS, probe VESA BIOS extension information. If mode is specified, the output shows only information for that mode; otherwise, all available VBE modes are listed.

C:> loadlin c:\vmlinuz root=/dev/hda2

LOADLIN @<tt>params</tt>

C:> LOADLIN @myparam vga=normal

LOADLIN [<tt>zimage_file</tt>] [<tt>options</tt>] [<tt>boot_params</tt>]

C:> LOADLIN | more

The Loadlin options are:

-clone

Bypass certain checks. Read the LOADLIN User Guide that comes with the Loadlin distribution before using.

-d file

Debug mode. Like **-**t, but sends output to file as well as to standard output.

-dskreset

Cause disks to be reset after loading but before booting Linux.

-noheap

For use by serious Linux hackers only; disable use of the setup heap.

-t

Test mode. Go through the loading process but don't actually start Linux. Also sets -v.

-txmode

Set the screen to text mode (80x25) on startup.

-v

Verbose. Print parameter and configuration information to standard output.

-wait=nn

After loading, wait nn (DOS) ticks before booting Linux.

In addition to these Loadlin options, the help message prints a number of kernel boot options that you can specify. The boot options that it prints are only a few of the many available boot options; see the BootPrompt HOWTO for a more complete list.

boot=/dev/hda2

title Red Hat Linux (2.4.18-14)

root (hd0,1)

kernel /vmlinuz-2.4.18-14 ro root=LABEL=/
hdc=ide-scsi

initrd /initrd-2.4.18-14.img

% dd if=/dev/hda2 of=/bootsect.lnx bs=512
count=1

% mount -t msdos /dev/fd0 /mnt

% cp /bootsect.lnx /mnt

% umount /mnt

C:> copy a:\bootsect.lnx c:\bootsect.lnx

C:> attrib -s -r c:\boot.ini

C:\bootsect.lnx="Linux"

[boot loader]

timeout=30

default=multi(0)disk(0)rdisk(0)partition(1)\WINNT

[operating systems]

multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windo
ws NT Server Version 4.

00"

multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windo
ws NT Server Version 4.

00 [VGA mode]" /basevideo /sos

C:\bootsect.lnx="Linux"

default=C:\bootsect.lnx

C:> attrib +s +r c:\boot.ini

OS Loader V4.00

Please select the operating system to start:

Windows NT Workstation Version 4.00

Windows NT Workstation Version 4.00 [VGA mode]

Linux

Select Linux, and the NT loader will read the Linux boot sector and transfer control to LILO or GRUB on the Linux partition.

If you are using LILO and you later modify /etc/lilo.conf or rebuild the kernel, you need to rerun the lilo command, create a new bootsect.lnx file, and replace the version of bootsect.lnx on the Windows NT partition with the new version. In other words, you need to rerun Steps 2-6.

If you have any problems or you simply want to remove LILO or GRUB later, you can reverse the installation procedure: boot to Windows NT, change the system and read-only attributes on boot.ini, re-edit boot.ini to remove the Linux entry, save the file, restore the system and read-only attributes, and

remove the Linux boot sector from the NT partition.

LILO boot: <press Tab>

linux test dos

boot: linux

boot: linux single

mem=128M

Because mem would have to be included on the command line for every boot, it often is specified on a command line saved with lock or with append to be added to the parameters passed to the kernel.

noinitrd

> When set, disable the two-stage boot and preserve the contents of /dev/initrd so the data is available after the kernel has booted. /dev/initrd can be read only once, and then its contents are returned to the system.

number

Start Linux at the runlevel specified by number. A runlevel is an operating state that the system can be booted to, such as a multiuser system or a system configuration running the X Window System. A runlevel is generally one of the numbers from 1 to 6; the default usually is 3. The runlevels and their corresponding states are defined in the file /etc/inittab. See the manpage for /etc/inittab for more information.

ro

Mount the root filesystem read-only. Used for doing system maintenance, such as checking the filesystem integrity, when you don't want anything written to the filesystem.

rw

Mount the root filesystem read/write. If neither ro nor rw is specified, the default value (usually rw) stored in the kernel image is used.

single

Start Linux in single-user mode. This option is used for system administration and recovery. It

gives you a root prompt as soon as the system boots, with minimal initialization. No other logins are allowed.

image=/vmlinuz

initrd=/boot/initrd # The file to load as the contents of /dev/initrd

...

kernel /vmlinuz-2.4.18-14 ro root=LABEL=/ hdc=ide-scsi

initrd /initrd-2.4.18-14.img

loadlin c:\linux\vmlinuz initrd=c:\linux\initrd

Chapter 5. Red Hat and Debian Package Managers

This chapter describes the two major Linux packaging systems: the Red Hat Package Manager (RPM) and the Debian GNU/Linux Package Manager.

When you install applications on your Linux system, most often you'll find a binary or a source package containing the application you want, instead of (or in addition to) a *.tar.gz* file. A package is a file containing the files necessary to install an application. However, while the package contains the files you need for installation, the application might require the presence of other files or packages that are not included, such as particular libraries (and even specific versions of the libraries), to actually be able to run. Such requirements are known as

dependencies.

Package management systems offer many benefits. As a user, you may want to query the package database to find out what packages are installed on the system and their versions. As a system administrator, you need tools to install and manage the packages on your system. And if you are a developer, you need to know how to build a package for distribution.

Among other things, package managers do the following:

- Provide tools for installing, updating, removing, and managing the software on your system.

- Allow you to install new or upgraded software directly across a network.

- Tell you what software package a particular file belongs to or what files a package contains.

- Maintain a database of packages on the system and their state, so you can find out what packages or versions are installed on your system.

- Provide dependency checking, so you don't mess up your system with incompatible software.

- Provide PGP, MD5, or other signature verification tools.

- Provide tools for building packages.

Any user can list or query packages. However, installing, upgrading, or removing packages generally requires superuser privileges. This is because the packages normally are installed in systemwide directories that are writable only by root. Sometimes you can specify an alternate directory to install a package into your home directory or into a project directory where you have write permission.

Both RPM and the Debian Package Manager back up old files before installing an updated package. Not only does this let you go back if there is a problem, but also ensures that you don't lose your changes (to configuration files, for example).

5.1 The Red Hat Package Manager

The Red Hat Package Manager (RPM) is a freely available packaging system for software distribution and installation. In addition to Red Hat and Red Hat-based distributions, both SuSE and Caldera are among the Linux distributions that use RPM.

Using RPM is straightforward. A single command, **rpm**, has options to perform all package management functions except building packages.[1] For example, to find out if the Emacs editor is installed on your system, you could say:

[1] In older versions of RPM, the build options were part of the **rpm** command.

```
% rpm -q emacs

emacs-21.2-18
```

The **rpmbuild** command is used to build both binary and source packages.

5.1.1 The rpm Command

RPM packages are built, installed, and queried with the **rpm** command. RPM package names usually end with a *.rpm* extension. **rpm** has a set of modes, each with its own options. The format of the **rpm** command is: **rpm** [*options*] [*packages*]

With a few exceptions, as noted in the lists of options that follow, the first option specifies the **rpm** mode (install, query, update, etc.), and any remaining options affect that mode.

Options that refer to packages are sometimes specified as *package-name* and sometimes as *package-file*. The package name is the

name of the program or application, such as **gif2png**. The package file is the name of the RPM file, such as *gif2png-2.4.6-1.i386.rpm*.

RPM provides a configuration file for specifying frequently used options. The default global configuration is usually */usr/lib/rpm/rpmrc*, the local system configuration file is */etc/rpmrc*, and users can set up their own *$HOME/.rpmrc* files. You can use the **--showrc** option to show the values RPM will use for all the options that may be set in an *rpmrc* file: **rpm --showrc**

The **rpm** command includes FTP and HTTP clients, so you can specify an *ftp://* or *http://* URL to install or query a package across the Internet. You can use an FTP or HTTP URL wherever *package-file* is specified in the commands presented here.

Any user can query the RPM database. Most of the other functions require superuser privileges.

5.1.1.1 General options

The following options can be used with all modes:

--dbpath *path*

Use *path* as the path to the RPM database instead of the default */var/lib/rpm*.

-?, **--help**

Print a long usage message (running **rpm** with no options gives a shorter usage message).

--pipe *command*

Pipe the **rpm** output to *command*.

--quiet

Display only error messages.

--rcfile *filelist*

Get configuration from the files in the colon-separated *filelist*. If **--rcfile** is specified, there must be at least one file in the list and the file must exist. *filelist* defaults to */var/lib/rpm/rpmrc:/usr/lib/rpm/redhat/rpmrc:~/.rpmrc*.

--root *dir*

Perform all operations within the directory tree rooted at *dir*.

-v

Verbose. Print progress messages.

--version

Print the version number of **rpm**.

-vv

Print debugging information.

5.1.1.2 Install, upgrade, and freshen options

Use the **install** command to install or upgrade an RPM package. The **install** syntax is: **rpm -i** [*install-options*] *package_file* ... **rpm --install** [*install-options*] *package_file* ...

To install a new version of a package and remove an existing version at the same time, use the **upgrade** command instead:

```
rpm -U  [install-options ] package_file  ...
```

rpm --upgrade [*install-options*] *package_file* ...

If the package doesn't already exist on the system, **-U** acts like **-i** and installs it. To prevent that behavior, you can **freshen** a package instead; in that case, **rpm** upgrades the package only if an earlier version is already installed. The **freshen** syntax is: **rpm -F** [*install-options*] *package_file* ... **rpm --freshen** [*install-options*] *package_file* ...

package-file can be specified as an FTP or HTTP URL to download the file before installing it. See Section 5.1.1.11.

The installation and upgrade options are:

--aid

If **rpm** suggests additional packages, add them to the list of package files.

--allfiles

Install or upgrade all files.

--badreloc

Used with **--relocate** to force relocation even if the package is not relocatable.

--excludedocs

Don't install any documentation files.

--excludepath *path*

Don't install any file whose filename begins with *path*.

--force

Force the installation. Equivalent to using all of **--replacepkgs**, **--replacefiles**, and **--oldpackage**.

-h, **--hash**

Print 50 hash marks as the package archive is unpacked. Use with **-v** or **--verbose** for a nicer display.

--ignorearch

Install even if the binary package is intended for a different architecture.

--ignoreos

Install binary package even if the operating systems don't match.

--ignoresize

Don't check disk space availability before installing.

--includedocs

Install documentation files. This is needed only if **excludedocs: 1** is specified in an *rpmrc* file.

--justdb

Update the database only; don't change any files.

--nodeps

Don't check whether this package depends on the presence of other packages.

--nodigest

Don't verify package or header digests.

--noorder

Don't reorder packages to satisfy dependencies before installing.

--nopost

Don't execute any post-install script.

--nopostun

Don't execute any post-uninstall script.

--nopre

Don't execute any pre-install script.

--nopreun

Don't execute any pre-uninstall script.

--noscripts

Don't execute any pre-install or post-install scripts. Equivalent to specifying all of **--nopre**, **--nopost**, **--nopreun**, and **--nopostun**.

--nosignature

Don't verify package or header signatures.

--nosuggest

Don't suggest packages that provide a missing dependency.

--notriggerin

Don't execute any install trigger scriptlet.

--notriggerun

Don't execute any uninstall trigger scriptlet.

--notriggerpostun

Don't execute any post-uninstall trigger scriptlet.

--notriggers

Don't execute any scripts triggered by package installation.

--oldpackage

Allow an upgrade to replace a newer package with an older one.

--percent

Print percent-completion messages as files are unpacked. Useful for running **rpm** from other tools.

--prefix *path*

Set the installation prefix to *path* for relocatable binary packages.

--relocate *oldpath*=*newpath*

For relocatable binary files, change all file paths from *oldpath* to *newpath*. Can be specified more than once to relocate multiple paths.

--repackage

Repackage the package files before erasing. Rename the package as specified by the macro **%_repackage_name_fmt** and save it in the directory specified by the macro **%_repackage_dir** (by default */var/tmp*).

--replacefiles

Install the packages even if they replace files from other installed packages.

--replacepkgs

Install the packages even if some of them are already installed.

--test

Go through the installation to see what it would do, but don't actually install the package. This option lets you test for problems before doing the installation.

5.1.1.3 Query options

The syntax for the **query** command is: **rpm -q** [*package-options*] [*information-options*] **rpm --query** [*package-options*] [*information-options*]

There are two subsets of query options. *Package selection* options determine what packages to query, and *information selection* options determine what information to provide.

5.1.1.4 Package selection options

package_name

Query the installed package *package_name*.

-a, **--all**

Query all installed packages.

-f *file*, **--file** *file*

Find out what package owns *file*.

--fileid *md5*

Query package with the specified MD5 digest.

-g *group*, **--group** *group*

Find out what packages have group *group*.

--hdrid *sha1*

Query package with the specified SHA1 digest in the package header.

-p *package_file*, **--package** *package_file*

Query the uninstalled package *package_file*, which can be a URL. If *package_file* is not a binary package, it is treated as a text file containing a package manifest, with each line of the manifest containing a path or one or more whitespace-separated glob expressions to be expanded to paths. These paths are then used instead of *package_file* as the query arguments. The manifest can contain comments that begin with a hash mark (#).

--pkgid *md5*

Query the package with a package identifier that is the given MD5 digest of the combined header and contents.

--querybynumber *num*

Query the *num*th database entry. Useful for debugging.

-qf, **--queryformat** *num*

Specify the format for displaying the query output, using tags to represent different types of data (e.g., NAME, FILENAME, DISTRIBUTION). The format specification is a variation of the standard **printf** formatting, with the type specifier omitted and replaced by the name of the header tag inclosed in brackets (**{ }**). For example:

```
%{NAME}
```

The tag names are case-insensitive. Use **--querytags** (see Section 5.1.1.10) to view a list of available tags. The tag can be followed by :*type* to get a different output format type. The possible types are:

armor

Wrap a public key in ASCII armor.

base64

Encode binary data as base64.

date

Use **strftime(3)** "%c" format.

day

Use **strftime(3)** "%a %b %d %Y" format.

depflags

Format dependency flags.

fflags

Format file flags.

hex

Use hexadecimal format.

octal

Use octal format.

perms

Format file permissions.

shescape

Escape single quotes for use in a script.

triggertype

Display trigger suffix.

--specfile *specfile*

Query *specfile* as if it were a package. Useful for extracting information from a spec file.

--tid *tid*

List packages with the specified transaction identifier (*tid*). The tid is a Unix timestamp. All packages installed or erased in a single transaction have the same tid.

--triggeredby *pkg*

List packages that are triggered by the installation of package *pkg*.

--whatrequires *capability*

List packages that require the given capability to function.

--whatprovides *capability*

List packages that provide the given capability.

5.1.1.5 Information selection options

-c, **--configfiles**

List configuration files in the package. Implies **-l**.

--changelog

Display the log of change information for the package.

-d, **--docfiles**

List documentation files in the package. Implies **-l**.

--dump

Dump information for each file in the package. This option must be used with at least one of **-l**, **-c**, or **-d**. The output includes the following information in this order:

```
path size mtime md5sum mode owner group
isconfig isdoc rdev symlink
```

--filesbypkg

List all files in each package.

-i, **--info**

Display package information, including the name, version, and description. Formats the results according to **--queryformat** if specified.

-l, **--list**

List all files in the package.

--last

List packages by install time, with the latest packages listed first.

--provides

List the capabilities this package provides.

-R, **--requires**

List any packages this package depends on.

-s, **--state**

List each file in the package and its state. The possible states are **normal**, **not installed**, or **replaced**. Implies **-l**.

--scripts

List any package-specific shell scripts used during installation and uninstallation of the package.

--triggers, **--triggerscript**

Display any trigger scripts in the package.

5.1.1.6 Uninstall options

The syntax for **erase**, the uninstall command, is: **rpm -e** *package_name* ... **rpm --erase** *package_name* ...

The uninstall options are:

--allmatches

Remove all versions of the package. Only one package should be specified; otherwise, an error results.

--nodeps

Don't check dependencies before uninstalling the package.

--nopostun

Don't run any post-uninstall scripts.

--nopreun

Don't run any pre-uninstall scripts.

--noscripts

Don't execute any pre-uninstall or post-uninstall scripts. Equivalent to **--nopreun --nopostun**.

--notriggerpostun

Don't execute any post-uninstall scripts triggered by the removal of this package.

--notriggers

Don't execute any scripts triggered by the removal of this package. Equivalent to **--notriggerun --notriggerpostun**.

--notriggerun

Don't execute any uninstall scripts triggered by the removal of this package.

--repackage

Repackage the files before uninstalling them. Rename the package as specified by the macro **%_repackage_name_fmt**

and save it in the directory specified by the macro **%_repackage_dir** (by default */var/tmp*).

--test

Don't really uninstall anything; just go through the motions. Use with **-vv** for debugging.

5.1.1.7 Verify options

The syntax for the **verify** command is: **rpm -V|-y|--verify** [*package-selection-options*] [*verify-options*]

Verify mode compares information about the installed files in a package with information about the files that came in the original package, and displays any discrepancies. The information compared includes the size, MD5 sum, permissions, type, owner, and group of each file. Uninstalled files are ignored.

The package selection options include those available for query mode. In addition, the following **verify** options are available:

--nodeps

Ignore package dependencies.

--nodigest

Ignore package or header digests.

--nofiles

Ignore attributes of package files.

--nogroup

Ignore group ownership errors.

--nolinkto

Ignore symbolic link errors.

--nomd5

Ignore MD5 checksum errors.

--nomode

Ignore file mode (permissions) errors.

--nordev

Ignore major and minor device number errors.

--nomtime

Ignore modification time errors.

--noscripts

Ignore any verify script.

--nosignature

Ignore package or header signatures.

--nosize

Ignore file size errors.

--nouser

Ignore user ownership errors.

The output is formatted as an eight-character string, possibly followed by an attribute marker, and then the filename. The possible attribute markers are:

c	Configuration file
d	Documentation file
g	Ghost file (contents not included in package)
l	License file
r	Readme file

Each of the eight characters in the string represents the result of comparing one file attribute to the value of that attribute from the RPM database. A period (.) indicates that the file passed that test. The following characters indicate failure of the corresponding test:

5	MD5 sum
D	Device
G	Group
L	Symlink
M	Mode (includes permissions and file type)

S	File size
T	Mtime
U	User

5.1.1.8 Database rebuild options

The syntax of the command to rebuild the RPM database is:

```
rpm --rebuilddb  [options ]
```

You also can build a new database: **rpm --initdb** [*options*]

The options available with the database rebuild mode are the **--dbpath**, **--root**, and **-v** options described earlier under Section 5.1.1.1.

5.1.1.9 Signature check options

RPM packages may have a PGP signature built into them. PGP configuration information is read from the *rpmrc* file. There are three types of digital signature options: you can check signatures, add signatures to packages, and import signatures.

The syntax of the signature check mode is: **rpm --checksig** *package_file* ... **rpm -K** *package_file* ...

The signature checking options **-K** and **--checksig** check the digests and signatures contained in the specified packages to insure the integrity and origin of the packages. Note that RPM now

automatically checks the signature of any package when it is read; this option is still useful, however, for checking all headers and signatures associated with a package.

The following options are available for use with signature check mode:

--nogpg

Don't check any GPG signatures.

--nomd5

Don't check any MD5 signatures.

--nopgp

Don't check any PGP signatures.

The syntax for adding signatures to binary packages is: **rpm --addsign** *binary-pkgfile* ... **rpm --resign** *binary-pkgfile* ...

Both **--addsign** and **--resign** generate and insert new signatures, replacing any that already exist in the specified binary packages.[2]

[2] In older versions of RPM, **--addsign** was used to add new signatures without replacing existing ones, but currently both options work the same way and replace any existing signatures.

The syntax for importing signatures is: **rpm --import** *public-key*

The **--import** option is used to import an ASCII public key to the RPM database so that digital signatures for packages using that key can be verified. Imported public keys are carried in headers, and keys are kept in a ring, which can be queried and managed like any package file.

5.1.1.10 Miscellaneous options

Several additional **rpm** options are available:

--querytags

Print the tags available for use with the **--queryformat** option in query mode.

--setperms *packages*

Set file permissions of the specified packages to those in the database.

--setugids *packages*

Set file owner and group of the specified packages to those in the database.

--showrc

Show the values **rpm** will use for all options that can be set in an *rpmrc* file.

5.1.1.11 FTP/HTTP options

The following options are available for use with FTP and HTTP URLs in install, update, and query modes.

--ftpport *port*

Use *port* for making an FTP connection on the proxy FTP server instead of the default port. Same as specifying the macro **%_ftpport**.

--ftpproxy *host*

Use *host* as the proxy server for FTP transfers through a firewall that uses a proxy. Same as specifying the macro

%_ftpproxy.

--httpport *port*

Use *port* for making an HTTP connection on the proxy HTTP server instead of the default port. Same as specifying the macro **%_httpport**.

--httpproxy *host*

Use *host* as the proxy server for HTTP transfers. Same as specifying the macro **%_httpproxy**.

5.1.2 The rpmbuild Command

The **rpmbuild** command is used to build RPM packages. The syntax for **rpmbuild** is: **rpmbuild -** [**b** |**t**]*step* [*build-options*] *spec-file* ...

Specify **-b** to build a package directly from a spec file, or **-t** to open a tarred, gzipped file and use its spec file.

Both forms take the following single-character *step* arguments, listed in the order they would be performed:

p

Perform the prep stage, unpacking source files and applying patches.

l

Do a list check, expanding macros in the files section of the spec file and verifying that each file exists.

c

Perform the build stage. Done after the prep stage; generally equivalent to doing a **make**.

i

Perform the install stage. Done after the prep and build stages; generally equivalent to doing a **make install**.

b

Build a binary package. Done after prep, build, and install.

s

Build a source package. Done after prep, build, and install.

a

Build both binary and source packages. Done after prep, build, and install.

The general **rpm** options described earlier in Section 5.1.1.1 can be used with **rpmbuild**.

The following additional options can also be used when building an **rpm** file with **rpmbuild**:

--buildroot *dir*

Override the **BuildRoot** tag with *dir* when building the package.

--clean

Clean up (remove) the build files after the package has been made.

--nobuild

Go through the motions, but don't execute any build stages. Used for testing spec files.

--rmsource

Remove the source files when the build is done. Can be used as a standalone option with **rpm** to clean up files separately from creating the packages.

--rmspec

Remove the spec file when the build is done. Like **--rmsource**, **--rmspec** can be used as a standalone option with **rpmbuild**.

--short-circuit

Can be used with **-bc** and **-bi** to skip previous stages.

--sign

Add a GPG signature to the package for verifying its identity and origin.

--target *platform*

When building the package, set the macros **%_target**, **%_target_arch**, and **%_target_os** to the value indicated by *platform*.

Two other options can be used standalone with **rpmbuild** to recompile or rebuild a package:

--rebuild *source-pkgfile*...

Like **--recompile**, but also build a new binary package. Remove the build directory, the source files, and the spec file once the build is complete.

--recompile *source-pkgfile*...

Install the named source package, and prep, compile, and install the package.

Finally, the **--showrc** option is used to show the current **rpmbuild** configuration:

```
rpmbuild --showrc
```

This option shows the values that will be used for all options that can be set in an *rpmrc* file.

5.1.3 RPM Examples

Query the RPM database to find Emacs-related packages: % **rpm -q -a | grep emacs**

Query an uninstalled package, printing information about the package and listing the files it contains:

```
% rpm -qpil ~/downloads/bash2-doc-2.03-8.i386.rpm
```

Install a package (assumes superuser privileges):

```
% rpm -i sudo-1.5.3-6.i386.rpm
```

apt-cache [<tt>options</tt>] <tt>command</tt>

apt-cdrom [<tt>options</tt>] <tt>command</tt>

apt-config [<tt>options</tt>] shell <tt>args</tt>

apt-config [<tt>options</tt>] dump

apt-extracttemplates [<tt>options</tt>] <tt>files</tt>

<tt>package version template-file config-script</tt>

apt-ftparchive [<tt>options</tt>] <tt>command</tt>

apt-get [<tt>options</tt>] <tt>command</tt> [<tt>package</tt>...]

apt-get -f install

apt-get dist-upgrade

apt-sortpkgs [<tt>options</tt>] <tt>indexfiles</tt>

dpkg [<tt>options</tt>] <tt>action</tt>

dpkg-deb <tt>action</tt> [<tt>options</tt>]

dpkg-query [<tt>option</tt>] <tt>command</tt>

dpkg-split [<tt>action</tt>] [<tt>options</tt>]

dselect [<tt>options</tt>] [<tt>action</tt>]

* 0. [A]ccess Choose the access method to use.

1. [U]pdate Update list of available packages, if possible.

2. [S]elect Request which packages you want on your system.

3. [I]nstall Install and upgrade wanted packages.

4. [C]onfig Configure any packages that are unconfigured.

5. [R]emove Remove unwanted software.

6. [Q]uit Quit dselect.

% dselect access

screenpart:[<tt>fgcolor</tt>],[<tt>bgcolor</tt>][:<tt>attr</tt>[+<tt>attr</tt>+...]]

This option can be specified multiple times, to override the default colors for different screen parts. Rather than having to specify the colors on the command line each time you run dselect, you might prefer to set them in the configuration file. The possible screen parts (going from the top of the screen to the bottom) are:

title

The screen title.

listhead

The header line above the package list.

list

The scrolling list of packages and some help text.

listsel

The selected item in the list.

pkgstate

The text showing the current state of each package.

pkgstatesel

The text showing the current state of the selected package.

infohead

The header line showing the state of the selected package.

infodesc

The short description of the package.

info

The text that displays information such as the package description.

infofoot

The last line of the screen when selecting packages.

query

Query lines.

helpscreen

The color of help screens.

Either the foreground color, the background color, or both can be specified for each screen part. The colors are given as the standard curses colors. After the color specification, you can specify a list of attributes separated by plus signs (+). The possible attributes are normal, standout, underline, reverse, blink, bright, dim, and bold. Not all attributes work on all terminals.

--expert

Run in expert mode; don't print help messages.

-D [file], --debug [file]

Turn on debugging. Send output to file if specified.

--help

Print help message and exit.

--license, licence

Print license information and exit.

--version

Print version information and exit.

Chapter 6. The Linux Shells: An Overview

The *shell* is a program that acts as a buffer between you and the operating system. In its role as a command interpreter, it should (for the most part) act invisibly. It can also be used for simple programming.

6.1 Purpose of the Shell

There are three main uses for the shell: interactive use, customizing your Linux session, and programming.

6.1.1 Interactive Use

When the shell is used interactively, it waits for you to issue commands, processes them (to interpret special characters such as wildcards), and executes them. Shells also provide a set of commands, known as *built-ins*, to supplement Linux commands.

6.1.2 Customizing Your Linux Session

A Linux shell defines

variables, such as the locations of your home directory and mail spool, to control the behavior of your session. Some variables are preset by the system; you can define others in startup files that your shell reads when you log in, or interactively for a single session. Startup files

can also contain Linux or shell commands, for execution immediately after login.

6.1.3 Programming

A series of individual commands (be they shell commands or other Linux commands available on the system) combined into one executable file is called a *shell*

script. Batch files in MS-DOS are a similar concept. **bash** is considered a powerful programming shell, while scripting in **tcsh** is rumored to be hazardous to your health.

$ exec tcsh

6.3 Common Features

The following table is a sampling of features that are common to **bash** and **tcsh**.

Symbol/Command	Meaning/Action
>	Redirect output.
>>	Append output to file.
<	Redirect input.
<<	"Here" document (redirect input).
|	Pipe output.
&	Run process in background.
;	Separate commands on same line.
*****	Match any character(s) in filename.
?	Match single character in filename.

Symbol/Command	Meaning/Action
!*n*	Repeat command number *n*.
[ ]	Match any characters enclosed.
( )	Execute in subshell.
" "	Partial quote (allows variable and command expansion).
' '	Preserve literal value of enclosed characters.
****	Quote following character.
$*var*	Use value for variable.
$$	Process ID.
$0	Command name.
$*n*	*n*th argument (0<*n*≤9).
$*	All arguments.

Symbol/Command	Meaning/Action
$?	Exit status.
#	Begin comment.
Tab	Complete current word.
bg	Background execution.
break	Break from loop statements.
cd	Change directories.
continue	Resume a program loop.
echo	Display output.
eval	Evaluate arguments.
exec	Execute a new shell or other program.
fg	Foreground execution.

Symbol/Command	Meaning/Action
jobs	Show active jobs.
kill	Terminate running jobs.
newgrp	Change to a new group.
shift	Shift positional parameters.
stop	Suspend a background job.
suspend	Suspend a foreground job.
umask	Set or list permissions on files to be created.
unset	Erase variable or function definitions.
wait	Wait for a background job to finish.

6.4 Differing Features

The following table is a sampling of features that differ between the two shells:

bash	tcsh	Meaning/Action
$	**%**	Default prompt.
>|	**>!**	Force redirection.
	>>!	Force append.
var=val	**set** *var=val*	Variable assignment.
export *var=val*	**setenv** *var val*	Set environment variable.
$#	**$#argv**	Number of arguments.
$(*command***),** `` ` ``	`` ` ``	Command substitution.
. *file*, **source** *file*	**source** *file*	Execute commands in *file*.
done	**end**	End a loop statement.

bash	tcsh	Meaning/Action
esac	**endsw**	End **case** or **switch**.
for/do	**foreach**	Loop through variables.
if [$i -eq 5]	**if ($i= =5)**	Sample **if** statement.
fi	**endif**	End **if** statement.
ulimit	**limit**	Set resource limits.
read	**$<**	Read from terminal.
readonly	**set -r**	Make a variable read-only.
Tab Tab		Show possible completions.
trap 2	**onintr**	Ignore interrupts.
until/do	**until**	Begin **until** loop.
while/do	**while**	Begin **while** loop.

```
# Set bash variable to keep 50 commands in history.


HSTSIZE=50
```

#

Set prompt to show current working directory and history number of

command.

PS1='\w: Command \!$ '

#

Set path to search for commands in my directories, then standard ones.

PATH=~/bin:~/scripts:$PATH

#

Keep group and others from writing my newly created files.

umask 022

#

Show color-coded file types.

alias ls='ls --color=yes'

#

Make executable and .o files ugly yellow so I can find and delete them.

export LS_COLORS="ex=43:*.o=43"

```
#

# Quick and dirty test of a single-file program.

function gtst ( ) {

  g++ -o $1 $1.C && ./$1

}
```

#

Remove .o files.

alias clean='find ~ -name *.o -exec rm { } \;'

bash provides the following features:

- Input/output redirection
- Wildcard characters (metacharacters) for filename abbreviation
- Shell variables for customizing your environment
- Powerful programming capabilities
- Command-line editing (using vi- or Emacs-style editing commands)
- Access to previous commands (command history)

- Integer arithmetic
- Arithmetic expressions
- Command name abbreviation (aliasing)
- Job control
- Integrated programming features
- Control structures
- Directory stacking (using pushd and popd)
- Brace/tilde expansion
- Key bindings

bash [options] [arguments]

bash can execute commands from a terminal (when -i is specified), from a file (when the first argument is an executable script), or from standard input (if no arguments remain or if -s is specified).

7.1.1 Options

Options that appear here with double hyphens also work when entered with single hyphens, but using double hyphens is standard coding procedure.

-, --

> Treat all subsequent strings as arguments, not options.

-D, --dump-strings

> For execution in non-English locales, dump all strings that bash translates.

--dump-po-strings

Same as --dump-strings, but uses the GNU gettext po (portable object) format suitable for scripting.

-c str

Read commands from string str.

--help

Print usage information and exit.

-i

Create an interactive shell (prompt for input).

-l, --login

Behave like a login shell; try to process /etc/profile on startup. Then process ~/.bash_profile, ~/.bash_login, or ~/.profile (searching for each file only if the previous file is not found).

--noediting

Disable line editing with arrow and control keys.

--noprofile

Do not process /etc/profile, ~/.bash_profile, ~/.bash_login, or ~/.profile on startup.

--norc

Do not process ~/.bashrc on startup.

--posix

Conform to POSIX standard.

-r, --restricted

Restrict users to a very secure, limited environment; for instance, they cannot change out of the startup directory or use the > sign to redirect output.

--rcfile file

Substitute file for .bashrc on startup.

-s

Read commands from standard input. Output from built-in commands goes to file descriptor 1; all other shell output goes to file descriptor 2.

-v, --verbose

Print each line as it is executed (useful for tracing scripts).

--version

Print information about which version of bash is installed.

-x

Turn on debugging, as described under the -x option to the set built-in command later in this chapter.

The remaining options to bash are listed under the set built-in command.

7.1.2 Arguments

Arguments are assigned, in order, to the positional parameters $1, $2, and so forth. If the first argument is an executable script, it is assigned to $0; then commands are read from it, and remaining arguments are assigned to $1, $2, and so on.

7.2 Syntax

This subsection describes the many symbols peculiar to **bash**. The topics are arranged as follows:

- Special files
- Filename metacharacters
- Command-line editing
- Quoting
- Command forms
- Redirection forms
- Coprocesses

7.2.1 Special Files

File	Purpose
/etc/profile	Executed automatically at login.
$HOME/.bash_profile	Executed automatically at login.

File	Purpose
$HOME/.bashrc	Executed automatically at shell startup.
$HOME/.bash_logout	Executed automatically at logout.
$HOME/.bash_history	Record of last session's commands.
$HOME/.inputrc	Initialization file for reading input in an interactive shell.
/etc/passwd	Source of home directories for **~***name* abbreviations.

7.2.2 Filename Metacharacters

Characters	Meaning
*****	Match any string of zero or more characters.
?	Match any single character.
[*abc***...]**	Match any one of the enclosed characters; a hyphen can be used to specify a range (e.g., a-z, A-Z, 0-9).

Characters	Meaning
[!*abc***...]**	Match any character *not* among the enclosed characters.
[^*abc***...]**	Same as **[!***abc***...]**.
{*str1***,...}**	Brace expansion: match any of the enclosed strings.
~*name*	HOME directory of user *name*. With no *name*, HOME directory of current user.
~+	Current working directory (PWD).
~-	Previous working directory from directory stack (OLDPWD; see also the **pushd** built-in command).
~+*n*	The *n*th entry in the directory stack, counting from the start of the list with the first entry being 0.
~-*n*	The *n*th entry in the directory stack, counting from the end of the list with the last entry being 0.

Patterns can be a sequence of patterns separated by **|**. If any of the subpatterns match, the entire sequence is considered matching. This extended syntax resembles that of **egrep** and **awk**.

7.2.2.1 Examples

```
$ ls new*              List new   and new.1

$ cat ch?              Match ch9   but not ch10

$ vi [D-R]*            Match files that begin with
uppercase D through R
```

7.2.3 Command-Line Editing

Command lines can be edited like lines in either Emacs or **vi**. Emacs is the default. See Section 7.5.1 later in this chapter for more information.

vi mode has two submodes, input mode and command mode. The default is input mode; you can go to command mode by pressing Esc. In command mode, typing **a** (append) or **i** (insert) returns you to input mode.

Some users discover that the Del or Backspace key on the terminal does not delete the character before the cursor, as it should. Sometimes this problem can be solved by issuing one of the following commands (or placing it in your *.bashrc* file): stty erase ^? stty erase ^H

See the **stty** command in Chapter 3 for more information. On the X Window System, an alternative solution is to use the **xmodmap** command; this is not described here, as it requires some research about your terminal.

Tables Table 7-1 through Table 7-14 show various Emacs and **vi** commands.

Table 7-1. Basic Emacs-mode commands

Command	Description
Ctrl-b	Move backward one character (without deleting).
Ctrl-f	Move forward one character.
Del	Delete one character backward.
Ctrl-d	Delete one character forward.

Table 7-2. Emacs-mode word commands

Command	Description
M-b	Move one word backward.
M-f	Move one word forward.
M-Del	Kill one word backward.

Command	Description
M-d	Kill one word forward.
Ctrl-y	Retrieve (**yank**) last item killed.

Table 7-3. Emacs-mode line commands

Command	Description
Ctrl-a	Move to beginning of line.
Ctrl-e	Move to end of line.
Ctrl-k	Kill forward to end of line.

Table 7-4. Emacs-mode commands for moving through the history file

Command	Description

Command	Description
Ctrl-p	Move to previous command.
Ctrl-n	Move to next command.
Ctrl-r	Search backward.
M-<	Move to first line of history file.
M->	Move to last line of history file.

Table 7-5. Emacs-mode completion commands

Command	Description
Tab	Attempt to perform general completion of the text.
M-?	List the possible completions.
M-/	Attempt filename completion.

Command	Description
Ctrl-x /	List the possible filename completions.
M-~	Attempt username completion.
Ctrl-x ~	List the possible username completions.
M-$	Attempt variable completion.
Ctrl-x $	List the possible variable completions.
M-@	Attempt hostname completion.
Ctrl-x @	List the possible hostname completions.
M-!	Attempt command completion.
Ctrl-x !	List the possible command completions.
M-Tab	Attempt completion from previous commands in the history list.

Table 7-6. Miscellaneous Emacs-mode commands

Command	Description
Ctrl-j	Same as Return.
Ctrl-l	Clear the screen, placing the current line at the top of the screen.
Ctrl-m	Same as Return.
Ctrl-o	Same as Return, then display next line in command history.
Ctrl-t	Transpose character left of and under the cursor.
Ctrl-u	Kill the line from the beginning to point.
Ctrl-v	Insert keypress instead of interpreting it as a command.
Ctrl-[	Same as Esc (most keyboards).
M-c	Capitalize word under or after cursor.
M-u	Change word under or after cursor to all capital letters.

Command	Description
M-l	Change word under or after cursor to all lowercase letters.
M-.	Insert last word in previous command line after point.
M-_	Same as **M-.** .

Table 7-7. Editing commands in vi input mode

Command	Description
Del	Delete previous character.
Ctrl-W	Erase previous word (i.e., erase until a blank).
Ctrl-V	Insert next keypress instead of interpreting it as a command.
Esc	Enter command mode (see Table 7-8).

Table 7-8. Basic vi command-mode commands

Command	Description
h	Move left one character.
l	Move right one character.
b	Move left one word.
w	Move right one word.
B	Move to beginning of preceding nonblank word.
W	Move to beginning of next nonblank word.
e	Move to end of current word.
E	Move to end of current nonblank word.
0	Move to beginning of line.
^	Move to first nonblank character in line.

Command	Description
$	Move to end of line.

Table 7-9. Commands for entering vi input mode

Command	Description
i	Insert text before current character (insert).
a	Insert text after current character (append).
I	Insert text at beginning of line.
A	Insert text at end of line.
r	Replace current character with next keypress.
R	Overwrite existing text.

Table 7-10. Some vi-mode delete commands

Command	Description
dh	Delete one character backward.
dl	Delete the current character.
db	Delete one word backward.
dw	Delete one word forward.
dB	Delete one nonblank word backward.
dW	Delete one nonblank word forward.
d$	Delete to end-of-line.
d0	Delete to beginning of line.

Table 7-11. Abbreviations for vi-mode delete commands

Command	Description

Command	Description
D	Delete to end of line (equivalent to **d$**).
dd	Delete entire line (equivalent to **0d$**).
C	Delete to end of line; enter input mode (equivalent to **c$**).
cc	Delete entire line; enter input mode (equivalent to **0c$**).
X	Delete character backward (equivalent to **dh**).
x	Delete the current character (equivalent to **dl**).

Table 7-12. vi-mode commands for searching the command history

Command	Description
k or **-**	Move backward one line.
j or **+**	Move forward one line.

Command	Description
G	Move to first line in history.
/string	Search backward for *string*.
?*string*	Search forward for *string*.
n	Repeat search in same direction as previous.
N	Repeat search in opposite direction of previous.

Table 7-13. vi-mode character-finding commands

Command	Description
f*x*	Move right to next occurrence of *x*.
F*x*	Move left to previous occurrence of *x*.
t*x*	Move right to next occurrence of *x*, then back one space.

Command	Description
T*x*	Move left to previous occurrence of *x*, then forward one space.
;	Redo last character-finding command.
,	Redo last character-finding command in opposite direction.

Table 7-14. Miscellaneous vi-mode commands

Command	Description
~	Invert (toggle) case of current character(s).
_	Insert last word of previous command after cursor; enter input mode.
Ctrl-L	Clear the screen and redraw the current line on it; good for when your screen becomes garbled.

Command	Description
#	Prepend # (comment character) to the line and send it to the history file; useful for saving a command to be executed later, without having to retype it.

7.2.4 Quoting

Quoting disables a character's special meaning and allows it to be used literally, as itself. The following characters have special meaning to **bash**:

Character	Meaning
;	Command separator
&	Background execution
( )	Command grouping (enter a subshell)
{ }	Command block
|	Pipe
> < &	Redirection symbols

Character	Meaning
*** ? [ ] ~ !**	Filename metacharacters
**" ' **	Used in quoting other characters
`	Command substitution
$	Variable substitution (or command substitution)
newline space tab	Word separators
#	Comment

The following characters can be used for quoting:

Character	Action

Character	Action
" "	Everything between " and " is taken literally, except for the following characters that keep their special meaning: **$** Variable substitution will occur. ` Command substitution will occur. " This marks the end of the double quote.
' '	Everything between ' and ' is taken literally, except for another '.
****	The character following **** is taken literally. Use within " " to escape ", **$**, and '. Often used to escape itself, spaces, or newlines.

7.2.4.1 Examples

```
$ echo 'Single quotes "protect" double quotes'

Single quotes "protect" double quotes
```

```
$ echo "Well, isn't that \"special\"?"
Well, isn't that "special"?

$ echo "You have `ls|wc -l` files in `pwd`"
You have  43 files in /home/bob

$ echo "The value of \$x is $x"
The value of $x is 100
```

7.2.5 Command Forms

Syntax	Effect
cmd **&**	Execute *cmd* in background.
cmd1 **;** *cmd2*	Command sequence; execute multiple *cmds* on the same line.
(*cmd1* **;** *cmd2***)**	Subshell; treat *cmd1* and *cmd2* as a command group.
cmd1 **|** *cmd2*	Pipe; use output from *cmd1* as input to *cmd2*.

Syntax	Effect
cmd1 `` `*cmd2*` ``	Command substitution; use *cmd2* output as arguments to *cmd1*.
cmd1 **$(***cmd2***)**	Command substitution; nesting is allowed.
cmd1 **&&** *cmd2*	AND; execute *cmd2* only if *cmd1* succeeds.
cmd1 **||** *cmd2*	OR; execute *cmd2* only if *cmd1* fails.
{ *cmd1* **;** *cmd2* **}**	Execute commands in the current shell.

7.2.5.1 Examples

```
$ cd; ls                                    Execute
sequentially

$ (date; who; pwd) > logfile                All
output is redirected

$ sort file | pr -3 | lp                    Sort
file, page output, then print

$ vi `grep -l ifdef *.c`                    Edit
```

```
files found by grep

$ egrep '(yes|no)' `cat list`            Specify a
list of files to search

$ egrep '(yes|no)' $(cat list)           Same as
previous using bash command


substitution

$ egrep '(yes|no)' $(<list)              Same, but
faster

$ grep XX file && lp file                Print
file if it contains the pattern

$ grep XX file || echo "XX not found"    Echo an
error message if pattern not found
```

7.2.6 Redirection Forms

File descriptor	Name	Common abbreviation	Typical default
0	Standard input	stdin	Keyboard
1	Standard output	stdout	Screen
2	Standard error	stderr	Screen

The usual input source or output destination can be changed as shown in Table 7-15.

Table 7-15. I/O redirectors

Redirector	Function
> *file*	Direct standard output to *file*.
< *file*	Take standard input from *file*.
cmd1 **|** *cmd2*	Pipe; take standard output of *cmd1* as standard input to *cmd2*.
>> *file*	Direct standard output to *file*; append to *file* if it already exists.
>| *file*	Force standard output to *file* even if **noclobber** is set.
*n***>|** *file*	Force output from the file descriptor *n* to *file* even if **noclobber** is set.
<> *file*	Use *file* as both standard input and standard output.

Redirector	Function
<< *text*	Read standard input up to a line identical to *text* (*text* can be stored in a shell variable). Input is usually typed on the screen or in the shell program. Commands that typically use this syntax include **cat**, **echo**, **ex**, and **sed**. If *text* is enclosed in quotes, standard input will not undergo variable substitution, command substitution, etc.
*n***>** *file*	Direct file descriptor *n* to *file*.
*n***<** *file*	Set *file* as file descriptor *n*.
>&*n*	Duplicate standard output to file descriptor *n*.
<&*n*	Duplicate standard input from file descriptor *n*.
&>*file*	Direct standard output and standard error to *file*.
<&-	Close the standard input.
>&-	Close the standard output.
*n***>&-**	Close the output from file descriptor *n*.

Redirector	Function
n<&-	Close the input from file descriptor *n*.

7.2.6.1 Examples

```
$ cat part1 > book

$ cat part2 part3 >> book

$ mail tim < report

$ grep Chapter part* 2> error_file


$ sed 's/^/XX /' << END_ARCHIVE

> This is often how a shell archive is "wrapped",

> bundling text for distribution. You would
normally

> run sed from a shell program, not from the
command line.

> END_ARCHIVE

XX This is often how a shell archive is "wrapped",

XX bundling text for distribution. You would
normally
```

```
XX run sed from a shell program, not from the
command line.
```

To redirect standard output to standard error:

```
$ echo "Usage error:  see administrator" 1>&2
```

The following command sends output (files found) to *filelist* and sends error messages (inaccessible files) to file *no_access*:

```
$ find / -print > filelist 2>no_access
```

7.3 Variables

Preface a variable by a dollar sign (**$**) to reference its value. You can also optionally enclose it in braces (**{ }**). You can assign a value to a variable through an equals sign (**=**) with no whitespace on either side of it:

```
$ TMP=temp.file
```

By default, variables are seen only within the shell itself; to pass variables to other programs invoked within the shell, see the **export** built-in command.

If enclosed by brackets (**[ ]**), the variable is considered an array variable. For instance:

```
$ DIR_LIST[0]=src

$ DIR_LIST[1]=headers

$ ls ${DIR_LIST[1]}
```

The contents of **headers** are listed. Many substitutions and commands in this chapter handle arrays by operating on each element separately.

7.3.1 Variable Substitution

In the following substitutions, braces (**{ }**) are optional, except when needed to separate a variable name from following characters that would otherwise be considered part of the name.

Variable	Meaning
`${var}`	Value of variable *var*.
`$0`	Name of the program.
`${n}`	Individual arguments on command line (positional parameters); $1 \leq n \leq 9$.
`$#`	Number of arguments on command line.
`$*`	All arguments on command line.
`$@`	Same as $*, but contents are split into words when the variable is enclosed in double quotes.
`$$`	Process number of current shell; useful as part of a filename for creating temporary files with unique names.
`$?`	Exit status of last command (normally 0 for success).
`$!`	Process number of most recently issued background command.

Variable	Meaning
$-	Current execution options (see the **set** built-in command). By default, **hB** for scripts and **himBH** for interactive shells.
$_	Initially set to name of file invoked for this shell, then set for each command to the last word of the previous command.

Tables Table 7-16 through Table 7-18 show various types of operators that can be used with **bash** variables.

Table 7-16. Substitution operators

Operator	Substitution
${*varname***:-***word***}**	If *varname* exists and isn't null, return its value; otherwise, return *word*.
Purpose:	Returning a default value if the variable is undefined.
Example:	**${count:-0}** evaluates to 0 if **count** is undefined.

Operator	Substitution
${*varname***:=***word***}**	If *varname* exists and isn't null, return its value; otherwise set it to *word* and then return its value. Positional and special parameters cannot be assigned this way.
Purpose:	Setting a variable to a default value if it is undefined.
Example:	**${count:=0}** sets **count** to 0 if it is undefined.
${*varname***:?***message***}**	If *varname* exists and isn't null, return its value; otherwise, print *varname*: followed by *message*, and abort the current command or script (noninteractive shells only). Omitting *message* produces the default message "parameter null or not set."
Purpose:	Catching errors that result from variables being undefined.
Example:	**{count:?"undefined"}** prints "count: undefined" and exits if **count** is undefined.
${*varname***:+***word***}**	If *varname* exists and isn't null, return *word*; otherwise, return null.

Operator	Substitution
Purpose:	Testing for the existence of a variable.
Example:	**${count:+1}** returns 1 (which could mean **true**) if **count** is defined.
${*#varname***}**	Return the number of characters in the value of *varname*.
Purpose:	Preparing for substitution or extraction of substrings.
Example:	If **${USER}** currently expands to **root**, **${#USER}** expands to 4.

Table 7-17. Pattern-matching operators

Operator	Meaning
${*variable#pattern}*	If the pattern matches the beginning of the variable's value, delete the shortest part that matches and return the rest.

Operator	Meaning
${*variable***##***pattern***}**	If the pattern matches the beginning of the variable's value, delete the longest part that matches and return the rest.
${*variable***%***pattern***}**	If the pattern matches the end of the variable's value, delete the shortest part that matches and return the rest.
${*variable***%%***pattern***}**	If the pattern matches the end of the variable's value, delete the longest part that matches and return the rest.
${*var***/***pat***/***sub***}**	Return *var* with the first occurrence of *pat* replaced by *sub*. Can be applied to **$*** or **$@**, in which case each word is treated separately. If *pat* starts with # it can match only the start of *var*; if *pat* ends with **%** it can match only the end of *var*.
${*var***//***pat***/***sub***}**	Return *var* with every occurrence of *pat* replaced by *sub*.
${*variable***:***n***}**	Truncate the beginning of the variable and return the part starting with character number *n*, where the first character is 0.

Operator	Meaning
${*variable*:*n*:*l***}**	Starting with character number *n*, where the first character is 0, return a substring of length *l* from the variable.

Table 7-18. Expression evaluation

Operator	Meaning
$((*arithmetic-expression***))**	Return the result of the expression. Arithmetic operators are described under Section 7.4.
Example:	**TODAY=`date +%-d`; echo $(($TODAY+7))** stores the number of the current day in **$TODAY** and then prints that number plus 7 (the number of the same day next week).
[[$*condition***]]**	Return 1 if *condition* is true and 0 if it is false. Conditions are described under the **test** built-in command.

7.3.2 Built-in Shell Variables

Built-in variables are set automatically by the shell and are typically used inside shell scripts. Built-in variables can use the variable substitution patterns shown earlier. When setting variables, you do not include dollar signs, but when referencing their values later, the dollar signs are necessary.

Tables Table 7-19 through Table 7-22 show the commonly used built-in variables in **bash**.

Table 7-19. Behavior-altering variables

Variable	Meaning
auto_resume	Allows a background job to be brought to the foreground simply by entering a substring of the job's command line. Values can be **substring** (resume if the user's string matches part of the command), **exact** (string must exactly match command), or another value (string must match at beginning of command).
BASH_ENV	Startup file of commands to execute, if **bash** is invoked to run a script.
CDPATH	Colon-separated list of directories to search for the directory passed in a **cd** command.
EDITOR	Pathname of your preferred text editor.

Variable	Meaning
IFS	Word separator; used by shell to parse commands into their elements. The default separators are space, tab, and newline.
IGNOREEOF	If nonzero, don't allow use of a single **Ctrl-D** (the end-of-file or EOF character) to log off; use the **exit** command to log off.
PATH	Colon-separated list of directories to search for each command.
PROMPT_COMMAND	Command that **bash** executes before issuing a prompt for a new command.
PS1	Prompt displayed before each new command; see Section 7.5.4 for ways to introduce into the prompt dynamically changing information such as the current working directory or command history number.
PS2	Prompt displayed before a new line if a command is not finished.
PS3	Prompt displayed by **select** built-in command.

Variable	Meaning
PS4	Prompt displayed by **-x** debugging (see Section 7.1) and the **set** built-in command).

Table 7-20. History variables

Variable	Meaning
FCEDIT	Pathname of editor to use with the **fc** command.
HISTCMD	History number of the current command.
HISTCONTROL	If **HISTCONTROL** is set to the value of **ignorespace**, lines beginning with a space are not entered into the history list. If set to **ignoredups**, lines matching the last history line are not entered. Setting it to **ignoreboth** enables both options.
HISTFILE	Name of history file on which the editing modes operate.
HISTFILESIZE	Maximum number of lines to store in the history file. The default is 500.

Variable	Meaning
HISTSIZE	Maximum number of commands to remember in the command history. The default is 500.

Table 7-21. Mail variables

Variable	Meaning
MAIL	Name of file to check for incoming mail.
MAILCHECK	How often, in seconds, to check for new mail (default is 60 seconds).
MAILPATH	List of filenames, separated by colons (:), to check for incoming mail.

Table 7-22. Status variables

Variable	Meaning

Variable	Meaning
BASH	Pathname of this instance of the shell you are running.
BASH_VERSION	Version number of the shell you are running.
COLUMNS	Number of columns your display has.
DIRSTACK	List of directories manipulated by **pushd** and **popd** commands.
EUID	Effective user ID of process running this shell, in the form of the number recognized by the system.
GROUPS	Groups to which user belongs, in the form of the numbers recognized by the system.
HOME	Name of your home (login) directory.
HOSTNAME	Host the shell is running on.
HOSTTYPE	Short name indicating the type of machine the shell is running on; for instance, **i486**.

Variable	Meaning
LINES	The number of lines your display has.
MACHTYPE	Long string indicating the machine the shell is running on; for instance, **i486-pc-linux-gnu**.
OLDPWD	Previous directory before the last **cd** command.
OSTYPE	Short string indicating the operating system; for instance, **linux-gnu**.
PPID	Process ID of parent process that invoked this shell.
PWD	Current directory.
SECONDS	Number of seconds since the shell was invoked.
SHELL	Pathname of the shell you are running.
SHLVL	Depth to which running shells are nested.
TERM	The type of terminal that you are using.

Variable	Meaning
UID	Real user ID of process running this shell, in the form of the number recognized by the system.

7.4 Arithmetic Expressions

The **let** command performs integer arithmetic. **bash** provides a way to substitute integer values (for use as command arguments or in variables); base conversion is also possible.

Expression	Meaning
((*expr***))**	Use the value of the enclosed arithmetic expression.

7.4.1 Operators

bash uses arithmetic operators from the C programming language; the following list is in decreasing order of precedence. Use parentheses to override precedence.

Operator	Meaning
-	Unary minus
! ~	Logical negation; binary inversion (one's complement)
*** / %**	Multiplication; division; modulus (remainder)

Operator	Meaning
+ -	Addition; subtraction
<< >>	Bitwise left shift; bitwise right shift
<= >=	Less than or equal to; greater than or equal to
< >	Less than; greater than
= = !=	Equality; inequality (both evaluated left to right)
&	Bitwise AND
^	Bitwise exclusive OR
|	Bitwise OR
&&	Logical AND
||	Logical OR
=	Assign value

Operator	Meaning
+= -=	Reassign after addition/subtraction
***= /= %=**	Reassign after multiplication/division/remainder
&= ^= |=	Reassign after bitwise AND/XOR/OR
<<= >>=	Reassign after bitwise shift left/right

7.4.2 Examples

See the **let** built-in command for more information and examples.

```
let "count=0" "i = i + 1"          Assign i and
count

let "num % 2"; echo $?                    Test for an
even number
```

7.5 Command History

bash lets you display or modify previous commands. Commands in the history list can be modified using:

- Line-edit mode
- The **fc** command

In addition, the command substitutions described in Chapter 8 also work in **bash**.

7.5.1 Line-Edit Mode

Line-edit mode lets you emulate many features of the **vi** and Emacs editors. The history list is treated like a file. When the editor is invoked, you type editing keystrokes to move to the command line you want to execute. On most terminals, arrow keys work in both Emacs mode and **vi** command mode. You can also change the line before executing it. See Table 7-23 for some examples of common line-edit commands. When you're ready to issue the command, press Return.

The default line-edit mode is Emacs. To enable **vi** mode, enter: $ **set -o vi**

Note that **vi** starts in input mode; to type a **vi** command, press Esc first.

The mode you use for editing **bash** commands is entirely separate from the editor that is invoked for you automatically within many commands (for instance, the editor invoked by mail readers when you ask them to create a new mail message). To change the default

editor, set the **VISUAL** or **EDITOR** variable to the filename or full pathname of your favorite editor: $ **export EDITOR=emacs**

Table 7-23. Common editing keystrokes

vi	Emacs	Result
k	**Ctrl-p**	Get previous command.
j	**Ctrl-n**	Get next command.
/string	**Ctrl-r** *string*	Get previous command containing *string*.
h	**Ctrl-b**	Move back one character.
l	**Ctrl-f**	Move forward one character.
b	**M-b**	Move back one word.
w	**M-f**	Move forward one word.
X	**Del**	Delete previous character.
x	**Ctrl-d**	Delete one character.

vi	Emacs	Result
dw	**M-d**	Delete word forward.
db	**M-Ctrl-h**	Delete word back.
xp	**Ctrl-t**	Transpose two characters.

7.5.2 The fc Command

Use **fc -l** to list history commands, and **fc -e** to edit them. See the **fc** built-in command for more information.

7.5.2.1 Examples

```
$ history              Display the command history
list

$ fc -l 20 30          List commands 20 through 30

$ fc -l -5             List the last five commands

$ fc -l cat            List the last command
beginning with cat

$ fc -ln 5 > doit      Save command 5 to file doit

$ fc -e vi 5 20        Edit commands 5 through 20
using vi
```

```
$ fc -e emacs          Edit previous command using
Emacs

$ !!                   Reexecute previous command

$ !cat                 Reexecute last cat command

$ !cat foo-file        Reexecute last command,
adding foo-file to

                       the end of the argument list
```

7.5.3 Command Substitution

Syntax	Meaning
!	Begin a history substitution.
!!	Previous command.
!*N*	Command number *N* in history list.
!-*N*	*N*th command back from current command.
!*string*	Most recent command that starts with *string*.
!?*string***?**	Most recent command that contains *string*.

Syntax	Meaning
!?*string***?%**	Most recent command argument that contains *string*.
!$	Last argument of previous command.
!#	The current command up to this point.
!!*string*	Previous command, then append *string*.
!*N string*	Command *N*, then append *string*.
!{*s1***}***s2*	Most recent command starting with string *s1*, then append string *s2*.
^*old***^***new***^**	Quick substitution; change string *old* to *new* in previous command, and execute modified command.

7.5.4 Variables in Prompt

Using the following variables, you can display information about the current state of the shell or the system in your **bash** prompt. Set the **PS1** variable to a string including the desired variables. For instance, the following command sets **PS1** to a string that includes the **\w** variable to display the current working directory, and the **\!**

variable to display the number of the current command. The next line is the prompt displayed by the change.

```
$ PS1='\w: Command \!$ '

~/book/linux: Command 504$
```

Variable	Meaning
\a	Alarm (bell)
\d	Date in the format "Mon May 8"
\e	Escape character (terminal escape, not backslash)
\h	Hostname
\j	Number of background jobs (active or stopped)
\l	Current terminal name
\n	Newline inserted in the prompt
\r	Carriage return inserted in the prompt
\s	Current shell

Variable	Meaning
\t	Time in 24-hour format, where 3:30 p.m. appears as 15:30:00
\u	User's account name
\v	Version and release of **bash**
\w	Current working directory
\A	Time in 24-hour format, where 3:30 p.m. appears as 15:30
\D{*format***}**	Time in the specified format interpreted by **strftime**; an empty format displays the locale-specific current time
\H	Like **\h**
\T	Time in 12-hour format, where 3:30 p.m. appears as 03:30:00
\V	Version, release, and patch level of **bash**
\W	Last element (following last slash) of current working directory

Variable	Meaning
****	Single backslash inserted in the prompt
\!	Number of current command in the command history
\#	Number of current command, where numbers start at 1 when the shell starts
\@	Time in 12-hour format, where 3:30 p.m. appears as 03:30 p.m.
\$	Indicates whether you are **root**: displays # for **root**, **$** for other users
\[	Starts a sequence of nonprinting characters, to be ended by **\]**
\]	Ends the sequence of nonprinting characters started by **\[**
*****nnn*	The character in the ASCII set corresponding to the octal number *nnn* inserted into the prompt

bash -m -i

set -m

set -o monitor

Many job control commands take jobID as an argument. This argument can be specified as follows:

%n

Job number n

%s

Job whose command line starts with string s

%?s

Job whose command line contains string s

%%

Current job

%+

Current job (same as preceding)

%-

Previous job

bash provides the following job control commands. For more information on these commands, see Section 7.7 earlier in this chapter.

bg

Put a job in the background.

fg

Put a job in the foreground.

jobs

List active jobs.

kill

Terminate a job.

stop

Suspend a background job.

stty tostop

Stop background jobs if they try to send output to the terminal.

wait

Wait for background jobs to finish.

Ctrl-Z

Suspend a foreground job, and use bg or fg to restart it in the background or foreground. (Your terminal may use something other than Ctrl-Z as the suspend character.)

#

#!<tt>
shell</tt>

#!/bin/bash

:

if who | grep -w $1 > /dev/null

  then : # do nothing

  # if pattern is found

  else echo "User $1 is not logged in"

fi

. <tt>file</tt> [<tt>arguments</tt>]

alias [<span
class="docEmphBold">-p] [<tt>
name</tt>[=
<tt>cmd</tt>]]

bg [<tt>
jobIDs</tt>]

bind [<tt>
options</tt>]

bind [<tt>
options</tt>] <tt>key</tt>:<tt>
function</tt>

$ bind "\C-t":copy-forward-word

break [<tt>
n</tt>]

builtin <tt>
command</tt> [<tt>arguments</tt>]

case <tt>
string</tt>

  in

  <tt>regex</tt>)

<tt>commands</tt>

;;

...

esac

cd [<tt>options</tt>] [<tt>dir</tt>]

command [<tt>options</tt>] <tt>command</tt> [<tt>arguments</tt>]

compgen [<tt>options</tt>] [<tt>word</tt>]

complete [<tt>options</tt>] <tt>names</tt>

continue [<tt>n</tt>]

declare [<tt>options</tt>] [<tt>name</tt>[=<tt>value</tt>]]

typeset [<tt>options</tt>] [<tt>name</tt>[=<tt>value</tt>]]

dirs [<tt>options</tt>]

disown [<tt>options</tt>] [<tt>jobIDs</tt>]

echo [<tt>options</tt>] [<tt>strings</tt>]

enable [<tt>options</tt>] [<tt>built-in .</tt>..]

eval [<tt>command args.</tt>..]

exec [<tt>options</tt>] [<tt>command</tt>]

trap 'exec 2>&-' 0   Close standard error when shell script

 exits (signal 0)

$ exec /bin/tcsh   Replace current shell with extended C shell

$ exec < infile   Reassign standard input to infile

exit [<tt>n</tt>]

if [$# -eq 0]; then

 echo "Usage: $0 [-c] [-d] file(s)"

 exit 1 # Error status

fi

export [<tt>
options</tt>] [<tt>variables</tt>]

export [<tt>
options</tt>] [<tt>name</tt><span
class="docEmphBold">=[<tt>value
</tt>]]<tt>.</tt>..

fc [<tt>
options</tt>] [<tt>first</tt>] [<tt>
last</tt>]

fc -s [<tt>
oldpattern</tt><span
class="docEmphBold">=<tt>
newpattern</tt>] [<tt>command</tt>]

fg [<tt>
jobIDs</tt>]

for <tt>x</tt> [in <tt>list</tt>]

do

<tt>commands</tt>

done

for file in *

do

 pr $file > $file.tmp

done

for item in `cat program_list`

do

 echo "Checking chapters for"

 echo "references to program $item..."

 grep -c "$item.[co]" chap*

done

function <tt>command</tt>{ ...}

getopts <tt>string name</tt> [<tt>args</tt>]

hash [<tt>options</tt>] [<tt>commands</tt>]

help [-s] [<tt>string</tt>]

history [<tt>options</tt>]

history [<tt>lines</tt>]

if <tt>test-cmds</tt>

if <tt>test-cmds</tt> if <tt>test-cmds</tt> if <tt>test-cmds</tt>

then then then

<tt>cmds1</tt> <tt>cmds1</tt> <tt>cmds1</tt>

fi else elif <tt>test-cmds</tt>

<tt>cmds2</tt> then

fi <tt>cmds2</tt>

...

else

<tt>cmdsn</tt>

fi

jobs [<tt>
options</tt>] [<tt>jobIDs</tt>]

kill [<tt>
options</tt>] <tt>IDs</tt>

let <tt>
expressions</tt>

let i=i+1

let "i = i + 1"

local [<tt>options</tt>] [<tt>variable</tt> [=<tt>value</tt>]] [<tt>variable2</tt> [=<tt>value</tt>]] <tt>.</tt>..

logout [<tt>status</tt>]

popd [<tt>options</tt>]

printf <tt>string</tt> [<tt>arguments</tt>]

printf "Previous command: %i\n" "$(($HISTCMD-1))"

Previous command: 534

$ echo $PAGER

less -E

$ printf "%q\n" "\t$PAGER"

\\tless\ -E

pushd [<tt>directory</tt>]

pushd [<tt>options</tt>]

pwd [option]

read [<tt>options</tt>] [<tt>variable1</tt> <tt>

variable2</tt> ...]

read first last address

Sarah Caldwell 123 Main Street

$ echo "$last, $first\n$address"

Caldwell, Sarah

123 Main Street

$ read -sp "Enter password (will not appear on screen)" user_pw

Enter password (will not appear on screen)

$ echo $user_pw

You weren't supposed to know!

IFS=:

cat /etc/passwd |

while

read account pw user group gecos home shell

do

echo "Account name $account has user info: $gecos"

done

readonly [<tt>options</tt>] [<tt>variable1</tt> <tt>variable2</tt>...]

return [<tt>n</tt>]

select <tt>name</tt> [in <tt>wordlist</tt> ;]

do

<tt>commands</tt>

done

set [<tt>options</tt>] [<tt>arg1</tt> <tt>arg2</tt> <tt>.</tt>..]

set -- "$num" -20 -30   Set $1 to $num, $2 to -20, $3 to -30

set -vx   Read each command line; show it; execute it;

show it again (with arguments)

set +x   Stop command tracing

set -o noclobber   Prevent file overwriting

set +o noclobber   Allow file overwriting again

shift [<tt>n</tt>]

shopt [<tt>options</tt>] [<tt>optnames</tt>]

source <tt>file</tt> [<tt>arguments</tt>]

suspend [-f]

test <tt>condition</tt>

[<tt>condition</tt>]

while test $# -gt 0   While there are arguments . . .

while [-n "$1"]   While the first argument is nonempty . . .

if [$count -lt 10]   If $count is less than 10 . . .

if [-d RCS]   If the RCS directory exists . . .

if ["$answer" != "y"]   If the answer is not y . . .

if [! -r "$1" -o ! -f "$1"]   If the first argument is not a

 readable file or a regular file . . .

times

trap [<tt> option</tt>] [<tt>commands</tt>] [<tt>signals</tt>]

trap "" 2   Ignore signal 2 (interrupts)

trap 2   Obey interrupts again

trap "rm -f $tmp; exit" 0 1 2 15

type [<tt>options</tt>] <tt>commands</tt>

type mv read

mv is /bin/mv

read is a shell built-in

typeset

ulimit [<tt>options</tt>] [<tt>n</tt>]

umask [<tt>options</tt>] [<tt>nnn</tt>]

unalias [-a] <tt>names</tt>

unset [<tt>options</tt>] <tt>names</tt>

until

<tt>test-commands</tt>

do

<tt>commands</tt>

done

wait [<tt>ID</tt>]

ait $! Wait for last background process to finish

while

<tt>test-commands</tt>

do

<tt>commands</tt>

done

Execute test-commands (usually a test or [ ] command); if the exit status is 0, perform commands. Repeat.

Chapter 8. tcsh: An Extended C Shell

This chapter describes **tcsh**, an enhanced version of the C shell. On many versions of Linux, **tcsh** is also used as the C shell; in that case, the **tcsh** features described in this chapter work even when you run **csh**. The C shell was so named because many of its programming constructs and symbols resemble those of the C

programming language.

The default shell on Linux systems is **bash**. If you want to use **tcsh**, you first need to change your default. Each user's shell preference is kept in the password table. If you are creating an account, you can set the default shell when you add the user. If the account already exists, use the **chsh** command to change the shell (see the command descriptions in Chapter 3).

8.1 Overview of Features

Features of **tcsh** include:

- Input/output redirection
- Wildcard characters (metacharacters) for filename abbreviation
- Shell variables for customizing your environment
- Integer arithmetic
- Access to previous commands (command history)
- Command-name abbreviation (aliasing)
- A built-in command set for writing shell programs
- Job control
- Command-line editing and editor commands
- Word completion (tab completion)
- Spellchecking
- Scheduled events, such as logout or terminal locking after a set idle period and delayed commands
- Read-only variables

tcsh [options] [arguments]

tcsh uses syntax resembling C and executes commands from a terminal or a file. The options -n, -v, and -x are useful when debugging scripts.

8.2.1 Options

-b

Allow the remaining command-line options to be interpreted as options to a specified command rather than as options to tcsh.

-c

Read and execute commands specified from the argument that follows and place any remaining arguments in the argv shell variable.

-d

Load directory stack from ~/*.cshdirs* even if not a login shell.

-e

Exit if a command produces errors.

-f

Fast startup; start without executing .tcshrc.

-i

Invoke interactive shell (prompt for input) even if not on a terminal.

-l

Login shell (must be the only option specified).

-m

Load *~/.tcshrc* even if effective user is not the owner of the file.

-n

Parse commands, but do not execute.

-q

Accept SIGQUIT when used under a debugger. Disables job control.

-s

Read commands from the standard input.

-t

Exit after executing one line of input (which may be continued with a \ to escape the newline).

-v

Display commands before executing them; expand history substitutions, but not other substitutions (e.g., filename, variable, and command). Same as setting verbose.

-V

Same as -v, but also display .tcshrc.

-x

Display commands before executing them, but expand all substitutions. Same as setting echo.

-X

Same as -x, but also display .tcshrc.

8.2.2 Arguments

Arguments are assigned, in order, to the positional parameters $1, $2, and so on. If the first argument is an executable script, commands are read from it, and remaining arguments are assigned to $1, $2, and so forth.

% ls new*   Match new and new.1 % cat ch?   Match ch9 but not ch10 % vi [D-R]*   Match files that begin with uppercase D through R % ls {ch,app}?   Expand, then match ch1, ch2, app1, app2 % cd ~tom   Change to tom's home directory

% echo 'Single quotes "protect" double quotes'

Single quotes "protect" double quotes

% echo "Well, isn't that "\""special?"\"

Well, isn't that "special"?

% echo "You have `ls|wc -l` files in `pwd`"

You have 43 files in /home/bob

% echo The value of \$x is $x

The value of $x is 100

% cd; ls   Execute sequentially % (date; who; pwd) > logfile   All output is redirected % sort file | pr -3 | lp   Sort file, page output, then print % vi `grep -l ifdef *.c`   Edit files found by grep % egrep '(yes|no)' `cat list`   Specify a list of files to search % grep XX file && lp file   Print file if it contains the pattern % grep XX file || echo XX not found   Echo an error message if XX not found

% cat part1 > book   Copy part1 to book % cat part2 part3 >> book   Append parts 2 and 3 to same file as part1 % mail tim < report   Take input to message from report % cc calc.c >& error_out   Store all messages, including errors % cc newcalc.c >&! error_out   Overwrite old file % grep Unix ch* |& pr   Pipe all messages, including errors % (find / -print > filelist) >& no_access   Separate error messages from list of files % sed 's/^/XX /' << "END_ARCHIVE"   Supply text right after command This is often how a shell archive is "wrapped", bundling text for distribution. You would normally run sed

from a shell program, not from the command line.
 "END_ARCHIVE"

8.4 Variables

This subsection describes the following:

- Variable substitution
- Variable modifiers
- Predefined shell variables
- Formatting for the *prompt* variable
- Sample *.tcshrc* file
- Environment variables

8.4.1 Variable Substitution

In the following substitutions, braces (**{ }**) are optional, except when needed to separate a variable name from following characters that would otherwise be considered part of the name.

Variable	Description
${*var***}**	The value of variable *var*.

Variable	Description
${*var***[***i***]}**	Select word or words in position *i* of *var*. *i* can be a single number, a range *m*-*n*, a range -*n* (missing *m* implies 1), a range *m*- (missing *n* implies all remaining words), or ***** (select all words). *i* also can be a variable that expands to one of these values.
${#*var***}**	The number of words in *var*.
${#argv}	The number of arguments.
$0	Name of the program.
${argv[*n***]}**	Individual arguments on command line (positional parameters); $1 \leq n \leq 9$.
${n}	Same as **${argv[***n***]}**.
${argv[*]}	All arguments on command line.
$*	Same as **{$argv[*]}**.
$argv[$#argv]	The last argument.
${?*var***}**	Return 1 if *var* is set, 0 if not.

Variable	Description
$$	Process number of current shell; useful as part of a filename for creating temporary files with unique names.
${?*name***}**	Return 1 if *name* is set, 0 if not.
$?0	Return 1 if input filename is known, 0 if not.

8.4.1.1 Examples

Sort the third through last arguments and save the output in a file whose name is unique to this process: **sort $argv[3-] > tmp.$$**

Process *.tcshrc* commands only if the shell is interactive (i.e., the **prompt** variable must be set):

```
if ($?prompt) then
   set commands,
   alias commands,
   etc.
endif
```

8.4.2 Variable Modifiers

Except for **$?***var*, **$$**, and **$?0**, the variable substitutions in the preceding section may be followed by one of these modifiers (when braces are used, the modifier goes inside them):

:r

Return the variable's root (the portion before the last dot).

:e

Return the variable's extension.

:h

Return the variable's header (the directory portion).

:t

Return the variable's tail (the portion after the last slash).

:gr

Return all roots.

:ge

Return all extensions.

:gh

Return all headers.

:gt

Return all tails.

:q

Quote a wordlist variable, keeping the items separate. Prevents further substitution. Useful when the variable contains filename metacharacters that should not be expanded.

:x

Quote a pattern, expanding it into a wordlist.

8.4.2.1 Examples using pathname modifiers

The following table shows the effect of pathname modifiers if the **aa** variable is set as follows: set aa=(/progs/num.c /book/chap.ps)

Variable portion	Specification	Output result
Normal variable	**echo $aa**	*/progs/num.c /book/chap.ps*
Second root	**echo $aa[2]:r**	*/book/chap*
Second header	**echo $aa[2]:h**	*/book*
Second tail	**echo $aa[2]:t**	*chap.ps*
Second extension	**echo $aa[2]:e**	*ps*
Root	**echo $aa:r**	*/progs/num /book/chap.ps*

Variable portion	Specification	Output result
Global root	**echo $aa:gr**	*/progs/num /book/chap*
Header	**echo $aa:h**	*/progs /book/chap.ps*
Global header	**echo $aa:gh**	*/progs /book*
Tail	**echo $aa:t**	*num.c /book/chap.ps*
Global tail	**echo $aa:gt**	*num.c chap.ps*
Extension	**echo $aa:e**	*c /book/chap.ps*
Global extension	**echo $aa:ge**	*c ps*

8.4.2.2 Examples using quoting modifiers

Unless quoted, the shell expands variables to represent files in the current directory:

```
% set a="[a-z]*" A="[A-Z]*"

% echo "$a" "$A"

[a-z]* [A-Z]*
```

```
% echo $a $A
at cc m4 Book Doc

% echo $a:x $A
[a-z]* Book Doc

% set d=($a:q $A:q)
% echo $d
at cc m4 Book Doc

% echo $d:q
[a-z]* [A-Z]*

% echo $d[1] +++ $d[2]
at cc m4 +++ Book Doc

% echo $d[1]:q
[a-z]*
```

8.4.3 Predefined Shell Variables

Variables can be set in one of two ways; by assigning a value:

`set var=value`

or by simply turning the variable on: set *var*

In the following table, variables that accept values are shown with the equals sign followed by the type of value they accept; the value is then described. (Note, however, that variables such as **argv**, **cwd**, and **status** are never explicitly assigned.) For variables that are turned on or off, the table describes what they do when set. **tcsh** automatically sets (and, in some cases, updates) the variables **addsuffix**, **argv**, **autologout**, **command**, **cwd**, **dirstack**, **echo-style**, **edit**, **gid**, **home**, **loginsh**, **logout**, **oid**, **owd**, **path**, **prompt**, **prompt2**, **prompt3**, **shell**, **shlvl**, **status**, **tcsh**, **term**, **tty**, **uid**, **user**, and **version**.

Variable	Description
addsuffix	Append **/** to directories and a space to files during tab completion to indicate a precise match.
afsuser	Set value to be used instead of the local username for Kerberos authentication with the **autologout** locking feature.

Variable	Description
ampm	Display all times in 12-hour format.
argv=(*args*)	List of arguments passed to current command; default is **( )**.
autocorrect	Check spelling before attempting to complete commands.
autoexpand	Expand history (such as **!** references) during command completion.
autolist[**=ambiguous]**	Print possible completions when correct one is ambiguous. If **ambiguous** is specified, print possible completions only when completion adds no new characters.

Variable	Description
autologout=*logout-minutes* [*locking-minutes*]	Log out after *logout-minutes* of idle time. Lock the terminal after *locking-minutes* of idle time, requiring a password before continuing. Not used if the **DISPLAY** environment variable is set.
backslash_quote	Always allow backslashes to quote ****, ', and ".
catalog	Use *tcsh.${catalog}* as the filename of the message catalog. The default is *tcsh*.
cdpath=*dirs*	List of alternate directories to search when locating arguments for **cd**, **popd**, or **pushd**.

Variable	Description
color	Turn on color for **ls-F**, **ls**, or both. Setting to nothing is equivalent to setting for both.
colorcat	Enable color escape sequence for Native Language System (NLS) support and display NLS messages in color.
command	If set, hold the command passed to the shell with the **-c** option.
complete=enhance	When **enhance**, ignore case in completion, treat ., **-**, and _ as word separators, and consider _ and **-** to be the same.
continue=*cmdlist*	If set to a list of commands, continue those commands instead of starting new ones.

Variable	Description
continue_args=*cmdlist*	Like **continue**, but execute the following: `echo `pwd` $argv > ~/.cmd_pause; %cmd`
correct={cmd|complete|all}	When **cmd**, spellcheck commands. When **complete**, complete commands. When **all**, spellcheck whole command line.
cwd=*dir*	Full pathname of current directory.
dextract	When set, the **pushd** command extracts the desired directory and puts it at the top of the stack instead of rotating the stack.
dirsfile=*file*	History file consulted by **dirs -S** and **dirs -L**. Default is *~/.cshdirs*.

Variable	Description
dirstack	Directory stack, in array format. **dirstack[0]** is always equivalent to **cwd**. The other elements can be artificially changed.
dspmbyte=*code*	Enable use of multibyte code; for use with Kanji. See the **tcsh** manpage for details.
dunique	Make sure that each directory exists only once in the stack.
echo	Redisplay each command line before execution; same as **csh -x** command.
echo_style={bsd|sysv|both|none}	Don't echo a newline with **-n** option (**bsd**), parse escaped characters (**sysv**), do both, or do neither.

Variable	Description
edit	Enable command-line editor. Set by default for interactive shells.
ellipsis	For use with **prompt** variable. Use ... to represent skipped directories.
fignore=*suffs*	List of filename suffixes to ignore during filename completion.
gid	User's group ID.
group	User's group name.
histchars=*ab*	A two-character string that sets the characters to use in history substitution and quick substitution (default is **!^**).

Variable	Description
histdup={all|prev|erase}	Maintain a record only of unique history events (**all**), do not enter a new event when it is the same as the previous one (**prev**), or remove an old event that is the same as the new one (**erase**).
histfile=*file*	History file consulted by **history -S** and **history -L**. Default is *~/.history*.
histlit	Do not expand history lines when recalling them.
history=*n format*	The first word indicates the number of commands to save in the history list. The second indicates the format with which to display that list (see Section 8.4.4 for possible formats).

Variable	Description
home=*dir*	Home directory of user, initialized from HOME. The **~** character is shorthand for this value.
ignoreeof	Ignore an end-of-file (EOF) from terminals; prevents accidental logout.
implicitcd	If directory name is entered as a command, **cd** to that directory. Can be set to **verbose** to echo the **cd** to standard output.
inputmode={insert|overwrite}	Control editor's mode.
killdup={all|prev|erase}	Enter only unique strings in the kill ring (**all**), do not enter new string when it is the same as the current killed string (**prev**), or erase from the kill ring an old string that is the same as the current string (**erase**).

Variable	Description
killring=*num*	Set the number of killed strings to keep in memory to *num*. The default is 30. If unset or set to a number less than 2, keep only the most recent killed string.
listflags=*flags*	One or more of the **x**, **a**, or **A** options for the **ls-F** built-in command. Second word can be set to path for **ls** command.
listjobs[=long]	When a job is suspended, list all jobs (in long format, if specified).
listlinks	In **ls -F** command, include type of file to which links point.
listmax=*num*	Do not allow **list-choices** to print more than *num* choices before prompting.

Variable	Description
listmaxrows=*num*	Do not allow **list-choices** to print more than *num* rows of choices before prompting.
loginsh	Set if shell is a login shell.
logout	Indicates status of an imminent logout (**normal**, **automatic**, or **hangup**).
mail=(*n files***)**	One or more files checked for new mail every 5 minutes or (if *n* is supplied) every *n* seconds.

Variable	Description
matchbeep= {never|nomatch|ambiguous|notunique}	Specifies circumstances under which completion should beep: never, if no match exists, if multiple matches exist, or if multiple matches exist and one is exact. If unset, **ambiguous** is used.
nobeep	Disable beeping.
noclobber	Don't redirect output to an existing file; prevents accidental destruction of files.
noding	Don't print "DING!" in prompt time specifiers when the hour changes.
noglob	Turn off filename expansion; useful in shell scripts.
nokanji	Disable Kanji (if supported).

Variable	Description
nonomatch	Treat filename metacharacters as literal characters if no match exists (e.g., **vi ch*** creates new file **ch*** instead of printing "No match").
nostat=*directory-list*	Do not stat *directory-list* during completion.
notify	Declare job completions when they occur.
owd	Old working directory.
path=(*dirs***)**	List of pathnames in which to search for commands to execute. Initialized from **PATH**; the default is **. /usr/ucb /usr/bin**.
printexitvalue	Print all nonzero exit values.

Variable	Description
prompt='*str*'	String that prompts for interactive input; default is **%**. See Section 8.4.4 later in this chapter for formatting information.
prompt2='*str*'	String that prompts for interactive input in **foreach** and **while** loops and continued lines (those with escaped newlines). See Section 8.4.4 for formatting information.
prompt3='*str*'	String that prompts for interactive input in automatic spelling correction. See Section 8.4.4 for formatting information.
promptchars=*cc*	Use the two characters specified as *cc* with the **%# prompt** sequence to indicate normal users and the superuser, respectively.

Variable	Description
pushdsilent	Do not print directory stack when **pushd** and **popd** are invoked.
pushdtohome	Change to home directory when **pushd** is invoked without arguments.
recexact	Consider completion to be concluded on first exact match.
recognize_only_executables	When command completion is invoked, print only executable files.
rmstar	Prompt before executing the command **rm ***.
rprompt=*string*	The string to print on the right side of the screen while the prompt is displayed on the left. Specify as for **prompt**.

Variable	Description
savedirs	Execute **dirs -S** before exiting.
savehist=*max* [**merge**]	Execute **history -S** before exiting. Save no more than *max* lines of history. If specified, merge those lines with previous history saves, and sort by time.
sched=*string*	Format for **sched**'s printing of events. See Section 8.4.4 for formatting information.
shell=*file*	Pathname of the shell program.
shlvl	Number of nested shells.
status=*n*	Exit status of last command. Built-in commands return 0 (success) or 1 (failure).

Variable	Description
symlinks={chase|ignore|expand}	Specify manner in which to deal with symbolic links. Expand them to real directory name in *cwd* (**chase**), treat them as real directories (**ignore**), or expand arguments that resemble pathnames (**expand**).
tcsh	Version of **tcsh**.
term	Terminal type.
time='*n %c*'	If command execution takes more than *n* CPU seconds, report user time, system time, elapsed time, and CPU percentage. Supply optional *%c* flags to show other data.
tperiod	Number of minutes between executions of **periodic** alias.

Variable	Description
tty	Name of tty, if applicable.
uid	User ID.
user	Username.
verbose	Display a command after history substitution; same as **tcsh -v**.
version	Shell's version and additional information, including options set at compile time.
visiblebell	Flash screen instead of beeping.
watch=([*n*] *user terminal...***)**	Watch for *user* logging in at *terminal*, where *terminal* can be a *tty* name or **any**. Check every *n* minutes, or 10 by default.

Variable	Description
who=*string*	Specify information to be printed by **watch**.
wordchars=*chars*	List of all nonalphanumeric characters that may be part of a word. Default is ***?_-.[ ]~=**.

8.4.4 Formatting for the Prompt Variable

tcsh provides a list of substitutions that can be used in formatting the prompt. The list of available substitutions includes:

%%

Literal %

%*l*

The present working directory

%~

The present working directory, in **~** notation

%#

for the superuser, **>** for others

%?

Previous command's exit status

%$*var*

The value of the shell or environment variable *var*

%*{string***%}**

Include *string* as a literal escape sequence to change terminal attributes (but should not move the cursor location); cannot be the last sequence in the prompt

\c, **^c**

Parse **c** as in the **bindkey** built-in command

%b

End boldfacing

%c*[*[**0***]n]*, **%.[[0]**n*]*

The last *n* (default 1) components of the present working directory; if 0 is specified, replace removed components with **/<skipped>**

%d

Day of the week (e.g., Mon, Tue)

%h, **%!**, **!**

Number of current history event

%j

The number of jobs

%l

Current tty

%m

First component of hostname

%n

Username

%p

Current time, with seconds (12-hour mode)

%s

End standout mode (reverse video)

%t, **%@**

Current time (12-hour format)

%u

End underlining

%w

Month (e.g., Jan, Feb)

%y

Year (e.g., 99, 00)

%B

Begin boldfacing

%C

Similar to **%c**, but use full pathnames instead of **~** notation

%D

Day of month (e.g., 09, 10)

%L

Clear from the end of the prompt to the end of the display or the line.

%M

Fully qualified hostname

%P

Current time, with seconds (24-hour format)

%R

In **prompt2**, the parser status; in **prompt3**, the corrected string; and in **history**, the history string

%S

Begin standout mode (reverse video)

%T

Current time (24-hour format)

%U

Begin underlining

%W

Month (e.g., 09, 10)

%Y

Year (e.g., 1999, 2000)

8.4.5 Sample .tcshrc File

```
# PREDEFINED VARIABLES


set path=(~ ~/bin /usr/ucb /bin /usr/bin . )

set mail=(/usr/mail/tom)


if ($?prompt) then              # settings for
interactive use

  set echo

  set noclobber ignoreeof


  set cdpath=(/usr/lib /usr/spool/uucp)

# Now I can type cd macros

# instead of cd /usr/lib/macros


  set history=100
```

```
  set prompt='tom \!% '          # includes history
number

  set time=3



# MY VARIABLES



  set man1="/usr/man/man1"      # lets me do   cd
$man1, ls $man1

  set a="[a-z]*"                # lets me do   vi
$a

  set A="[A-Z]*"                # or           grep
string $A



# ALIASES



  alias c "clear; dirs"         # use quotes to
protect ; or |

  alias h "history|more"

  alias j jobs -l

  alias ls ls -sFC              # redefine ls
command
```

```
  alias del 'mv \!* ~/tmp_dir' # a safe
alternative to rm

endif
```

8.4.6 Environment Variables

tcsh maintains a set of *environment variables*, which are distinct from shell variables and aren't really part of the shell. Shell variables are meaningful only within the current shell, but environment variables are exported automatically, making them available globally. For example, shell variables are accessible only to a particular script in which they're defined, whereas environment variables can be used by any shell scripts, mail utilities, or editors you might invoke.

Environment variables are assigned as follows: setenv *VAR value*

By convention, environment variable names are all uppercase. You can create your own environment variables, or you can use the predefined environment variables that follow.

The following environment variables have corresponding **tcsh** shell variables. When either one changes, the value is copied to the other.

AFSUSER

> Alternative to local user for Kerberos authentication with **autologout** locking; same as **afsuser**.

GROUP

> User's group name; same as **group**.

HOME

Home directory; same as **home**.

PATH

Search path for commands; same as **path**.

SHLVL

Number of nested shell levels; same as **shlvl**.

TERM

Terminal type; same as **term**.

USER

User's login name; same as **user**.

Other environment variables, which do not have corresponding shell variables, include the following:

COLUMNS

Number of columns on terminal.

DISPLAY

Identifies user's display for the X Window System. If set, the shell doesn't set **autologout**.

EDITOR

Pathname to default editor. See also **VISUAL**.

HOST

Name of machine.

HOSTTYPE

Type of machine. Obsolete; will be removed eventually.

HPATH

Colon-separated list of directories to search for documentation.

LANG

Preferred language. Used for native language support.

LC_CTYPE

The locale, as it affects character handling. Used for native language support.

LINES

Number of lines on the screen.

LOGNAME

Another name for the **USER** variable.

LS_COLORS

Colors for use with the **ls** command. See the **tcsh** manpage for detailed information.

MACHTYPE

Type of machine.

MAIL

The file that holds mail. Used by mail programs. This is not the same as the shell variable **mail**, which only checks for new mail.

NOREBIND

Printable characters not rebound. Used for native language support.

OSTYPE

Operating system.

PWD

The current directory; the value is copied from **cwd**.

REMOTEHOST

Machine name of remote host.

SHELL

Undefined by default; once initialized to **shell**, the two are identical.

TERMCAP

The file that holds the cursor-positioning codes for your terminal type. Default is */etc/termcap*.

VENDOR

System vendor.

VISUAL

Pathname to default full-screen editor. See also **EDITOR**.

8.5 Expressions

Expressions are used in @, **if**, and **while** statements to perform arithmetic, string comparisons, file testing, and so on. **exit** and **set** also specify expressions, as can the **tcsh** built-in command **filetest**. Expressions are formed by combining variables and constants with operators that resemble those in the C programming language. Operator precedence is the same as in C and can be remembered as follows:

1. *** / %**
2. **+ -**

Group all other expressions inside parentheses. Parentheses are required if the expression contains <, >, &, or |.

8.5.1 Operators

Operators can be one of the following types.

8.5.1.1 Assignment operators

Operator	Description
=	Assign value.
+= -=	Reassign after addition/subtraction.

Operator	Description
***= /= %=**	Reassign after multiplication/division/remainder.
&= ^= |=	Reassign after bitwise AND/XOR/OR.
++	Increment.
--	Decrement.

8.5.1.2 Arithmetic operators

Operator	Description
*** / %**	Multiplication; integer division; modulus (remainder).
+ -	Addition; subtraction.

8.5.1.3 Bitwise and logical operators

Operator	Description

Operator	Description
~	Binary inversion (one's complement).
!	Logical negation.
<< >>	Bitwise left shift; bitwise right shift.
&	Bitwise AND.
^	Bitwise exclusive OR.
|	Bitwise OR.
&&	Logical AND.
||	Logical OR.
{ *command* **}**	Return 1 if command is successful, 0 otherwise. Note that this is the opposite of *command*'s normal return code. The **$status** variable may be more practical.

8.5.1.4 Comparison operators

Operator	Description
= = !=	Equality; inequality.
<= >=	Less than or equal to; greater than or equal to.
< >	Less than; greater than.

8.5.1.5 File inquiry operators

Command substitution and filename expansion are performed on *file* before the test is performed. Operators can be combined (e.g., **-ef**). The following is a list of the valid file inquiry operators:

Operator	Description
-b *file*	The file is a block special file.
-c *file*	The file is a character special file.
-d *file*	The file is a directory.
-e *file*	The file exists.

Operator	Description
-f *file*	The file is a plain file.
-g *file*	The file's **set-group-ID** bit is set.
-k *file*	The file's sticky bit is set.
-l *file*	The file is a symbolic link.
-L *file*	Apply any remaining operators to symbolic link, not the file it points to.
-o *file*	The user owns the file.
-p *file*	The file is a named pipe (FIFO).
-r *file*	The user has read permission.
-s *file*	The file has nonzero size.
-S *file*	The file is a socket special file.
-t *file*	*file* is a digit and is an open file descriptor for a terminal device.

Operator	Description
-u *file*	The file's **set-user-ID** bit is set.
-w *file*	The user has write permission.
-x *file*	The user has execute permission.
-X *file*	The file is executable and is in the path, or is a shell built-in.
-z *file*	The file has 0 size.
!	Reverse the sense of any preceding inquiry.

Finally, **tcsh** provides the following operators, which return other kinds of information:

Operator	Description
-A[:] *file*	Last time file was accessed, as the number of seconds since the epoch. With a colon (:), the result is in timestamp format.

Operator	Description
-C[:] *file*	Last time inode was modified. With a colon (:), the result is in timestamp format.
-D *file*	Device number.
-F *file*	Composite file identifier, in the form *device*:*inode*.
-G[:] *file*	Numeric group ID for the file. With a colon (:), the result is the group name if known, otherwise the numeric group ID.
-I *file*	Inode number.
-L *file*	The name of the file pointed to by symbolic link *file*.
-M[:] *file*	Last time file was modified. With a colon (:), the result is in timestamp format.
-N *file*	Number of hard links.
-P[:] *file*	Permissions in octal, without leading 0. With a colon (:), the result includes a leading 0.

Operator	Description
-P*mode*[:] *file*	Equivalent to **-P** *file* ANDed to *mode*. With a colon (:), the result includes a leading 0.
-U[:] *file*	Numeric user ID of the file's owner. With a colon (:), the result is the username if known, otherwise the numeric user ID.
-Z *file*	The file's size, in bytes.

8.5.2 Examples

The following examples show @ commands and assume n = 4:

Expression	Value of $x
@ x = ($n > 10 || $n < 5)	1
@ x = ($n >= 0 && $n < 3)	0
@ x = ($n << 2)	16
@ x = ($n >> 2)	1

Expression	Value of $x
@ x = $n % 2	0
@ x = $n % 3	1

The following examples show the first line of **if** or **while** statements:

Expression	Meaning
while ($#argv != 0)	While there are arguments . . .
if ($today[1] = = "Fri")	If the first word is "Fri". . .
if (-f $argv[1])	If the first argument is a plain file. . .
if (! -d $tmpdir)	If **tmpdir** is not a directory. . .

8.6 Command History

Previously executed commands are stored in a history list. You can access this list to verify commands, repeat them, or execute modified versions of them. The **history** built-in command displays the history list; the predefined variables **histchars** and **history** also affect the history mechanism. There are a number of ways to use the history list:

- Rerun a previous command
- Edit a previous command
- Make command substitutions
- Make argument substitutions (replace specific words in a command)
- Extract or replace parts of a command or word

The easiest way to take advantage of the command history is to use the arrow keys to move around in the history, select the command you want, and then rerun it or use the editing features described in Section 8.7.5, later in this chapter, to modify the command. The arrow keys are:

Key	Description
Up arrow	Previous command.

Key	Description
Down arrow	Next command.
Left arrow	Move left in command line.
Right arrow	Move right in command line.

The next sections describe some tools for editing and rerunning commands. With the C shell, which does not have the command-line editing features of **tcsh**, these tools are important for rerunning commands. With **tcsh**, they are less often used, but they still work.

8.6.1 Command Substitution

Command	Description
!	Begin a history substitution.
!!	Previous command.
!*N*	Command number *N* in history list.
!-*N*	*N*th command back from current command.

Command	Description
!*string*	Most recent command that starts with *string*.
!?*string***?**	Most recent command that contains *string*.
!?*string***?%**	Most recent command argument that contains *string*.
!$	Last argument of previous command.
!!*string*	Previous command, then append *string*.
!*N string*	Command *N*, then append *string*.
!{*s1***}***s2*	Most recent command starting with string *s1*, then append string *s2*.
^*old***^***new***^**	Quick substitution; change string *old* to *new* in previous command, and execute modified command.

8.6.2 Command Substitution Examples

The following command is assumed:

```
%3 vi cprogs/01.c ch002 ch03
```

Event number	Command typed	Command executed
4	**^00^0**	**vi cprogs/01.c ch02 ch03**
5	**nroff !***	**nroff cprogs/01.c ch02 ch03**
6	**nroff !$**	**nroff ch03**
7	**!vi**	**vi cprogs/01.c ch02 ch03**
8	**!6**	**nroff ch03**
9	**!?01**	**vi cprogs/01.c ch02 ch03**
10	**!{nr}.new**	**nroff ch03.new**
11	**!!|lp**	**nroff ch03.new | lp**
12	**more !?pr?%**	**more cprogs/01.c**

8.6.3 Word Substitution

Colons may precede any word specifier.

Specifier	Description
:0	Command name
:*n*	Argument number *n*
^	First argument
$	Last argument
:*n*-*m*	Arguments *n* through *m*
-*m*	Words 0 through *m*; same as **:0-***m*
:*n*-	Arguments *n* through next-to-last
:*n**	Arguments *n* through last; same as *n*-**$**
*	All arguments; same as **^-$** or **1-$**
#	Current command line up to this point; fairly useless

8.6.4 Word Substitution Examples

The following command is assumed:

```
%13 cat ch01 ch02 ch03 biblio back
```

Event number	Command typed	Command executed
14	**ls !13^**	**ls ch01**
15	**sort !13:***	**sort ch01 ch02 ch03 biblio back**
16	**lp !cat:3***	**more ch03 biblio back**
17	**!cat:0-3**	**cat ch01 ch02 ch03**
18	**vi !-5:4**	**vi biblio**

8.6.5 History Modifiers

Command and word substitutions can be modified by one or more of the following modifiers:

8.6.5.1 Printing, substitution, and quoting

Modifier	Description

Modifier	Description
:p	Display command, but don't execute.
:s/*old***/***new*	Substitute string *new* for *old*, first instance only.
:gs/*old***/***new*	Substitute string *new* for *old*, all instances.
:&	Repeat previous substitution (**:s** or **^** command), first instance only.
:g&	Repeat previous substitution, all instances.
:q	Quote a wordlist.
:x	Quote separate words.

8.6.5.2 Truncation

Modifier	Description
:r	Extract the first available pathname root (the portion before the last period).

Modifier	Description
:gr	Extract all pathname roots.
:e	Extract the first available pathname extension (the portion after the last period).
:ge	Extract all pathname extensions.
:h	Extract the first available pathname header (the portion before the last slash).
:gh	Extract all pathname headers.
:t	Extract the first available pathname tail (the portion after the last slash).
:gt	Extract all pathname tails.
:u	Make first lowercase letter uppercase.
:l	Make first uppercase letter lowercase.
:a	Apply modifier(s) following **a** as many times as possible to a word. If used with **g**, **a** is applied to all words.

8.6.6 History Modifier Examples

From the preceding, command number 17 is:

```
%17 cat ch01 ch02 ch03
```

Event number	Command typed	Command executed
19	**!17:s/ch/CH/**	**cat CH01 ch02 ch03**
20	**!17g&**	**cat CH01 CH02 CH03**
21	**!more:p**	**more cprogs/01.c** *(displayed only)*
22	**cd !$:h**	**cd cprogs**
23	**vi !mo:$:t**	**vi 01.c**
24	**grep stdio !$**	**grep stdio 01.c**
25	**^stdio^include stdio^:q**	**grep "include stdio" 01.c**
26	**nroff !21:t:p**	**nroff 01.c** *(is that what I wanted?)*

Event number	Command typed	Command executed
27	**!!**	**nroff 01.c** *(execute it)*

8.6.7 Special Aliases

Certain special aliases can be set in **tcsh**. The aliases are initially undefined. Once set, the commands they specify are executed when specific events occur. The following is a list of the special aliases and when they are executed:

beepcmd

At beep.

cwdcmd

When the current working directory changes.

jobcmd

Before running a command or before its state changes. Like **postcmd**, but does not print built-ins.

helpcommand

Invoked by the **run-help** editor command.

periodic

Every few minutes. The exact amount of time is set by the **tperiod** shell variable.

precmd

Before printing a new prompt.

postcmd

Before running a command.

shell *shell*

If a script does not specify a shell, interpret it with *shell*, which should be a full pathname.

8.7 Command-Line Manipulation

tcsh offers a certain amount of functionality in manipulating the command line, including word or command completion and the ability to edit a command line.

8.7.1 Completion

The shell automatically completes words and commands when you press the Tab key, and notifies you when a completion is finished by appending a space to complete filenames or commands and a ***/*** to complete directories.

In addition, **tcsh** recognizes **~** notation for home directories; it assumes that words at the beginning of a line and subsequent to **|**, **&**, **;**, **||**, or **&&** are commands, and modifies the search path appropriately. Completion can be done midword; only the letters to the left of the prompt are checked for completion.

8.7.2 Related Shell Variables

- **autolist**
- **fignore**
- **listmax**
- **listmaxrows**

8.7.3 Related Command-Line Editor Commands

- **complete-word-back**
- **complete-word-forward**
- **expand-glob**
- **list-glob**

8.7.4 Related Shell Built-ins

- **complete**
- **uncomplete**

8.7.5 Command-Line Editing

tcsh lets you move your cursor around in the command line, editing the line as you type. There are two main modes for editing the command line, based on the two most common text editors: Emacs and **vi**. Emacs mode is the default; you can switch between the modes with:

```
bindkey -e      Select Emacs bindings

bindkey -v      Select vi bindings
```

The main difference between the Emacs and **vi** bindings is that the Emacs bindings are modeless (i.e., they always work). With the **vi** bindings, you must switch between input and command modes; different commands are useful in each mode. Additionally:

- Emacs mode is simpler; **vi** mode allows finer control.
- Emacs mode allows you to yank cut text and set a mark; **vi** mode does not.
- The command-history searching capabilities differ.

8.7.5.1 Emacs mode

Tables Table 8-1 through Table 8-3 describe the various editing keystrokes available in Emacs mode.

Table 8-1. Cursor positioning (Emacs mode)

Command	Description
Ctrl-B	Move cursor back (left) one character.
Ctrl-F	Move cursor forward (right) one character.
M-b	Move cursor back one word.

Command	Description
M-f	Move cursor forward one word.
Ctrl-A	Move cursor to beginning of line.
Ctrl-E	Move cursor to end of line.

Table 8-2. Text deletion (Emacs mode)

Command	Description
Del or **Ctrl-H**	Delete character to left of cursor.
Ctrl-D	Delete character under cursor.
M-d	Delete word.
M-Del or **M-Ctrl-H**	Delete word backward.
Ctrl-K	Delete from cursor to end-of-line.

Command	Description
Ctrl-U	Delete entire line.

Table 8-3. Command history (Emacs mode)

Command	Description
Ctrl-P	Previous command.
Ctrl-N	Next command.
Up arrow	Previous command.
Down arrow	Next command.
cmd-fragment **M-p**	Search history for *cmd-fragment*, which must be the beginning of a command.
cmd-fragment **M-n**	Like **M-p**, but search forward.

Command	Description
M-*num*	Repeat next command *num* times.
Ctrl-Y	Yank previously deleted string.

8.7.5.2 vi mode

vi mode has two submodes, input mode and command mode. The default mode is input. You can toggle modes by pressing Esc; alternatively, in command mode, typing **a** (append) or **i** (insert) will return you to input mode.

Tables Table 8-4 through Table 8-10 describe the editing keystrokes available in vi mode.

Table 8-4. Command history (vi input and command modes)

Command	Description
Ctrl-P	Previous command
Ctrl-N	Next command

Command	Description
Up arrow	Previous command
Down arrow	Next command
Esc	Toggle mode

Table 8-5. Editing (vi input mode)

Command	Description
Ctrl-B	Move cursor back (left) one character.
Ctrl-F	Move cursor forward (right) one character.
Ctrl-A	Move cursor to beginning of line.
Ctrl-E	Move cursor to end-of-line.
DEL or **Ctrl-H**	Delete character to left of cursor.

Command	Description
Ctrl-W	Delete word backward.
Ctrl-U	Delete from beginning of line to cursor.
Ctrl-K	Delete from cursor to end-of-line.

Table 8-6. Cursor positioning (vi command mode)

Command	Description
h or **Ctrl-H**	Move cursor back (left) one character.
l or **SPACE**	Move cursor forward (right) one character.
w	Move cursor forward one word.
b	Move cursor back one word.
e	Move cursor to next word ending.

Command	Description
W, **B**, **E**	Like **w**, **b**, and **e**, but treat only whitespace as word separator instead of any nonalphanumeric character.
^ or **Ctrl-A**	Move cursor to beginning of line (first nonwhitespace character).
0	Move cursor to beginning of line.
$ or **Ctrl-E**	Move cursor to end-of-line.

Table 8-7. Text insertion (vi command mode)

Command	Description
a	Append new text after cursor until **Esc**.
i	Insert new text before cursor until **Esc**.
A	Append new text after end of line until **Esc**.
I	Insert new text before beginning of line until **Esc**.

Table 8-8. Text deletion (vi command mode)

Command	Description
x	Delete character under cursor.
X or **Del**	Delete character to left of cursor.
d*m*	Delete from cursor to end of motion command *m*.
D	Same as **d$**.
Ctrl-W	Delete word backward.
Ctrl-U	Delete from beginning of line to cursor.
Ctrl-K	Delete from cursor to end of line.

Table 8-9. Text replacement (vi command mode)

Command	Description

Command	Description
c*m*	Change characters from cursor to end of motion command *m* until **Esc**.
C	Same as **c$**.
r*c*	Replace character under cursor with character *c*.
R	Replace multiple characters until **Esc**.
s	Substitute character under cursor with characters typed until **Esc**.

Table 8-10. Character-seeking motion (vi command mode)

Command	Description
f*c*	Move cursor to next instance of *c* in line.
F*c*	Move cursor to previous instance of *c* in line.

Command	Description
t*c*	Move cursor just before next instance of *c* in line.
T*c*	Move cursor just after previous instance of *c* in line.
;	Repeat previous **f** or **F** command.
,	Repeat previous **f** or **F** command in opposite direction.

8.8 Job Control

Job control lets

you place foreground jobs in the background, bring background jobs to

the foreground, or suspend (temporarily stop) running jobs. The shell

provides the following commands for job control. For more information

on these commands, see Section 8.9.

bg

Put a job in the background.

fg

Put a job in the foreground.

jobs

List active jobs.

kill

Terminate a job.

notify

Notify when a background job finishes.

stop

Suspend a background job.

Ctrl-Z

Suspend the foreground job.

Many job control commands take *jobID* as an

argument. This argument can be specified as follows:

%*n*

Job number *n*.

%*s*

Job whose command line starts with string *s*.

%?*s*

Job whose command line contains string *s*.

%%

Current job.

%

Current job (same as preceding).

%+

Current job (same as preceding).

%-

Previous job.

@ [variable[n]=expression]

#

#!shell

#!/bin/tcsh -f

:

alias [name [command]]

alias R 'set noglob; eval `resize` unset noglob'

alias | grep ls

alias ms 'nroff -ms \!*'

alias back 'cp \!^ \!^.old'

% \ls

alloc

bg [jobIDs]

4% nroff -ms report Ctrl-Z

5% bg

5% bg %   Current job

5% bg %1   Job number 1

5% bg %nr   Match initial string nroff

5% % &

bindkey [options]

[key] [command]

break

breaksw

built-ins

bye

case pattern :

cd [options] [dir]

chdir [dir]

complete [string [word/pattern/list[:select]/[suffix]]]

continue

default :

dirs [options]

echo [-n] string

echotc [options] arguments

else

end

endif

endsw

eval args

set noglob eval `tset -s xterm` unset noglob

% set b='$a'

% set a=hello

% echo $b   Read the command line once

$a

% eval echo $b   Read the command line twice

hello

exec command

exit [(expr)]

fg [jobIDs]

% %

% fg

% fg %

% fg %vi   Match initial string

filetest -op files

foreach name (wordlist) commandsend

% foreach i ([A-Z]*)

? mv $i $i.new

? end

foreach arg ($argv)

 # does it begin with - ?

 if ("$arg" =~ -*) then

 echo "Argument is an option"

 else

 echo "Argument is a filename"

 endif

end

glob wordlist

goto string

hashstat

history [options]

history -h 5 > do_it

source do_it

hup [command]

if

if (<tt>expr</tt>) <tt>cmd</tt>

if (<tt>expr</tt>) then if (<tt>expr</tt>) then if (<tt>expr</tt>) then

  <tt>cmds</tt> <tt>cmds1</tt> <tt>cmds1</tt>

endif else else if (<tt>expr</tt>) then

<tt>cmds2</tt> <tt>cmds2</tt>

endif else

<tt>cmds3</tt>

endif

if ($#argv = = 0) then

 echo "No filename given. Sending to Report."

 set outfile = Report

else

 set outfile = $argv[1]

endif

jobs [-l]

kill [options] IDs

% kill -l

HUP INT QUIT ILL TRAP ABRT BUS FPE KILL USR1 SEGV USR2

PIPE ALRM TERM STKFLT CHLD CONT STOP TSTP TTIN TTOU URG

XCPU XFSZ VTALRM PROF WINCH POLL PWR SYS RTMIN RTMIN+1

RTMIN+2 RTMIN+3 RTMAX-3 RTMAX-2 RTMAX-1 RTMAX

44% nroff -ms report &

45% kill 19536   Process ID

45% kill %   Current job

45% kill %1   Job number 1

45% kill %nr   Initial string

45% kill %?report   Matching string

limit [-h] [resource [limit]]

log

login [user|-p]

logout

ls-F [options] [files]

newgrp [-] [group]

nice [+n] command

nohup [command]

notify [jobID]

onintr label

onintr -

onintr

onintr cleanup   Go to "cleanup" on interrupt

 .

 .   Shell script commands

 .

cleanup: Label for interrupts

  onintr -   Ignore additional interrupts

  rm -f $tmpfiles   Remove any files created

  exit 2   Exit with an error status

popd [options]

printenv [variable]

pushd name

pushd [options]

pushd

% dirs

/home/bob /usr

% pushd /etc   Add /etc to directory stack

/etc /home/bob /usr

% pushd +2   Switch to third directory

/usr /etc /home/bob

% pushd   Switch top two directories

/etc /usr /home/bob

% popd   Discard current entry; go to next

/usr /home/bob

rehash

repeat n command

% repeat 3 pr memo | lp

% repeat 10 line > item_list

% repeat 50 cat template >> report

sched [options]

sched time command

set variable=value

set [options] variable[n]=value

set variableset

% set list=(yes no maybe)   Assign a wordlist

% set list[3]=maybe   Assign an item in existing wordlist

% set quote="Make my day"   Assign a variable

% set x=5 y=10 history=100   Assign several variables

% set blank   Assign a null value to blank

setenv [name [value]]

settc capability value

setty [options] [+|-mode]

shift [variable]

while ($#argv)   While there are arguments

  if (-f $argv[1])

 wc -l $argv[1]

 else

 echo "$argv[1] is not a regular file"

 endif

 shift   Get the next argument

end

source [-h] script [args]

source ~/.cshrc

stop jobIDs

suspend

switch

switch (string) switch ($argv[1])

  case pattern1 : case -[nN]:

  commands nroff $file | lp

  breaksw breaksw

  case pattern2 : case -[Pp]:

  commands pr $file | lp

  breaksw breaksw

case pattern3 : case -[Mm]:

commands more $file

breaksw breaksw

. case -[Ss]:

. sort $file

. breaksw

default: default:

commands echo "Error--no such option"

exit 1

breaksw breaksw

endsw endsw

telltc

time [command]

umask [nnn]

unalias pattern

uncomplete pattern

unhash

unlimit [-h] [resource]

unset variables

unsetenv variable

wait

watchlog

where command

which command

while (expression)

  commands

end

set user = (alice bob carol ted)

while ($argv[1] != $user[1])   Cycle through each user, checking

 for a match

  shift user   If we cycled through with no match...

if ($#user = = 0) then

 echo "$argv[1] is not on the list of users"

 exit 1

 endif

end

Chapter 9. Pattern Matching

A number of Linux text editing utilities let you search for (and, in some cases, change) text patterns rather than fixed strings.

These utilities include the editing programs **ed**, **ex**, **vi**, and **sed**, the **gawk** scripting language, and the commands **grep** and **egrep**. Text patterns (also called *regular expressions*) contain normal characters mixed with special characters (also called

metacharacters).

Perl's regular expression support is so rich that it does not fit into this book; you can find a description in the O'Reilly books *Mastering Regular*

Expressions, *Perl in a Nutshell*, *Perl 5 Pocket Reference*, or *Programming Perl*. The Emacs editor also provides regular expressions similar to those shown in this chapter.

ed and **ex** are hardly ever used as standalone, interactive editors nowadays. But **ed** can be found as a batch processor invoked from shell scripts, and **ex** commands are often invoked within **vi** through the colon (:) command. We use **vi** in this chapter to refer to the regular expression features supported by both **vi** and the **ex** editor

on which it is based. **sed** and **gawk** are widely used in shell scripts and elsewhere as filters to alter text.

$ grep [A-Z]* chap[12]

$ grep Array.c Bug.c Comp.c chap1 chap2

$ grep "[A-Z]*" chap[12]

Double quotes suffice in most cases, but single quotes are the safest bet.

Note also that * and ? have subtly different meanings in pattern matching and filename expansion by the shell.

9.2 Metacharacters, Listed by Linux Program

Some metacharacters are valid for one program but not for another. Those available to a given program are marked by a checkmark (✓) in the following table. Notes are provided after the table, and full descriptions of metacharacters are in the following section.

Symbol	ed	vi	sed	gawk	grep	egrep	Action
.	✓	✓	✓	✓	✓	✓	Match any character (can match newline in **gawk**).
*	✓	✓	✓	✓	✓	✓	Match zero or more preceding elements.
^	✓	✓	✓	✓	✓	✓	Match beginning of line or string.
$	✓	✓	✓	✓	✓	✓	Match end of line or string.
\	✓	✓	✓	✓	✓	✓	Escape following character.
[ ]	✓	✓	✓	✓	✓	✓	Match one from a list or range.
\(\)	✓	✓	✓				Store pattern for later replay.

Symbol	ed	vi	sed	gawk	grep	egrep	Action
\n	✓	✓	✓				Reuse matched text stored in *n*th \(\).
{ }				✓			Match a range of instances.
\{ \}	✓	✓	✓		✓		Match a range of instances.
\<		✓					Match word's beginning.
\>		✓					Match word's end.
+				✓	✓	✓	Match one or more preceding elements.
?				✓	✓	✓	Match zero or one preceding elements.
|				✓		✓	Separate choices to match.
( )				✓		✓	Group expressions to match.

On some Linux systems, **grep** is a link to **egrep**, so whenever you run **grep** you actually get **egrep** behavior.

In **ed**, **vi**, and **sed**, when you perform a search-and-replace (substitute) operation, the metacharacters in this table apply to the pattern you are searching for but not to the string replacing it.

In **gawk**, **{ }** is specified in the POSIX standard and is supported by **gawk** if you run it with the **-Wre-interval** option.

In **ed**, **vi**, and **sed**, the following additional metacharacters are valid only in a replacement pattern:

Symbol	ex	sed	ed	Action
****	✓	✓	✓	Escape following character.
\n	✓	✓	✓	Reuse matched text stored in *n*th **\(\)**.
&	✓	✓		Reuse previous search pattern.
~	✓			Reuse previous replacement pattern.
\e	✓			Turn off previous **\L** or **\U**.
\E	✓			Turn off previous **\L** or **\U**.
\l	✓			Change single following character to lowercase.
\L	✓			Change following characters to lowercase until **\E** is encountered.

Symbol	ex	sed	ed	Action
\u	✓			Change single following character to uppercase.
\U	✓			Change following characters to uppercase until **\E** is encountered.

9.3 Metacharacters

The following characters have special meaning in search patterns:

Character	Action
.	Match any single character except newline.
*	Match any number (or none) of the single character that immediately precedes it. The preceding character also can be a regular expression (e.g., since . (dot) means any character, **.*** means match any number of any characterexcept newlines).
^	Match the beginning of the line or string.
$	Match the end of the line or string.
[ ]	Match any one of the enclosed characters. A hyphen (**-**) indicates a range of consecutive characters. A circumflex (**^**) as the first character in the brackets reverses the sense: it matches any one character *not* in the list. A hyphen or close bracket (**]**) as the first character is treated as a member of the list. All other metacharacters are treated as members of the list.
[^]	Match anything except enclosed characters.

Character	Action
\{*n*,*m***\}**	Match a range of occurrences of the single character that immediately precedes it. The preceding character also can be a regular expression. **\{***n***\}** matches exactly *n* occurrences, **\{***n***,\}** matches at least *n* occurrences, and **\{***n*,*m***\}** matches any number of occurrences between *n* and *m*.
{*n*,*m***}**	Like **\{***n*,*m***\}**. Available in **grep** by default and in **gawk** with the **-Wre-interval** option.
****	Turn off the special meaning of the character that follows.
\(\)	Save the matched text enclosed between **\(** and **\)** in a special holding space. Up to nine patterns can be saved on a single line. They can be "replayed" in the same pattern or within substitutions by the escape sequences **\1** to **\9**.
*****n*	Reuse matched text stored in *n*th **\(\)**.
\<	Match the beginning of a word.
\>	Match the end of a word.

Character	Action
+	Match one or more instances of preceding regular expression.
?	Match zero or one instance of preceding regular expression.
|	Match the regular expression specified before or after.
( )	In **egrep** and **gawk**, group regular expressions.

Many utilities support POSIX character lists, which are useful for matching non-ASCII characters in languages other than English. These lists are recognized only within **[ ]** ranges. A typical use would be **[[:lower:]]**, which in English is the same as **[a-z]**.

The following table lists POSIX character lists:

Notation	Matches
[:alnum:]	Alphanumeric characters
[:alpha:]	Alphabetic characters, uppercase and lowercase

Notation	Matches
[:blank:]	Printable whitespace: spaces and tabs but not control characters
[:cntrl:]	Control characters, such as **^A** through **^Z**
[:digit:]	Decimal digits
[:graph:]	Printable characters, excluding whitespace
[:lower:]	Lowercase alphabetic characters
[:print:]	Printable characters, including whitespace but not control characters
[:punct:]	Punctuation, a subclass of printable characters
[:space:]	Whitespace, including spaces, tabs, and some control characters
[:upper:]	Uppercase alphabetic characters
[:xdigit:]	Hexadecimal digits

The following characters have special meaning in replacement patterns:

Character	Action
****	Turn off the special meaning of the character that follows.
*****n*	Restore the *n*th pattern previously saved by **\(** and **\)**. *n* is a number from 1 to 9, matching the patterns searched sequentially from left to right.
&	Reuse the search pattern as part of the replacement pattern.
~	Reuse the previous replacement pattern in the current replacement pattern.
\e	End replacement pattern started by **\L** or **\U**.
\E	End replacement pattern started by **\L** or **\U**.
\l	Convert first character of replacement pattern to lowercase.
\L	Convert replacement pattern to lowercase.

Character	Action
\u	Convert first character of replacement pattern to uppercase.
\U	Convert replacement pattern to uppercase.

9.4 Examples of Searching

When used with **grep** or **egrep**, regular expressions are normally surrounded by quotes to avoid interpretation by the shell. (If the pattern contains a **$**, you must use single quotes, as in '**$200**', or escape the **$**, as in "**\$200**".) When used with **ed**, **vi**, **sed**, and **gawk**, regular expressions are usually surrounded by **/** (although any delimiter works). Here are some sample patterns:

Pattern	Matches
bag	The string "bag"
^bag	"bag" at beginning of line or string
bag$	"bag" at end of line or string
^bag$	"bag" as the only text on line
[Bb]ag	"Bag" or "bag"
b[aeiou]g	Second character is a vowel
b[^aeiou]g	Second character is not a vowel
b.g	Second character is any character except newline

Pattern	Matches
^...$	Any line containing exactly three characters
^\.	Any line that begins with a dot
^\.[a-z][a-z]	Same, followed by two lowercase letters (e.g., **troff** requests)
^\.[a-z]\{2\}	Same as previous (**grep** or **sed** only)
^[^.]	Any line that doesn't begin with a dot
bugs*	"bug", "bugs", "bugss", etc
"**word**"	The string "word" in quotes
"***word"***	The string "word", with or without quotes
[A-Z][A-Z]*	One or more uppercase letters
[A-Z]+	Same (**egrep** or **gawk** only)
[A-Z].*	An uppercase letter, followed by zero or more characters

Pattern	Matches
[A-Z]*	Zero or more uppercase letters
[a-zA-Z]	Any letter
[0-9A-Za-z]+	Any alphanumeric sequence

egrep or gawk pattern	Matches
[567]	One of the numbers *5*, *6*, or *7*
five|six|seven	One of the words *five*, *six*, or *seven*
80[23]?86	*8086*, *80286*, or *80386*
compan(y|ies)	*company* or *companies*

vi pattern	Matches
\<the	Words like *theater* or *the*

vi pattern	Matches
the\>	Words like *breathe* or *the*
\<the\>	The word *the*

sed or grep pattern	Matches
0\{5,\}	Five or more zeros in a row
[0-9]\{3\}-[0-9]\{2\}-[0-9]\{4\}	Social security number (*nnn-nn-nnnn*)

9.4.1 Examples of Searching and Replacing

The following examples show the metacharacters available to **sed** and **vi**. We have shown **vi** commands with an initial colon because that is how they are invoked within **vi**. A space is marked by a ·; a tab is marked by *tab*.

Command	Result
s/.*/(&)/	Reproduce the entire line, but add parentheses.

Command	Result
s/.*/mv & &.old/	Change a wordlist (one word per line) into **mv** commands.
/^$/d	Delete blank lines.
:g/^$/d	Same as previous, in **vi** editor.
/^[·*tab*]***$/d**	Delete blank lines, plus lines containing spaces or tabs.
:g/^[·*tab*]***$/d**	Same as previous, in **vi** editor.
s/··*/·/g	Turn one or more spaces into one space.
:%s/··*/·/g	Same as previous, in **vi** editor.
:s/[0-9]/Item &:/	Turn a number into an item label (on the current line).
:s	Repeat the substitution on the first occurrence.
:&	Same as previous.

Command	Result
:sg	Same, but for all occurrences on the line.
:&g	Same as previous.
:%&g	Repeat the substitution globally.
:.,$s/Fortran/\U&/g	Change word to uppercase, on current line to last line.
:%s/.*/\L&/	Lowercase entire file.
:s/\<./\u&/g	Uppercase first letter of each word on current line. (Useful for titles.)
:%s/yes/No/g	Globally change a string ("yes") to another string ("No").
:%s/Yes/~/g	Globally change a different string to "No" (previous replacement).

Finally, here are some **sed** examples for transposing words. A simple transposition of two words might look like this:

```
s/die or do/do or die/          Transpose words
```

The real trick is to use hold buffers to transpose variable patterns. For example:

```
s/\([Dd]ie\) or \([Dd]o\)/\2 or\1/
Transpose using hold buffers
```

emacs

emacs files

10.1 Emacs Concepts

This section describes some Emacs terminology that may be unfamiliar if you haven't used Emacs before.

10.1.1 Modes

One of the features that makes Emacs popular is its editing modes. The modes set up an environment designed for the type of editing you are doing, with features like having appropriate key bindings available, and automatically indenting according to standard conventions for that type of document. There are two types of modes, major and minor. The major modes include modes for various programming languages like C or Perl, for text processing (e.g., SGML or even straight text), and many more. One particularly useful major mode is Dired (Directory Editor), which has commands that let you manage directories. Minor modes set or unset features that are independent of the major mode, such as auto-fill (which controls word wrapping), insert versus overwrite, and auto-save. For a full discussion of modes, see *Learning GNU Emacs* or the Emacs Info documentation system

(**C-h i**).

10.1.2 Buffer and Window

When you open a file in Emacs, the file is put into a *buffer*

so you can edit it. If you open another file, that file goes into another buffer. The view of the buffer contents that you have at any point in time is called a

window. For a small file, the window might show the entire file; for a large file, it shows only a portion of a file. Emacs allows multiple windows to be open at the same time, to display the contents of different buffers or different portions of a single buffer.

10.1.3 Point and Mark

When you are editing in Emacs, the position of the cursor is known as

point. You can set a *mark* at another place in the text to operate on the region between point and mark. This is a very useful feature for such operations as deleting or moving an area of text.

10.1.4 Kill and Yank

Emacs uses the terms

kill and *yank* for the concepts more commonly known today as cut and paste. You cut text in Emacs by killing it, and paste it by yanking it back. If you do multiple kills in a row, you can yank them back all at once.

(keyboard-translate ?\C-h ?\C-?)

(keyboard-translate ?\C-\\ ?\C-h)

Now the Del or Backspace kill should work, and you can invoke help by pressing C-\ (an arbitrarily chosen key sequence).

Another potential problem is that on some systems, C-s causes the terminal to hang. This is due to an old-fashioned handshake protocol between the terminal and the system. You can restart the terminal by pressing C-q, but that doesn't help you enter commands that contain the sequence C-s. The solution (aside from using a more modern dial-in protocol) is to create new key bindings that replace C-s or to enter those commands as M-x command-name. This is not specifically an Emacs problem, but it can cause problems when you run Emacs in a terminal window because C-s and C-q are commonly used Emacs key sequences.

10.3 Notes on the Tables

Emacs commands use the Ctrl key and the Meta key. Most modern terminals provide a key named Alt that functions as a Meta key. In this section, the notation **C-** indicates that you should hold down the Ctrl key and press the character that follows, while **M-** indicates the same for the Meta or Alt key. As an alternative to Meta or Alt, you can press the Esc key, release it, and press the character. You might want to do this if you have problems with controlling windows capturing the Alt key (which sometimes happens).

In the command tables that follow, the first column lists the keystroke and the last column describes it. When there is a middle column, it lists the command name. The command can be executed by typing **M-x** followed by the command name; you have to do this when the binding is listed as "(none)". If you're unsure of the full command name, you can type a space, tab, or carriage return, and Emacs will list possible completions of what you've typed so far.

Because Emacs is such a comprehensive editor, containing hundreds of commands, some commands must be omitted for the sake of preserving a "quick" reference. You can browse the full command set by typing **C-h** (for help) and then **b** to get a list of the key bindings,[1] or **M-x** followed by a space or tab to get the command names.

[1] If you want to learn to create your own key bindings, see *Learning GNU Emacs* (O'Reilly).

10.3.1 Absolutely Essential Commands

If you're just getting started with Emacs, here's a short list of the most important commands to know:

Binding	Action
C-h	Enter the online help system.
C-x C-s	Save the file.
C-x C-c	Exit Emacs.
C-x u	Undo last edit (can be repeated).
C-g	Get out of current command operation.
C-p	Up by one line.
C-n	Down by one line.
C-f	Forward by one character.
C-b	Back by one character.
C-v	Forward by one screen.
M-v	Backward by one screen.

Binding	Action
C-s	Search forward for characters.
C-r	Search backward for characters.
C-d	Delete current character.
Del	Delete previous character.
Backspace	Delete previous character.

10.4 Summary of Commands by Group

Tables list keystrokes, command name, and description. **C-** indicates the Ctrl key; **M-** indicates the Meta key.

10.4.1 File Handling Commands

Binding	Command	Action
C-x C-f	find-file	Find file and read it.
C-x C-v	find-alternate-file	Read another file; replace the one read currently in the buffer.
C-x i	insert-file	Insert file at cursor position.
C-x C-s	save-buffer	Save file. (If terminal hangs, **C-q** restarts.)
C-x C-w	write-file	Write buffer contents to file.
C-x C-c	save-buffers-kill-emacs	Exit Emacs.
C-z	suspend-emacs	Suspend Emacs (use **exit** or **fg** to restart).

10.4.2 Cursor Movement Commands

In addition to the key bindings shown in this table, you can use the arrow keys to move around in Emacs. When you are running Emacs in a graphical display environment (e.g., in the X Window System), you can also use the mouse for operations such as moving the cursor or selecting text.

Binding	Command	Action
C-f	forward-char	Move forward one character (right).
C-b	backward-char	Move backward one character (left).
C-p	previous-line	Move to previous line (up).
C-n	next-line	Move to next line (down).
M-f	forward-word	Move one word forward.
M-b	backward-word	Move one word backward.
C-a	beginning-of-line	Move to beginning of line.
C-e	end-of-line	Move to end of line.

Binding	Command	Action
M-a	backward-sentence	Move backward one sentence.
M-e	forward-sentence	Move forward one sentence.
M-{	backward-paragraph	Move backward one paragraph.
M-}	forward-paragraph	Move forward one paragraph.
C-v	scroll-up	Move forward one screen.
M-v	scroll-down	Move backward one screen.
C-x [	backward-page	Move backward one page.
C-x]	forward-page	Move forward one page.
M->	end-of-buffer	Move to end-of-file.
M-<	beginning-of-buffer	Move to beginning of file.

Binding	Command	Action
(none)	goto-line	Go to line *n* of file.
(none)	goto-char	Go to character *n* of file.
C-l	recenter	Redraw screen with current line in the center.
M-*n*	digit-argument	Repeat the next command *n* times.
C-u *n*	universal-argument	Repeat the next command *n* times.

10.4.3 Deletion Commands

Binding	Command	Action
Del	backward-delete-char	Delete previous character.
C-d	delete-char	Delete character under cursor.
M-Del	backward-kill-word	Delete previous word.

Binding	Command	Action
M-d	kill-word	Delete the word the cursor is on.
C-k	kill-line	Delete from cursor to end-of-line.
M-k	kill-sentence	Delete sentence the cursor is on.
C-x Del	backward-kill-sentence	Delete previous sentence.
C-y	yank	Restore what you've deleted.
C-w	kill-region	Delete a marked region (see Section 10.4.4).
(none)	backward-kill-paragraph	Delete previous paragraph.
(none)	kill-paragraph	Delete from the cursor to the end of the paragraph.

10.4.4 Paragraphs and Regions

Binding	Command	Action
C-@	set-mark-command	Mark the beginning (or end) of a region.
C-Space	(Same as preceding)	(Same as preceding)
C-x C-p	mark-page	Mark page.
C-x C-x	exchange-point-and-mark	Exchange location of cursor and mark.
C-x h	mark-whole-buffer	Mark buffer.
M-q	fill-paragraph	Reformat paragraph.
(none)	fill-region	Reformat individual paragraphs within a region.
M-h	mark-paragraph	Mark paragraph.
M-{	backward-paragraph	Move backward one paragraph.
M-}	forward-paragraph	Move forward one paragraph.

Binding	Command	Action
(none)	backward-kill-paragraph	Delete previous paragraph.
(none)	kill-paragraph	Delete from the cursor to the end of the paragraph.

10.4.5 Stopping and Undoing Commands

Binding	Command	Action
C-g	keyboard-quit	Abort current command.
C-x u	advertised-undo	Undo last edit (can be done repeatedly).
(none)	revert-buffer	Restore buffer to the state it was in when the file was last saved (or auto-saved).

10.4.6 Transposition Commands

Binding	Command	Action
C-t	transpose-chars	Transpose two letters.
M-t	transpose-words	Transpose two words.
C-x C-t	transpose-lines	Transpose two lines.
(none)	transpose-sentences	Transpose two sentences.
(none)	transpose-paragraphs	Transpose two paragraphs.

10.4.7 Capitalization Commands

Binding	Command	Action
M-c	capitalize-word	Capitalize first letter of word.
M-u	upcase-word	Uppercase word.
M-l	downcase-word	Lowercase word.

Binding	Command	Action
M- - M-c	negative-argument; capitalize-word	Capitalize previous word.
M- - M-u	negative-argument; upcase-word	Uppercase previous word.
M- - M-l	negative-argument; downcase-word	Lowercase previous word.
(none)	capitalize-region	Capitalize initial letters in region.
C-x C-u	upcase-region	Uppercase region.
C-x C-l	downcase-region	Lowercase region.

10.4.8 Incremental Search Commands

Binding	Command	Action
C-s	isearch-forward	Start or repeat incremental search forward.

Binding	Command	Action
C-r	isearch-backward	Start or repeat incremental search backward.
Return	(none)	Exit a successful search.
C-g	keyboard-quit	Cancel incremental search; return to starting point.
Del	(none)	Delete incorrect character of search string.
M-C-r	isearch-backward-regexp	Incremental search backward for regular expression.
M-C-s	isearch-forward-regexp	Incremental search forward for regular expression.

10.4.9 Word Abbreviation Commands

Binding	Command	Action
(none)	abbrev-mode	Enter (or exit) word abbreviation mode.

Binding	Command	Action
C-x a -	inverse-add-global-abbrev	Define previous word as global (mode-independent) abbreviation.
C-x a i l	inverse-add-mode-abbrev	Define previous word as mode-specific abbreviation.
(none)	unexpand-abbrev	Undo the last word abbreviation.
(none)	write-abbrev-file	Write the word abbreviation file.
(none)	edit-abbrevs	Edit the word abbreviations.
(none)	list-abbrevs	View the word abbreviations.
(none)	kill-all-abbrevs	Kill abbreviations for this session.

10.4.10 Buffer Manipulation Commands

Binding	Command	Action
C-x b	switch-to-buffer	Move to specified buffer.

Binding	Command	Action
C-x C-b	list-buffers	Display buffer list.
C-x k	kill-buffer	Delete specified buffer.
(none)	kill-some-buffers	Ask about deleting each buffer.
(none)	rename-buffer	Change buffer name to specified name.
C-x s	save-some-buffers	Ask whether to save each modified buffer.

10.4.11 Window Commands

Binding	Command	Action
C-x 2	split-window-vertically	Divide the current window in two vertically, resulting in one window on top of the other.
C-x 3	split-window-horizontally	Divide the current window in two horizontally, resulting in two side-by-side windows.

Binding	Command	Action
C-x >	scroll-right	Scroll the window right.
C-x <	scroll-left	Scroll the window left.
C-x o	other-window	Move to the other window.
C-x 0	delete-window	Delete current window.
C-x 1	delete-other-windows	Delete all windows but this one.
(none)	delete-windows-on	Delete all windows on a given buffer.
C-x ^	enlarge-window	Make window taller.
(none)	shrink-window	Make window shorter.
C-x }	enlarge-window-horizontally	Make window wider.

Binding	Command	Action
C-x {	shrink-window-horizontally	Make window narrower.
M-C-v	scroll-other-window	Scroll other window.
C-x 4 f	find-file-other-window	Find a file in the other window.
C-x 4 b	switch-to-buffer-other-window	Select a buffer in the other window.
C-x 5 f	find-file-other-frame	Find a file in a new frame.
C-x 5 b	switch-to-buffer-other-frame	Select a buffer in another frame.
(none)	compare-windows	Compare two buffers; show first difference.

10.4.12 Special Shell Mode Characters

The following table shows commands that can be used in Shell mode. To enter Shell mode, run the command **M-x shell**.

Binding	Command	Action
C-c C-c	interrupt-shell-subjob	Terminate the current job.
C-c C-d	shell-send-eof	End-of-file character.
C-c C-u	kill-shell-input	Erase current line.
C-c C-w	backward-kill-word	Erase the previous word.
C-c C-z	stop-shell-subjob	Suspend the current job.

10.4.13 Indentation Commands

Binding	Command	Action
C-x .	set-fill-prefix	Prepend each line in paragraph with characters from beginning of line up to cursor column; cancel prefix by typing this command in column 1.

Binding	Command	Action
(none)	indented-text-mode	Major mode: each tab defines a new indent for subsequent lines.
(none)	text-mode	Exit indented text mode; return to text mode.
M-C-\	indent-region	Indent a region to match first line in region.
M-m	back-to-indentation	Move cursor to first character on line.
M-^	delete-indentation	Join this line to the previous line.
M-C-o	split-line	Split line at cursor; indent to column of cursor.
(none)	fill-individual-paragraphs	Reformat indented paragraphs, keeping indentation.

10.4.14 Centering Commands

Binding	Command	Action

Binding	Command	Action
(none)	center-line	Center line that cursor is on.
(none)	center-paragraph	Center paragraph that cursor is on.
(none)	center-region	Center currently defined region.

10.4.15 Macro Commands

Binding	Command	Action
C-x (	start-kbd-macro	Start macro definition.
C-x)	end-kbd-macro	End macro definition.
C-x e	call-last-kbd-macro	Execute last macro defined.
M-*n* C-x e	digit-argument and call-last-kbd-macro	Execute last macro defined *n* times.
C-u C-x (	start-kbd-macro	Execute last macro defined, then add keystrokes.

Binding	Command	Action
(none)	name-last-kbd-macro	Name last macro you created (before saving it).
(none)	insert-last-keyboard-macro	Insert the macro you named into a file.
(none)	load-file	Load macro files you've saved.
(none)	*macroname*	Execute a keyboard macro you've saved.
C-x q	kbd-macro-query	Insert a query in a macro definition.
C-u C-x q	(none)	Insert a recursive edit in a macro definition.
M-C-c	exit-recursive-edit	Exit a recursive edit.

10.4.16 Detail Information Help Commands

Binding	Command	Action

Binding	Command	Action
C-h a	command-apropos	What commands involve this concept?
(none)	apropos	What commands, functions, and variables involve this concept?
C-h c	describe-key-briefly	What command does this keystroke sequence run?
C-h b	describe-bindings	What are all the key bindings for this buffer?
C-h k	describe-key	What command does this keystroke sequence run, and what does it do?
C-h l	view-lossage	What are the last 100 characters I typed?
C-h w	where-is	What is the key binding for this command?
C-h f	describe-function	What does this function do?
C-h v	describe-variable	What does this variable mean, and what is its value?

Binding	Command	Action
C-h m	describe-mode	Tell me about the mode the current buffer is in.
C-h s	describe-syntax	What is the syntax table for this buffer?

10.4.17 Help Commands

Binding	Command	Action
C-h t	help-with-tutorial	Run the Emacs tutorial.
C-h i	info	Start the Info documentation reader.
C-h n	view-emacs-news	View news about updates to Emacs.
C-h C-c	describe-copying	View the Emacs General Public License.
C-h C-d	describe-distribution	View information on ordering Emacs from the FSF.

Binding	Command	Action
C-h C-w	describe-no-warranty	View the (non)warranty for Emacs.

10.5 Summary of Commands by Key

Emacs commands are presented next in two alphabetical lists. Tables list keystrokes, command name, and description. **C-** indicates the Ctrl key; **M-** indicates the Meta key.

10.5.1 Control-Key Sequences

Binding	Command	Action
C-@	set-mark-command	Mark the beginning (or end) of a region.
C-Space	(Same as preceding)	(Same as preceding)
C-]	abort-recursive-edit	Exit recursive edit and exit query-replace.
C-a	beginning-of-line	Move to beginning of line.
C-b	backward-char	Move backward one character (left).

Binding	Command	Action
C-c C-c	interrupt-shell-subjob	Terminate the current job.
C-c C-d	shell-send-eof	End-of-file character.
C-c C-u	kill-shell-input	Erase current line.
C-c C-w	backward-kill-word	Erase previous word.
C-c C-z	stop-shell-subjob	Suspend current job.
C-d	delete-char	Delete character under cursor.
C-e	end-of-line	Move to end of line.
C-f	forward-char	Move forward one character (right).
C-g	keyboard-quit	Abort current command.

Binding	Command	Action
C-h	help-command	Enter the online help system.
C-h a	command-apropos	What commands involve this concept?
C-h b	describe-bindings	What are all the key bindings for this buffer?
C-h c	describe-key-briefly	What command does this keystroke sequence run?
C-h C-c	describe-copying	View the Emacs General Public License.
C-h C-d	describe-distribution	View information on ordering Emacs from the FSF.
C-h C-w	describe-no-warranty	View the (non)warranty for Emacs.
C-h f	describe-function	What does this function do?

Binding	Command	Action
C-h i	info	Start the Info documentation reader.
C-h k	describe-key	What command does this keystroke sequence run, and what does it do?
C-h l	view-lossage	What are the last 100 characters I typed?
C-h m	describe-mode	Tell me about the mode the current buffer is in.
C-h n	view-emacs-news	View news about updates to Emacs.
C-h s	describe-syntax	What is the syntax table for this buffer?
C-h t	help-with-tutorial	Run the Emacs tutorial.
C-h v	describe-variable	What does this variable mean, and what is its value?
C-h w	where-is	What is the key binding for this command?

Binding	Command	Action
C-k	kill-line	Delete from cursor to end-of-line.
C-l	recenter	Redraw screen with current line in the center.
C-n	next-line	Move to next line (down).
C-p	previous-line	Move to previous line (up).
C-q	quoted-insert	Insert next character typed. Useful for inserting a control character.
C-r	isearch-backward	Start or repeat nonincremental search backward.
C-r	(none)	Enter recursive edit (during query replace).
C-s	isearch-forward	Start or repeat nonincremental search forward.
C-t	transpose-chars	Transpose two letters.

Binding	Command	Action
C-u *n*	universal-argument	Repeat the next command *n* times.
C-u C-x (	start-kbd-macro	Execute last macro defined, then add keystrokes.
C-u C-x q	(none)	Insert recursive edit in a macro definition.
C-v	scroll-up	Move forward one screen.
C-w	kill-region	Delete a marked region.
C-x (	start-kbd-macro	Start macro definition.
C-x)	end-kbd-macro	End macro definition.
C-x [	backward-page	Move backward one page.
C-x]	forward-page	Move forward one page.

Binding	Command	Action
C-x ^	enlarge-window	Make window taller.
C-x {	shrink-window-horizontally	Make window narrower.
C-x }	enlarge-window-horizontally	Make window wider.
C-x <	scroll-left	Scroll the window left.
C-x >	scroll-right	Scroll the window right.
C-x .	set-fill-prefix	Prepend each line in paragraph with characters from beginning of line up to cursor column; cancel prefix by typing this command in column 1.
C-x 0	delete-window	Delete current window.

Binding	Command	Action
C-x 1	delete-other-windows	Delete all windows but this one.
C-x 2	split-window-vertically	Divide current window in two vertically, resulting in one window on top of the other.
C-x 3	split-window-horizontally	Divide current window in two horizontally, resulting in two side-by-side windows.
C-x 4 b	switch-to-buffer-other-window	Select a buffer in the other window.
C-x 4 f	find-file-other-window	Find a file in the other window.
C-x 5 b	switch-to-buffer-other-frame	Select a buffer in another frame.

Binding	Command	Action
C-x 5 f	find-file-other-frame	Find a file in another frame.
C-x a -	inverse-add-global-abbrev	Define previous word as global (mode-independent) abbreviation.
C-x a i l	inverse-add-mode-abbrev	Define previous word as mode-specific abbreviation.
C-x b	switch-to-buffer	Move to the buffer specified.
C-x C-b	list-buffers	Display the buffer list.
C-x C-c	save-buffers-kill-emacs	Exit Emacs.
C-x C-f	find-file	Find file and read it.
C-x C-l	downcase-region	Lowercase region.

Binding	Command	Action
C-x C-p	mark-page	Place cursor and mark around whole page.
C-x C-q	(none)	Toggle read-only status of buffer.
C-x C-s	save-buffer	Save file. (If terminal hangs, **C-q** restarts.)
C-x C-t	transpose-lines	Transpose two lines.
C-x C-u	upcase-region	Uppercase region.
C-x C-v	find-alternate-file	Read an alternate file, replacing the one currently in the buffer.
C-x C-w	write-file	Write buffer contents to file.
C-x C-x	exchange-point-and-mark	Exchange location of cursor and mark.
C-x Del	backward-kill-sentence	Delete previous sentence.

Binding	Command	Action
C-x e	call-last-kbd-macro	Execute last macro defined.
C-x h	mark-whole-buffer	Place cursor and mark around whole buffer.
C-x i	insert-file	Insert file at cursor position.
C-x k	kill-buffer	Delete the buffer specified.
C-x o	other-window	Move to the other window.
C-x q	kbd-macro-query	Insert a query in a macro definition.
C-x s	save-some-buffers	Ask whether to save each modified buffer.
C-x u	advertised-undo	Undo last edit (can be done repeatedly).
C-y	yank	Restore killed text.

Binding	Command	Action
C-z	suspend-emacs	Suspend Emacs (use **exit** or **fg** to restart).

10.5.2 Meta-Key Sequences

Binding	Command	Action
M-- M-c	negative-argument; capitalize-word	Capitalize previous word.
M-- M-l	negative-argument; downcase-word	Lowercase previous word.
M-- M-u	negative-argument; upcase-word	Uppercase previous word.
M-$	spell-word	Check spelling of word after cursor.
M-%	query-replace	Search for and replace a string.
M-!	shell-command	Prompt for a shell command and run it.

Binding	Command	Action
M-<	beginning-of-buffer	Move to beginning of file.
M->	end-of-buffer	Move to end-of-file.
M-{	backward-paragraph	Move backward one paragraph.
M-}	forward-paragraph	Move forward one paragraph.
M-^	delete-indentation	Join this line to the previous one.
M-*n*	digit-argument	Repeat the next command *n* times.
M-*n* C-x e	digit-argument; call-last-kbd-macro	Execute the last defined macro *n* times.
M-a	backward-sentence	Move backward one sentence.
M-b	backward-word	Move one word backward.
M-c	capitalize-word	Capitalize first letter of word.

Binding	Command	Action
M-C-\	indent-region	Indent a region to match first line in region.
M-C-c	exit-recursive-edit	Exit a recursive edit.
M-C-o	split-line	Split line at cursor; indent to column of cursor.
M-C-r	isearch-backward-regexp	Incremental search backward for regular expression.
M-C-s	isearch-forward-regexp	Incremental search forward for regular expression.
M-C-v	scroll-other-window	Scroll other window.
M-d	kill-word	Delete word that cursor is on.
M-Del	backward-kill-word	Delete previous word.
M-e	forward-sentence	Move forward one sentence.
M-f	forward-word	Move one word forward.

Binding	Command	Action
(none)	fill-region	Reformat individual paragraphs within a region.
M-h	mark-paragraph	Place cursor and mark around whole paragraph.
M-k	kill-sentence	Delete sentence that cursor is on.
M-l	downcase-word	Lowercase word.
M-m	back-to-indentation	Move cursor to first nonblank character on line.
M-q	fill-paragraph	Reformat paragraph.
M-t	transpose-words	Transpose two words.
M-u	upcase-word	Uppercase word.
M-v	scroll-down	Move backward one screen.
M-x	(none)	Execute a command by typing its name.

10.6 Summary of Commands by Name

The following Emacs commands are presented alphabetically by command name. Use **M-x** to access the command name. Tables list command name, keystroke, and description. **C-** indicates the Ctrl key; **M-** indicates the Meta key.

Command	Binding	Action
macroname	(none)	Execute a keyboard macro you've saved.
abbrev-mode	(none)	Enter (or exit) word abbreviation mode.
abort-recursive-edit	C-]	Exit recursive edit and query replace.
advertised-undo	C-x u	Undo last edit (can be done repeatedly).
apropos	(none)	What functions and variables involve this concept?
back-to-indentation	M-m	Move cursor to first nonblank character on line.

Command	Binding	Action
backward-char	C-b	Move backward one character (left).
backward-delete-char	Del	Delete previous character.
backward-kill-paragraph	(none)	Delete previous paragraph.
backward-kill-sentence	C-x Del	Delete previous sentence.
backward-kill-word	C-c C-w	Delete previous word.
backward-kill-word	M-Del	Delete previous word.
backward-page	C-x [	Move backward one page.
backward-paragraph	M-{	Move backward one paragraph.

Command	Binding	Action
backward-sentence	M-a	Move backward one sentence.
backward-word	M-b	Move backward one word.
beginning-of-buffer	M-<	Move to beginning of file.
beginning-of-line	C-a	Move to beginning of line.
call-last-kbd-macro	C-x e	Execute last macro defined.
capitalize-region	(none)	Capitalize region.
capitalize-word	M-c	Capitalize first letter of word.
center-line	(none)	Center line that cursor is on.
center-paragraph	(none)	Center paragraph that cursor is on.

Command	Binding	Action
center-region	(none)	Center currently defined region.
command-apropos	C-h a	What commands involve this concept?
compare-windows	(none)	Compare two buffers; show first difference.
delete-char	C-d	Delete character under cursor.
delete-indentation	M-^	Join this line to previous one.
delete-other-windows	C-x 1	Delete all windows but this one.
delete-window	C-x 0	Delete current window.
delete-windows-on	(none)	Delete all windows on a given buffer.

Command	Binding	Action
describe-bindings	C-h b	What are all the key bindings for in this buffer?
describe-copying	C-h C-c	View the Emacs General Public License.
describe-distribution	C-h C-d	View information on ordering Emacs from the FSF.
describe-function	C-h f	What does this function do?
describe-key	C-h k	What command does this keystroke sequence run, and what does it do?
describe-key-briefly	C-h c	What command does this keystroke sequence run?
describe-mode	C-h m	Tell me about the mode the current buffer is in.
describe-no-warranty	C-h C-w	View the (non)warranty for Emacs.

Command	Binding	Action
describe-syntax	C-h s	What is the syntax table for this buffer?
describe-variable	C-h v	What does this variable mean, and what is its value?
digit-argument	M-*n*	Repeat next command *n* times.
downcase-region	C-x C-l	Lowercase region.
downcase-word	M-l	Lowercase word.
edit-abbrevs	(none)	Edit word abbreviations.
end-kbd-macro	C-x)	End macro definition.
end-of-buffer	M->	Move to end-of-file.
end-of-line	C-e	Move to end-of-line.

Command	Binding	Action
enlarge-window	C-x ^	Make window taller.
enlarge-window-horizontally	C-x }	Make window wider.
exchange-point-and-mark	C-x C-x	Exchange location of cursor and mark.
exit-recursive-edit	M-C-c	Exit a recursive edit.
fill-individual-paragraphs	(none)	Reformat indented paragraphs, keeping indentation.
fill-paragraph	M-q	Reformat paragraph.
fill-region	(none)	Reformat individual paragraphs within a region.

Command	Binding	Action
find-alternate-file	C-x C-v	Read an alternate file, replacing the one currently in the buffer.
find-file	C-x C-f	Find file and read it.
find-file-other-frame	C-x 5 f	Find a file in another frame.
find-file-other-window	C-x 4 f	Find a file in another window.
forward-char	C-f	Move forward one character (right).
forward-page	C-x]	Move forward one page.
forward-paragraph	M-}	Move forward one paragraph.
forward-sentence	M-e	Move forward one sentence.

Command	Binding	Action
forward-word	M-f	Move forward one word.
goto-char	(none)	Go to character *n* of file.
goto-line	(none)	Go to line *n* of file.
help-command	C-h	Enter the online help system.
help-with-tutorial	C-h t	Run the Emacs tutorial.
indent-region	M-C-\	Indent a region to match first line in region.
indented-text-mode	(none)	Major mode: each tab defines a new indent for subsequent lines.
info	C-h i	Start the Info documentation reader.
insert-file	C-x i	Insert file at cursor position.

Command	Binding	Action
insert-last-keyboard-macro	(none)	Insert the macro you named into a file.
interrupt-shell-subjob	C-c C-c	Terminate the current job (shell mode).
inverse-add-global-abbrev	C-x a -	Define previous word as global (mode-independent) abbreviation.
inverse-add-mode-abbrev	C-x a i l	Define previous word as mode-specific abbreviation.
isearch-backward	C-r	Start incremental search backward.
isearch-backward-regexp	M-C-r	Same, but search for regular expression.
isearch-forward	C-s	Start incremental search forward.

Command	Binding	Action
isearch-forward-regexp	M-C-s	Same, but search for regular expression.
kbd-macro-query	C-x q	Insert a query in a macro definition.
keyboard-quit	C-g	Abort current command.
kill-all-abbrevs	(none)	Kill abbreviations for this session.
kill-buffer	C-x k	Delete the buffer specified.
kill-line	C-k	Delete from cursor to end-of-line.
kill-paragraph	(none)	Delete from cursor to end of paragraph.
kill-region	C-w	Delete a marked region.
kill-sentence	M-k	Delete sentence the cursor is on.

Command	Binding	Action
kill-shell-input	C-c C-u	Delete current line.
kill-some-buffers	(none)	Ask about deleting each buffer.
kill-word	M-d	Delete word the cursor is on.
list-abbrevs	(none)	View word abbreviations.
list-buffers	C-x C-b	Display buffer list.
load-file	(none)	Load macro files you've saved.
mark-page	C-x C-p	Place cursor and mark around whole page.
mark-paragraph	M-h	Place cursor and mark around whole paragraph.
mark-whole-buffer	C-x h	Place cursor and mark around whole buffer.

Command	Binding	Action
name-last-kbd-macro	(none)	Name last macro you created (before saving it).
negative-argument; capitalize-word	M-- M-c	Capitalize previous word.
negative-argument; downcase-word	M-- M-l	Lowercase previous word.
negative-argument; upcase-word	M-- M-u	Uppercase previous word.
next-line	C-n	Move to next line (down).
other-window	C-x o	Move to the other window.
previous-line	C-p	Move to previous line (up).

Command	Binding	Action
query-replace	M-%	Search for and replace a string.
query-replace-regexp	(none)	Query-replace a regular expression.
quoted-insert	C-q	Insert next character typed. Useful for inserting a control character.
recenter	C-l	Redraw screen, with current line in center.
rename-buffer	(none)	Change buffer name to specified name.
replace-regexp	(none)	Replace a regular expression unconditionally.
re-search-backward	(none)	Simple regular-expression search backward.
re-search-forward	(none)	Simple regular-expression search forward.

Command	Binding	Action
revert-buffer	(none)	Restore buffer to the state it was in when the file was last saved (or auto-saved).
save-buffer	C-x C-s	Save file. (If terminal hangs, C-q restarts.)
save-buffers-kill-emacs	C-x C-c	Exit Emacs.
save-some-buffers	C-x s	Ask whether to save each modified buffer.
scroll-down	M-v	Move backward one screen.
scroll-left	C-x <	Scroll the window left.
scroll-other-window	M-C-v	Scroll other window.
scroll-right	C-x >	Scroll the window right.
scroll-up	C-v	Move forward one screen.

Command	Binding	Action
set-fill-prefix	C-x .	Prepend each line in paragraph with characters from beginning of line up to cursor column; cancel prefix by typing this command in column 1.
set-mark-command	C-@ or C-Space	Mark the beginning (or end) of a region.
shell-command	M-!	Prompt for a shell command and run it.
shell-send-eof	C-c C-d	End-of-file character (shell mode).
shrink-window	(none)	Make window shorter.
shrink-window-horizontally	C-x {	Make window narrower.
spell-buffer	(none)	Check spelling of current buffer.
spell-region	(none)	Check spelling of current region.

Command	Binding	Action
spell-string	(none)	Check spelling of string typed in minibuffer.
spell-word	M-$	Check spelling of word after cursor.
split-line	M-C-o	Split line at cursor; indent to column of cursor.
split-window-horizontally	C-x 3	Divide current window horizontally into two.
split-window-vertically	C-x 2	Divide current window vertically into two.
start-kbd-macro	C-x (	Start macro definition.
stop-shell-subjob	C-c C-z	Suspend current job.
suspend-emacs	C-z	Suspend Emacs (use **fg** to restart).
switch-to-buffer	C-x b	Move to the buffer specified.

Command	Binding	Action
switch-to-buffer-other-frame	C-x 5 b	Select a buffer in another frame.
switch-to-buffer-other-window	C-x 4 b	Select a buffer in another window.
text-mode	(none)	Enter text mode.
transpose-chars	C-t	Transpose two characters.
transpose-lines	C-x C-t	Transpose two lines.
transpose-paragraphs	(none)	Transpose two paragraphs.
transpose-sentences	(none)	Transpose two sentences.
transpose-words	M-t	Transpose two words.

Command	Binding	Action
unexpand-abbrev	(none)	Undo the last word abbreviation.
universal-argument	C-u *n*	Repeat the next command *n* times.
upcase-region	C-x C-u	Uppercase region.
upcase-word	M-u	Uppercase word.
view-emacs-news	C-h n	View news about updates to Emacs.
view-lossage	C-h l	What are the last 100 characters I typed?
where-is	C-h w	What is the key binding for this command?
write-abbrev-file	(none)	Write the word abbreviation file.
write-file	C-x C-w	Write buffer contents to file.

Command	Binding	Action
yank	C-y	Restore what you've deleted.

Chapter 11. The vi Editor

vi is the classic screen-editing program for Unix. A number of enhanced versions exist, including **nvi**, **vim**, **vile**, and **elvis**. On Linux, the **vi** command is usually a link to one of these programs. The Emacs editor, covered in Chapter 10, has several **vi** modes that allow you to use the same commands covered in this chapter.

The **vi**

editor operates in two modes, command mode and insert mode. The dual mode makes **vi** an attractive editor for users who separate text entry from editing. For users who edit as they type, the modeless editing of **emacs** can be more comfortable.

vi is based on an older line editor called **ex**. A user can invoke powerful editing capabilities within **vi** by typing a colon (:), entering an **ex** command, and pressing the Return key. Furthermore, you can place **ex** commands in a startup file called *~/.exrc*, which **vi** reads at the beginning of your editing session. Because **ex** commands are still an important part of **vi**, they are also described in this chapter.

One of the most common versions of **vi** found on Linux systems is Bram Moolenaar's Vi IMproved, or **vim**. On some Linux distributions, **vim** is the default version of **vi** and runs when you invoke **vi**. **vim** changes some of the basic features of **vi**, most notoriously changing the undo key to support multiple levels of undo. While seasoned users of **vi** find **vim**'s changes disturbing, those new to **vi** find **vim**'s extensive features attractive.

Fully documenting **vim** is beyond the scope of this chapter, but we do cover some of its most commonly used options and features. Beyond what we cover here, **vim** offers enhanced support to programmers through an integrated build and debugging process, syntax

highlighting, extended **ctags**

support, and support for Perl and Python, as well as GUI fonts and menus, function key mapping, independent mapping for each mode, and more. Fortunately, **vim** comes with a powerful help program you can use to learn more about the things we just couldn't fit into this chapter.

For more information, see the O'Reilly book

Learning the vi Editor.

11.1 Review of vi Operations

This section provides a review of the following:

- Command-line options
- **vi** modes
- Syntax of **vi** commands
- Status-line commands

11.1.1 Command Mode

Once the file is opened, you are in command mode. From command mode, you can:

- Invoke insert mode
- Issue editing commands
- Move the cursor to a different position in the file
- Invoke **ex** commands
- Invoke a Linux shell
- Save or exit the current version of the file

11.1.2 Insert Mode

In insert mode, you can enter new text in the file. Press the Esc or Ctrl-[keys to exit insert mode and return to command mode. The following commands invoke insert mode:

a

Append after cursor

A

Append at end-of-line

c

Begin change operation (must be followed by a movement command)

C

Change to end-of-line

i

Insert before cursor

I

Insert at beginning of line

o

Open a line below current line

O

Open a line above current line

r

Replace character under cursor

R

Begin overwriting text

s

Substitute a character

S

Substitute entire line

11.1.3 Syntax of vi Commands

In **vi**, commands have the following general form: [*n*] *operator* [*m*] *object*

The basic editing *operators* are:

c

Begin a change

d

Begin a deletion

y

Begin a yank (or copy)

If the current line is the object of the operation, the operator is the same as the object: **cc**, **dd**, **yy**. Otherwise, the editing operators act

on objects specified by cursor-movement commands or pattern-matching commands. *n* and *m* are the number of times the operation is performed or the number of objects the operation is performed on. If both *n* and *m* are specified, the effect is *n* x *m*.

An object can represent any of the following text blocks:

Word

Includes characters up to a space or punctuation mark. A capitalized object is a variant form that recognizes only blank spaces.

Sentence

Extends to ., **!**, or **?** followed by two spaces.

Paragraph

Extends to next blank line or **nroff/troff** paragraph macro (defined by **para=** *option*).

Section

Extends to next **nroff/troff** section heading (defined by **sect=** *option*).

11.1.3.1 Examples

2cw

Change the next two words

d}

Delete up to next paragraph

d^

Delete back to beginning of line

5yy

Copy the next five lines into temporary buffer (for future pasting)

y]]

Copy up to the next section into temporary buffer (for future pasting)

11.1.4 Status-Line Commands

Most commands are not echoed on the screen as you input them. However, the status line at the bottom of the screen is used to echo input for the following commands:

/

Search forward for a pattern

?

Search backward for a pattern

:

Invoke an **ex** command

!

Pipe the text indicated by a subsequent movement command through the following shell command, and replace the text with the output of the shell command

Commands that are input on the status line must be entered by pressing the Return key. In addition, error messages and output from the **Ctrl-G** command are displayed on the status line.

vi file

vi + n file

vi +/ pattern file

You can open file for editing, optionally at line n or at the first line matching pattern. If no file is specified, vi opens with an empty buffer. The command-line options that can be used with vi are as follows (vim-only options are labeled):

+[num]

Start editing at line number num, or the last line of the file if num is omitted.

+/pattern

Start editing at the first line matching pattern. (Fails if nowrapscan is set in your .exrc startup file.)

-b

Edit the file in binary mode. {vim}

-c command

Run the given vi command upon startup. Only one -c option is permitted. ex commands can be invoked by prefixing them with a colon. An older form of this option, +command, is still supported.

--cmd command

Like -c, but execute the command before any resource files are read. {vim}

-d

Run in diff mode. Works like vimdiff. {vim}

-e

Run as ex (line editing rather than full-screen mode).

-h

Print help message, then exit.

-i file

Use the specified file instead of the default .viminfo to save or restore vim's state. {vim}

-l

Enter LISP mode for running LISP programs (not supported in all versions).

-m

Start the editor with the write option turned off so the user cannot write to files. {vim}

-n

Do not use a swap file; record changes in memory only. {vim}

--noplugin

Do not load any plug-ins. {vim}

-o[n]

Start vim with n open windows. The default is to open one window for each file. {vim}

-r [file]

Recovery mode; recover and resume editing on file after an aborted editor session or system crash. Without file, list files available for recovery.

-s, -s scriptfile

When running in ex mode (-e), suppress prompts or informative messages sent to the console. Otherwise, read and execute commands given in the specified scriptfile as if they were typed in from the keyboard. {vim}

-t tag

Edit the file containing tag and position the cursor at its definition. (See ctags in Chapter 3 for more information.)

-u file

Read configuration information from the specified resource file instead of default .vimrc resource files. If the file argument is NONE, vim will read no resource files, load no plug-ins, and

run in compatible mode. If the argument is NORC, it will read no resource files but it will load plug-ins. {vim}

-v

Run in full-screen mode (default).

--version

Print version information, then exit.

-w rows

Set the window size so rows lines at a time are displayed; useful when editing over a slow dial-up line.

-x

Prompt for a key that will be used to try to encrypt or decrypt a file using crypt (not supported in all versions).

-y

Modeless vi; run vim in insert mode only, without a command mode. This is the same as invoking vim as evim. {vim}

-C

Same as -x, but assume the file is encrypted already (not supported in all versions). For vim this option starts the editor in vi-compatible mode.

-D

Debugging mode for use with scripts. {vim}

-L

List files that were saved due to an aborted editor session or system crash (not supported in all versions). For vim this option is the same as -r.

-M

Do not allow text in files to be modified. {vim}

-N

Run vim in a non-vi-compatible mode. {vim}

-O[n]

Start vim with n open windows arranged vertically on the screen. {vim}

-R

Edit files read-only.

-S commandfile

Source commands given in commandfile after loading any files for editing specified on the command line. Shorthand for the option -c source. {vim}

-T type

Set the terminal type. This value overrides the $TERM environment variable. {vim}

-V[n]

Verbose mode; print messages about what options are being set and what files are being read or

written. You can set a level of verbosity to increase or decrease the number of messages received. The default value is 10 for high verbosity. {vim}

-W scriptfile

Write all typed commands from the current session to the specified scriptfile. The file created can be used with the -s command. {vim}

-Z

Start vim in restricted mode. Do not allow shell commands or suspension of the editor. {vim}

11.3 ex Command-Line Options

While most people know **ex** commands only by their use within

vi, the editor also exists as a

separate program and can be invoked from the shell (for instance, to

edit files as part of a script). Within **ex**, you can enter the **vi** or **visual**

command to start **vi**. Similarly,

within **vi**, you can enter **Q** to quit the **vi** editor and enter **ex**.

If you invoke **ex** as a standalone

editor, you can include the following options:

+[*num*]

Start editing at line number *num*, or the last

line of the file if *num* is omitted.

+/*pattern*

Start editing at the first line matching

pattern. (Fails if **nowrapscan** is set in your

.exrc startup file.)

-c *command*

Run the given **ex** command upon

startup. Only one **-c** option is

permitted. An older form of this option, **+***command*, is still

supported.

-e

Run as a line editor rather than full-screen **vi** mode (default).

-l

Enter LISP mode for running LISP programs (not supported in all

versions).

-r [*file***]**

Recover and resume editing on *file* after an

aborted editor session or system crash. Without

file, list files available for recovery.

-s

Silent; do not display prompts. Useful when running a script. This

behavior also can be set through the older **-** option.

-t *tag*

Edit the file containing *tag* and position the

cursor at its definition (see **ctags**

in Chapter 3 for more information).

-v

Run in full-screen mode (same as invoking **vi**).

-w *rows*

Set the window size so *rows* lines at a time are

displayed; useful when editing by a slow dial-up line.

-x

Prompt for a key that will be used to try to encrypt or decrypt a file using **crypt** (not supported in all versions).

-C

Same as **-x**, but assume the file is encrypted already (not supported in all versions).

-L

List files that were saved due to an editor or system crash (not supported in all versions).

-R

Edit files read-only; do not allow changes to be saved.

You can exit **ex** in several ways:

:x

Exit (save changes and quit).

:q!

Quit without saving changes.

:vi

Enter the **vi** editor.

11.4 Movement Commands

Some versions of **vi** do not recognize extended keyboard keys (e.g., arrow keys, Page Up, Page Down, Home, Insert, and Delete); some do. All, however, recognize the keys in this section. Many users of **vi** prefer to use these keys, as it helps them keep their fingers on the home row of the keyboard. A number preceding a command repeats the movement. Movement commands are also objects for change, delete, and yank operations.

11.4.1 Character

Command	Action
h, **j**, **k**, **l**	Left, down, up, right (, , , →)
Spacebar	Right
Backspace	Left
Ctrl-H	Left

11.4.2 Text

Command	Action
w, **b**	Forward, backward by word (treating punctuation marks as words).
W, **B**	Forward, backward by word (recognizing only whitespace, not punctuation, as separators).
e	End of word (treating a punctuation mark as the end of a word).
E	End of word (recognizing only whitespace as the end of a word).
ge	End of previous word (treating a punctuation mark as the end of a word). {vim}
gE	End of previous word (recognizing only whitespace as the end of a word). {vim}
), **(**	Beginning of next, current sentence.
}, **{**	Beginning of next, current paragraph.
]], **[[**	Beginning of next, current section.

Command	Action
][, **[ ]**	End of next, current section. {vim}

11.4.3 Lines

Long lines in a file may show up on the screen as multiple lines. While most commands work on the lines as defined in the file, a few commands work on lines as they appear on the screen.

Command	Action
0, **$**	First, last position of current line.
^, _	First nonblank character of current line.
+, **-**	First character of next, previous line.
Return	First nonblank character of next line.
***n*|**	Column *n* of current line.
g0, **g$**	First, last position of screen line. {vim}

Command	Action
g^	First nonblank character of screen line. {vim}
gm	Middle of screen line. {vim}
gk, **gj**	Move up, down one screen line. {vim}
H	Top line of screen.
M	Middle line of screen.
L	Last line of screen.
nH	*n* lines after top line of screen.
nL	*n* lines before last line of screen.

11.4.4 Screens

Command	Action
Ctrl-F, Ctrl-B	Scroll forward, backward one screen.

Command	Action
Ctrl-D, Ctrl-U	Scroll down, up one-half screen.
Ctrl-E, Ctrl-Y	Show one more line at bottom, top of window.
z Return	Reposition line with cursor to top of screen.
z.	Reposition line with cursor to middle of screen.
z-	Reposition line with cursor to bottom of screen.
Ctrl-L	Redraw screen (without scrolling).

11.4.5 Searches

Command	Action
/*pattern*	Search forward for *pattern*.
/	Repeat previous search forward.
/*pattern****/+****n*	Go to line *n* after *pattern*.

Command	Action
?*pattern*	Search backward for *pattern*.
?	Repeat previous search backward.
?*pattern***?-***n*	Go to line *n* before *pattern*.
n	Repeat previous search.
N	Repeat previous search in opposite direction.
%	Find match of current parenthesis, brace, or bracket.
*****	Search forward for word under cursor. Matches only exact words. {vim}
#	Search backward for word under cursor. Matches only exact words. {vim}
g*	Search backward for word under cursor. Matches the characters of this word when embedded in a longer word. {vim}

Command	Action
g#	Search backward for word under cursor. Matches the characters of this word when embedded in a longer word. {vim}
f*x*	Move forward to *x* on current line.
F*x*	Move backward to *x* on current line.
t*x*	Move forward to just before *x* in current line.
T*x*	Move backward to just after *x* in current line.
,	Reverse search direction of last **f**, **F**, **t**, or **T**.
;	Repeat last character search (**f**, **F**, **t**, or **T**).
:noh	Suspend search highlighting until next search. {vim}.

11.4.5.1 Line numbering

Command	Action

Command	Action
Ctrl-G	Display current filename and line number.
gg	Move to first line in file. {vim}
*n***G**	Move to line number *n*.
G	Move to last line in file.
:*n*	Move to line number *n*.

11.4.5.2 Marking position

Command	Action
m*x*	Mark current position with character *x*.
`*x*	(backquote) Move cursor to mark *x*.
'*x*	(apostrophe) Move to start of line containing *x*.

Command	Action
``	(backquotes) Return to previous mark (or location prior to search).
''	(apostrophes) Like preceding, but return to start of line.
'''	(apostrophe quote) Move to position when last editing the file. {vim}
`[,`]	(backquote bracket) Move to beginning/end of previous text operation. {vim}
'[,']	(apostrophe bracket) Like preceding, but return to start of line where operation occurred. {vim}
`.	(backquote period) Move to last change in file. {vim}
'.	(apostrophe period) Like preceding, but return to start of line. {vim}
:marks	List active marks. {vim}

11.5 Edit Commands

Recall that **c**, **d**, and **y** are the basic editing operators.

11.5.1 Inserting New Text

Command	Action
a	Append after cursor.
A	Append to end of line.
i	Insert before cursor.
I	Insert at first nonblank character of line.
gI	Insert at beginning of line. {vim}
o	Open a line below cursor.
O	Open a line above cursor.
Esc	Terminate insert mode.

The following commands work in insert mode.

Command	Action
Tab	Insert a tab.
Backspace	Delete previous character.
Ctrl-E	Insert character found just below cursor. {vim}
Ctrl-Y	Insert character found just above cursor. {vim}
Ctrl-H	Delete previous character (same as Backspace).
Delete	Delete current character.
Ctrl-W	Delete previous word. {vim}
Ctrl-A	Repeat last insertion. {vim}
Ctrl-I	Insert a tab.
Ctrl-N	Insert next completion of the pattern to the left of the cursor. {vim}

Command	Action
Ctrl-P	Insert previous completion of the pattern to the left of the cursor. {vim}
Ctrl-T	Shift line right to next shift width. {vim}
Ctrl-D	Shift line left to previous shift width. {vim}
Ctrl-U	Delete current line.
Ctrl-V	Insert next character verbatim.
Ctrl-[	Terminate insert mode.

Some of the control characters listed in the previous table are set by **stty**. Your terminal settings may differ.

11.5.2 Changing and Deleting Text

The following table is not exhaustive, but illustrates the most common operations.

Command	Action

Command	Action
cw	Change through end of current word.
cc	Change line.
c$	Change text from current position to end-of-line.
C	Same as **c$**.
dd	Delete current line.
d$	Delete remainder of line.
D	Same as **d$**.
*n***dd**	Delete *n* lines.
dw	Delete a word.
d}	Delete up to next paragraph.
d^	Delete back to beginning of line.

Command	Action
d/*pattern*	Delete up to first occurrence of pattern.
dn	Delete up to next occurrence of pattern.
df*a*	Delete up to and including *a* on current line.
dt*a*	Delete up to (not including) *a* on current line.
dL	Delete up to last line on screen.
dG	Delete to end-of-file.
gqap	Reformat current paragraph to **textwidth**. {vim}
g~w	Switch case of word. {vim}
guw	Change word to lowercase. {vim}
gUw	Change word to uppercase. {vim}
p	Insert last deleted or yanked text after cursor.

Command	Action
gp	Same as **p**, but leave cursor at end of inserted text. {vim}
]p	Same as **p**, but match current indention. {vim}
[p	Same as **P**, but match current indention. {vim}
P	Insert last deleted or yanked text before cursor.
gP	Same as **P**, but leave cursor at end of inserted text. {vim}
r*x*	Replace character with *x*.
R*text*	Replace *text* beginning at cursor.
s	Substitute character.
*n***s**	Substitute *n* characters.
S	Substitute entire line.
u	Undo last change.

Command	Action
Ctrl-R	Redo last change. {vim}
U	Restore current line.
x	Delete current character.
X	Delete back one character.
*n***X**	Delete previous *n* characters.
.	Repeat last change.
~	Reverse case.
&	Repeat last substitution.
Y	Copy (yank) current line to temporary buffer.
yy	Same as **Y**.
"*x***yy**	Copy current line to buffer *x*.

Command	Action
ye	Copy text to end of word into temporary buffer.
yw	Same as **ye**.
y$	Copy rest of line into temporary buffer.
"*x***dd**	Delete current line into buffer *x*.
"*X***dd**	Delete current line and append to buffer *x*.
"*x***p**	Put contents of buffer *x*.
J	Join previous line to current line.
gJ	Same as **J**, but without inserting a space. {vim}
:j!	Same as **J**.
Ctrl-A	Increment number under cursor. {vim}
Ctrl-X	Decrement number under cursor. {vim}

11.6 Saving and Exiting

Writing a file means saving the edits and updating the file's modification time.

Command	Action
ZZ	Quit **vi**, writing the file only if changes were made.
:x	Same as **ZZ**.
:wq	Write and quit file.
:w	Write file.
:w *file*	Save copy to *file*.
:*n1*,*n2***w** *file*	Write lines *n1* to *n2* to new *file*.
:*n1*,*n2***w >>** *file*	Append lines *n1* to *n2* to existing *file*.
:w!	Write file (overriding protection).
:w! *file*	Overwrite *file* with current buffer.

Command	Action
:w %.*new*	Write current buffer named *file* as *file.new*.
:q	Quit file.
:q!	Quit file (discarding edits).
Q	Quit **vi** and invoke **ex**.
:vi	Return to **vi** after **Q** command.
%	Current filename.
#	Alternate filename.

11.7 Accessing Multiple Files

Command	Action
:e *file*	Edit *file*; current file becomes the alternate file.
:e!	Restore last saved version of current file.
:e+ *file*	Begin editing at end of new *file*.
:e+ *n file*	Open new *file* at line *n*.
:e#	Open to previous position in alternate (previously edited) file.
:ta *tag*	Edit file containing *tag* at the location of the tag.
:n	Edit next file.
:n!	Force next file into buffer (don't save changes to current file).
:n *files*	Specify new list of *files*.

Command	Action
:args	Display multiple files to be edited.
:rew	Rewind list of multiple files to top.

11.8 Window Commands

The following table lists common commands for controlling windows in vim. See also the **split**, **vsplit**, and **resize** commands in Chapter 3. For brevity, control characters are marked in the following list by ^.

Command	Action
:new	Open a new window.
:new *file*	Open *file* in a new window.
:sp *file*	Split the current window.
:sv*file*	Same as **:sp**, but make new window read-only.
:sn*file*	Edit next file in new window.
:clo	Close current window.
:hid	Hide current window, unless it is the only visible window.
:on	Make current window the only visible one.

Command	Action
:res *n*	Resize window to *n* lines.
:wa	Write all changed buffers to file.
:qa	Close all buffers and exit.
^W s	Same as **:sp**.
^W n	Same as **:new**.
^W ^	Open new window with alternate (previously edited) file.
^W c	Same as **:clo**.
^W o	Same as **:only**.
^W j, ^W k	Move cursor to next/previous window.
^W p	Move cursor to previous window.
^W h, ^W l	Move cursor to window on left/right.

Command	Action
^W t, ^W b	Move cursor to window on top/bottom of screen.
^W K, ^W B	Move current window to top/bottom of screen.
^W H, ^W L	Move current window to far left/right of screen.
^W r, ^W R	Rotate windows down/up.
^W +, ^W -	Increase/decrease current window size.
^W =	Make all windows same height.

11.9 Interacting with the Shell

Command	Action
:r *file*	Read in contents of *file* after cursor.
:r !_command_	Read in output from *command* after current line.
:nr !_command_	Like preceding, but place after line *n* (0 for top of file).
:!_command_	Run *command*, then return.
!_object command_	Send *object*, indicated by a movement command, as input to shell command *command*; replace *object* with command output.
:_n1_,_n2_**!** _command_	Send lines *n1* through *n2* to *command*; replace with output.
_n_**!!**_command_	Send *n* lines to *command*; replace with output.
!!	Repeat last system command.

Command	Action
!!*command*	Replace current line with output of *command*.
:sh	Create subshell; return to file with EOF.
Ctrl-Z	Suspend editor; resume with **fg**.
:so *file*	Read and execute **ex** commands from *file*.

11.10 Macros

Command	Action
:ab *in* *out*	Use *in* as abbreviation for *out*.
:unab *in*	Remove abbreviation for *in*.

Command	Action
:ab	List abbreviations.
:map *c* *sequence*	Map character *c* as *sequence* of commands.
:unmap *c*	Disable map for character *c*.
:map	List characters that are mapped.
:map! *c* sequence	Map character *c* to input mode *sequence*.

Command	Action
:unmap! *c*	Disable input mode map (you may need to quote the character with Ctrl-V).
:map!	List characters that are mapped to input mode.
qx	Record typed characters into register specified by letter *x*. If letter is uppercase, append to register.{vim}

Command	Action
q	Stop recording. {vim}
@x	Execute the register specified by letter *x*. {vim}

In vi, the following

characters are unused in command mode and can be mapped as

user-defined commands:

Letters

g K q V v

Control keys

^K ^O ^T ^W ^X

Symbols

_ * \ =

The = is used by **vi** if LISP mode is

set. **vim** uses all of these

characters, but you can create macros for function keys and multiple

character commands. See **:help :map**

for details. Other versions of **vi**

may use some of these characters as well, so test them before using

them.

11.11 Miscellaneous Commands

Command	Action
<	Shift line left to position indicated by following movement command.
>	Shift line right to position indicated by following movement command.
<<	Shift line left one shift width (default is 8 spaces).
>>	Shift line right one shift width (default is 8 spaces).
>}	Shift right to end of paragraph.
<%	Shift left until matching parenthesis, brace, bracket, etc. (Cursor must be on the matching symbol.)
=	Indent line in C-style, or using program specified in **equalprg** option. {vim}
K	Look up word under cursor in manpages (or program defined in **keywordprg**). {vim}

Command	Action
^[	Abort command or end input mode.
^]	Perform a tag lookup on the text under the cursor.
^	Enter **ex** line-editing mode.
^^	(Caret key with Ctrl key pressed) Return to previously edited file.

11.12 Alphabetical List of Keys in Command Mode

For brevity, control characters are marked by **^**.

Command	Action
a	Append text after cursor.
A	Append text at end-of-line.
^A	Search for next occurrence of word under cursor. Increment number in **vim** when cursor is on a number.
b	Back up to beginning of word in current line.
B	Back up one word, treating punctuation marks as words.
^B	Scroll backward one window.
c	Change text up to target of next movement command.
C	Change to end of current line.

Command	Action
^C	End insert mode; interrupts a long operation.
d	Delete up to target of next movement command.
D	Delete to end of current line.
^D	Scroll down half-window; in insert mode, unindent to **shiftwidth** if **autoindent** is set (or when using **vim**).
e	Move to end of word.
E	Move to end of word, treating punctuation as part of word.
^E	Show one more line at bottom of window.
f	Find next character typed forward on current line.
F	Find next character typed backward on current line.
^F	Scroll forward one window.

Command	Action
g	Unused in **vi**. Begins many multiple-character commands in **vim**.
G	Go to specified line or end-of-file.
^G	Print information about file on status line.
h	Left arrow cursor key.
H	Move cursor to home position.
^H	Left arrow cursor key; Backspace key in insert mode.
i	Insert text before cursor.
I	Insert text before first nonblank character on line.
^I	Unused in command mode; in insert mode, same as Tab key.
j	Down arrow cursor key.
J	Join previous line to current line.

Command	Action
^J	Down arrow cursor key; in insert mode, move down a line.
k	Up arrow cursor key.
K	Unused in **vi**. Look up word using **keywordprg** in **vim**.
^K	Unused in **vi**. Insert multiple-keystroke character in **vim**.
l	Right arrow cursor key.
L	Move cursor to last position in window.
^L	Redraw screen.
m	Mark the current cursor position in register (a-z).
M	Move cursor to middle position in window.
^M	Move to beginning of next line.
n	Repeat the last search command.

Command	Action
N	Repeat the last search command in reverse direction.
^N	Down arrow cursor key.
o	Open line below current line.
O	Open line above current line.
^O	Unused in **vi**. Return to previous jump position in **vim**.
p	Put yanked or deleted text after or below cursor.
P	Put yanked or deleted text before or above cursor.
^P	Up arrow cursor key.
q	Unused in **vi**. Record keystrokes in **vim**.
Q	Quit **vi** and enter **ex** line-editing mode.
^Q	Unused in **vi**. Same as **^V** in **vim** (On some terminals, resume data flow.)

Command	Action
r	Replace character at cursor with the next character you type.
R	Replace characters.
^R	Redraw the screen.
s	Change the character under the cursor to typed characters.
S	Change entire line.
^S	Unused. (On some terminals, stop data flow.)
t	Find next character typed forward on current line and position cursor before it.
T	Find next character typed backward on current line and position cursor after it.
^T	Unused in command mode for **vi**. Pop tag from tagstack in **vim**. In insert mode, move to next tab setting.

Command	Action
u	Undo the last change made. In **vi**, a second undo redoes an undone command. **vim** supports multiple levels of undo. To redo, use **Ctrl-R**.
U	Restore current line, discarding changes.
^U	Scroll the screen upward a half-window.
v	Unused in **vi**. Enter visual mode in **vim**.
V	Unused in **vi**. Enter linewise visual mode in **vim**.
^V	Unused in command mode for **vi**. Enter blockwise visual mode in **vim**. In insert mode, insert next character verbatim.
w	Move to beginning of next word.
W	Move to beginning of next word, treating punctuation marks as words.
^W	Unused in command mode in **vi**. Begins window commands in **vim**. In insert mode, back up to beginning of word.

Command	Action
x	Delete character under cursor.
X	Delete character before cursor.
^X	Unused in **vi**. Decrement number in **vim** when cursor is on a number. In insert mode in **vim**, begins several commands.
y	Yank or copy text up to target of following movement command into temporary buffer.
Y	Make copy of current line.
^Y	Show one more line at top of window.
z	Reposition line containing cursor. **z** must be followed by Return (reposition line to top of screen), . (reposition line to middle of screen), or **-** (reposition line to bottom of screen).
ZZ	Exit the editor, saving changes.

11.13 Syntax of ex Commands

To enter an **ex** command from **vi**, type: :[*address*] *command* [*options*]

An initial : indicates an **ex** command. As you type the command, it is echoed on the status line. Enter the command by pressing Return. *address* is the line number or range of lines that are the object of *command*. *options* and *addresses* are described in the following sections. **ex** commands are described in the alphabetical summary.

11.13.1 Options

!

> Indicates a variant command form, overriding the normal behavior.

count

> The number of times the command is to be repeated. Unlike **vi** commands, the *count* comes after the command, not before it. Numbers preceding an **ex** command are considered to be part of the *address*. For example, **3d** deletes line 3; **d3** deletes 3 lines beginning with the current line.

file

> The name of a file that is affected by the command. **%** stands for current file; # stands for previous file.

11.13.2 Addresses

If no address is given, the current line is the object of the command. If the address specifies a range of lines, the format is: *x*,*y*

where *x* and *y* are the first and last addressed lines (*x* must precede *y* in the buffer). *x* and *y* may be line numbers or symbols. Using ; instead of , sets the current line to *x* before interpreting *y*.

11.13.3 Address Symbols

Symbol	Meaning
1,$	All lines in the file
%	All lines; same as **1,$**
x,*y*	Lines *x* through *y*
x;*y*	Lines *x* through *y*, with current line reset to *x*
0	Top of file
.	Current line
n	Absolute line number *n*
$	Last line

Symbol	Meaning
x-*n*	*n* lines before *x*
x+*n*	*n* lines after *x*
-[*n*]	One or *n* lines previous
+[*n*]	One or *n* lines ahead
'*x*	Line marked with *x*
''	Previous mark
/*pattern*/	Forward to line matching *pattern*
?*pattern*?	Backward to line matching *pattern*

See Chapter 9 for more information on using patterns.

ab [<tt>
string text</tt>]

:ab ora O'Reilly & Associates, Inc.

:ab id Name:^MRank:^MPhone:

[<tt>address</tt>] <span
class="docEmphBold">a[<span
class="docEmphBold">!]

a   Begin appending to current line.

Append this line

and this line too.

.   Terminate input of text to append.

ar

cd <tt>
dir</tt>

chdir <tt>
dir</tt>

[<tt>n</tt>] <span
class="docEmphBold">bd[<span
class="docEmphBold">!] [<tt>n
</tt>]

[<tt>n</tt>] <span
class="docEmphBold">b[<span
class="docEmphBold">!] [<tt>n
</tt>]

buffers[<span
class="docEmphBold">!]

[<tt>address</tt>] <span
class="docEmphBold">ce [<tt>width
</tt>]

[<tt>address</tt>] <span
class="docEmphBold">c[<span

class="docEmphBold">!] <tt>text
</tt>

clo[<span
class="docEmphBold">!]

[<tt>address</tt>] <span
class="docEmphBold">co <tt>
destination</tt>

1,10 co 50   Copy first 10 lines to just
after line 50

[<tt>address</tt>] <span
class="docEmphBold">d [<tt>buffer
</tt>]

/Part I/,/Part II/-1d   Delete to line above
"Part II"

:/main/+d   Delete line below "main"

:.,$d   Delete from this line to last
line

e[!] [+<tt>n</tt>] [<tt>file</tt>]

e file

:e#   Return to editing the previous file

:e!   Discard edits since last save

exu [<tt> command</tt>]

f [<tt> filename</tt>]

f %.new

<tt>address</tt> fo

[<tt>address</tt>] foldc[!]

[<tt>address</tt>] foldo[!]

[<tt>address</tt>] g[!]/<tt>pattern</tt>/[<tt>commands</tt>]

g/Unix/p   Print all lines containing "Unix"

:g/Name:/s/tom/Tom/   Change "tom" to "Tom" on all lines

 containing "Name:"

h

hid

<tt>address</tt> i[!]

[<tt>address</tt>] j[!] [<tt>count</tt>]

1,5j!   Join first five lines, preserving whitespace

ju

[<tt>address</tt>] k <tt>char</tt>

[<tt>address</tt>] l [<tt>count</tt>]

[<tt>address</tt>] le [<tt>count</tt>]

map[!] [<tt>char commands</tt>]

map K dwwP   Transpose two words

:map q :w^M:n^M   Write current file; go to next

:map! + ^[bi(^[ea)   Enclose previous word in parentheses

[<tt>address</tt>] ma <tt>char</tt>

marks [<tt>chars</tt>]

marks abc   Print marks a, b and c.

mk[!] <tt>file

</tt>

[<tt>address</tt>] m <tt>destination</tt>

.,/Note/m /END/   Move text block after line containing "END"

[<tt>count</tt>]new

n[!] [[+<tt>command</tt>] <tt>filelist</tt>]

n chap*   Start editing all "chapter" files

noh

[<tt>address</tt>] nu [<tt>count</tt>]

[<tt>address</tt>] o [/<tt>pattern</tt>/]

pre

prev[!]

[<tt>address</tt>] p [<tt>count</tt>]

[<tt>address</tt>] P [<tt>count</tt>]

100;+5p   Show line 100 and the next 5 lines

[<tt>address</tt>] pu [<tt>char</tt>]

qa[!]

q[!]

[<tt>address</tt>] r <tt>file</tt>

0r $HOME/data   Read file in at top of current file

[<tt>address</tt>] r !<tt> command</tt>

$r !cal   Place a calendar at end-of-file

rec [<tt> file</tt>]

red

res [[+|-]<tt>n</tt>]

rew[!]

[<tt>address</tt>] le [<tt>width</tt>]

[<tt>n</tt>] sb [<tt>n</tt>]

[<tt>count</tt>] sbn [<tt>count</tt>]

[<tt>count</tt>] sn [[+<tt>n</tt>] <tt>filelist</tt>]

[<tt>count</tt>] sp [+<tt>n</tt>] [<tt>filename</tt>]

[<tt>count</tt>] spr [<span

class="docEmphBold">+<tt>n</tt>]

sc[!] [<tt>file
</tt>]

se <tt>
parameter1 parameter2 ..</tt>.

set nows wm=10

:set all

sh

so <tt>
file</tt>

so $HOME/.exrc

st

[<tt>address</tt>] s [/<tt>pattern

</tt>/<tt>
replacement</tt>/] [<tt>
options</tt>] [<tt>count</tt>]

1,10s/yes/no/g   Substitute on first 10 lines

:%s/[Hh]ello/Hi/gc   Confirm global substitutions

:s/Fortran/\U&/ 3   Uppercase first instance of

 "Fortran" on next three lines

:g/^[0-9][0-9]*/s//Line &:/   For every line beginning with one or

 more digits, add the "Line" and a colon

su

[<tt>count</tt>] sv [+<tt>n</tt>] [<tt>filename</tt>

[<tt>address</tt>] t <tt> destination</tt>

%t$ Copy the file and add it to the end

[<tt>address</tt>] ta[!] <tt>tag </tt>

:!ctags *.c

:tag myfunction

tags

una <tt> word</tt>

u

[<tt>count</tt>] unh

unm[!] <tt>char</tt>

[<tt>address</tt>] v/<tt>pattern</tt>/[<tt>commands</tt>]

v/#include/d   Delete all lines except "#include" lines

ve

vi [+<tt>n</tt>] <tt>file</tt>

vie[[+<tt>n</tt>] <tt>filename</tt>]

[<tt>address</tt>] vi [<tt>type</tt>] [<tt>count</tt>]

viu [<tt>key</tt>]

[<tt>count</tt>] vs [+<tt>n</tt>] [<tt>filename</tt>]

wa[!]

[<tt>count</tt>] wn[!] [[+<tt>n</tt>] <tt>filename</tt>]

wq[!]

wqa[!]

[<tt>address</tt>] w[!] [[>>>] <tt>file</tt>]

[<tt>address</tt>] w !<tt>command</tt>

1,10w name_list   Copy first 10 lines to name_list

:50w >> name_list   Now append line 50

X

x

[<tt>address</tt>] ya [<tt>char</tt>] [<tt>count</tt>]

101,200 ya a

[<tt>address</tt>] z [<tt>type</tt>] [<tt>count</tt>]

[<tt>address</tt>] !<tt>command</tt>

!ls   List files in the current directory

:11,20!sort -f   Sort lines 11-20 of current file

[<tt>address</tt>] =

[<tt><i>address</i></tt>]<[<tt><i>count</i></tt>]

[<tt>address</tt>]>[<tt>count</tt>]

<tt>address</tt>

<tt>Return</tt>

[<tt>address</tt>] @ [<tt>char</tt>]

[<tt>address</tt>] @

& [<tt>options</tt>] [<tt>count</tt>]

s/Overdue/Paid/   Substitute once on current line

:g/Status/&   Redo substitution on all "Status" lines

[<tt>address</tt>] ~ [<tt>count</tt>]

Replace the previous regular expression with the previous replacement pattern from a substitute (s) command.

11.15 vi Configuration

This section describes the following:

- The **:set** command
- Options available with **:set**
- Sample *~/.exrc* file

11.15.1 The :set Command

The **:set** command lets you specify options that change characteristics of your editing environment. Options may be put in the *~/.exrc* file or set during a **vi** session.

The colon should not be typed if the command is put in *~/.exrc*.

Command	Action
:set *x*	Enable option *x*.
:set no*x*	Disable option *x*.
:set *x*=*val*	Give *value* to option *x*.

Command	Action
:set	Show changed options.
:set all	Show all options.
:set *x*?	Show value of option *x*.

11.15.2 Options Used by :set

The following table describes the options to **:set**. The first column includes the optional abbreviation, if there is one, and uses an equals sign to show that the option takes a value. The second column gives the default, and the third column describes the behavior of the enabled option.

Option	Default	Description
autoindent (ai)	**noai**	In insert mode, indent each line to the same level as the line above or below.
autoprint (ap)	**ap**	Display changes after each editor command. (For global replacement, display last replacement.)

Option	Default	Description
autowrite (**aw**)	**noaw**	Automatically write (save) file if changed, before opening another file with **:n** or before giving a Linux command with **:!**.
background (bg)		Describe the background so the editor can choose appropriate highlighting colors. Default value of **dark** or **light** depends on the environment in which the editor is invoked. {vim}
backup (bk)	**nobackup**	Create a backup file when overwriting an existing file. {vim}
backupdir= (bdir)	**.,~/tmp/,~/**	Name directories in which to store backup files if possible. The list of directories is comma-separated and in order of preference. {vim}
backupext= (bex)	**~**	String to append to filenames for backup files. {vim}
beautify (bf)	**nobf**	Ignore all control characters during input (except tab, newline, or formfeed).

Option	Default	Description
cindent (cin)	**nocindent**	Insert indents in appropriate C format. {vim}
compatible (cp)	**cp**	Make **vim** behave more like **vi**. Default is **nocp** when a *~/.vimrc* file is found. {vim}
directory= (dir)	**/tmp**	Name the directory in which **ex** stores buffer files. (Directory must be writable.)
edcompatible	**noed-compatible**	Use **ed**-like features on substitute commands.
equalprg= (ep)		Use the specified program for the **=** command. When the option is blank (the default), the key invokes the internal C indention function or the value of the **indentexpr** option. {vim}
errorbells (eb)	**errorbells**	Sound bell when an error occurs.
exrc (ex)	**noexrc**	Allow the execution of *~/.exrc* files that reside outside the user's home directory.

Option	Default	Description
formatprg= (fp)		The **gq** command will invoke the named external program to format text. It will call internal formatting functions when this option is blank (the default). {vim}
gdefault (gd)	**nogdefault**	Set the **g** flag on for substitutions by default. {vim}
hardtabs= (ht)	**8**	Define boundaries for terminal hardware tabs.
hidden (hid)	**nohidden**	Hide buffers rather than unload them when they are abandoned. {vim}
hlsearch (hls)	**hlsearch**	Highlight all matches of most recent search.
history= (hi)	**20**	Number of **ex** commands to store in the history table. {vim}
ignorecase (ic)	**noic**	Disregard case during a search.

Option	Default	Description
incsearch (is)	**noincsearch**	Highlight matches to a search pattern as it is typed. {vim}
lisp	**nolisp**	Insert indents in appropriate LISP format. **( )**, **{ }**, **[[**, and **]]** are modified to have meaning for LISP.
list	**nolist**	Print tabs as **^I**; mark ends of lines with **$**. (Use **list** to tell if tabs or spaces are at the end of a line.)
magic	**magic**	Wildcard characters . (dot), ***** (asterisk), and **[ ]** (brackets) have special meaning in patterns.
mesg	**mesg**	Permit system messages to display on terminal while editing in **vi**.
mousehide (mh)	**mousehide**	When characters are typed, hide the mouse pointer. {vim}
number (nu)	**nonu**	Display line numbers on left of screen during editing session.

Option	Default	Description
paste	**nopaste**	Change the defaults of various options to make pasting text into a terminal window work better. All options are returned to their original value when the **paste** option is reset. {vim}
redraw (re)	**noredraw**	Terminal redraws screen whenever edits are made (in other words, insert mode pushes over existing characters, and deleted lines immediately close up). Default depends on line speed and terminal type. **noredraw** is useful at slow speeds on a dumb terminal: deleted lines show up as @, and inserted text appears to overwrite existing text until you press Esc.
remap	**remap**	Allow nested map sequences.
report=	**5**	Display a message on the prompt line whenever you make an edit that affects at least a certain number of lines. For example, **6dd** reports the message "6 lines deleted."

Option	Default	Description
ruler (ru)	**ruler**	Show line and column numbers for the current cursor position. {vim}
scroll=	**<1/2 window>**	Amount of screen to scroll.
sections= (sect)	**SHNHH HUnhsh**	Define section delimiters for **[[ ]]** movement. The pairs of characters in the value are the names of **nroff/troff** macros that begin sections.
shell= (sh)	**/bin/sh**	Pathname of shell used for shell escape (**:!**) and shell command (**:sh**). Default value is derived from **SHELL** variable.
shiftwidth= (sw)	**8**	Define number of spaces used by the indent commands (**^T**, **^D**, **>>**, and **<<**).
showmatch (sm)	**nosm**	In **vi**, when **)** or **}** is entered, cursor moves briefly to matching **(** or **{**. (If the match is not on the screen, rings the error message bell.) Very useful for programming.

Option	Default	Description
showmode	**noshowmode**	In insert mode, displays a message on the prompt line indicating the type of insert you are making, such as "Open Mode" or "Append Mode."
slowopen (slow)		Hold off display during insert. Default depends on line speed and terminal type.
smartcase (scs)	**nosmartcase**	Override the **ignorecase** option when a search pattern contains uppercase characters. {vim}
tabstop= (ts)	**8**	Define number of spaces that a tab indents during editing session. (Printer still uses system tab of 8.)
taglength= (tl)	**0**	Define number of characters that are significant for tags. Default (0) means that all characters are significant.
tags=	**tags** */usr/lib/tags*	Define pathname of files containing tags (see the **ctags** command in Chapter 3). By default, the system looks for files **tags** (in the current directory) and */usr/lib/tags*.

Option	Default	Description
term=		Set terminal type.
terse	**noterse**	Display shorter error messages.
timeout (to)	**timeout**	Keyboard maps timeout after 1 second.
ttytype=		Set terminal type. Default is inherited from **TERM** environment variable.
undolevels= (ul)	**1000**	Number of changes that can be undone. {vim}
warn	**warn**	Display the message "No write since last change."
window= (w)		Show a certain number of lines of the file on the screen. Default depends on line speed and terminal type.
wrapmargin= (wm)	**0**	Define right margin. If greater than 0, automatically insert carriage returns to break lines.

Option	Default	Description
wrapscan (ws)	**ws**	Searches wrap around either end of file.
writeany (wa)	**nowa**	Allow saving to any file.
writebackup (wb)	**wb**	Back up files before attempting to overwrite them. Remove the backup when the file has been sucessfully written.

11.15.3 Sample ~/.exrc File

The following lines of code are an example of a customized *.exrc* file:

```
set nowrapscan wrapmargin=7

set sections=SeAhBhChDh nomesg

map q :w^M:n^M

map v dwElp

ab ORA O'Reilly & Associates, Inc.
```

Chapter 12. The sed Editor

sed is a noninteractive (**s**tream-oriented) **ed**itor that interprets and performs the actions in a script. **sed** is said to be stream-oriented because, as with many Unix programs, input flows through the program and is directed to standard output. For example, **sort** is stream-oriented; **vi** is not. **sed**'s input typically comes from a file but can be directed from the keyboard. Output goes to the screen by default but can be captured in a file instead.

Typical uses of **sed** include:

- Editing one or more files automatically
- Simplifying repetitive edits to multiple files
- Writing conversion programs

sed operates as follows:

- Each line of input is copied into a pattern space.
- All editing commands in a **sed** script are applied in order to each line of input.
- Editing commands are applied to all lines (globally) unless line addressing restricts the lines affected.
- If a command changes the input, subsequent commands are applied to the changed line, not to the original input line.

- The original input file is unchanged because the editing commands modify a copy of the original input line. The copy is sent to standard output (but can be redirected to a file).

For more information on **sed**, see *sed & awk* (O'Reilly).

sed [options] 'command' file(s)

sed [options] -f scriptfile file(s)

The first form allows you to specify an editing command, surrounded by single quotes, on the command line. The second form allows you to specify a scriptfile, which is a file containing sed commands. If no files are specified, sed reads from standard input.

The following options are recognized:

-e cmd, --expression=cmd

> Next argument is an editing command; not needed unless specifying two or more editing commands.

-f scriptfile, --file=scriptfile

> Next argument is a file containing editing commands.

-n, --silent, --quiet

Suppress the default output; sed displays only those lines specified with the p command or with the p flag of the s command.

-V, --version

Display version number.

-h, --help

Display brief help message with command-line options.

[address [,address]][!] command [arguments]

s/xx/yy/g Substitute on all lines (all occurrences) /BSD/d Delete lines containing BSD

/^BEGIN/,/^END/p Print all lines between lines that begin with BEGIN and END, inclusive

/SAVE/!d Delete any line that doesn't contain SAVE /BEGIN/,/END/!s/xx/yy/g Substitute on all lines, except between BEGIN and END 1~2 p Print all odd numbered lines

2~3 d Delete every third line beginning with the second.

[/address/[,/address/]]{

command1

command2

}

The opening curly brace must end a line, and the closing curly brace must be on a line by itself. Be sure there are no blank spaces after the braces.

12.3 Group Summary of sed Commands

In the

following tables, the **sed** commands

are grouped by function and are described tersely. Full descriptions,

including syntax and examples, can be found afterward in the

alphabetical summary.

12.3.1 Basic Editing

Command	Action
a	Append text after a line.
c	Replace text (usually a text block).
i	Insert text before a line.

Command	Action
d	Delete lines.
s	Make substitutions.
y	Translate characters (like **tr** in Chapter 3).

12.3.2 Line Information

Command	Action
=	Display line number of a line.
l	Display control characters in ASCII.
p	Display the line.

12.3.3 Input/Output Processing

Command	Action

Command	Action
n	Skip current line and go to line below.
r	Read another file's contents into the input.
w	Write input lines to another file.
q	Quit the **sed** script (no further output).

12.3.4 Yanking and Putting

Command	Action
h	Copy pattern space into hold space; wipe out what's there.

Command	Action
H	Copy pattern space into hold space; append to what's there.
g	Get the hold space back; wipe out the pattern space.
G	Get the hold space back; append to the pattern space.
x	Exchange contents of hold space and pattern space.

12.3.5 Branching Commands

Command	Action
b	Branch to *label* or to end of script.

Command	Action
t	Same as **b**, but branch only after substitution.
:*label*	Label branched to by **t** or **b**.

12.3.6 Multiline Input Processing

Command	Action
N	Read another line of input (creates embedded newline).
D	Delete up to the embedded newline.
P	Print up to the embedded newline.

#

:label

[/pattern/]=

[address]a\

text

$a\

This goes after the last line in the file\

(marked by $). This text is escaped at the\

end of each line, except for the last one.

[address1[,
address2]]
b[<span
class="docEmphasis">label]

<span
class="docEmphBold">/^\.TS/,/^\.TE/b

[address1[,
address2]]
c\<span
class="docEmphasis">text

1,100c\

\

<First 100 names to be
supplied>

[address1[,
address2]]
d

/^$/d

[address1[,
address2]]
D

/^$/{

N

/^\n$/D

}

[address1[,
address2]]
g

/Item:/H

/<Replace this line with the item list>/g

[address1[, address2]]
G

/Item:/{

H

d

}

/Summary of items:/G

[address1[,
address2]]
h

Edit a line; print the change; replay the original

/Linux/{

h

s/.* Linux \(.*\) .*/\1:/

p

x

}

This describes the Linux ls command.

This describes the Linux cp command.

ls:

This describes the Linux ls command.

cp:

This describes the Linux cp command.

[address1[, address2]] H

[address1]
i\

text

/Item 1/i\

The five items are listed below:

[address1[,
address2]]
l

[address1[,
address2]]
n

/^\.NH/{

n

p

}

[address1[,
address2]]
N

/^\.NH/{

N

p

}

/^\.NH/{

N

s/\n/ /

p

}

[address1[,
address2]]
p

[address1[,
address2]]
P

[address]<span
class="docEmphBold">q

/Garbled text
follows:/q

50q

[address]r file

/The list of items follows:/r item_file

[address1[, address2]] s/pattern/replacement/[flags]

Change third and fourth quote to (and):

/function/{

s/"/(/3

s/"/)/4

}

Remove all quotes on a given line:

/Title/s/"//g

Remove first colon or all quotes; print resulting lines:

s/://p

s/"//gp

Change first "if" but leave "ifdef" alone:

/ifdef/!s/if/ if/

[address1[,
address2]]
t [<span
class="docEmphasis">label]

ID: 1 Name: greg Rate: 45

ID: 2 Name: dale

ID: 3

ID: 1 Name: greg Rate: 45 Phone: ??

ID: 2 Name: dale Rate: ?? Phone: ??

ID: 3 Name: ???? Rate: ?? Phone: ??

/ID/{

s/ID: .* Name: .* Rate: .*/& Phone: ??/p

t

s/ID: .* Name: .*/& Rate: ?? Phone: ??/p

t

s/ID: .*/& Name: ?? Rate: ?? Phone: ??/p

}

[address1[, address2]] w file

Store tbl and eqn blocks in a file:

/^\.TS/,/^\.TE/w troff_stuff

/^\.EQ/,/^\.EN/w troff_stuff

[address1[, address2]] x

[address1[, address2]] y/abc/xyz/

Change item 1, 2, 3 to Item A, B, C ...

/^item [1-9]/y/123456789/ABCDEFGHI/

f

Chapter 13. The gawk Scripting Language

gawk is the GNU version of **awk**, a powerful pattern-matching program for processing text files that may be composed of fixed- or variable-length records separated by some delineator (by default, a newline character). **gawk** may be used from the command line or in **gawk** scripts. You should normally be able to invoke this utility using either **awk** or **gawk** on the shell command line.

With **gawk**, you can:

- Conveniently process a text file as though it were made up of records and fields in a textual database.
- Use variables to change those records and fields.
- Execute shell commands from a script.
- Perform arithmetic and string operations.
- Use programming constructs such as loops and conditionals.
- Define your own functions.
- Process the result of shell commands.
- Produce formatted reports.

For more information on **gawk**, see *sed & awk* (O'Reilly) or *Effective gawk Programming* (O'Reilly).

gawk [options] 'script' var=value file(s)

gawk [options] -f scriptfile var=value file(s)

gawk -F : '{print $1; print $2; print $3}' /etc/passwd

Numerous examples are shown later in Section 13.2.

13.1.1 Options

All options exist in both traditional POSIX (one-letter) format and GNU-style (long) format. Some recognized options are:

--

Treat all subsequent text as commands or filenames, not options.

-f scriptfile, --file=scriptfile

Read gawk commands from scriptfile instead of command line.

-v var=value, --assign=var=value

Assign a value to variable var. This allows assignment before the script begins execution.

-F c, --field-separator=c

Set the field separator to character c. This is the same as setting the variable FS. c may be a regular expression. Each input line, or record, is divided into fields by whitespace (blanks or tabs) or by some other user-definable record separator. Fields are referred to by the variables $1, $2,..., $n. $0 refers to the entire record.

-W option

All -W options are specific to gawk, as opposed to awk. An alternate syntax is --option (i.e., --

compat). option may be one of:

compat, traditional

Behave exactly like traditional (non-GNU) awk.

copyleft, copyright

Print copyleft notice and exit.

dump-variables[=file]

Print the name, type, and value of all global variables to the specified file, or to the file awkvars.out in the current directory if no file is specified.

help, usage

Print syntax and list of options, then exit.

lint[=fatal]

Warn about commands that might not port to other versions of awk or that gawk considers

problematic. When fatal is specified, warnings are treated as fatal errors.

lint-old

Like lint, but compares to an older version of awk used on Version 7 Unix.

non-decimal-data

When reading data, interpret numbers beginning with 0 to be octal, and those beginning with 0x to be hexadecimal. (To print nondecimal numbers, use the printf command, as print prints only string representations of nondecimal numbers.)

posix

Expect exact compatibility with POSIX; disable all gawk extensions as if traditional had been specified. Ignore \x escape sequences, **, **=, the keyword func, and single-tab field separators. Disallow newlines after `?` or : and the fflush function.

profile[=file]

Write a pretty printed version of the script being executed to the specified file, or to the file awkprof.out in the current directory if no other file is specified. When gawk is invoked as pgawk and passed this version of the program with the -f option, it will add profile data to the file inserting execution counts to the left of each statement in the program.

re-interval

Allow use of {n,m} intervals in regular expressions.

source=script

Treat script as gawk commands. Like the 'script' argument, but lets you mix commands from files (using -f options) with commands on the gawk command line.

version

Print version information and exit.

13.2 Patterns and Procedures

gawk scripts consist of patterns and procedures:

```
pattern {procedure}
```

Both are optional. If *pattern* is missing, **{***procedure***}** is applied to all records. If **{***procedure***}** is missing, the matched record is printed. By default, each line of input is a record, but you can specify a different record separator through the **RS** variable.

13.2.1 Patterns

A pattern can be any of the following:

```
/regular expression/

relational expression

pattern-matching expression

pattern,pattern

BEGIN

END
```

Some rules regarding patterns include:

- Expressions can be composed of quoted strings, numbers, operators, functions, defined variables, or any of the predefined variables described later under "gawk System Variables."

- Regular expressions use the extended set of metacharacters and are described in Chapter 9.

- In addition, **^** and **$** can be used to refer to the beginning and end of a field, respectively, rather than the beginning and end of a record.

- Relational expressions use the relational operators listed under "Operators" later in this chapter. Comparisons can be either string or numeric. For example, **$2 > $1** selects lines for which the second field is greater than the first.

- Pattern-matching expressions use the operators **~** (match) and **!~** (don't match). See "Operators" later in this chapter.

- The **BEGIN** pattern lets you specify procedures that take place before the first input record is processed. (Generally, you set global variables here.)

- The **END** pattern lets you specify procedures that take place after the last input record is read.

- If there are multiple **BEGIN** or **END** patterns, their associated actions are taken in the order in which they appear in the script.

- *pattern*,*pattern* specifies a range of lines. This syntax cannot include **BEGIN** or **END** as a pattern.

Except for **BEGIN** and **END**, patterns can be combined with the Boolean operators **||** (OR), **&&** (AND), and **!** (NOT).

In addition to other regular-expression operators, GNU **gawk** supports POSIX character lists, which are useful for matching non-ASCII characters in languages other than English. These lists are

recognized only within **[ ]** ranges. A typical use is **[[:lower:]]**, which in English is the same as **[a-z]**. See Chapter 9 for a complete list of POSIX character lists.

13.2.2 Procedures

Procedures consist of one or more commands, functions, or variable assignments, separated by newlines or semicolons and contained within curly braces. Commands fall into four groups:

- Variable or array assignments
- Printing commands
- Built-in functions
- Control-flow commands

13.2.3 Simple Pattern/Procedure Examples

1. Print first field of each line (no pattern specified):

   ```
   { print $1 }
   ```

2. Print all lines that contain "Linux":

   ```
   /Linux/
   ```

3. Print first field of lines that contain "Linux":

   ```
   /Linux/{ print $1 }
   ```

4. Print records containing more than two fields:

```
NF > 2
```

5. Interpret each group of lines up to a blank line as a single input record:

```
BEGIN { FS = "\n"; RS = "" }
```

6. Print fields 2 and 3 in switched order, but only on lines whose first field matches the string "URGENT":

```
$1 ~ /URGENT/ { print $3, $2 }
```

7. Count and print the number of instances of "ERR" found:

```
/ERR/ { ++x }; END { print x }
```

8. Add numbers in second column and print total:

```
{total += $2 }; END { print "column total is",
total}
```

9. Print lines that contain fewer than 20 characters:

```
length( ) < 20
```

10. Print each line that begins with "Name:" and that contains exactly seven fields:

```
NF =  = 7 && /^Name:/
```

11. Reverse the order of fields:

```
{ for (i = NF; i >= 1; i--) print $i }
```

13.3 gawk System Variables

Variable	Description
$*n*	*n*th field in current record; fields are separated by **FS**
$0	Entire input record
ARGC	Number of arguments on command line
ARGIND	Current file's place in command line (starting with 0)
ARGV	An array containing the command-line arguments
CONVFMT	Conversion format for numbers (default is **%.6g**)
ENVIRON	An associative array of environment variables
ERRNO	Description of last system error
FIELDWIDTHS	List of field widths (whitespace-separated)
FILENAME	Current filename

Variable	Description
FNR	Like **NR**, but relative to the current file
FS	Field separator (default is any whitespace; null string separates into individual characters)
IGNORECASE	If true, make case-insensitive matches
LINT	If true, run as if **-W lint** had been specified on the command line; if value is **FATAL**, generate fatal errors
NF	Number of fields in current record
NR	Number of the current record
OFMT	Output format for numbers (default is **%.6g**)
OFS	Output field separator (default is a blank)
ORS	Output record separator (default is a newline)
RLENGTH	Length of the string matched by **match** function

Variable	Description
RS	Record separator (default is a newline)
RT	The input text matching the record separator (RS) for the current record
RSTART	First position in the string matched by **match** function
SUBSEP	Separator character for array subscripts (default is **\034**)

13.4 PROCINFO Array

gawk stores information about the current process in an array named **PROCINFO**. The array contains the following fields:

Field	Description
egid	Effective group ID
euid	Effective user ID
FS	The type of field splitting being used, either **FS** or **FIELDWIDTHS**
gid	Group ID of the calling process
group*n*	Supplementary user group IDs associated with the process (**group1**, **group2**, etc.)
pgrpid	Process group ID
pid	Process ID
ppid	Parent process ID

Field	Description
uid	User ID of the calling process

13.5 Operators

The following table lists the operators, in order of increasing precedence, that are available in **gawk**.

Symbol	Meaning
= += -= *= /= %= ^= **=	Assignment
?:	C style conditional (*expression* ? *if-true-action* : *if-false-action*)
||	Logical OR
&&	Logical AND
in	Array membership (see **for** command)
~ !~	Match regular expression and negation
< <= > >= != == >> | | &	Relational and redirection operators
(blank)	Concatenation

Symbol	Meaning
+ -	Addition, subtraction
*** / %**	Multiplication, division, and modulus
+ - !	Unary plus and minus and logical negation
^ **	Exponentiation
++ --	Increment and decrement, either prefix or postfix
$	Field reference

```
FS = ","

/<tt><i>pattern</i></tt>/ { <tt><i>array</i></tt>
["/<tt><i>pattern</i></tt>/"]++ }



END { print <tt><i>array</i></tt>["/<tt>
<i>pattern</i></tt>/"] }

var ""

var + 0
```

13.7 Group Listing of gawk Commands

Arithmetic functions	String functions	Control flow statements	Input/output processing	Time functions	Miscellaneous
atan2	asort	break	close	mktime	and
cos	gensub	continue	fflush	strftime	compl
exp	gsub	do/while	getline	systime	delete
int	index	exit	print		function
log	length	for	printf		lshift
rand	match	if	sprintf		or
sin	split	next			rshift
sqrt	strtonum	nextfile			system
srand	sub	return			xor
	substr	while			

Arithmetic functions	String functions	Control flow statements	Input/output processing	Time functions	Miscellaneous
	tolower				
	toupper				

and(x,y)

asort(array[,outarray])

atan2(y,x)

break

close(filename-expr)

close(command-expr)

compl(value)

continue

cos(x)

delete array[element]

delete array

do

body

while (expr)

exit

exp(arg)

fflush(filename)

for (i=lower ; i<=upper ; i++)

command

for (item in array)

command

function name(parameter-list) {

statements

}

gensub(r,s,n,t)

getline [varhairsp;] [<file]

command | getline [var]

gsub(r,s,t)

if (condition)

command1

[else

command2]

if ($1 ~ /[Aa]*/)

index(<span
class="docEmphasis">substr,<span
class="docEmphasis">str<span
class="docEmphBold">)

int(<span
class="docEmphasis">arg<span
class="docEmphBold">)

length(<span
class="docEmphasis">arg<span
class="docEmphBold">)

log(<span
class="docEmphasis">arg<span
class="docEmphBold">)

lshift(value, n)

match(s,r)

mktime(datestring)

next

nextfile

or(x,y)

print [args] [destination]

printf[format [, expressions]]

{printf ("The sum on line %s is %d.\n", NR, $1+$2)}

5 5

The sum on line 1 is 10.

rand( )

return [expr]

rshift(value, n)

sin(x)

split(string,<span

class="docEmphasis">array[,<span
class="docEmphasis">sep]<span
class="docEmphBold">)

sprintf [<span
class="docEmphasis">format[, <span
class="docEmphasis">expression(s)]]

sqrt(<span
class="docEmphasis">arg<span
class="docEmphBold">)

srand(<span
class="docEmphasis">expr<span
class="docEmphBold">)

strftime(
[format[,<span
class="docEmphasis">timestamp]]<span
class="docEmphBold">)

strtonum(
str<span
class="docEmphBold">)

sub(<span
class="docEmphasis">r,<span

class="docEmphasis">s,<span
class="docEmphasis">t<span
class="docEmphBold">)

substr(<span
class="docEmphasis">string,<span
class="docEmphasis">m[,<span
class="docEmphasis">n]<span
class="docEmphBold">)

system(<span
class="docEmphasis">command<span
class="docEmphBold">)

systime( )

BEGIN {

  now = systime( )

  mesg =
strftime("Started at %m/%d/%Y %H:%M:%S", now)

print mesg

}

process data ...

END {

now = systime( )

mesg =
strftime("Ended at %m/%d/%Y %H:%M:%S", now)

print mesg

}

tolower(str)

toupper(str)

while (condition)

command

xor(x,y)

Return the result of a bitwise XOR of x and y.

Chapter 14. RCS

The Revision Control System (RCS) offers *version*

control (or *revision control*), a system that maintains information about a project's evolution so that project members can retrieve prior versions, track changes, andmost importantly coordinate the efforts of the team. In a typical interaction, a developer checks out the most current code from the repository, makes changes, tests the results, and then commits those changes back to the repository when they are deemed satisfactory.

RCS can keep track of multiple file revisions, thereby reducing the amount of storage space needed. With RCS you can automatically store and retrieve revisions, merge or compare revisions, keep a complete history (or log) of changes, and identify revisions using symbolic keywords.

RCS preceded CVS and performs a similar role. RCS is more limited because it was designed to be used within a single directory and maintains a separate repository in each directory. Although RCS is no longer actively developed, it is still useful for small projects, offering fewer administrative issues than CVS. Projects with large directory structures or with team members on many different computer systems would benefit from using CVS, described in Chapter 15.

RCS uses a *locking*

model to coordinate the efforts of multiple developers by serializing file modifications. Before making changes to a file, a developer must not only obtain a copy of it, but also request and obtain a lock on it from the system. This lock serves to prevent (or really, dissuade) multiple developers from working on the same file at the same time. When the changes are committed, the developer unlocks the file, permitting other developers to gain access to it.

14.1 Overview of RCS Commands

The three

most important RCS commands are:

ci

Check in revisions (put a file under RCS control).

co

Check out revisions.

rcs

Set up or change attributes of RCS files.

Two additional commands provide information about RCS files:

ident

Extract keyword values from an RCS file.

rlog

Display a summary (log) about the revisions in an RCS file.

You can compare RCS files with these commands:

rcsdiff

Report differences between revisions.

rcsmerge

Incorporate changes from two RCS files into a third RCS file.

The following command helps with configuration management:

rcsclean

Remove working files that have not been changed.

14.2 Basic RCS Operations

Normally, you maintain RCS files in a subdirectory of your working directory called *RCS*, so the first step in using RCS should be:

```
mkdir RCS
```

Next, you place an existing file (or files) under RCS control by running the checkin command:

```
ci  file
```

This creates a file called *file,v* in directory *RCS*. *file,v* is called an RCS file, and it will store all future revisions of *file*. When you run **ci** on a file for the first time, you are prompted to describe the contents. **ci** then deposits *file* into the RCS directory as revision 1.1.

To edit a new revision, check out a copy:

```
co -l  file
```

This causes RCS to extract a copy of *file* from the RCS directory. You must lock the file with **-l** to make it writable by you. This copy is called a working file. When you're done editing, you can record the changes by checking the working file back in again:

```
ci  file
```

This time, you are prompted to enter a log of the changes made, and the file is deposited as revision 1.2. Note that a checkin normally removes the working file. To retrieve a read-only copy, do a checkout without a lock:

```
co  file
```

This is useful when you need to keep a copy on hand for compiling or searching. As a shortcut to the previous **ci**/**co**, you could type:

```
ci -u  file
```

This checks in the file but immediately checks out a read-only copy. To compare changes between a working file and its latest revision, you can type:

```
rcsdiff  file
```

Another useful command is **rlog**, which shows a summary of log messages.

System administrators can use the **rcs** command to set up default behavior for RCS.

$ keyword $

$ keyword : value
$

Author: daniel $

$Date: 2003/02/25 18:21:10 $

$Header: /projects/new/chapter3,v 1.2 2003/02/25 18:21:10 daniel \

Exp Locker: daniel $

$Id: chapter3,v 1.2 2003/02/25 18:21:10 daniel Exp daniel $

$Locker: daniel $

$Log: chapter3,v $

#Revision 1.2 2003/02/25 18:21:10 daniel

#Added section on error-handling

#

#Revision 1.1 2003/02/25 16:49:59 daniel

#Initial revision

#

$RCSfile: chapter3,v $

$Revision: 1.2 $

$Source: /projects/new/chapter3,v $

$State: Exp $

co -l -r1.4 ch01

ci -r2 ch01

ci -r1.4.1 ch01

ci -u -nPrototype *.c

rcsdiff -r$ ch01

rcs -nDraft:$ ch*

2002/10/16 02:00:00 (year/month/day time)

6:00 pm lt (assuming today is Oct. 16, 2002)

2:00 AM, Oct. 16, 2002

Wed Oct 16 18:00:00 2002 LT

Wed Oct 16 18:00:00 PST 2002

ci -x,v/ ch01   (second suffix is blank)

RCS/ch01,v

ch01,v

RCS/ch01

-z[zone]

Specify the format of the date in keyword substitution. If empty, the default is to output the UTC time with no zone indication. With an argument of LT, the local time zone will be used to output an ISO 8601 format, with an indication of the separation from UTC. You may also specify a numeric UTC offset. For example, -z+4:30 would output a string such as "2002-11-24 02:30:00+4:30".

ci [options] files

ci -m'First round edits' chap*

ci -u prog.c

ci -r2 -nPrototype prog.c

co [options] files

co -p file | sort

co -l [A-Z]*

co -r3 *,v Latest revisions of release 3

co -r3 -wjim *,v Same, but only if checked in by jim

co -d'May 5, 2 pm LT' *,v Latest revisions that were modified

on or before the date

co -rPrototype *,v Latest revisions named Prototype

ident [<tt>option</tt>] [<tt>files</tt>]

char rcsID[ ] = "$Author: daniel $";

% ident prog.c prog.o

prog.c:

$Author: daniel $

prog.o:

$Author: daniel $

co -p RCS/*,v | ident -q

rcs [options] files

rcs -nTo_customer: RCS/*

rcs -ageorge,paul,ringo beatle_deals

rcs -o1.2-1.5 doc

echo "$description" | rcs -t file

rcsclean [options] [files]

rcsclean *.c *.h

rcsdiff [options] [diff_options] files

rcsmerge [options] file

co -l prog.c

(edit latest revision by adding revision 1.3 updates, then:)

rcsmerge -p -r1.3 -r1.6 prog.c > prog.updated.c

rcsmerge -r3.5 -r3.2 chap08

rlog [options] files

rlog RCS/*,v | more

rlog -R -L -ldaniel RCS/*

rlog -t calc.c

Chapter 15. CVS

The Concurrent Versions System (CVS) is the most popular revision control system among users of free and open source software. It is particularly appropriate for highly distributed projects, with developers working on many different computer systems and even in different parts of the world.

This chapter is based on CVS Version 1.11.2.

15.1 Basic Concepts

To accommodate large projects using a hierarchy of several directories, CVS defines the concepts

repository and *sandbox*.

The *repository* (also called an *archive*) is the centralized storage area that stores the projects' files. It is managed by the version control system and the repository administrator, and contains information required to reconstruct historical versions of the files in a project. An administrator sets up and controls the repository using the procedures and commands described later in Section 15.5.

A *sandbox* (also called a *working directory*) contains copies of versions of files from the repository. New development occurs in sandboxes, and any number of sandboxes may be created from a single repository. The sandboxes are independent of one another and may contain files from different stages of the development of the same project. Users set up and control sandboxes using the procedures and commands found in the later Section 15.6.

In a typical interaction with CVS, a developer checks out the most current code from the repository, makes changes, tests the results, and then commits those changes back to the repository when they are deemed satisfactory.

15.1.1 Locking and Merging

Some systems, such as RCS (Revision Control System) and the older SCCS (Source Code Control System), use a *locking model* that coordinates the efforts of multiple developers by serializing file modifications. This was described at the beginning of the previous chapter. The locking model is pessimistic: it assumes that conflicts *must* be avoided. Serialization of file modifications through locks prevents conflicts. But it is cumbersome to have to lock files for editing when bug-hunting. Often, developers will circumvent the lock mechanism to keep working, which is an invitation to trouble.

To handle work by multiple developers on a single file, CVS uses a *merging model* that allows everyone to have access to the files at all times and supports concurrent development. The merging model is optimistic: it assumes that conflicts are not common and that when they do occur, it usually isn't

difficult to resolve them.

CVS is capable of operating under a locking model via the **-L** and **-l** options to the **admin** command. Also, CVS has special commands (**edit** and **watch**) for those who want additional development coordination support. CVS uses locks internally to prevent corruption when multiple people are accessing the repository simultaneously, but this is different from the user-visible locks of the locking model.

15.1.2 Conflicts and Merging

In the event that two developers commit changes to the same version of a file, CVS automatically defers the commit of the second committer's file.

The second committer then issues the **cvs update** command, which merges the first

committer's changes into the local file. In many cases, the changes are in different areas of the file, and the merge is successful. However, if both developers have made changes to the same area of the file, the second to commit will have to resolve the conflict. This involves examining the problematic areas of the file and selecting among the multiple versions or making changes that resolve the conflict.

CVS detects only textual conflicts, but conflict resolution is concerned with keeping the project as a whole logically consistent.

Therefore, conflict resolution sometimes involves changing files other than the one CVS complained about.

For example, if one developer adds a parameter to a function definition, it may be necessary for all the calls to that function to be modified to pass the additional parameter. This is a logical conflict, so its detection and resolution is the job of the developers (with support from tools like compilers and debuggers); CVS won't notice the problem.

In any merge situation, whether or not there was a conflict, the second developer to commit will often want to retest the resulting version of the project because it has changed since the original

commit. Once it passes the test, the developer will need to recommit the file.

15.1.3 Tagging

CVS tracks file versions by revision number, which can be used to retrieve a particular revision from the repository. In addition, it is possible to create symbolic tags so that a group of files (or an entire project) can be referred to by a single identifier even when the revision numbers of the files are not the same (which is most often the case). This capability is often used to keep track of released versions or other important project milestones.

For example, the symbolic tag **hello-1_0** might refer to revision number 1.3

of *hello.c* and revision number 1.1 of

Makefile (symbolic tags are created with the **tag** and **rtag** commands).

15.1.4 Branching

The simplest form of development is *linear*

development, in which there is a succession of revisions to a file, each derived from the prior revision. Many projects can get by with a completely linear development process, but larger projects (as measured by number of files, number of developers, and/or size of the user community) often run into maintenance issues that require additional capabilities. Sometimes it is desirable to do some speculative development while the main line of development continues uninterrupted. Other times, bugs in the currently released version must be fixed while work on the next version is already underway. In both of these cases, the solution is to create a *branch* (or *fork*) from an appropriate point in the development of the project. If, at a future point, some or all of the changes on the branch are needed on the main line of development (or elsewhere), they can be *merged* together

(*joined*).

Branches are forked with the **tag -b** command; they are joined with the **update -j** command.

cvs <tt>global_options</tt> <tt>
command</tt> <tt>command_options
</tt>

$ cvs -d /usr/local/cvsrep init $ cd
~/work/hello

$ cvs -d /usr/local/cvsrep import -m 'Import' hello
vendor start $ cd ..

$ mv hello hello.bak $ cvs -d
/usr/local/cvsrep checkout hello $ cd
hello

$ vi hello.c

$ cvs commit -m 'Fixed a typo' $ cvs tag
hello-1_0 $ cvs remove -f Makefile $
cvs commit -m 'Removed old Makefile' $
cvs upd -r hello-1_0 $ cvs upd -A

Some global options are common to both user and administrator commands, and some are specific to each. The common global options are described in the next section, and the administrator and user

options are described in Section 15.5 and Section 15.6, respectively.

15.3 Common Global Options

Table 15-1 lists the

global options that apply to both user and administrator commands.

Table 15-1. Common global options

Option	Description

Option	Description
-b *bindir*	Location of external RCS programs. This option is obsolete, having been deprecated at CVS versions above 1.9.18.
-T *tempdir*	Absolute path for temporary files. Overrides the setting of **$TMPDIR**.
-v, **--version**	Display version and copyright information.

15.4 Gotchas

This section clarifies a few aspects of CVS that can sometimes cause confusion.

File orientation

> While directories are supported, they are not versioned in the same way as traditional files are. This is particularly important in the early evolutionary stages of a project, when the structure may be in flux. Also, if the project is undergoing major changes, the structure is likely to change. See Section 15.5.4.

Text orientation

> There is no equivalent to **diff** for binary files, although CVS's support for binary files is usually sufficient. Use **admin -kb** to tell CVS a file is binary.

Line orientation

> Moving a segment of code from one place in a file to another is seen as a delete (from the old location) and an unrelated add (to the new location).

CVS is not syntax-aware

> As far as CVS is concerned, small formatting changes are equivalent to sweeping logic changes in the same line ranges.

$ export CVSROOT=/usr/local/cvsrep

$ cvs init

$ cvs -d /usr/local/cvsrep init

cvspserver 2401/tcp

cvspserver stream tcp nowait root /usr/bin/cvs cvs --allow-root=/usr/

local/cvsroot pserver

cvspserver stream tcp nowait root /usr/sbin/tcpd /usr/bin/cvs

--allow-root=/usr/local/cvsroot pserver

```
service cvspserver

{

  port = 2401

  socket_type = stream

  protocol = tcp

  wait = no

  user = root

  passenv = PATH

  server = /usr/local/bin/cvs
```

server_args = -f --allow-root=/usr/local/cvsroot pserver

}

cvs commit: Rebuilding administrative file database

ALL $CVSROOT/CVSROOT/commit-ALL.pl

DEFAULT $CVSROOT/CVSROOT/commit-DEFAULT.pl

CVSROOT$ $CVSROOT/CVSROOT/commit-CVSROOT.pl

*.jpg -k b

/usr/local/cvsrep/hello Makefile,1.1,1.2 hello.c,1.8,1.9

DEFAULT mail -s %s developers@company.com

<tt>module_name</tt> -a <tt>alias_module </tt>...

<tt>module_name</tt> [<tt>options</tt>] <tt>directory</tt> <tt>file</tt>

...

<tt>module_name</tt> [<tt>options</tt>] &<tt>other_module</tt> ...

h -a hello

project -a library client server

header library library.h

project &library &client &server

ALL mail %s -s "CVS notification"

anonymous:TY7QWpLw8bvus:cvsnoname

john:john@somecompany.com

jane:jane@anothercompany.com

$ tar xvzf foo-1.0.tar.gz

$ tar xvzf foo-1.1.tar.gz

$ tar xvzf foo-2.0.tar.gz

$ mkdir foo

$ cp -R -p foo-1.0/* foo

$ cd foo

$ cvs import -m 'Imported version 1.0' foo vendor start

$ cd ..

$ mv foo foo.bak

$ cvs checkout foo

$ cd foo

$ cvs tag foo-1_0

$ cd ..

$ diff -Naur foo-1.0 foo-1.1 | (cd foo; patch -Np1)

$ cd foo

$ cvs commit -m 'Imported version 1.1'

$ cvs tag foo-1_1

$ cd ..

$ diff -Naur foo-1.1 foo-2.0 | (cd foo; patch -Np1)

$ cd foo

$ cvs commit -m 'Imported version 2.0'

$ cvs tag foo-2_0

$ mv foo foo-rcs

$ mkdir foo

$ cd foo

$ cvs import -m 'New empty project' foo vendor start

$ cd ..

$ mv foo foo.bak

$ cvs checkout foo

$ cd foo

$ mkdir dir

$ cvs add dir

$ cd ..

$ cp -p foo-rcs/*,v $CVSROOT/foo

$ cp -p foo-rcs/dir/*,v $CVSROOT/foo/dir

$ cd foo

$ cvs upd

$ cd /usr/local/bar

$ cvs import bar vendor start

$ cd ..

$ mv bar bar.bak

$ cvs checkout bar

admin

[-b[<tt>rev</tt>]]

[-c<tt>string</tt>]

[-k<tt>kflag</tt>]

[-l[<tt>rev</tt>]]

[-L]

[-m<tt>rev</tt>:<tt>msg</tt>]

[-n<tt>name</tt>[:[<tt>rev</tt>]]]

[-N<tt>name</tt>[:[<tt>rev</tt>]]]

[-o<tt>range</tt>]

[-q]

[-s<tt>state</tt>[:<tt>rev</tt>]

[-t[<tt>file</tt>]]

[-t-<tt>string</tt>]

[-u[<tt>rev</tt>]]

[-U]

[<tt>files </tt>...]

init

kserver

pserver

server

The CVS client runs this command on the remote machine when connecting to a repository specified by an :ext: repository locator (usually via RSH or SSH).

$ alias cvs="export CVS_RSH=ssh; cvs" $ alias cvr="export CVS_RSH=rsh; cvs"

$ export CVSROOT=/usr/local/cvsroot $ cd ~/work

$ cvs checkout hello

#

foo.pl

#

$Log: ch15.xml,v $

#

#

Revision 1.2 2000/06/09 22:10:23 me

Fixed the new bug introduced when the last one was fixed.

#

Revision 1.1 2000/06/09 18:07:51 me

Fixed the last remaining bug in the system.

#

add

[-k <tt>kflag</tt>]

[-m <tt>message</tt>]

<tt>file </tt>...

annotate

[[-D <tt>date</tt> | -r <tt>rev</tt>] -f]

[-F]

[-l | -R]

<tt>file </tt>...

checkout

[-A]

[-c | -s]

[-d <tt>dir</tt> [-N]]

[[-D <tt>date</tt> | -r <tt>rev</tt>] -f]

[-j <tt>rev1</tt> [-j <tt>rev2</tt>]]

[-k <tt>kflag</tt>]

[-l | -R]

[-n]

[-p]

[-P]

<tt>module </tt>...

commit

[-f | [-l | -R]]

[-F <tt>file</tt> | -m <tt>message</tt>]

[-n]

[-r <tt>revision</tt>]

[<tt>file</tt> ...]

diff

[-k <tt>kflag</tt>]

[-l | -R] [format]

[[-r <tt>rev1</tt> | -D <tt>date1</tt>] [-r <tt>rev2</tt> | -D <tt>date2</tt>]]

[<tt>file </tt>...]

edit

[-a <tt>action</tt>]

[-l | -R]

[<tt>file </tt>...]

editors

[-l | -R]

[<tt>file </tt>...]

export

[-d <tt>dir</tt> [-N]]

[-D <tt>date</tt> | -r <tt>rev</tt>]

[-f] [-k <tt>kflag</tt>]

[-l | -R]

[-n]

[-P]

<tt>module </tt>...

$ cvs export -r foo-1_0 -d foo-1.0 foo $ tar czf foo-1.0.tar.gz foo-1.0

help

history

[-a | -u <tt>user</tt>]

[-b <tt>string</tt>]

[-c]

[-D <tt>date</tt>]

[-e | -x <tt>type</tt>]

[-f <tt>file</tt> | -m <tt>module</tt> | -n <tt>module</tt> | -p <tt> repository</tt>]...

[-l]

[-o]

[-r <tt>rev</tt>]

[-t <tt>tag</tt>]

[-T]

[-w]

[-x <tt>types</tt>]

[-z <tt>zone</tt>]

[<tt>file </tt>...]

cvs history -p CVSROOT -p hello

import

[-b <tt>branch</tt>]

[-d]

[-I <tt>pattern</tt>]

[-k <tt>kflag</tt>]

[-m <tt>message</tt>]

[-W <tt>spec</tt>]

<tt>module</tt> <tt>vendor_tag</tt> <tt>release_tag </tt>...

log

[-b]

[-d <tt>dates</tt>]

[-h]

[-N]

[-r<tt>revisions</tt>]

[-R]

[-s <tt>states</tt>]

[-t]

[-w<tt>logins</tt>]

[<tt>file </tt>...]

login

logout

rannotate

[[-D <tt>date</tt> | -r <tt>rev</tt>] -f]

[-F]

[-l | -R]

<tt>module </tt>...

rdiff

[-c | -s | -u]

[{ { -D <tt>date1</tt> | -r <tt>rev1</tt> } [-D <tt>date2</tt> | -r <tt>rev2</tt>] } | -t]

[-f]

[-l | -R]

[-V <tt>vn</tt>]

<tt>file </tt>...

release

[-d]

<tt>directory </tt>...

remove

[-f]

[-l | -R]

[<tt>file </tt>...]

rlog

[-b]

[-d <tt>dates</tt>]

[-h] [-N] [-r<tt>revisions</tt>]

[-R]

[-s <tt>state</tt>]

[-t]

[-w<tt>logins</tt>]

[<tt>module </tt>...]

rtag

[-a]

[-b]

[-B]

[-d]

[-D <tt>date</tt> | -r <tt>rev</tt>]

[-f]

[-F]

[-l | - R]

[-n]

<tt>tag</tt> <tt>file </tt>...

status

[-l | -R]

[-v]

[<tt>file </tt>...]

tag

[-b]

[-c]

[-d]

[-D <tt>date</tt> | -r <tt>rev</tt>]

[-f]

[-F]

[-l | R]

<tt>tag</tt> [<tt>file </tt>...]

unedit

[-l | -R]

[<tt>file </tt>...]

update

[-A]

[-C]

[-d]

[-D <tt>date</tt> | -r <tt>rev</tt>]

[-f]

[-I <tt>pattern</tt>]

[-j <tt>rev1</tt> [-j <tt>rev2</tt>]]

[-k <tt>kflag</tt>]

[-l | -R]

[-p]

[-P]

[-W <tt>spec</tt>]

[<tt>file </tt>...]

cvs -n -q update -AdP

$ cvs update -j 1.2.2.1 -j 1.2.2.2 foo.c

version

watch

{ { on | off } | { add | remove } [-a <tt>action</tt>] }

[-l | -R]

<tt>file </tt>...

watchers

[-l | -R]

[<tt>file </tt>...]

Display a list of users watching the specified files. This is determined by checking which users have run the watch command on a particular file (or set of files). If the watch command has not been used, no results will be displayed.

The standard meanings of the common client options -l and -R apply.

See also watch.

Chapter 16. Graphical Desktop Overview

Linux was criticized in the past for lacking an easy-to-use graphical desktop, but that has changed significantly. These days, there are several such desktops, as well as advanced alternatives aimed at experts, developers, and high-performance enthusiasts. This book covers three graphical configurations: two complete environments with their own application suites (GNOME in Chapter 17

and KDE in Chapter 18), and one traditional window manager (**fvwm2** in Chapter 19).

GNOME and KDE are the most commonly used desktops on Linux systems. KDE has been around longer, and as the default desktop for SuSE, has more of a European following. The GNOME

project was started more recently, and its desktop, used as the default for Red Hat Linux, is more common in the United States. Many operating system vendors have produced unified themes for the two desktops, so that applications written with one toolkit appear consistent with those from the other.

16.1 Desktop Environments and Window Managers

The simplest graphical desktops consist of the X Window System, which displays the windows and graphics, and a *window manager*, which determines where windows are placed and how users interact with applications. A window manager, such as **fvwm2**, determines window "focus" (that is, which window is currently accepting input) and some keyboard shortcuts. Window managers often include some sort of control panel or task bar, but not always.

A desktop environment, in contrast, attempts to provide a complete experience, offering many of the tools someone would need for a typical day at the office. A desktop environment also includes more extensive cooperation between applications, ensuring that cutting and pasting text and dragging and dropping objects around the desktop work as expected, even between applications. A minimal installation would include a window manager with control panel, a file manager, and a few sample applications such as a text editor. Additional items such as games, email, calendars, office tools, and software development tools are usually available and work closely with the desktop core.

Both GNOME and KDE should look familiar to Windows and Macintosh users: they feature a desktop background with icons for files and folders, a bar with buttons and a clock at the top or bottom of the screen, and a central menu to access everything from applications to system settings. Both have more settings available than either Windows or the Macintosh OS, including support for virtual desktops, customizable key bindings, and window focus behavior. They also include or share a series of applications: office suites for word processing, spreadsheets, and presentations; groupware tools for email, calendar, and address-book management; and image processing, web, and software

development tools for artists, programmers, and system administrators.

While GNOME and KDE offer entire suites of applications and configuration tools and serve as both software and software development platform, **fvwm2** focuses strictly on handling windows and the desktop background. **fvwm2** does not include other applications, and is customized with configuration files the way that all Linux and Unix applications used to be. It is, in fact, almost endlessly customizableas long as you are willing to edit the right files. Some developers of software for GNOME or KDE will admit to using **fvwm2** or another window manager on their own systems, because they have customized it to work exactly as they wish.

16.2 Desktop Differences: Development

The differences between KDE and GNOME matter more to developers than to the average user. If you choose to develop software based on one of these platforms, your choice of toolkit and desktop environment will probably be based on two factors: language and license. The GNOME

libraries (the main widget set is called Gtk+) are written in C, and the KDE libraries (notably Qt) in C++. For both desktops, bindings for other languages, such as Perl, Python, and Java, are available and popular, so the choice comes down to the architecture you prefer.

Licensing is slightly more complicated.

Qt, the base toolkit for KDE software, is controlled by TrollTech, Inc., and is available under a dual licensing scheme: free for the development of open software, and proprietary for the development of proprietary software. Most GNOME

libraries are available for open or proprietary development at no charge. Visit

http://developer.gnome.org and http://developer.kde.org for more details on building software with these tools.

Chapter 17. GNOME

One of the two popular desktop environments used with Linux, GNOME is provided as the default desktop for Red Hat, Debian, and several other popular distributions.

As a graphical environment, GNOME provides users with a highly customizable user interface and consistent functionality of common GUI features such as menus, toolbars, and buttons. In addition, it offers users a growing set of native applications to create a productive computing system. The number and quality of applications are a testament to the developer-friendly GNOME libraries; many GNOME

technologies are also used in nongraphical or totally unrelated software. GNOME is distributed with most Linux distributions, and you can also get it from the GNOME web site (http://www.gnome.org) or from Ximian (http://www.ximian.com), a company that specializes in the GNOME desktop.

GNOME stands for "GNU Network Object Model Environment," and although the name is

admittedly obscure, it does point to one of GNOME's

core technologies: its CORBA-based objects. CORBA (Common Object Request Broker Architecture) specifies methods that allow interaction among applications through the sharing and embedding of component objects.

For example, a spreadsheet created by Gnumeric (a GNOME spreadsheet program) can be placed as an object into an AbiWord document, and the Nautilus file browser can display images, web pages, and so forth by embedding an image viewer and HTML display engine. GNOME uses two libraries to do this: ORBit, (http://orbit-resource.sourceforge.net), which provides an Object Request Broker (ORB), and

Bonobo

(http://developer.ximian.com/articles/whitepapers/bonobo), which is designed to simplify the task of creating reusable software components and compound documents.

This edition of *Linux in a Nutshell* covers a GNOME desktop based on the GNOME 2 platform, which is included in Red Hat 8.0, Mandrake 9.0, and SuSE 8.1, among other distributions. The new platform differs from the previous version, GNOME 1.4, in a number of significant ways. Overall, the new platform has brought increased performance and stability, more coherent and powerful developer tools, and a friendlier, simpler interface.

You do not need to be familiar with earlier versions of GNOME to use GNOME 2. If you are familiar with earlier versions and want to know what's new, or if you are curious about the history

of the project, see Section 17.5 at the end of this chapter for additional background information.

17.1 Desktop Overview

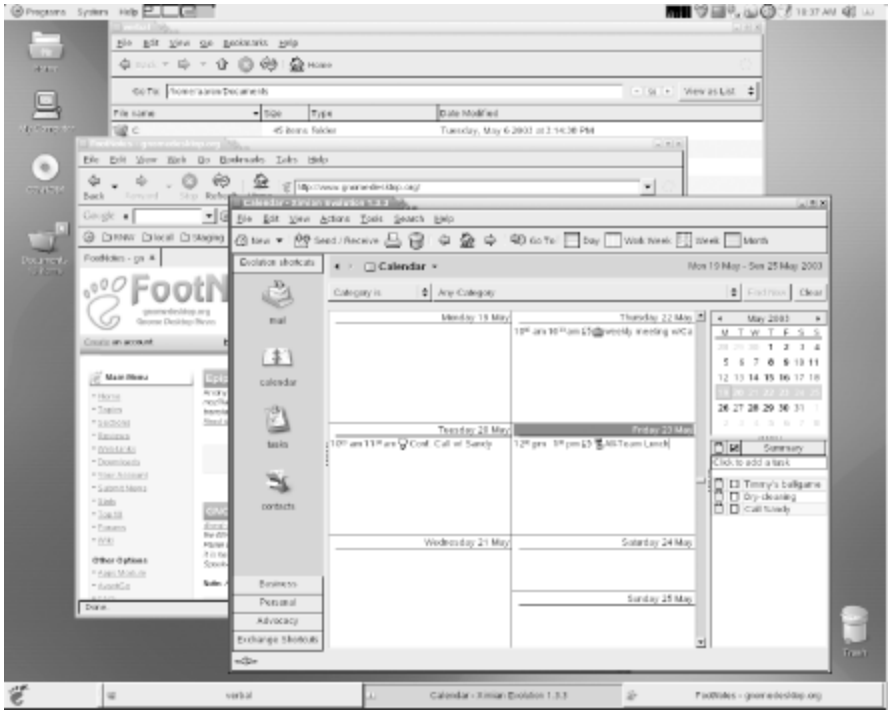
Figure 17-1 shows the default GNOME desktop. The left side of the screen contains icons that are shortcuts to open applications, files, or URLs. The top icon is a link to the user's home folder; when double-clicked, it launches

the Nautilus file manager to display the folder's

contents. The other icons include shortcuts to the floppy drive and CD-ROM and links to web pages. In general, double-clicking on an icon leads to the most logical operation, so clicking folder icons displays folder contents, clicking application icons launches applications, and clicking a file opens the file in its most appropriate application (see Section 17.4.4.2 for information on how to choose which one). A button or icon you can press to start an application is called a

launcher. You can drag and drop the icons around the desktop to arrange them as you like, or drag them onto the Trash icon to get them out of your way.

Figure 17-1. The GNOME desktop



You can also right-click on any blank space in the desktop and get a context menu that allows you to:

- Open a new file manager window.
- Create a new folder.
- Create an application launcher.

- Open a new terminal window.
- Run scripts or executable files you place in your scripts folder (*~/.gnome2/nautilus-scripts*).
- Clean up the icons on the desktop.
- Copy, paste, or cut selected files.
- Change your desktop background.
- Display the contents of any mounted disk in the file manager.

The bars across the top and bottom of the screen are called *panels*. Some distributions and configurations use only one, and others use more than two. GNOME allows you to create a variety of panels and choose their placement, size, and behavior. The one at the top, called the menu panel, is perhaps the most common and important.

It may be slightly different in some distributions of GNOME, but it generally consists of two menus on the left, and a clock and application switcher on the right. Some systems may also include other small applications, called

applets; see Section 17.2.3 for more information.

If you do not have a menu panel, the desktop probably offers the GNOME menu, which is similar to the Start menu in Microsoft Windows. On most systems, it appears in the lower left corner and is designated with a GNOME foot logo, earning it the nickname "the foot menu." In Red

Hat 8.0, click the red hat to get the same menu with a list of applications, help, and a few other tools, plus the logout and screen-locking tools. If you want to add this GNOME menu to a panel, right-click on any blank space in the panel and select Add to Panel →GNOME Menu.

In general, the panel is a primary means of finding and opening applications and managing your desktop. You can add buttons to the panel to launch any application on your system; you can also include small applications, called applets, in the panel. For more information, see Section 17.2.

GNOME allows you an enormous number of configuration options for your desktop environment. You can right-click on just about anything and get a pop-up menu (called a *context menu*) containing specific actions for that item and a way to configure its properties. General configuration settings are contained in the GNOME Control Center. You can access this tool by selecting System and then Settings in your menu panel (on some systems, look for Desktop Preferences under the Programs or Applications menu). For more information about settings, see Section 17.4.

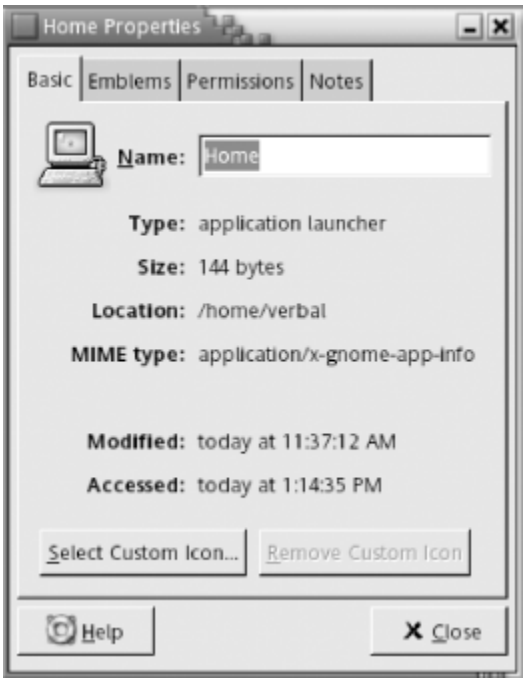
17.1.1 Adding Desktop Icons

Desktop icons offer convenient double-click access to your most important files, applications, and links. The items displayed on your desktop exist as files in the *.gnome-desktop* directory of your home directory. Anything you add to that directory will appear on the desktop.

To add an icon that launches an application, select New Launcher.

This opens the Desktop Entry Properties dialog box as shown in Figure 17-2. Provide the name of the launcher (this will be the text displayed underneath the icon), a comment (the tooltip that appears when the pointer is over the icon), and the command used to run the application. After you click OK, the new launcher icon appears on your desktop.

Figure 17-2. Desktop entry properties



A convenient use of desktop icons is to provide shortcuts to frequently used files or folders. Adding shortcuts is easiest from the file manager (Nautilus). Display the directory containing the item for which you want a shortcut on your desktop, right-click on the item, and select Make Link. Then click on the new link and drag it to the desktop, which has the underlying effect of moving it to your *~/.gnome-desktop* folder. You can copy an

item instead of moving it by pressing the Ctrl key while selecting and dragging the item.

17.2 The Panel

The GNOME panel can contain several different types of objects. The most obvious are the buttons for the menu and application launchers. You also can use a button to open a

drawer, which is a subpanel containing additional launchers. There are a few special types of buttons used for logging out of the session and locking the screen. Finally, small programs called applets can be run on the panel. The Workspace Switcher, the Window List, and the clock are examples of panel applets.

As with many other things in GNOME, you can find settings for the panel by looking in the Panel menu under the GNOME menu, or by right-clicking on the panel itself. In both cases, there are options to add new launchers or applets to the panel; to adjust the style, size, and display of the panel; or to create new panels on the desktop. The menu panel does not have a properties dialog, as it is preconfigured. If you wish, you can remove it or replace it with an edge panel that is more to your liking.

17.2.1 Additional Panels

You can create more than one panel on your desktop. This is useful if you have different sets of applications used for specific but common tasks. For example, if you do a lot of work on graphics, you can dedicate a panel to launch your favorite graphics tools. To create a new panel, right-click on the default panel and select Add New Panel, or, from the GNOME menu, select Panel ⟶ Add New Panel. There are five different types of panels available from the submenu:

Edge panel

> The style of the default panel. It stretches across one entire edge of the screen. Arrow buttons on each end of the panel are used to collapse the panel to the side of the screen. Clicking on the remaining visible arrow button of a collapsed panel will cause the panel to appear again in full.

Corner panel

> A panel that is anchored to one corner of the screen and extends just enough to fit the buttons and applets it contains. A corner panel can be hidden by clicking the arrow button that is at the edge of the screen. The arrow button farthest from the edge will anchor an aligned panel on the opposite side of the screen.

Sliding panel

> Like a corner panel, except that instead of being anchored to a corner, it can be placed anywhere along the edge of the screen. This is similar to the Dock in Macintosh OS X.

Floating panel

> A panel that can be placed anywhere on the screen.

Menu panel

> A special type of preconfigured edge panel. It is a thin bar that stretches across the top of the screen and contains drop-down menus and a clock. Users of the Macintosh will find it quite familiar.

All of the panels except the menu panel can be moved by middle-clicking (or clicking the left and right buttons simultaneously) and dragging the panel to another part of the screen. They can also be configured individually from the Panel menu in its context (right-click) menu.

Right-click and select Panel → Properties to open a properties dialog. Your options vary depending on panel type, but the complete list is:

Position

> This option does not appear for floating panels. For other types, choose a position by clicking the location on a diagram of a screen.
>
> You can also move all four configurable types of panels without this dialog by middle-clicking and dragging to a new location.

Orientation

> This option is available only for floating panels. It lets you choose whether the panel should be vertical or horizontal. Other panels are aligned with the screen edges.

Autohide

> Check this box to have the panel hide itself automatically. A small amount of the panel will be left showing; bring the mouse pointer over that bit, and the panel will reappear. This functions similarly to the Microsoft Windows task bar.

Show Hide Buttons, Arrows on Hide Buttons

> At each end of the panel is a hide button with an arrow. Click the button, and the panel shrinks into it. Click again, and the panel reappears. Uncheck these boxes to get rid of the buttons or to remove the arrow labels from them. The buttons are slightly smaller if they are unlabeled.

A second tab of the preferences window allows you to choose a background for your panel. Normally, a panel is light gray, but you can choose a color or even an image to go behind your buttons. If you choose an image, you can also scale or stretch it to fit the panel more accurately.

17.2.2 Adding an Application Launcher to the Panel

One of the conveniences of the panel is creating launcher icons that allow you one-click access to frequently used applications. To add an application, right-click the panel and select Add to Panel ⟶ New Launcher.

You can also right-click launchers in your menus and choose Add This Launcher to

Panel, or simply drag launchers to the panel.

To customize the launcher, right-click on it and select Properties.

Here, you can enter a name for the application, a comment to be used as a tooltip, and the command used to open the application. (This is similar to creating a launcher, as shown in Section 17.1.1.) Click on the icon button to select the image to be used for the button on the panel. If the application is to be run in a terminal windowthat is, if it is a text-based program normally invoked from the shellcheck the box labeled Run in Terminal.

Launcher buttons can be placed in any position and order you want on the panel. To move a launcher button, right-click it and select Move, or middle-click and drag it. The mouse pointer will change shape, indicating that you can drag the button to another position. Click to set the new position of the button.

The middle mouse button has two features that are often overlooked. First, it can be used to paste whatever text is currently selected, without a copy command. Second, it can be used to move items such as the Workspace Switcher Applet that cannot be moved with the left mouse button. (In the Workspace Switcher Applet, left clicking and

dragging moves individual windows among the workspaces, rather than moving the applet itself.)

17.2.3 GNOME Panel Applets

As mentioned earlier, GNOME offers small applications that run inside the panel, called panel applets. Some will already be in your panel by default. To add more, right-click on a blank space in the panel and select one of the items under Add to Panel ⟶

Applet. The two most important GNOME panel applets are the Workspace Switcher Applet and the Window List Applet.

17.2.3.1 Workspaces and the Workspace Switcher

GNOME allows you to create extra screen real estate by multiplying your desktop with workspaces. You can keep a few windows open in each workspace instead of choosing between clutter and closing windows. For example, you might keep your mail program open in one workspace with a few messages open in separate windows, then have one or two browser windows open in a second workspace, and a spreadsheet in a third.

The workspaces are drawn in miniature on your panel in the Workspace Switcher Applet, which at first looks like a series of small grey boxes. Drawn inside each workspace, you can see the individual windows you have open, and even drag them from one workspace to another with the left mouse button. You can navigate these workspaces with the keyboard shortcuts Ctrl+Alt+Left, Ctrl+Alt+Right, Ctrl+Alt+Up, and Ctrl+Alt+Down, or by clicking one of the workspaces in the Workspace Switcher Applet.

Most systems give you four workspaces by default, although some stick with one or two. To create more, open the Workspace Switcher Applet preferences tool by right-clicking on the Workspace Switcher Applet and selecting Preferences. There, you can select not only the number of workspaces, but also their names and their layout in columns and rows.

17.2.3.2 The Window List Applet

The Window List Applet applet lets you keep track of open windows. It displays a button on the panel for each window you have open. Clicking on a button in the tasklist will bring that window to the foreground, reopening it if it was minimized.

You can right-click on any of the buttons for an application to get a context menu that allows you to interact with the window for that application, and close, minimize, maximize, un-maximize, move, or resize it.

To set the way the Window List Applet behaves, right-click on an empty space in the window list or on the small

"handle" on its left side, and

select Preferences. This will open a dialog box that allows you to set the following options:

Window List content

You can show the windows from all your workspaces, or just from the workspace you are currently in.

When restoring minimized windows

This setting matters only if you choose to display windows from all your workspaces. The default setting, "Restore to current workspace," means that minimized windows you restore with the Window List Applet will appear on your current workspace. If you select "Restore to native

workspace," the windows will appear in the workspace where you were last using them. In other words, you can choose to have your windows appear either where you are now or where you were last using them.

Window grouping

One of the more useful Window List Applet features is window grouping. If you have several windows open for a particular application, the Window List Applet can save space by combining all the entries for that application. Then, instead of a button for each window, you'll have a menu for the whole

application. For example, if you have one Galeon web browser window open, you will have a button that shows the Galeon icon and the title of the page for that window. If you have five open, the Window List Applet will show "Galeon (5)," and

clicking the button will offer you a choice of which window to focus.

You can set the applet never to group windows, to group windows only when it runs out of space, or to group windows whenever it can.

Window List size

Choose a minimum and maximum size for your window list applet. The applet will grow between these sizes depending upon how many windows you have open.

17.2.3.3 Miscellaneous applets

In addition to the Window List and Workspace Switcher Applets, there are roughly a dozen other applets in five categories. You can find the complete list of applets by right-clicking on the panel and selecting Add to Panel ⟶ Applet.

Accessories

A clock, a dictionary, a stock ticker, and a weather report.

Amusements

A fish that tells your fortune, and some googly eyes that follow your mouse pointer.

Internet

A modem traffic monitor and a mail checker.

Multimedia

A CD player and volume control.

Utility

The largest category: a battery charge monitor, international character and keyboard layout tools, a mini command line, a system load monitor, a system event notification area, and the Workspace Switcher and Window List Applets.

17.3 The GNOME Menu and the Menu Panel Menus

Because different distributions organize their menus differently, you're best off exploring them yourself or referring to the documentation for your particular system. For the most part, however, the menus have similar contents: applications, help files, settings or preferences tools, utilities and system administration tools, and a few special items like Log Out and Run Command. The major difference is whether these items are split into several menus, as in the menu panel, or put in a single place, as in the GNOME menu.

There are, however, a number of common tasks related to menu items that do not vary across distributions, and we'll describe those here. First, there are context menus for individual menu items. To see them, right-click on a menu item (this is easier if you do not keep the left mouse button pressed while navigating your menus). The possible actions are:

Add this launcher to panel

> Add the item as a launcher in the panel. If you have more than one panel, the launcher will be added to the first panel. You can then drag the launcher to wherever you actually want it.

Put into run dialog

> Display this item as a shortcut in the Run Program dialog.

Entire menu

The two items here, "Add this as menu to panel" and "Add this as drawer to

panel," allow you to add the entire submenu containing the launcher to your panel as either a drawer (a portion of panel that stays open when a menu might close) or as a menu.

The menu customization tools are one of the simplest and most obvious improvements in the GNOME 2 platform. To alter your menus, just open your Nautilus file manager and type

"applications:///" into the

location bar. The Applications display represents your menus and submenus as folders, with item launchers in them. Drag items around and move, copy, delete, or rename them as you see fit.

To add a submenu, right-click in an empty spot in the folder and select New Folder, then rename the folder, or right-click on it to set its name and icon.

To add a launcher, right-click in an empty spot in the folder and select New Launcher, then enter the name of the application, type in any pop-up text you want to describe it, and enter or browse for the command that starts the application itself. If you're not sure where that is, look in the */usr/bin* directory, or use the **which** or **locate** commands (see Chapter 2) to find it.

17.4 The GNOME Control Center

The GNOME Control Center (Figure 17-3) is where most customization and configuration of your desktop environment takes place. There are several ways to open it: you can click the toolbox button on your panel, select items in the Preferences section of the GNOME menu, or choose Settings in the System menu. The Control Center is also one place where different distributions vary significantly. The tools covered in this section appear in most distributions, but you may find a slightly different set that is organized in a different fashion.

Figure 17-3. GNOME Control Center

17.4.1 Personal Settings

These settings control the system information pertaining most directly to you.

17.4.1.1 Accessibility

The Accessibility tool, new in GNOME 2, handles the various settings related to making a computer easier to use for people with disabilities.

There are three sections to the tool: Basic, Filters, and Mouse, plus the option to turn all the accessibility options on or off at once.

To set any option, you must first click the "Enable keyboard accessibility" checkbox.

In the Basic section, you can set the following options:

Features

> You can set the features to turn off automatically if they are unused, or to emit a keyboard beep when turned on or off. This helps prevent confusion if they are turned on accidentally.
>
> The Import Feature Settings button is relevant only to those migrating from CDE to GNOME 2, and allows you to import CDE
>
> accessibility settings.

Sticky Keys

> Sticky Keys allows you to press key combinations such as Ctrl+R by pressing only one key at a time. If you choose to enable the sticky keys feature, you can also set it to beep when used, and to turn off if a normal key combination is pressed.

Repeat Keys

Key repeat can be troublesome for people with limited mobility. To turn it off, uncheck this box. Alternately, you can use the sliders to adjust the delay before key repeating begins, and the speed at which keys are repeated.

The Filters section handles the way that the system can ignore accidental keypresses. Individuals with limited dexterity may wish to use these features to prevent extraneous key inputs.

Slow Keys

Check the Enable Slow Keys button to have the system accept only keys that are held down, and ignore keys that are tapped quickly. You can set an amount of time before a keypress is registered, and also set beeps for keypress, key acceptance, or key rejection.

Bounce Keys

The Bounce Keys feature ignores duplicate keypresses: if you hit a key and then accidentally hit it again, the system will ignore the second input.

You can set the amount of time the system will wait before accepting another key, and have it beep when it rejects a duplicate.

Toggle Keys

Some keypresses toggle individual features on and off. Enabling the Toggle Keys function will have the system beep once when a toggle is turned on, and twice when it is turned off.

The mouse accessibility settings allow you to control the mouse pointer with the keyboard arrows instead of with the mouse itself.

Click the Enable Mouse Keys checkbox to turn this feature on.

There are three settings for the Mouse Keys feature. You can set sliders for the maximum speed, the acceleration rate, and the delay between pressing the arrow key and the time that the mouse pointer actually begins to move.

17.4.1.2 Menus

The Menus button in the Control Center takes you to the special Nautilus view called Applications, which is a file display of your menu tree.

There, you can edit the menus as though they were files. You can drag and drop items to move, add, or delete them, and change their names or descriptions by right-clicking on them and selecting Properties.

17.4.1.3 Password

Open this tool to change your password. You will be asked for your current password and your new password. You will have to type the new password twice to confirm it.

The tool may reject passwords if they are not long enough or are too easy to guess.

17.4.2 Look and Feel Settings

These sections provide settings for the overall look of your desktop by letting you choose the background, screensaver, theme, and keyboard shortcuts.

17.4.2.1 Background

Selects images, colors, and dispay effects for the part of the display that lies behind windows and menus. To use an image, select one from the list of default backgrounds, or add another by clicking the Add Picture button and selecting a file from your disk. Then choose how you want the image fitted to your screen.

For very small images or tile patterns, choose Tiled; for larger images or those exactly the size of the screen, choose Centered.

Images that are a different shape from the screen may be scaled to fit, or scaled symmetrically until the entire image fits on the screen. You can also choose No Image and just use a color. If the image doesn't exactly fit the screen,

you'll see the background color behind it.

In the color section, select either a solid, single color, or a horizontal or vertical gradient of two colors. If you select a gradient, the color will fade from the first color to the second one, down or across the screen. Click the color box to select the colors.

17.4.2.2 Fonts

Here, choose the fonts you want GNOME to use in different situations, and how it should draw those fonts on your screen.

The top half of the font dialog displays the fonts your system uses; click on the current font to change it.

Application Font

The font used in dialog boxes, text areas, and menus.

Desktop Font

The font used to display desktop icon labels.

Window Title Font

The font used in window titles.

Terminal Font

The font used in the terminal emulator. This should be a monospaced font such as Courier.

The bottom half of the font dialog is taken up with font rendering options. Font rendering determines the method used to draw the actual fonts on your screen. You have four options:

Monochrome

Render the fonts only in black and white. This can lead to letters being somewhat jagged, but is faster.

Best Shapes

Also known as "antialiasing," this is considered the highest quality and is recommended for most standard monitors.

Best Contrast

Make sure that the letters are displayed in the sharpest possible contrast. Recommended for the visually impaired.

Subpixel Smoothing (LCDs)

This optimization method is best for flat-panel (LCD) monitors.

17.4.2.3 Screensaver

Contains settings for the screensaver. You can choose from a list of available screensavers (including a random setting). Input the number of minutes of inactivity before the screensaver starts and

whether you will be required to give your password before going back to the desktop. Power management settings are available here if your system is configured for them.

17.4.2.4 Keyboard Shortcuts

The Keyboard Shortcuts tool lets you choose the key combinations that will execute a wide range of actions, from opening the Run Command dialog or taking a screenshot to moving windows between workspaces.

To choose a new keyboard shortcut, click the action you want to change and then type a new accelerator. Setting a shortcut that conflicts with an existing GNOME shortcut will disable the old shortcut. Setting a shortcut that conflicts with one for any other application will usually override the application's shortcut.

17.4.2.5 Theme Selector

Themes provide a consistent overall style to the many widgets and components used by GNOME. A number of basic themes are included with the *gtk-engines* package, and you can download and install additional themes from http://gtk.themes.org.

Any themes installed on your system are listed in the Theme Selector.

You can select one, and the theme will be applied to the desktop immediately.

If you have downloaded a theme and wish to install it, click the Install New Theme button. Provide the name and location of the *.tar.gz* or *.tgz* file and

click OK. The new theme will be installed in the

/usr/share/themes directory and be available for you to use on your desktop.

17.4.3 Hardware Settings

In some systems this section is called Peripherals. It includes tools that allow you to control the way your system works with hardware and peripheral devices such as your mouse and keyboard. If you have installed PalmOS device software, you may also have Palm Pilot configuration tools. Some systems also include a printer configuration tool here.

17.4.3.1 Keyboard

Contains settings for keyboard autorepeat and sounds. You can set the repeat rate of a pressed key and the delay before it starts. You

also can enable keyboard clicks and their volume. Settings can be previewed by typing in the Test Settings textbox.

Three sliders adjust the volume, pitch, and duration of the keyboard bell. Click the Test button to hear the bell's new settings.

17.4.3.2 Mouse

Lets you configure the mouse for either a right-handed user or a left-handed user. If you configure for a left-handed user, all documentation about the left button applies to the right button, and vice versa.

You also can set the acceleration and threshold (sensitivity) of the mouse.

17.4.3.3 Sound

This tool sets up a sound scheme for various actions. You can enable or disable all system sounds by checking the box "Start the sound server when I log in." To add sounds to particular events, check the box "Play sounds for specific

events."

The rest of the window contains a listing of events, sorted by application. Each event with an associated sound has the name of

the file played when it occurs. Select an event from the listing and click Play to hear its assigned sound.

To change the sound for an event, click on the event and then enter the name of the file or click Browse to choose a file to play.

17.4.4 System Settings

The System Settings category contains settings that relate to the internal operation of your desktop and operating system, such as the time or the way that files are associated with applications.

17.4.4.1 Date and Time

You will need your system's root password to change the date and time, since time affects all users on the system. The appearance of the time tool itself may vary depending on the operating system you use, but it will present you with options for choosing the time, date, and time zone for your system.

In addition, most time and date tools allow you to set one or more network time servers with which to synchronize, so your clock will never be inaccurate.

17.4.4.2 File Associations (MIME Types)

This tool also varies between operating systems. On most it is called File Associations, but on some it is called MIME Types. It allows you to set the applications that open particular files or handle particular network protocols. There are a bewildering number of file types listed here, sorted by category. For example, HTML documents are in the World Wide Web section of the Documents category, and have filenames ending in *.htm* and

.html.

To add a new type of document, click the Add button. A dialog will open asking for the category/type listing for the file type and the extensions to associate with it. Then, choose an application (if any) that handles that document. Optionally, you can supply up to two regular expressions to identify the type. For example, if you installed the RealPlayer music software and wanted to add RealAudio music files to your system, you would list the

.ram, *.rm*, and

.ra file extensions (but not

.rpm, which is used for the RPM software package format), and place the MIME type in the Audio category. Then, you would enter the name of your RealAudio application.

To edit an existing file association, select its listing and click the Edit button. You will see a dialog box in which you can choose an icon to be used for the file type, add or remove file extensions, and supply commands that will open, view, and edit this type of file. To delete a type, select it from the list and click the Delete button.

You can also choose protocol handlers, the applications that work with various internet services. To change your web browser to Galeon, you would open the World Wide Web (*http://*) item in the Internet Services section and set it to run */usr/bin/galeon*. Some distributions replicate this ability in a tool called URL Handlers, described next.

17.4.4.3 URL Handlers

Not all distributions have this tool, as it replicates some of the functionality from the File Associations tool. The URL Handlers tool allows you to adjust the settings for special URL launchers used by the GNOME help system. The defaults for protocols such as HTTP, FTP, and Mail are already set and likely handled by your default web browser (for example, Galeon). The special URLs are *ghelp*,

info, and *man* (i.e., GNOME

help files, command *info* files, and

man files). The defaults use the help browser for these types of files.

17.4.4.4 Manage Software

Not all distributions have this tool, which opens the Ximian Red Carpet software management application. You can use Red Carpet to install, remove, or update the software on your computer. There isn't room to

describe this tool completely, but you can find complete instructions in its help menu.

17.4.4.5 Network Proxies

A *proxy* is a system that stands between your computer and the Internet, handling all the connections for you. If your system uses a proxy, you can name it here to allow all GNOME

applications to take advantage of it.

You may set a proxy for HTTP, Secure HTTP (HTTPS), FTP, and Socks transactions. To do so, click the "Manual proxy

configuration" radio button and enter the name of the proxy and, if necessary, the port number. Leave an entry blank if you do not use a proxy for that protocol.

Some networks have automatic proxy configuration, which allows you to enter a single URL and let the computer set the proxy names and port numbers. If this is the case, click the "Automatic proxy configuration" radio button and enter the URL

for your proxy configuration host.

17.4.4.6 Sessions

This tool controls session managementthat is, what happens when you log in and log out.

The options are divided into three categories: general session options, settings for the current session, and startup programs. If you create multiple session types, you can choose among them when you log in. Most distributions have at least a few different sessions created for you in advance; you can select among them at the login screen.

The Session Options tab contains:

Show splash screen on login

> Normally, GNOME displays a "splash screen" image to show you what's
>
> going on as it starts up. If you prefer not to have this appear, uncheck this box.

Prompt on logout

Normally, when you select the Log Out item from your System menu or GNOME menu, you will be asked if you really want to log out, and whether you would like to shut down, restart, or just log out. If you want the system to log out without confirmation, uncheck this box.

Automatically save changes to session

When you log out and leave applications running, GNOME will normally remember that the application was running, and start that application again the next time you log in. If you prefer not to have that happen, uncheck this box.

Sessions:

You can create alternate sessions and select them from the GNOME

login screen before you log in. You could, for example, create a "work" session that would start

your calendar and spreadsheet as soon as you logged in, and turn down the volume on your speakers. Another session might turn the volume up and start your web browser and a game. To make changes to a session beyond renaming it, log into that session and change it there.

The Current Session tab shows the programs that are running right now. Each application is assigned a number to indicate the order in which it is started (lower numbers are first). You can also assign a style to each application, which determines how it starts, restarts, or quits.

Normal

The application starts automatically when you log in and runs until you quit.

Restart

The application starts when you log in, and restarts itself if you stop it or if it crashes. The Metacity window manager and the panel have this style because they need to be running at all times.

Trash

The application doesn't start when you log in.

Settings

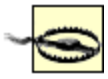
The application starts before other applications and is used to store configuration settings. Very few applications get this style.

The Startup Programs tab allows you to start an application every time you log in, whether or not it's running when you log out. To add a program to your list of startup applications, click the Add button and enter its name or select it by clicking Browse and selecting it in the filesystem.

17.4.5 Advanced Settings Control with GConf

GConf is the name of the GNOME configuration database. It acts as a central repository for settings and other shared system data, much like the Registry in Microsoft Windows. It uses XML files stored in

the *~/.gconf* directory to associate keys and values that multiple applications can access. It's possible to change those values with a text editor, but the gconf-editor application provides a much easier way to adjust variables and settings. It displays keys and their values in a convenient heirarchical fashion, together with any explanation of the key and its available values.



The gconf-editor program is not generally supported and is not the recommended tool for general-purpose configuration. It does not have an Undo feature; if you break something and can't figure out how to fix it, you can revert to the system default GConf database. To do so, log out, log into a command-line or non-GNOME

environment (Ctrl+Alt+F1 will get you a virtual console in most systems), and delete the *.gconf* directory in your home directory. The default settings for GNOME applications will be restored when you next start your applications. Do not delete your *.gconf* directory while running GNOME.

To begin working with GConf keys, run the "gconf-editor" command at the

command line or in the "Run

Program..." dialog. The editor has a filesystem tree on the left, and an area on the right that is used to display keys and values. There are five groups of keys at the top level, but only two are really relevant to most users: *apps*, where you'll set preferences for applications, and *desktop*, which stores most of the keys set by the GNOME Control Center. There are too many keys and values to list here, but the examples in the rest of the section should give you a feel for what you can do with them.

Note that most of

these configuration variables are loaded only when a program starts, so changes will not take effect until you restart the relevant application.

17.4.5.1 Window managers and window behavior

The window manager is the individual piece of software that handles the particulars of placement, movement, and border style of the windows on your desktop. The default GNOME 2 window manager is called Metacity.

Except for themes, almost none of the Metacity options are settable in the standard GNOME Control Center. To set them, select "apps" and then

"metacity" in the gconf-editor.

Metacity has a "general" category,

where most of the options are set, plus a number of key bindings for window and workspace control. The most commonly changed items are:

auto_raise

If the box is checked, windows that get mouse focus automatically come to the front of the screen. If you set the focus mode to **sloppy** and leave this box unchecked, you can type into windows that are in the background. In the default focus mode, this setting has no effect.

focus_mode

GNOME 1.2 and 1.4 users will remember this feature, which lets you choose how windows get the keyboard focus. The default behavior is **click**: you must click on a window to bring it into focus. Other possible values are **enter-exit** (window focus comes to a window when the mouse does, and leaves when the mouse does) and **sloppy** (window focus comes to a window when the mouse enters it, and stays until the mouse enters another window).

animation_style

Choose the style of animation for moving, resizing, and minimizing windows. Your options are **wireframe**, **thin**, **opaque**, and **none**.

wrap_style

This item is a good example of the sort of feature that has been removed, hidden, or abstracted away in GNOME 2. The settings chosen here determine what happens when you use the key combination (normally Ctrl-Alt-Arrow, and set in the **global_keybindings** section of the Metacity keys) to try to move beyond the defined borders of your virtual desktops. If the value is **none**, you will not move any further. If the value is **classic**, movement leads you to the next workspace in number: going right from the rightmost workspace in a row will take you to the first workspace in the next row, and moving down from the bottom workspace in a column will take you to the top workspace in the next column. If you select **toroidal**, the workspaces behave as though they were on a globe: moving from the end of one row will take you back to the beginning of the same row.

17.4.5.2 The panel

As mentioned earlier, many panel options have been left out of the Panel Preferences tool that you get when you right-click on the panel. Some of the items you can adjust with gconf-editor are:

panel_show_delay and **panel_hide_delay**

If you have a panel set to automatically hide itself when you leave it alone, these values are the time in milliseconds that you leave it alone before it hides itself, and the time it takes to appear when you move the mouse to its side of the screen.

tooltips-enabled

Some people don't like tooltips in their panels.

Setting this key to false (unchecking the box) will turn them off.

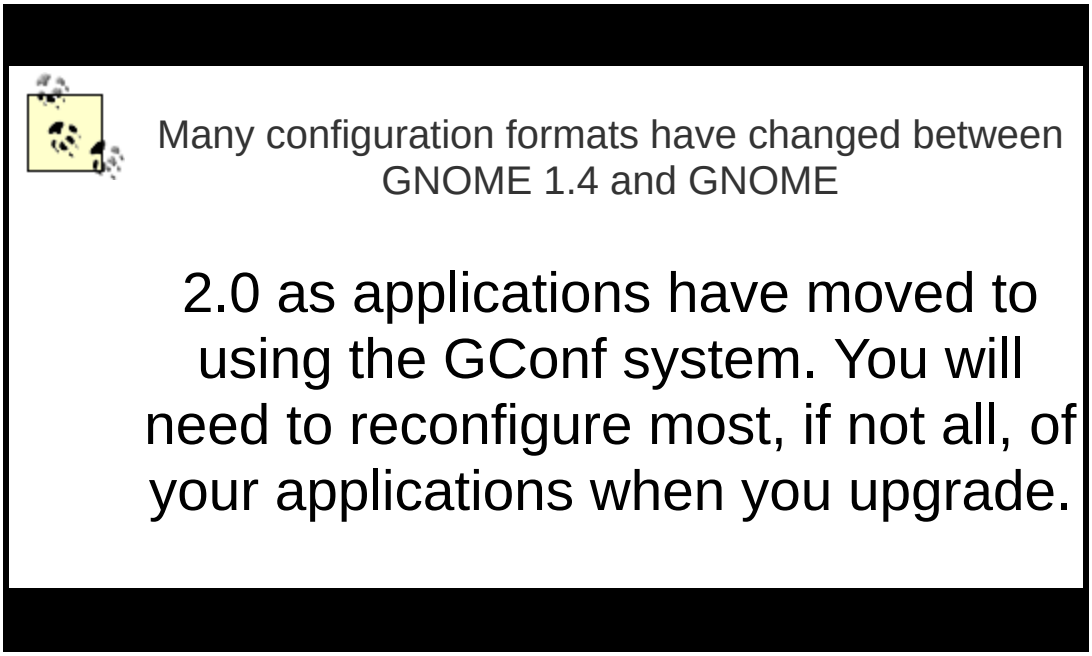
enable-animations and **panel_animation_speed**

When you hide or show a panel, it slides on and off the screen smoothly. These values allow you to turn off animation entirely or have it move faster or slower.

17.5 History and Changes in GNOME 2

Most new software releases add features, and a major complaint among users is of steadily increasing complexity, or "feature creep." The

GNOME 2 desktop is distinctive in that, while adding a few features, it has also removed some features.



New features include:

- Font smoothing (antialiased fonts).
- Marked speed increases in the Nautilus file manager.

- Tabbed browsing

  (press Ctrl-T to create a new tab) in both Galeon and Mozilla web browsers.

- Internationalization support, including proper handling of right-to-left languages.

- Accessibility tools for people with visual or mobility impairments.

- Control Center more smoothly integrated with the rest of the system.

Removed features

include:

- Edge flipping

  (dragging items from one desktop to another) is no longer available; use keyboard shortcuts or the Workspace Switcher Applet instead.

- Detachable menus have been disabled; you can turn them on with the gconf-editor tool.

- It is no longer possible to choose window placement algorithms. Some versions of GNOME ship without window focus options and leave them only as gconf keys.

- GNOME 1.x made a distinction between two types of virtual desktops. *Workspaces* assumed the creation of multiple distinct desktops, whereas *viewports* assumed that you were creating a single, large desktop and displaying only a portion of it. Few people ever fully understood the distinction, and it has been removed from Metacity and the Workspace Switcher. As a result, windows moved to the edge of the screen no longer appear in the next workspace over, and it is not possible to create a three-dimensional array of virtual desktop areas.

- Application

  display fonts, themes for window borders and application buttons, and the virtual filesystem's HTTP proxy have been given simplified configuration tools that appear in only one place. Several other Control Center tools have been eliminated, and the term "capplet" has been removed from the

  GNOME vocabulary.

- Both menu

  configuration tools (gmenu and the menu display options tool) have been replaced by

the ability to edit menus in Nautilus (type the location "applications:///" into

Nautilus).

- Many of the less

  important panel options are available only in the gconf-editor, including panel movement speed, the delay before hiding the panel when using the automatic hiding option, menu caching, and the option to display an image on the left side of the GNOME and Programs menus.

- There are far fewer panel applets. Notably, there is only one clock applet rather than five (nobody admits to thinking five clocks were ever necessary) and there are only two applets in the

  "Amusements" category: eyes that

  track the mouse, and the fish that tells your fortune.

- The Nautilus file manager also has fewer options. Notably, speed has been improved to the point that the Performance Tradeoffs are no longer necessary, the User Levels feature has

been removed, the theme, font, and HTTP proxy settings have been moved to the Control Center, and the Help display has been moved to a dedicated help browser called **yelp**.

Most of these feature removals stem from switching from Sawfish to the much simpler Metacity window manager. However, all of the features that ended up being removed had initially sprung from one of the major pitfalls of any software project, especially open projects where making an executive decision is harder. When developers disagree, it is often easier politically to create an option than to find an acceptable compromise or enforce a "correct" decision. However, each

preference, option, and feature adds another layer of complexity, making the software harder to support, debug, and integrate with the rest of the desktop.

The clearest example in GNOME is window management software. The window manager is important and complex software: it controls window placement and ordering, controls event handling and focus, and so forth. Without a window manager, your desktop would not be functional as you know it. At the same time, window managers have to be completely unobtrusive, or they interfere with the use of applications. Most computer users do not know or care what window managers are, and have certainly never started a window manager by hand themselves.

When GNOME

1.0 was released, it allowed users to select among various window managers, including **twm** and **icewm**. The majority of users and

distributions used a window manager called Enlightenment, which quickly grew from being a window manager to being a desktop environment in its own right. Running GNOME and Enlightenment at once became increasingly redundant, so for GNOME 1.2 and 1.4, users and distributors switched to Sawfish, which was small, fast, and highly configurable.

In fact, there were so many combinations of settings that it was hard to test, and it kept getting larger and more complex. By the time GNOME 2.0 was on the way, Sawfish was slow, large, and impossible to maintain. The Sawfish maintainer had abandoned it, and nobody was willing to continue the project.

The GNOME development team was determined not to repeat the process. Metacity, the GNOME 2 window manager, is small, fast, and simple. The GNOME Control Center is designed around the assumption that most users do not care to pick or know their window manager, and that developers do not care to bear the burden of code complexity caused by interacting with multiple window management systems.

It's still possible to switch window managers, and those who wish to do so can do it by editing their *~/.gnome-session* files.

The GNOME

2 series of releases (2.0 and 2.2 have been released as of this writing; 2.4 and 2.6 are

planned) aims to choose good defaults rather than offer limitless options, and considers simplicity and usability to be as important as adding additional behavior. The desktop is kept as simple as possible so that it does not distract from the actual functionality of the applications that are based upon it and that are its real reason for existence. The applications themselves are not shying away from new features or new capabilities, but for the most part share the goals of simplicity and ease of use.

Chapter 18. KDE

The K

Desktop Environment (KDE) is an open source software project that provides a consistent, user-friendly, contemporary desktop for Unix and Linux systems. KDE is not simply a window manager like **fvwm**, but a whole desktop environment including services to assist end users, application developers, and system administrators in standardizing the look and feel of their applications and configuring their systems. The KDE interface makes full use of drag-and-drop functionality so that, for instance, you can grab an icon for a text file in the file manager and drag it to a text editor window to open it. Full network integration of KDE

applications allows you to transparently access files from other computers or FTP sites and manipulate them as if they were local.

KDE also implements a standard help system based on HTML.

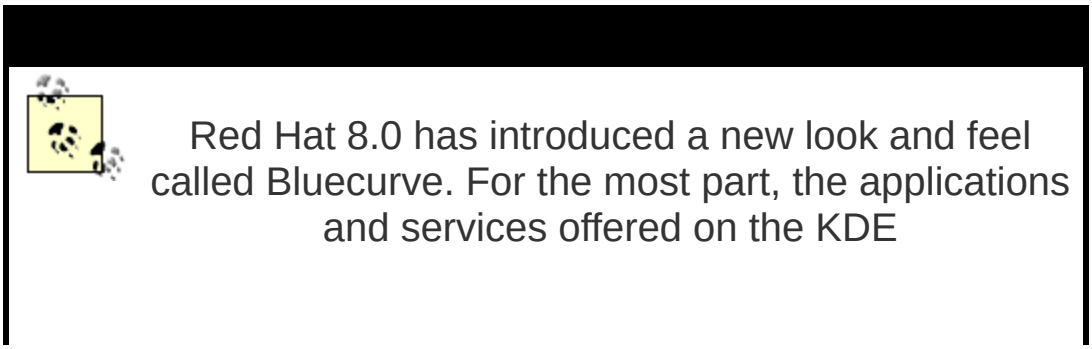
Applications that display a Help button offer a specific file of documentation for that application in the help viewer.

One goal of KDE is to provide the user with system information and configuration through easy-to-use graphical interfaces on the desktop. The KDE Control Center is a central utility for desktop and application configuration, as well as a source of information for important system components. The Information module of the Control Center can retrieve and display status information for your processor, memory, PCI bus, network devices, and many other hardware components on your system.

A wide variety of applications has been developed to take advantage of KDE's features and provide the user with a wealth of productive applications. The base package comes with programs such as a mail client, a calendar and organizer, a CD player, image viewers, chat programs, and more.

Most Linux distributions ship with KDE and allow you to set it up as the default session environment when you install the operating system. If you are installing KDE separately, download and install the KDE packages (you can find them at ftp://ftp.kde.org, among other places). You can also use this link if you want to upgrade your version of KDE

with the current official version.



desktop under Bluecurve are the same as those described here, but they may appear in different places or have different names and icons. For instance, the startup menu at the bottom left corner of the screen, which displays a large K in vanilla KDE, displays a red hat in Bluecurve.

To set KDE as your desktop environment, look for the X initialization files in your home directory. Depending on your distribution and how you log in, look for

.xinitrc, *.xsession*, or

.Xclients in your home directory. If none of these files exist, create a new *.xinitrc* file.

Edit the file to remove any window manager references that may exist and add **startkde** on a line at the end of the file. If you are logging in through the KDE or GNOME

display manager, there may be a Session tab or icon that allows you to select KDE as your desktop environment. In that case, you do not need to update or add an initialization file.

KDE Config File

[Desktop Entry]

Comment[en-US]=The Kate text editor Encoding=UTF-8

Exec=kate Icon=

MimeType=

Name[en-US]=Kate Path=

ServiceTypes=

SwallowExec=

SwallowTitle=

Terminal=false TerminalOptions=

Type=Application X-KDE-SubstituteUID=false X-KDE-Username=

...

As you can see, the syntax is simple and straightforward. The items filled in on the properties windows are listed on each line of the file. The Type line identifies the kind of link file. In this example, Kate is an application; Type=URL would indicate an Internet address link file. The Name line lists the name of the application, Comment lists the comment you added to describe the application, Exec lists the command, Icon identifies the icon image file for an icon other than the default, and so on. Unspecified options have empty values. The Name and Comment lines include the language in square brackets.

18.2 The Panel

The panel is the control bar across the bottom of the screen, used to find and launch applications and navigate among windows and desktops. It contains the K menu (identified by the KDE K logo), which organizes the installed KDE applications into submenus; the Desktop Access button, which minimizes all open windows to display a clean desktop; and the desktop pager and the taskbar. Additional buttons that open applications, directories, and URLs can be added to the panel.

18.2.1 The Desktop Pager

Like most window managers, **kwin** (KDE's window manager) can divide your workspace into multiple desktops. Different application windows can be open on each desktop, reducing the amount of clutter on your screen. You can switch among desktops by using the desktop buttons on the panel. The panel displays a grid of buttons called the *pager*. The pager has one button for each virtual desktop, and shows images of the open windows or their names (1, 2, 3, etc., by default). Clicking on a button switches your screen to the corresponding desktop.

Clicking the wider vertical bar to the left of the pager opens a Desktop Pager window. If your version of KDE does not have that bar, right-click on the pager itself and select Launch Pager. The Desktop Pager has larger buttons for each desktop and an actual icon for each application running on that desktop, making it easier to tell what applications are running.

You can configure the number of virtual desktops and their names in the Control Center by selecting Look & Feel ⟶

Desktop and clicking the Number of Desktops tab.

18.2.2 The Taskbar

The taskbar runs in the panel and helps you keep track of running applications. It contains buttons to identify each open application window. If the button for an application is clicked, that becomes the current active window. When you iconify a window, you can raise it again by clicking its button on the taskbar. If a window has been iconified, its taskbar button contains a parenthesized text label. If multiple instances of one type of window are running, they are represented by a single button in the taskbar, with a small up-arrow on the right. Click the button to display a list of the instances and select the one you want.

Clicking the wider vertical bar to the left of the taskbar displays a menu divided into sections for each desktop and items for each window they contain. (This window list is also accessible by middle-clicking on the desktop background.) For example, if desktop 2 contains an open Konqueror window, you can click on its entry in the window list, and the window manager switches you to desktop 2 and activates the Konqueror window.

You can add an external taskbar to the desktop, either in place of or in addition to the one running in the panel. The external taskbar runs along one edge of the desktop, usually the top or bottom. You can configure the location and other settings of the external taskbar in the Control Center by selecting Panel ⟶

Extensions.

18.2.3 Adding an Application Link to the Panel

The simplest way to add an application button to the panel is by dragging an icon from the desktop to the panel. This copies the link from the desktop. Any application listed in the K menu can be easily added to the panel. From the K menu, choose Panel ⟶ Add Application, then select from the submenus or items that are listed. The choices you have are the same items that appear on the K menu.

To remove a button from the panel, right-click on an empty place on the panel to bring up the panel menu, and select Remove. That displays another menu that lets you remove applets such as the pager or taskbar, buttons that start applications, or special buttons such as the K menu or the Desktop Access button. You can also right-click on an icon and select Remove.

18.2.4 Running an Application on the Panel

A swallowed application, or applet, is a program that runs on the panel instead of in a desktop window. A swallowed application can be a small utility that monitors network activity or provides mail notification, for example. The lock/login applet, the **klipper** clipboard applet, and the clock are examples of swallowed applications, as are the pager and the taskbar when they're running in the panel.

Add other applets by selecting Add → Applet from the panel menu and selecting the applet you want.

18.3 The KDE Control Center

The KDE Control Center contains a number of configuration tools, called *modules*, that allow you to configure and view information about your system.

You can configure the desktop, window behavior and decoration, input devices, and any other important part of your system here.

The Control Center is split into two windows: the left window shows a hierarchical list of installed modules, and the right window displays the modules when they are selected. (You can also run individual Control Center modules by selecting the one you want from the Preferences option on the K menu.) Start the Control Center from the panel button with an icon of a circuit board in front of a terminal, or from the K menu. Many of the modules can also be accessed directly from their applications, by right-clicking on the application icon and selecting Preferences, or by right-clicking on the titlebar of the running application window and selecting Configure.

Context-sensitive help is available for Control Center modules by clicking on the question mark on the titlebar and then on the setting for which you want help. You can get a brief general help message for the currently open module by clicking the Help tab in the left window; each help message also contains a link to the module's manual. Or you can

get help by opening the Help Center, selecting Applications →Preferences, and then choosing a module.

The middle tab in the left Control Center window lets you search. You can enter a term in the text box at the top to narrow the list of keywords, or scroll through the keywords. When you find the one you want, click on it to highlight it. The Results box at the bottom will display a list of modules for which the term applies; select a module from the list and that module will appear in the right-hand window.

For example, selecting the keyword **audio** displays the results Sound, Sound Server, System Bell, and System Notifications. If you select Sound Server from that list, you can view or change the Sound Server settings.

The following sections describe the modules. The number and type of available Control Center modules vary depending on your Linux distribution and which version of KDE you are running. You may find that some of the modules described here are not on your system, that you have others that we don't describe, or that some

of the specifics are different. In general, however, these

descriptions should give you a good idea of how to go about configuring your KDE desktop.

18.3.1 File Browsing

The File Browsing modules are used to set file associations and to configure Konqueror's

file management features.

18.3.1.1 File Associations

This module associates a filename extension with an application so that clicking on a file with that extension automatically opens the application with the file loaded. The utility divides files into various types based on the MIME standard used for email and the Web. Thus, under the text types you can find the **html** type. The file extensions *.htm*, *.HTM*, *.html*, and *.HTML* are associated with the **html** file type.

The window initially has only a text box where you can enter a filename pattern to search for the matching application and a list of known applications. You can also add and remove types from the list.

When you select an application from the Known Types list, the right side of the window displays options for that file type. The General tab shows the patterns associated with the application; you can add others or select one of the patterns to remove it from the list.

Clicking on the icon image opens the Select Icon window for choosing a different icon to use with that file type. Application Preference Order lists the applications associated with that type. You can add applications to the list with the Add

button; select an entry to activate the other buttons. You can then move the entry up or down on the list, edit it, or remove it from the list.

On the Embedding tab, the set of radio buttons determines whether a file is viewed in Konqueror or in an application window.

18.3.1.2 File Manager

This module contains configuration settings for the Konqueror file manager. The Behavior tab sets some global options, such as whether opening a directory also opens a new Konqueror window or if you want a dialog box to show the progress of a network file download. On the Appearance tab, you can set your choice of font, font size, and text color. Other options on this tab include whether to wrap or truncate icon text, underline filenames on the desktop, or display the file size in bytes (otherwise kilobytes or megabytes are shown for large files) in tree, detailed list, and text view modes. The Trash tab determines whether you'll be asked for confirmation

when moving a file to the trash, deleting it, or shredding it. For every file type checked on the Previews tab, the Konqueror file manager displays a mini-preview of the file contents instead of an icon.

18.3.2 Information

The Information modules allow you to view status information about various system components. There are no configuration settings here, but if you need to see information about your processor or what PCI devices you have installed, use these modules. Information is provided for the following system components:

- Block devices
- Devices
- DMA-Channels
- Interrupts
- IO-Ports
- Memory
- Network interfaces
- Partitions

- **PCI**
- **PCMCIA**
- Processor
- Protocols
- Samba status

- **SCSI**
- Sound
- USB devices
- X-Server

18.3.3 Look & Feel

The Look & Feel modules determine the appearance of the KDE desktop and configure it to work in a way that is comfortable for you.

18.3.3.1 Background

This module sets the desktop background. It allows you to specify a background for each virtual desktop or just one background for all of them. Select the desktop you want to configure from the list, or check the Common Background box.

On the Background tab at the bottom of the window, select whether you want a flat (Solid color) background, a pattern, or one of several gradients from the Mode drop-down box. If you select a pattern, click the Setup box, and the Select Background Pattern window will open.

Select a pattern from the choices or add your own, and click OK. For a pattern or gradient, choose colors for Color 1 and Color 2 by clicking on the color bars. For a flat background, choose a color from the Color 1 color bar.

On the Wallpaper tab, you can choose No Wallpaper, Single Wallpaper, or Multiple Wallpaper. If you are using wallpaper, the Mode setting determines how the image file is laid out. You can choose a tiled layout, or various centered or scaled options. If you want to use an image file as wallpaper on the background, select it from the drop-down list, or click the Browse button to look for the image on the filesystem. To use multiple wallpapers, click the Setup Multiple button. Set the time interval that each wallpaper remains on the screen. The Mode button lets you cycle through wallpaper patterns in order or randomly. Choose the wallpapers to use from the list, or click the Add or Remove button to add or remove files from the list.

When viewing available wallpapers, right-click the display and choose Thumbnail Previews to see what they'll look like.

The Advanced tab sets a blending pattern for a multicolored background. The choices range from no blending to horizontal or vertical blending to various patterns and intensities.

18.3.3.2 Colors

The Colors module allows you to select the colors for the various window widgets, the components used to build windows. You can select the colors based on a scheme installed on your system. Select one of the available schemes from the list, and click Apply to change the color scheme. You can also set colors for individual components. The top portion of the tab shows sample window components. Click on the component you want to configure, such as the active titlebar or window background, to select it, or select a component from the widget color list. Click the color bar under the list to open the color selector dialog box and choose your color.

Other choices are to set the contrast of the component with the Contrast slider and to apply the colors to non-KDE applications.

18.3.3.3 Desktop

The Desktop module sets preferences for the visual display of your environment.

You can determine the appearance of your desktop and the number of virtual desktops, the actions taken by mouse buttons, and other options.

On the Desktop tab, select the options you want. Enable Desktop Menu places a copy of the desktop menu as a menubar at the top of the screen. Enable Icons on Desktop lets you place icons on the desktop for easy access. If you also set Show Hidden Files on

Desktop, all files in the Desktop directory are displayed, including hidden files.

Programs in Desktop Window lets you set a running application such as **xearth** as your desktop background.

From the list of file types under "Show Previews

for," select any for which you want a preview

displayed on the desktop instead of an icon. Under

"Clicks on the desktop," select the

action to be taken on a left, right, or middle mouse-button click.

From the Appearance tab, select your preferred font, font size, and text color. You can also select a background color if you want to view the background box of the label. By default, the background is transparent, allowing the desktop background to be visible. Finally, you can choose to have filenames underlined.

The Number of Desktops tab sets the number of virtual desktops you can divide your workspace into. The default is 4, and you can use up to 16. The slider selects the number of desktops. The desktops are listed, numbered 1 through 16. For each enabled desktop, the listing has a usable text box next to it. You can type in

a label for each desktop that will be shown in the pager display on the panel. The default labels are Desktop 1 through Desktop 16.

The Paths tab lets you set the path to the desktop, trash, and Autostart directories and to the default path for documents. In most cases, you won't need to change the defaults.

18.3.3.4 Fonts

The Fonts module sets the default fonts used on your display. You can set the font for the following listed selections: General, Fixed Width, Toolbar, Window Title, and Taskbar. Click the Choose button for the font you want to change, and the Select Font window appears. From this, you can select the font you want, choose if you want it to be bold or italic, and select the point size. A sample of the selected font is displayed at the bottom of the window. You can also check the box labeled "Use Anti-Aliasing for

fonts" to make the fonts appear smoother on the

screen.

18.3.3.5 Icons

This module controls the display and behavior of icons on the desktop. The module has two tabs, Theme and Advanced.

On the Theme tab, select an icon theme. The sample icons at the top are displayed in the current theme. After selecting a theme, select Install New Theme to install it. At the bottom of the window, you can add a new theme to the list by entering the path in the text box or clicking the button at the right to open a File dialog window. You can also remove any theme except the default.

Icons are used for many purposes: in the file manager, on the panel, and so on. The Advanced tab configures icons individually for different uses. First, select the use that you want to configure from the list. For that use, select a size from the Size list, or "Double-sized pixels" to get very

large, but somewhat blurry, icons. You can also choose to have the icons animated. The icon images at the bottom show how the icon will appear normally (default), when it is active (i.e., when the cursor is over it), and when it is disabled.

For each use, you can set special effects. Clicking the Set Effect button displays the Set Default Icon Effect window. From that window, select an effect and move the slider to determine the amount of that effect. For instance, choosing the To Gray effect for Active causes an icon to turn gray when you put the cursor on it. (To Gray is probably more appropriate for Disabled.) If you choose Colorize, the Color option becomes active so you can pick a color.

18.3.3.6 Launch Feedback

The Launch Feedback module has two options to provide visual feedback while an

application is starting, so you know something is happening. If you check the box Enable Busy Cursor, a mini-icon appears and tracks the cursor. Under the Enable Busy Cursor option, checking Enable Blinking causes the icon to blink. Enable Taskbar Notification causes the taskbar entry to show a revolving timer while the application starts.

Both options let you set a timeout, which has a maximum of 99

seconds. When the timeout period is over, the visual feedback stops, even if the application is not yet fully started.

18.3.3.7 Panel

The panel configuration module controls the panel's

appearance, the panel's behavior, and the K menu

layout, and it configures any extensions that are enabled.

On the Position tab, you can choose to place the panel at the top, bottom, left side, or right side of the screen by clicking the appropriate radio button. You can set the panel alignment to Left, Center, or Right. Five settings are available to set the size of the panel to either Tiny, Small, Medium, Large, or Custom. In the Length section, you can move the slider to set the amount of the width or height of the desktop that the panel occupies. If you set it to less than 100% and click the "Expand to fit required

size" button, the panel will expand to take up as much room as it needs to fit all the icons.

The Hiding tab, shown in Figure 18-7, is divided into sections that provide options for hiding the panel when you want more visible desktop space to use as a work area. The Hide Buttons section contains checkboxes that enable the display of a hide button at one or both sides of the panel. Clicking on a hide button causes the panel to roll up into that side and disappear, leaving only a button you can click to have the panel reappear. Move the slider in the Hide Button Size section to set the size of the hide button.

Checking the box in the Manual Hide Animation section enables the panel to use a sliding visual effect when you show or hide it with its side arrow buttons. The slider sets the speed of this animation.

You can have the panel autohide (i.e., disappear when not being used) by checking the "Enable automatic

hide" button and moving the slider to set the amount

of time after the pointer has left the panel that the panel will hide. The panel reappears when you move the mouse over it. Checking the "Show panel when switching

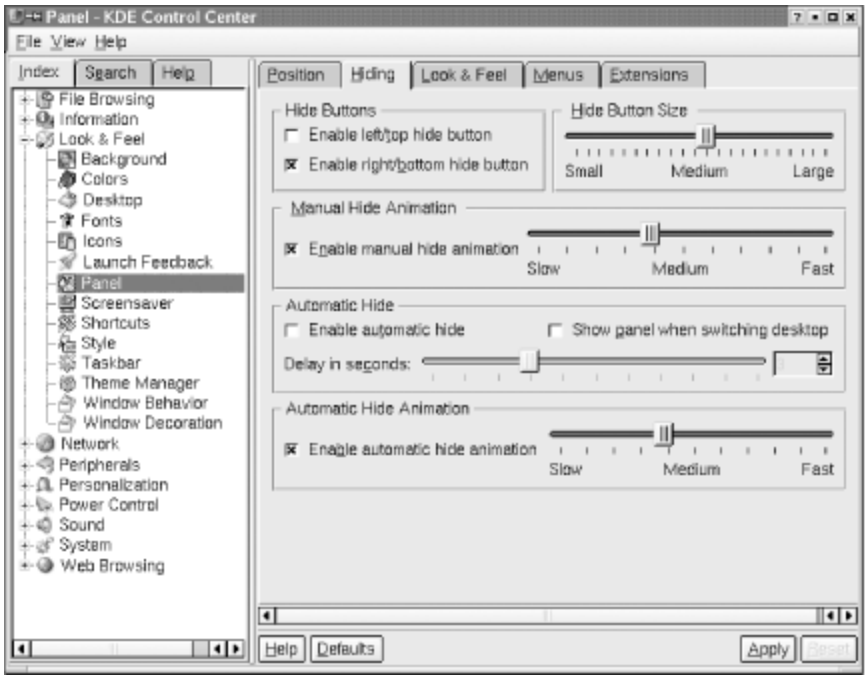
desktop" box causes the panel to reappear when you

move to a different desktop. It appears on the new desktop and then hides again. You can set "Enable automatic hide

animation" and move the slider to set the animation

speed.

Figure 18-7. Panel hiding configuration



The Look & Feel tab contains settings for the appearance of the panel and its icons. Each icon is on a tile that by default is transparent, so all you see is the icon. If you set

"Enable background tiles," the

tiles are shown, and the settings in the middle of the window are enabled so you can enable or disable different tile types and set the background colors individually. The tile types are K-Menu,

Quickbrowser, Application Launcher, Legacy Application Launcher, Window List, and Desktop Access.

Also on the Look & Feel tab, the "Enable icon zooming" setting causes icons to zoom large when you

pass the mouse over them, so they appear to pop out at you. The "Show tooltips" option displays

tooltips when you move the mouse over icons on the panel. To see an image on the panel background, set "Enable

background image" and either enter the path to the

image or browse to find one.

The Menus tab contains settings for the K menu and other menu layouts. Settings for the K menu include:

Show side image

Show a small image next to each menu entry.

Show "Bookmarks" submenu

Include a Bookmarks menu that shows your Konqueror bookmarks and lets you edit them.

Show "Recent Documents" submenu

Display a list of recently used files. Click the Clear History box to empty the list.

Show "Quick Browser" submenu

Add a browser menu for easy access to files without running the file manager. The top-level entries are Home Directory, Root Directory, and System Configuration to start from the user's

home directory, the root directory, and */etc*,

respectively.

Detailed menu entries

Include a brief description of each menu item. If the box is not checked, only the application name is shown.

Below these settings are two columns, Available Menus and Selected Menus. The first column contains menus that are not in the K menu but can be added. The second column contains those you have selected to add. Highlight an entry in the Available Menus column and click the right-pointing arrow to move it to the Selected Menus column; do the reverse to unselect an entry. Click the Apply button to add the selected menus to the K menu. Once added, they stay in the Selected list.

To have browser menus show hidden files (i.e., dot files), select "Show hidden files in browser

menus." Move the slider "Maximum

browser menu entries" to change the number of

entries a browser menu will display.

The Quick Start section is at the top of the K menu. Its entries consist of either the most recently used or the most frequently used applications, depending on which radio button is selected. Set the maximum number of entries shown in the Quick Start section if you want to change the default, which is 5.

The Extensions tab is active if you have added any extensions, such as a child panel or an external taskbar (by right-clicking on the panel and selecting Add ⟶ Extension). For each extension, you can locate it to the left, right, top, or bottom of the desktop, set automatic hide options, enable hide buttons, and specify the hide button size.

18.3.3.8 Screensaver

This module sets up your screensaver. A list of available screensavers is shown with a preview window. If you want to use a screensaver, check the Enable Screensaver box and select the screensaver you want to use.

The Setup button opens a dialog box that contains specific

configuration settings for each screensaver. For a full-screen test of the screensaver, click the Test button.

The Settings section allows you to set the amount of time the system is inactive before the screensaver starts. Type in the number of minutes in the Wait For box to set this time. If you check the

Require Password box, the user must supply a password before returning to the desktop. You can also check the box

"Show Password as Stars" to display

the password text as asterisks instead of leaving the field blank.

The Priority control lets you adjust the priority that the

screensaver process has when it is run. If you have lots of important server activity, for example, set the priority to low so the performance of other programs does not suffer.

18.3.3.9 Shortcuts

Shortcuts are key combinations that do something to your desktop or

application. By default, for example, Alt-F4 closes the current window. That's a global shortcut; there are also

shortcuts used in specific applications.

You can choose the scheme used for shortcuts by selecting the KDE

defaults or the current scheme if you have customized the shortcuts.

The bottom section of the tab allows you to edit the selected keyboard shortcut. You can choose no key for the action, the default key, or a custom key. If you select custom, the Define Shortcut dialog appears. Here you can set or modify the primary or alternate shortcut. Click the X in the box pointing to the current shortcut, then click Multi-Key if you want to use any modifiers (Shift, Ctrl, or Alt). Finally, press the key or key combination on the keyboard that you want to use. Click Apply to save the change to the current scheme, or click New Scheme and then Save to save them as a new scheme.

The Modifier Keys tab shows the modifier keys for the current keyboard. It is not used for configuration, but is there simply to provide information.

18.3.3.10 Style

The Style module contains settings for the display of windows and icons. Choose a style from the Widget Style list on the Style tab. The bottom half of the window previews the selected style. If you check Enable GUI Effects on the Effects tab, you can set effects such as fade or animation for ComboBox, ToolTip, and Menu effects. You can also make menus translucent and specify the type and degree of translucency or opacity. (Note that translucency uses more system resources, so you might not want to use these effects on a slower system.) The Miscellaneous tab offers several toolbar and visual appearance options.

18.3.3.11 Taskbar

The Taskbar module configures the taskbar, both in the panel and on the desktop. It is not used to set where on the desktop the taskbar appears (do that from Panel ⟶Extensions). To add or remove the taskbar from the panel, right-click on the panel and select Add or Remove and then Taskbar. To place the taskbar directly on the desktop or to remove it from the desktop, use the Extensions tab in the Panel module.

The Taskbar module settings include the following:

Show windows from all desktops

> The taskbar shows all windows, regardless of which desktop they are on. You can then select a task and move directly to that desktop with that window active.

Show window list button

Add a menu button at the left edge of the taskbar that displays, when clicked, a list of all desktops and their windows. This is

particularly useful if "Show windows from all

desktops" is unchecked, since it lets you select a

window on another desktop.

Group similar tasks

Combine multiple instances of any application into a single taskbar entry. The entry shows a small up-arrow. When you click on the arrow, a list of the instances appears and you can select the one you want.

Sort tasks by virtual desktop

Order the taskbar by desktop.

Show application icons

Show the application's icon along with the text on each taskbar entry.

The Actions section sets an action for each of the left, middle, and right mouse buttons. The possible actions for each are Show Task List, Show Operations Menu, Cycle Through Windows, Activate Task, Raise Task, Lower Task, and Minimize Task.

18.3.3.12 Theme Manager

Themes provide an overall visual style to your desktop, so you get an integrated look instead of having to configure items individually.

A theme can determine the color scheme of windows, the font styles, icons, background, and even sound events for your desktop. Several themes are installed by default with KDE, and many more are available at http://kde.themes.org for download.

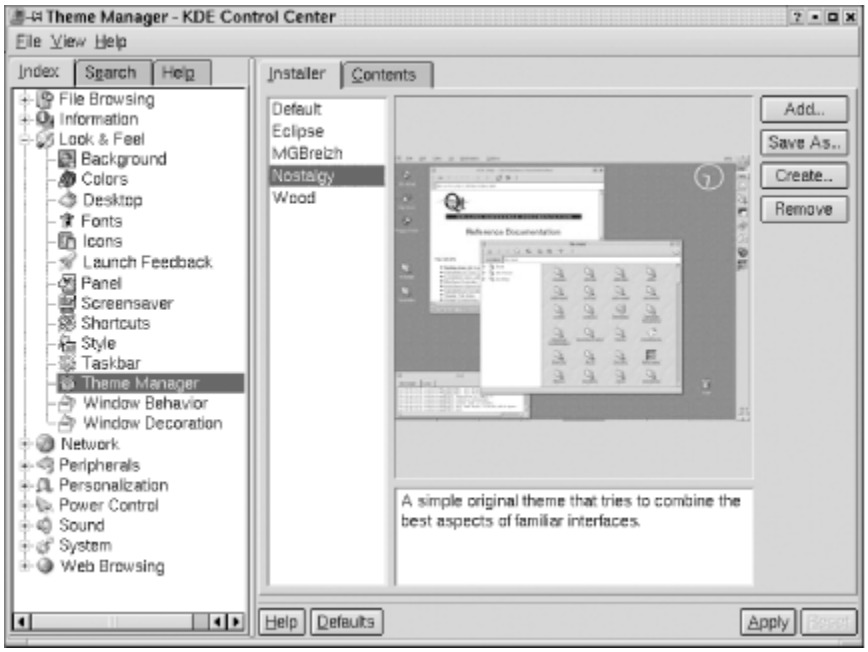
The Installer tab, shown in Figure 18-8, lists the themes you have installed on your system. These include global themes, which are stored in

$KDEDIR/share/apps/kthememgr/Themes, and local

themes, which you have installed or customized and stored in *~/.kde/share/apps/kthememgr/Themes*. You can edit

and save local themes. Global themes cannot be altered by individual users.

Figure 18-8. Theme Manager Installer tab



If you select a theme from the list, a sample desktop image using the theme is displayed on the tab with a short text description. To apply a new theme, select from the list and click the Apply button. The selected theme is copied to your theme manager work directory (*~/.kde/share/apps/kthememgr/Work*).

You can also manage installed themes on the Installer tab. To install a new theme that you have downloaded, click the Add button. Specify the filename and location of the theme's

.tgz file in the pop-up dialog box and click OK.

The Save As button saves the currently configured theme as a separate local package without altering the original theme. The Create button works similarly, allowing you to copy your current working theme as a new local theme

package. The Remove button deletes a local theme or inactivates a global theme.

The Contents tab shows the components that are configured by a theme.

A theme may not have settings available for all the listed

components. If a theme configures a specific component, it is listed as available. Otherwise, the component is listed as empty. Use the checkboxes to select which theme components you want to use. If you choose not to activate a specific component, information on that component from a previous theme will be used if its information is still in the theme manager work directory. If you

don't want this to happen, activate the component

even if it is listed as empty, and default settings will be used.

18.3.3.13 Window Behavior

The modules in the Window Behavior module allow you to set the look and functionality of window titlebars, mouse button actions, and focus and placement policy.

On the Focus tab, the Focus Policy section sets the policy for giving a window keyboard focus and an active titlebar. The drop-down list contains four focus styles:

Click to Focus

> This default focus policy requires a mouse click in a window to give it focus and raise it (bring it to the foreground).

Focus Follows Mouse

> This setting causes a window to receive focus when the mouse pointer enters it. It does not come to the foreground unless the Auto Raise button is checked. The window maintains focus until the pointer enters another window.

Focus Under Mouse

> This setting is similar to the Focus Follows Mouse setting, except that the window loses focus when the mouse pointer moves out of it.
>
> If the pointer is not in a window, no window has focus.

Focus Strictly Under Mouse

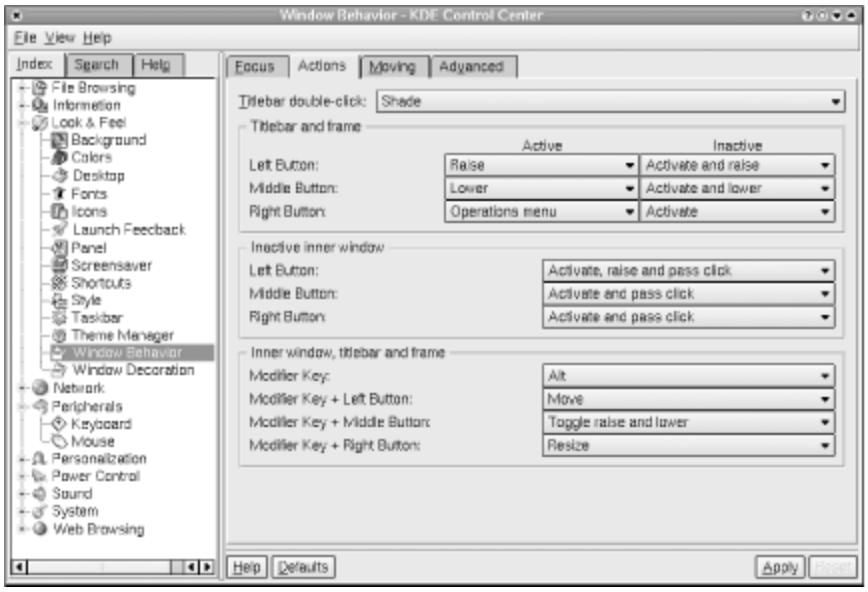
> This setting is similar to Focus Under Mouse, but is stricter in interpreting when the mouse leaves the window.

For all the focus policies except Click to Focus, the Auto Raise and Click Raise boxes are available. One of these must be checked to be able to raise windows to the foreground. Auto Raise raises a window after a short delay, which you can set with the Delay slider.

The Keyboard section determines behavior when you use the keyboard to switch windows or desktops. In KDE mode, a box appears showing the window you are about to switch to. In CDE mode, no box appears. Check "Traverse windows on all desktops"

if you want to move through all windows on all desktops. (CDE is an older desktop distributed with some commercial Unix systems.) The Actions tab (Figure 18-9) configures the actions of mouse buttons on the various window components. Drop-down lists contain several options, such as raise or lower, for the left, middle, and right buttons. For each of the components, you can choose what happens when a mouse button is clicked. For instance, by default, right-clicking in a titlebar brings up an Operations menu.

Figure 18-9. Windows Behavior Actions tab



The Moving tab sets window and snap zone options. The Windows section of the tab has the following settings:

Display Content in Moving Windows

Enable the display of window contents when the window is moved.

Display Content in Resizing Window

Enable the display of window contents when the window is resized.

Animate Minimize and Restore

Enable animation and activate the slider, which controls the animation speed.

Allow Moving and Resizing of Maximized Windows

Allow the use of the border on a maximized window to move or resize the window.

Placement

Choose Smart, Cascade, or Random window placement from the drop-down list to determine the initial placement of new windows on the desktop. Smart placement attempts to keep windows as uncluttered as possible. Cascade attempts to place windows in a cascaded pattern, so at least an edge of each window is visible. Random placement puts each new window randomly on the desktop.

The Snap Zones section sets the width of the zone within which a window will snap to the desktop border at the edge of the screen ("Border snap zone") or to another

window ("Window snap zone"). Both

zones are set with sliders and default to 10 pixels.

The Advanced tab sets options for configuring window shading, the use of Xinerama, and active desktop borders. In the Shading section, check Animate to animate window shading and unshading, and check Enable Hover to cause a shaded window to unshade if the mouse pointer remains over the titlebar for the number of milliseconds determined by moving the Delay slider.

Xinerama is a feature of XFree86 that allows multiple monitors to be treated as one big monitor. KDE supports Xinerama; the Xinerama section is available only if Xinerama support was compiled into KDE.[1] The following checkboxes configure the use of Xinerama:

[1] The screens also need to be configured in

/etc/X11/XF86Config.

Enable Xinerama Support

Enable the use of Xinerama with KDE.

Enable Window Resistance Support

Provide some resistance when a window moves to the edge of one monitor screen so that it doesn't accidentally move

onto the next.

Enable Window Placement Support

Provide support for positioning windows on any portion of the logical desktop, regardless of which monitor it falls on. If window placement support is enabled, a new window appears on the screen where the mouse pointer is.

Enable Window Maximize Support

Permit windows to be maximized across multiple monitors.

Active desktop borders enable you to switch between desktops by moving the mouse pointer to an adjacent screen edge. In the Active Desktop Borders section, the choices are:

Disabled

Turn off active desktop borders.

Only When Moving Windows

Enable active borders only when moving windows across the edge, not when moving the mouse across.

Always Enabled

Active borders are enabled both for moving windows and for moving the mouse across the edge to another desktop.

The Desktop switch-delay slider sets a delay time for the switch to the adjacent desktop. Set this time to a comfortable setting that doesn't cause an unwanted desktop switch every time

you move the pointer to the screen edge. The slider is active unless active borders have been disabled.

18.3.3.14 Window Decoration

The Window Decoration module sets preferences for the appearance of KDE windows. There are three tabs: General, Buttons, and Configure.

The Buttons tab configures the layout of buttons that appear on the titlebars of windows. It is available only if you check

"Use custom titlebar button

positions" on the General tab. There are five

buttons: minimize (dot), maximize (square), sticky (pushpin), close (X), and menu (dash or application-specified icon). Each button has three placement options, specified by radio buttons: left, right, or off. You can place no more than three buttons on one side of the titlebar.

The options available on the Configure tab vary according to the window decoration style selected on the General tab. Depending on the style, they affect the appearance of the titlebar and buttons, and the width of the window frame. The tab itself shows the name of the current style. For some styles, no configuration options are available.

18.3.4 Network

The Network modules configure a number of networking features on your system. These modules don't configure the network itself, but set your

personal preferences and configuration options. The primary networking modules in the Control Center include Email, Preferences, SOCKS, and Window Shares. These modules are described here. Depending on your Linux distribution and what applications you have

installed, you may also see modules such as Kisdndock, which configures ISDN

docking options; LAN Browsing to configure LISa, the LAN Information Server, which provides a TCP/IP-based "network

neighborhood" for your network; and News Ticker,

which is a news ticker applet that runs on the panel and provides running access to current news from sites you specify.

18.3.4.1 Email

The Email module provides a place for you to specify some basic pieces of information that KDE may use for email-related purposes. In the User Information section, you can specify your Full Name, Organization, Email Address, and Reply-To Address. In the Preferred Email Client section, enter the name of the email client you want KDE to use on your behalf. The default client is KDE's KMail, but you can enter another client or

click the Browse button to find available clients. Check the Run in Terminal box if you use a text-

based email client such as Pine.

18.3.4.2 Preferences

The Preferences module sets timeout values for "Socket read,"

"Proxy connect,"

"Server connect," and

"Server response." If your

connection is slow and you find that you are getting frequent timeouts, you might want to increase the values. This module also has an FTP Options section where you can check "Enable

passive mode (PASV)" if you want to permit passive

FTP transfers because you are behind a

firewall, and you can check "Mark partially uploaded files" so

that they appear in a directory listing with a

.part extension until the transfer is complete.

18.3.4.3 SOCKS

The SOCKS module enables SOCKS support in KDE applications. SOCKS is a protocol that executes proxy requests on behalf of a client. If you have SOCKS installed and working, you can check the box "Enable SOCKS

support." At that point, additional options become

available. Click "Auto detect" to

have KDE try to determine which SOCKS implementation you are using, or click NEC SOCKS or Dante if you are using one of those. Finally, if KDE can't detect your SOCKS implementation, you

can enter one or more paths in the "Additional

library search paths" section for KDE to search.

Click the Test button to test your SOCKS support.

18.3.4.4 Windows Shares

The Windows Shares module configures your Samba client if you are running Samba on your network to communicate with Windows systems. This module configures the client, not the Samba server, so that Konqueror can access shared Windows filesystems from other computers.

Enter your default username, default password, and workgroup in the three text boxes at the top of the window. Check the box

"Show hidden shares" if you want to

see shares whose names end with a dollar sign ($).

Note that entering your password and having KDE store it on the system is a security risk. If you don't enter your

name or password and they are required to access the shared Windows system, you will be prompted for them. That's less

convenient, but safer.

18.3.5 Password

The Password module is available in some distributions, including Red Hat 8 with Bluecurve, allowing you to change your login password. This module prompts you for your current password and new password, and then asks you to re-enter the new password.

18.3.6 Peripherals

The modules here control the configuration of the keyboard and mouse.

These settings affect only your use of KDE; they

don't modify the global X server settings. Some

distributions may include other modules in this section, such as a Digital Camera module that lets you add and configure one or more digital cameras.

18.3.6.1 Keyboard

This module configures your primary keyboard layout and any additional layouts (if you work in more than one language, for example). Use it also to configure other keyboard options.

The Layout tab determines your keyboard layouts if you check "Enable keyboard layouts." In the

Configuration section, three drop-down lists are available to set your keyboard model, primary country and language layout, and a primary variant. The options on the variant drop-down list vary depending on which primary layout you select, but generally include different keymaps for the same language.

In the "Additional layouts"

section, select any other layouts you may want to use. You can also select a variant for each. The additional layouts include variations such as the Dvorak keyboard layout, in addition to language layouts.

The list of layouts shows a flag image for each layout. When you enable layouts, a button is added to the system tray in the panel that allows you to switch between your selected

layouts. The button contains the flag for the current layout. Click the flag to cycle through the layouts.

The Options tab sets the Switching Policy to one of Global, Window Class, or Window. The policy determines whether a layout switch happens globally, by window class, or on a window-by-window basis.

The Xkb Options section sets Xkb (X keyboard) options; you can check the box Reset Old Options to reset your options to the defaults as defined in your */etc/X11/XF86Config* file. The rest of this section consists of five drop-down lists used to set the behavior of certain keys.[2] Each can be left blank or set to one of the listed options. The drop-down lists are:

[2] Depending on your Linux distribution, you may not see all the lists described here.

CapsLock Key Behavior

> Sets whether the Caps Lock key uses internal capitalization or acts as a shift lock key, in each case with or without canceling caps.

Third-Level Choosers

Sets the key or keys that can be used as an ISO 9995 third-level chooser. ISO 9995 is a multipart ISO standard that specifies keyboard layouts for text and office systems. The choices are Any of the Win-keys, Left Win-key, Right Control, Right Win-key, or Menu key.

Control Key Position

Determines the position of the Control key. The choices are Control Key at Bottom Left, Control Key at Left of

"A", Make CapsLock an Additional

Control, or Swap Control and CapsLock.

Use keyboard LED to show alternative group

Sets one of Scroll_Lock LED, Caps_Lock LED, or Num_Lock LED to show an alternative group as defined by ISO 9995.

Group Shift/Lock behavior

Determines what key is used to switch groups. Some of the options change the group until it is changed again, while others change the group only while the key is pressed. The choices are: Left Win-key changes group, Both Win-keys switch group while pressed, R-Alt switches group while pressed, Both Shift keys together change group, Right Win-key changes group, Left Win-key switches group while pressed, Caps Lock key changes group, Right Alt key changes group, or Alt+Shift changes group.

The Advanced tab configures the keyboard repeat (i.e., holding a key down repeats the character until you release the key). Check the "Keyboard repeat" box to enable

keyboard repeat, and use the slider to set the keyclick volume. If you don't want to hear keyclicks, set the slider to

0. Note that your keyboard may not support changing the keyclick volume. The section "NumLock on KDE

startup" determines the state of the NumLock key

when KDE starts. Click Turn on, Turn off, or Leave unchanged.

18.3.6.2 Mouse

This module configures the movement and button layout of your mouse.

On the General tab, the Button Mapping section sets the mouse to be right-handed or left-handed. In the Icons section, select either "Single-click to open files and

folders" or "Double-click to open

files and folders." If you select single-click, you

also have the option to check "Change pointer shape

over icons" and "Automatically

select icons." If you select

"Automatically select icons," move

the slider to determine how long the pointer can remain over the icon before it is selected. If you select double-click to open, then a single click selects an icon. Check "Visual feedback

on activation" or "Large

cursor" to select either of those options.

On the Advanced tab, the Pointer Acceleration slider determines how fast the pointer moves on your screen when you move your mouse. Lower values move the pointer more slowly for small mouse movements, giving you finer control. Higher values let you cross the screen more quickly. The Pointer Threshold slider sets the distance (in pixels) that the mouse must move before pointer movement occurs. The Double Click Interval slider sets the maximum time that can pass between two clicks and still be considered a double-click. The Drag Start Time slider sets the amount of time that has to pass after you click a button and start to move the mouse before it is considered a drag operation. The Drag Start Distance slider sets the distance (in pixels) that the mouse must move with a button clicked before it is considered a drag operation. The Mouse Wheel Scrolls

By slider determines how many lines are scrolled for one mouse wheel movement.

18.3.7 Personalization

The Personalization modules cover configuration options that don't fit neatly into any

of the other categories.

18.3.7.1 Accessibility

The Accessibility module sets configuration options to make the system more accessible. The Bell tab sets the audible bell to one of "Use System

bell" or "Use customized

bell." If you choose a customized bell, use the text

box to enter the path to the sound you want, or select Browse. You can set a visible "bell" by

checking "Use visible bell" and

then either "Invert screen" or

"Flash screen." Select a color by

clicking the color button and use the slider to set the duration of the inversion or flashing.

The Keyboard tab has three settings to make the keyboard easier to use:

Use sticky keys

> Pressing one of the Ctrl, Alt, or Shift keys and then releasing it to press another key is treated as a multi-key combination for the next keypress. For example, to enter Ctrl-Tab with sticky keys enabled, press Ctrl, release it, and then press Tab. You can also check "Lock sticky keys." In that case,
>
> pressing Ctrl, Alt, or Shift once behaves the same as
>
> "Use sticky keys"; pressing the key
>
> twice leaves it in the modified mode until you press it again to release it.

Use slow keys

> Each key must be held down for the amount of time specified by the Delay slider to be recognized as a key press. This protects you from accidental keystrokes.

Use bounce keys

> The amount of time specified by the Delay slider must pass between two key presses for the second one to be recognized. This protects you from accidentally hitting multiple keys if you type fast.

The Mouse tab sets values for using the arrow keys on the numeric keypad to move the mouse pointer instead of having to use a mouse, trackball, or other device. The possible settings are Acceleration Delay, Repeat Interval, Acceleration Time, Maximum Speed, and Acceleration Profile. If you

don't have a numeric keypad (e.g., on a laptop

computer), the numeric keypad keys are usually available on the keyboard, accessed with a function (fn) key.

18.3.7.2 Country & Language

The Country & Language module sets the preferred locale settings for your programs.

The Locale tab has a drop-down list for selecting your country. When you select a country, the language or languages generally used in that country appear in the Languages box. You can add a language to the list from the

Add Language drop-down list, or select a language and click Remove Language to remove one from the list. The bottom portion of the tab shows the default appearance of numbers, money, dates, and time for that country and language. Use the remaining tabs to change these defaults. As you change the settings, the examples at the bottom of the window (which are shown on all the tabs) also change, so you can see how they will look with the new settings.

The Numbers tab lets you set the character used for the decimal symbol, thousands separator, positive sign, and negative sign.

The Money tab lets you set the character to be used for the currency symbol, decimal symbol, thousands separator, and fract (fractional) digits, i.e., the number of digits after the decimal point. In the sections positive and negative, check the "Prefix

currency symbol" box to have the currency symbol

appear before the positive or negative sign, and then select where the sign should be placed from the drop-down list. The possible positions are:

Parentheses around

Display the value in parentheses. For example, you might select this option to display negative numbers in parentheses rather than showing a minus sign.

Before quantity money

Display the sign after the currency symbol and before the value.

After quantity money

Display the sign after the value and before the currency symbol.

Before money

Display the sign before both the value and the currency symbol.

After money

Display the sign after both the value and the currency symbol.

The Time & Dates tab has the following settings: Time format, Date format, Short date format, and Start week on Monday. Select the time and date formats from drop-down lists or enter a format into the appropriate text box. Check the "Start week on

Monday" box if you want Monday rather than Sunday to

start a new week. The time and date formats are specified with the following codes; the table

shows the time codes first, then the date codes.

Code	Replaced by
HH	The two-digit hour on a 24-hour clock.
hH	The hour on a 24-hour clock, using one or two digits as appropriate.
PH	The two-digit hour on a 12-hour clock.
pH	The hour on a 12-hour clock, using one or two digits as appropriate.
MM	The minute, using two digits.
SS	The second, using two digits.

Code	Replaced by
AMPM	Replaced with either "am" or "pm" as appropriate; primarily for use with a 12-hour clock setting.
YYYY	The four-digit year.
YY	The two-digit year.
MM	The two-digit month.
mM	The month, using one or two digits as appropriate.
MONTH	The full name of the month.
SHORTMONTH	The abbreviated name of the month.
DD	The two-digit day of the month.

Code	Replaced by
dD	The day of the month, using one or two digits as appropriate.
WEEKDAY	The full day of the week.
SHORTWEEKDAY	The abbreviated day of the week.

18.3.7.3 Crypto

The Crypto module controls the configuration of the OpenSSL

implementation of the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) network protocols. Use this module to configure SSL for use with KDE

applications, and to manage your personal certificates and known certificate authorities. The module has six tabs: SSL, OpenSSL, Your

Certificates, Authentication, Peer SSL Certificates, and SSL Signers.

If you do not have OpenSSL installed, each of the tabs displays the message "SSL ciphers cannot be configured because

this module was not linked with OpenSSL." If you do

have OpenSSL installed, you probably still should not make any configuration changes unless you are familiar with SSL and know what you are doing. For more information on using and configuring OpenSSL, see the O'Reilly book *Network Security with Open SSL*.

At the bottom of the SSL tab are some checkboxes. If OpenSSL is installed, you'll see five checkboxes; if it is not

installed, only the last three are shown and available. The checkboxes are:

Use EGD

> Use the entropy-gathering daemon (EGD) to initialize the pseudo-random number generator. Entropy is basically a measure of randomness; SSL requires random numbers for

such functions as creating public and private keys, and EGD is a Perl script that gathers entropy.

Use entropy file

Use the file whose path you specify in the text box or by browsing as the source of entropy for the pseudo-random number generator.

Warn on leaving SSL mode

See a warning when you leave an SSL-enabled site.

Warn on sending unencrypted data

See a warning when you are about to send unencrypted data over the network with a web browser.

At the top of the SSL tab are three checkboxes that enable TLS

support (if it's available), SSL v2 support, and SSL

v3 support. These options are followed by two columns containing a list of SSL v2 and SSL v3 ciphers. You can check or uncheck the ciphers individually to select the ones you want to use, or use the buttons below the lists to select them in groups: Most Compatible, US

Ciphers Only, Export Ciphers Only, or Enable All.

The OpenSSL tab lets you specify the path to your SSL libraries or use the Browse button to locate them. Then click on the Test button to test your settings.

The Your Certificates tab shows a list of certificates that KDE knows about and lets you manage them with a set of buttons on the right-hand side. The buttons are Import, Export, Remove, Unlock, Verify, and Change Password.

The Authentication tab lets you specify a default certificate from the drop-down button and a default action or policy. The possible policies are Send, Prompt, and Don't Send. You can

also add or remove hosts and host certificates, and set one of the same three actions for each.

The Peer SSL Certificates tab lets you Export, Remove, or Verify peer organizations, and set a policy of Accept, Reject, or Prompt for each.

The SSL Signers tab lets you Import, Remove, or Restore organizations to a list of signing organizations, and determine whether to accept for site signing, accept for email signing, or accept for code signing.

18.3.7.4 Konsole

The Konsole module configures the Konsole terminal emulator. It has four tabs: General, Schema, Session, and Write Daemon.

The General tab has settings to determine general Konsole behavior:

Use Konsole as default terminal application

Set Konsole as your default terminal.

Show Terminal Size when Resizing

Display the number of characters and lines as you resize the terminal. The display changes as you change the size, showing the current dimensions.

Show Frame

Put a frame around the terminal window. If the frame is turned on, a narrow edge appears inside the Konsole window around the portion that represents the terminal itself. This option does not affect the border around the entire Konsole window.

Warn for Open Sessions on Quit

Konsole warns you if you still have open sessions when you quit out of Konsole.

Blinking Cursor

The cursor blinks inside the Konsole window, making the cursor easier to find.

Require Ctrl key for drag and drop

Require the use of the Ctrl key for dragging and dropping text.

Konsole allows you to drag text and drop it into other applications.

Line Spacing

Use the up or down arrows to set the line spacing. The default value is Normal, which is single-spacing.

Double Click

The text box shows a default set of characters (besides letters and digits) that are treated as part of a word when you double-click to select a word. You can add or remove characters from this set.

The Schema tab sets up the appearance of the Konsole window. The Schema box in the center of the tab shows the available schemas with the current one highlighted. The current schema also is shown in the Title box at the top. Check the box under the list of schemas to set the highlighted one as the default or click the Remove Schema button to remove it from the list. You can also set up a color scheme and add a background image to your Konsole sessions.

The Session tab lets you set properties for the different types of Konsole sessions. Pick the session type from the list in the Session section of the tab, and then enter or change the name, the command to execute, and the default directory in the General section. You can select an icon by clicking on the image of the current icon, pick a font size from the Font drop-down menu, enter or change the setting of the **TERM** environment variable, set the keyboard type (e.g., Linux console, VT100, or XTerm), and set the color schema for that session type.

The Write Daemon tab has a single checkbox, whose effect is to start the write daemon when KDE starts up. You should generally leave this box checked.

18.3.7.5 Passwords

The Passwords module has three settings that determine how passwords are treated when you run privileged operations. Note that there are

security implications to these settings.

If you are not in a secure location, be careful of setting options that may make things easier for you but may also open security holes.

The settings are:

Echo characters as

Determines if and how your password is echoed on the screen as you enter it. Each character can be shown as one star, three stars, or not at all.

Remember passwords

KDE will remember passwords so they don't have to be entered each time.

Timeout

If "Remember passwords" is checked,

this option becomes available to set the period of time for the passwords to be remembered. For example, you might set a timeout period of an hour if you are temporarily doing work that requires superuser privileges.

18.3.7.6 Session Manager

The Session Manager module controls the configuration of your KDE sessions. Check the

"Confirm logout" box if you want

KDE to ask for confirmation before logging you out. The confirmation box lets you change your mind and cancel the logout, or proceed to log out, with or without saving your current session information. If you check the "Save session for future

logins" box, the logout confirmation box will also

have that option checked by default; you can uncheck it when you log out to have your next session revert to the previously saved session.

The section "Default action after

logout" has three options that control what action

takes place by default when you log out. Check one of: Login as Different User, Turn Off Computer, and Restart Computer. The default action applies only if you logged in through the **kdm** display manager. You can always choose a different option if you let the system prompt you for confirmation.

18.3.7.7 Spell Checking

The Spell Checking module configures the KDE spell checker, KSpell. You can check either or both of

"Create root/affix combinations not in

dictionary" and "Consider

run-together words as spelling errors." Select a

dictionary from the drop-down list of dictionaries available on your system; also select the language encoding to use from another drop-down list, and whether to use international ISpell or ASpell as the underlying spell-checking client program. International ISpell is

an interactive spell-checking program that understands many European languages. ASpell is similar to ISpell, but it is smarter about suggesting alternative spellings.

18.3.8 Power Control

The Power Control modules configure the energy-saving and battery settings for your system. The Energy module applies to all systems that have a monitor with energy-saving capabilities; the remaining modules apply to laptop or other battery-powered systems using the Linux Advanced Power Management (APM) or Advanced Configuration and Power Interface (ACPI) features.

The Laptop Power Control module is available only if your system supports APM and you set the setuid bit for the APM binary file. (Do this as root with the command **chmod u+s /usr/bin/apm**. But be aware that there are

security issues with turning on the setuid bit.) In addition, both the Low Battery Critical and Low Battery Warning modules add Suspend and Standby to the possible options if this file is setuid. Suspend mode is designed for times when you want to stop working (especially on a laptop), but want to be able to restart where you left off. Open applications are preserved in

memory, the CPU remains active at a low level, and the hard drive is shut down. Standby mode shuts the computer down almost completely, but continues to draw a small amount of power so that it wakes up quickly when you restart it. Nothing is saved in memory.

18.3.8.1 Battery Monitor

Check the Show Battery Monitor box in the Battery Monitor module to turn on the visual display of the battery's status.

Then you can set the frequency with which the status should be checked. You can also change the icons that appear when there is no battery, the battery is not charging, or the battery is plugged in and charging by clicking on the icon that you want to change and selecting a new one.

18.3.8.2 Energy

If your hardware supports power management, you can enable it by checking the Enable Display Energy Saving box in the Energy module.

Then use the sliders to set the amount of idle time before the system goes to standby mode, then to suspend mode, and finally turns off.

18.3.8.3 Laptop Power Control

This module configures how and when the automatic power-down feature takes effect. You can set the behavior separately for Not Powered, when the computer is running off the battery, and Powered, when it is plugged into a power supply. For each case you can set it to suspend, go into standby mode, or turn off. Then set the amount of idle time before the power management takes effect. It's

common to set a shorter delay for battery power to save the battery.

18.3.8.4 Low Battery Critical

The Low Battery Critical module sets a trigger time for notifying you that your battery is almost out of power. The time tells you how many minutes are left before the battery runs out of power. The remaining options determine what happens when that threshold is

reached. The first two choices are Run Command and Play Sound. Each has a text box where you can enter the path to the command or the sound. The remaining options are System Beep, Notify, Suspend, and Standby.

Suspend and standby are available only if

/usr/bin/apm has the setuid bit on.

18.3.8.5 Low Battery Warning

The Low Battery Warning module is like the Low Battery Critical module, except that it notifies you sooner when the battery is running low. The trigger time is set to a higher number, so if both warnings are set, you will first get a warning that the power is getting low, and then the power-critical warning when it is about to run out. The other options are the same.

18.3.9 Sound

The modules contained in this section configure the keyboard bell and other system sounds. The basic modules described here are Midi, Sound Server, System Bell, and System Notifications. You may also have other Sound modules, such as Audio CD IO-Slave, which configures the program that lets you make *.wav*, *.mp3*, or Ogg Vorbis

files from audio CDs or DVDs, or Mixer, which configures a sound mixer.

18.3.9.1 Midi

The Midi module sets the Midi device to use. Select the device you want from the list, and if you want to use a Midi map, check Use Midi Mapper and enter the path to the map file or click the Browse button to search for it.

18.3.9.2 Sound Server

The Sound Server module configures the aRts sound server, which is a simulation of an analog real-time synthesizer used by KDE for multimedia support. On the General tab, check "Start

aRts soundserver on KDE startup" to enable the aRts

server. This makes the rest of the checkboxes available:

Enable network transparency

Allow the server to respond to sound requests that come in over the network. Uncheck if you want the server to respond only to local requests.

Exchange security

Check this box if you also checked the "Enable network transparency" box. It allows security

information to be exchanged across the network.

Run soundserver with realtime priority

If your system supports real-time scheduling, check this box to have the sound server run at a high priority. Note that you may need root privileges for this option to take effect.

Autosuspend if idle

The sound server will suspend itself if it has been idle for the number of seconds that you specify in the combination box. Having the server suspend itself frees the sound card for other applications.

Display messages using

Specify the application that will display aRts messages and enter the application to change it from the default **artsmessage**.

Whether or not you enable the sound server, you can set the level of messages to be displayed from the Message Display drop-down list. The choices are Errors, Warnings, Informational, and Debug. Messages at the selected level or higher are displayed.

The Sound I/O tab configures the sound itself if the sound server is enabled. Choose the Sound I/O method from the drop-down list. The choices are Autodetect, Threaded Open Sound System, No Audio Input/Output, and Open Sound System. The default

Autodetect is usually fine. Similarly, you can set the Sound quality to Autodetect, 16 bits (high), or 8 bits (low). Between the two drop-down lists are four checkboxes: "Enable full duplex

operation," "Use custom sound

device," "Use custom sampling

rate," and "Other custom

options." Select full-duplex operation if you need

to record and play sound at the same time. Use the custom options if you have particular requirements that make the defaults unsuitable.

See the context-sensitive help to determine if you should check one of these boxes. The slider at the bottom of the tab labeled "Audio buffer size (response time)"

determines how responsive the sound is. Moving the slider to the left speeds up response time, while moving it to the right slows it down.

Faster response time has a cost, though, as it uses more of the CPU.

18.3.9.3 System Bell

This module configures the system bell. Checking "Use System Bell instead of System

Notification" enables the other options, which you

can set with sliders or directly in the combination boxes. Volume sets the volume of the bell, measured as a percentage of the maximum possible volume; Pitch sets the tone of the slider in Hz; and Duration determines how long the beep lasts, in milliseconds. To listen to your settings, click the Test button.

18.3.9.4 System Notifications

This module allows you to enable and configure sound events for different applications. Most of the window is taken up with a tree listing of applications and events. For each event, you can select among four choices for how you want to be notified:

Log to file

Log the event to a file, specified in the Filename text box below.

Play sound

Play the sound contained in the file specified in the Filename text box below. You can also check the box "Use external

player" to use a particular media player and enter

the path to the player in the text box.

Show messagebox

Display the notification in a message box displayed in the program's window.

Standard error output

Write the message to the standard error output.

You can select more than one option. For example, you might want to hear a sound and also have the event logged to a file.

The volume slider lets you determine the sound volume, and the long button at the bottom of the window switches between

"Enable all sounds" and

"Disable all sounds" to turn sound

on or off for all applications at once.

18.3.10 System

The system modules set configuration options that have a systemwide effect. For that reason, most of them require you to enter the root password before you can make changes.

Other modules, like the Font Installer module, let you make changes in your home directory without the root password, but require it for global changes. To enter the root password, click the Administrator Mode button in the lower-right corner of the window and enter the password in the window that appears. A red border appears around the module window to indicate that you are in administrator mode.

Administrator mode does not carry over across modules. You must enter the root password for each module you want to use in administrator mode.

In addition to the modules described here, other modules that you might find depending on your distribution and KDE version include Boot Manager (LILO), Linux Kernel Configurator, Alarm Daemon, and XML

RPC Daemon.

18.3.10.1 Date & Time

The Date & Time module lets you set the system date and time. The window has a calendar in the upper left corner of the window, and a clock in the upper right. Set the date by selecting a month from the drop-down list and setting the year in the combination box. Then select the correct date from the calendar. Set the time by entering the correct hour, minutes, and seconds in the boxes under the clock. The current time zone is displayed under the clock; you can choose a new time zone by selecting a location from the drop-down list and click Apply to see the new time zone displayed.

18.3.10.2 Font Installer

The Font Installer module installs new fonts onto the system. If you want to install a font globally, click the Administrator Mode button and enter the root password; otherwise the font will be installed in your home directory. The two columns on the Fonts tab show the directory from which the font will be installed on the left and the directory to which it will be installed on the right. To install the font from a different location, click the Change Folder button and select a new location. If you highlight a font in one of the columns, a preview appears in the box at the bottom of the window. To install a font, select it from the first column and click the Install button.

The Anti-Alias tab lets you set up options for antialiasing with the XRender Extension. If your system is not capable of using this extension, the checkboxes under the Configuration File text box are grayed out. If they are available, you can check

"Exclude range" and enter the end

points of the range to exclude certain point sizes from antialiasing, and check "Use sub-pixel hinting."

If you are in administrator mode, you can click the Advanced button to set up rules for matching font families and establish any desired include directives specifying additional font files. The Include Directives tab has two sections. Any files listed in the top section, Include, must be available. Files in the bottom section, Include If, are included if they are found, but aren't required.

The Settings tab has customization settings for five aspects of font installation:

Appearance

The Appearance settings customize the look and operation of the Font Installer module. The settings are:

Mode

Select Advanced or Basic mode. In advanced mode, the full font directory and file hierarchy are shown in the columns on the Fonts tab, and you can install Speedo and bitmap fonts in addition to TrueType and Type 1. In basic mode, only the font files are shown in the columns on the Fonts tab, and you can install only TrueType and Type 1 fonts.

Font Lists

Select whether to display the fonts on the Fonts tab in left and right columns or top and bottom columns.

Custom preview string

Check this box and enter the string if you want to see characters other than the alphabet for the font preview on the Fonts tab.

Folder & Files

When you select the Font Installer module, it searches for the correct folders (directories) and files. Use these settings to make any changes or add any missing entries for the fonts folder, the font configuration file, and the encodings folder. You can also check the Ghostscript, Fontmap file box and set the path to the file to enable configuration of Ghostscript. Check the CUPS folder box and set the path if you are using CUPS (the Common Unix Print System) and want to enable its configuration.

Install/Uninstall

To configure font install and uninstall operations, check "Fix TrueType Postscript names

table" to automatically fix the names table when you

install TrueType fonts; this corrects any problems that sometimes occur with TrueType fonts. You can also specify a file to which fonts should be moved when they are uninstalled.

StarOffice

Check the box to configure fonts for StarOffice, and specify a folder and a printer file. These options need to be set only for

versions of StarOffice before Version 6. OpenOffice and later versions of StarOffice don't need this section.

X System

This section sets some X Window System font configuration options:

Configure X to only use ... encoding

Check this box and select an encoding from the drop-down list to specify an encoding for X to use.

Generate AFMs, with ... encoding

Check this box and select an encoding from the drop-down list to generate the Adobe Font Metric (AFM) files used by programs such as StarOffice to display font sizes correctly.

Command to Refresh X's Font List

This section lets you select the command to use after adding or removing fonts to refresh the internal font list. The choices are:

- **xset fp rehash**: This command resets the font path, which causes the font server to reread its database.
- **/etc/rc.d/init.d/xfs restart**: This command restarts the X font server. You must be in administrator mode to use this command.
- Custom: Check this box to use a command of your own choosing, and enter the command in the text box.

18.3.10.3 Login Manager

The KDE login manager, **kdm** (KDE display manager), is the program that controls the graphical login screen. This module lets you configure the graphical style of the login screen and set some default display options, such as prelisted users and available session environments.

The Appearance tab lets you edit the greeting string displayed on the login screen. Certain special characters, such as **%s** and **%n** in the default greeting string, can be used; they are replaced as follows:

Sequence	Replaced by
%d	The current display
%h	The host name
%m	The machine (hardware) type
%n	The node name

Sequence	Replaced by
%r	The operating system version
%s	The operating system
%%	A single %

You can choose a logo if you want one, or select Show Clock to display a clock instead of a logo. To select a logo other than the default KDE logo, click the logo image and select another image file.

A drop-down list offers you a choice of GUI styles for the login screen. Use the Echo Mode drop-down list to choose whether to replace each character of the password with a single star, three stars, or no echo. The Language option lets you select the default character encoding for **kdm**.

The Fonts tab lets you choose the font style and size for the Greeting, Fail, and Standard screen messages. Select which type you want to configure from the drop-down list and click the Change Font button. The pop-up window shows a list of available fonts and lets you set the point size. Click the OK button to close the pop-up window. The font you have chosen is displayed in the Example area of the tab.

The Background tab lets you select the background for the login screen. See the earlier description of the Background module in the Look & Feel section for a detailed description of the options on this tab.

The Sessions tab configures session settings. There are two drop-down lists at the top that set who is allowed to shut down the system from the console and remotely. The choices for both are Everybody, Only root, and Nobody. The default for the console is Everybody; the default for remote shutdown is Only root.

The Commands section allows you to set the commands used for shutdown and restart. In the Lilo section, check "Show boot

options" to enable the LILO boot options to be

displayed in the Shutdown menu. The Session Types section configures the list of session types available to the user from the login screen. The Default list contains the various environments that are installed on your system, such as KDE and GNOME. You can add a new type or remove a type from the listing.

The Users tab lets you show a list of users on the login screen. The users are listed by username with a logo displayed for each user. To log in, you can simply click your logo to automatically enter your name into the login box, but you still must supply your password. The tab contains listings labeled "Remaining

users," "Selected

users," and "No-show

users." You can select names from the lists and use

the arrow buttons to move them from one box to another. Two options are available for who to display on the login screen. You can place names in the "Selected users" list

and click the "Show only selected

users" button, or you can click the

"All but no-show" button to display

users in both the "Remaining users"

and "Selected users" lists. No-show

users are user IDs that are used to control access-restricted system resources (e.g., **root**, **news**, and **nobody**). They are never displayed on the login screen. You can add the names of other users to the no-show list.

Check the "Sort users" box to

display the images in alphabetical order.

You can select a different default image by clicking on the image button and choosing another image file. You can also select a unique image for each user by highlighting the user in one of the lists and selecting an image file for that user.

Users whose image is not shown on the login screen can still log in; the only difference is that they have to enter their login name as well as their password.

The Convenience tab offers options that make logging in easier and more automatic. The drawback to the convenience options is that they are potential security risks, because they make it easy for someone without a valid login ID to access the system. Therefore, conveneince options should be used only on a system that is in a secure environment. The sections on the Convenience tab are:

Automatic login

If you check the box "Enable

auto-login" in this section, you log in normally to

KDE, but if you then use Ctrl-Alt-Backspace to kill the X server, **kdm** will log you in automatically when it comes up again. If you check the box "Truly

automatic login" and select a username from the

drop-down box, **kdm** will

automatically log you in to that account when it comes up.

Preselect User

kdm can be configured to start up with a username already entered in the Name field. Select None if you do not want this option, Previous to have the previously logged-in user preselected, or Specify to choose a user from the drop-down list. Check the Focus password box to leave the cursor in the password field after preselecting the user. The alternative is to leave it in the user box so you can select a different username.

Password-less login

Check the "Enable password-less

logins" box if you want users to be able to log in

without entering a password. Then move users from the

"Password required" list to the

"Skip password" list, to identify

users who can log in without entering their password.

Miscellaneous

Check the box "Automatically log in again after X

server crash" to have **kdm** log you in automatically when X restarts after a crash.

18.3.10.4 Printing Manager

The Printing Manager module configures the KDE print manager, KDEPrint. KDEPrint is a frontend to your system's printing subsystem that provides an

interface layer between KDE applications and the printing subsystem.

Therefore, the details for using the Printing Manager module may vary depending on the printing subsystem in use.

Any application can use KDEPrint by printing to

$KDEDIR/bin/kprinter instead of */usr/bin/lpr*. KDEPrint supports all the usual Linux printing subsystems, with an emphasis on CUPS.

The configuration window has a row of icons on the top. If you select Administrator Mode for this module, all the icons are available except for features you don't have installed.

Otherwise, you can select only certain icons. From left to right, the icons are:

- Add printer/class
- Add special (pseudo) printer
- Start/Stop printer

- Enable/Disable job spooling
- Set as local default
- Set as user default
- Remove
- Configure
- Test printer
- Printer tools
- Restart server
- Configure server
- Configure manager
- Refresh view
- Show/Hide printer details
- View
- Orientation

- Toggle printer filtering

Below the icons is a large window that lists all available printers or pseudo-printers. A pseudo-printer lets you send print output somewhere other than to a printer, such as a file or a fax. At the bottom of the window is a drop-down list of print systems that shows the print system in use. Below that is a window with four tabs. The tabs provide detailed information for the printer currently highlighted in the printer list. The Information tab shows

information such as the type, state, and location of the printer. The Jobs tab lists current jobs and their status. The Properties tab displays current property settings and lets you change them. The Instances tab shows individual printers and lets you configure them.

18.3.11 Web Browsing

The Web Browsing modules let you configure the Konqueror web browser.

18.3.11.1 Cache

The Cache module controls the use of a local cache to save copies of web pages you have visited. Using a cache lets Konqueror display pages without going back to the remote web site, which saves time and Internet access at the cost of using disk space and not always seeing the current version of a page. If you use the cache, you can always press Konqueror's Reload button to get the latest

version.

The Use Cache button is checked by default. Uncheck it if you do not want any pages stored in a cache on your system. The Policy section controls how up to date the cache is kept and whether to read first from the cache or the Web. Check one of these options:

Keep cache in sync

> Before displaying a page from the cache, verify that it is still valid and, if not, reread it from the remote site.

Use cache if possible

> Display a page from the cache if it is there, otherwise read it from the remote site.

Offline browsing mode

> Display a page from the cache if it is there, otherwise do not display it. Do not attempt to reread the page from the remote site.

In the "Disk cache size" box, enter

the cache size you want. Click the Clear Cache button at any time to empty the cache.

18.3.11.2 Cookies

The Cookies module sets Konqueror's cookie policy and lets you manage individual cookies. This module has two tabs: Policy and Management.

On the Policy tab, check the Enable Cookies box to enable web sites to store cookies on your system. Then check one of the options in the Default Policy section:

Ask for confirmation before accepting cookies

> Konqueror asks you to confirm your acceptance of every cookie. Each time a web site wants to store a cookie, a window pops up asking if you want to accept the cookie. At that time, you can choose to apply your response to just that cookie, all cookies from the domain, or all cookies.

Accept all cookies by default

> Konqueror accepts and stores all cookies. If you select this policy, you will never be asked for confirmation. You can go to the Management tab later to remove cookies you don't
>
> want.

Reject all cookies by default

Konqueror refuses to accept any cookies. Selecting this policy can result in your not being able to use certain web sites that require you to accept their cookies. You can change this policy later for all cookies or for a particular domain.

The Domain Specific Policy section lets you override the default policy for cookies from certain domains. This section lists domains for which you have specifically chosen to accept or reject cookies.

Click the New button to add a domain. Click on a domain in the list to change the policy for that domain or to delete it from the list.

The Delete All button deletes the entire list. The possible policies are Accept, Reject, and Ask.

The Management tab lets you manage cookies stored by Konqueror. The top portion lists the cookies hierarchically by domain and host.

Select a domain and click Delete to delete all cookies from that domain, or select an individual cookie from the Host column and click Delete to delete just that cookie. If an individual cookie is selected, the Cookie Details portion of the tab shows the content, expiration date, and other information for that cookie. The Delete All button deletes all cookies stored by Konqueror, and the Reload List button reloads the list of

stored cookies if more have been saved since you selected this module.

18.3.11.3 Enhanced Browsing

The Enhanced Browsing module controls the use of Internet keywords and web shortcuts by Konquerortwo features designed to make web browsing easier. Click the Enable Internet Keywords box to enable the use of Internet keywords. This lets you enter words such as company names without having to enter a full URL. For example, the use of Internet keywords would let you enter

oreilly instead of the full URL http://www.oreilly.com. If you enable Internet keywords, you can also use the Fallback Search Engine drop-down list to pick the search engine to use if what you entered isn't found as an Internet keyword.

Check the Enable Web Shortcuts box to enable the use of web shortcuts. These are shortcuts that are not known Internet keywords, but have much the same purpose. A predefined list of shortcuts is provided. For example, *gg* is a shortcut for Google. This means that you can enter *gg:KDE*

anywhere a URL is expected, and KDE searches Google for references to KDE. You

can add your own shortcuts by clicking Add, or select an entry in the list and click Change to change its associated shortcuts or Delete to remove it from the list. The character sequence **\{@}** in a shortcut causes the shortcut to prompt the user for a string (such as a term in a query) and to plug the user's string into the shortcut at

that point.

18.3.11.4 Konqueror Browser

This module configures settings used for Konqueror's web browsing functionality. It has five tabs: HTML, Appearance, Java, JavaScript, and Plugins.

On the HTML tab, checking the box "Enable completion of forms" causes Konqueror to store your responses

as you fill in forms and automatically fill in the same information in other forms. Move the "Maximum

completions" slider to set the maximum number of

responses Konqueror will remember. Check the box

"Change cursor over links" to have

Konqueror display a different cursor when the pointer is over a link.

The link cursor is a hand with a finger pointing to the link. If you check the box "Right click goes back in

history," right-clicking on a web page acts like the

Back button and takes you back a page. Check the box

"Automatically load images" to have

Konqueror load embedded images as it loads a web page. Check the box "Allow automatic delayed

reloading/redirecting" to allow Konqueror to honor

reload or redirect requests from the remote host. Choose an option from the Underline Links drop-down box to determine if and when links should be underlined. Select Enable to always underline links, Disable to never underline them, or Only on Hover to underline links only when the mouse pointer is directly over the link. Finally, you can enable or disable animations, or set them to be shown only once.

The Appearance tab sets fonts, font sizes, and the default encoding.

Use the top slider to set the minimum font size and the lower slider to set the medium font size. Then specify your preferred fonts. The text boxes let you set standard, fixed, serif, sans serif, cursive, and fantasy fonts. Use the Default Encoding drop-down box to set the default language encoding. Just above this box, you can set a font size adjustment for the encoding.

The Java tab controls whether you will permit web pages to run Java applets. This tab lets you turn Java on globally, or set a policy on a host- or domain-specific basis. You can also turn Java on globally and then selectively turn it off for certain hosts or domains. To set policy by domain, click the Add button to specify a domain. A dialog box appears for you to enter the host or domain name and specify a policy of Reject or Accept. If you change your mind, you

can click the Change or Delete button to change the policy or delete the entry.

If you enable Java globally, you can also set the following Java Runtime Settings.

Show Java Console

Open a Java console window for the use of Java applets.

Use Security Manager

Run a security manager that prevents Java applets from accessing your filesystem or perfoming other insecure actions. Normally, you should leave this box checked to protect your system.

Shutdown Applet Server when inactive

Shut down the Java Applet Server when no Java applet is running. This frees up resources when no applet is running, but causes a new applet to take slightly longer to start. If you choose this option, you can also move the Applet Server Timeout slider to set the number of seconds with no applet activity before the server shuts down.

Path to java executable, or "java"

Enter the path to the Java executable in the text box, or *java* to have Konqueror find Java in your path.

Additional Java Arguments

Enter any arguments for the Java virtual machine in this text box.

KDE does not implement any Java functionality itself; it relies on an external Java virtual machine. Therefore, you must already have Java installed on your system in order for these settings to have any effect.

The JavaScript tab controls whether you will permit web pages to run JavaScript code. Check the box to enable JavaScript globally, or set a policy on a host- or domain-specific basis, as you did for Java. As with Java, you can turn JavaScript on globally and off selectively for certain hosts or domains. Finally, set the policy you want for JavaScript web pop-ups. The possible policies are:

Allow

Allow all JavaScript pop-ups.

Ask

Ask for permission before displaying a JavaScript pop-up.

Deny

Don't allow any JavaScript pop-ups.

Smart

Allow only JavaScript pop-ups that have been activated by the user with a mouse click or keyboard operation.

On the Plugins tab, check Enable Plugins Globally to allow the use of plug-ins.

18.3.11.5 Netscape Plugins

The Konqueror browser can use Netscape plug-ins to display various types of web content. The Netscape Plugins module lets you tell Konqueror where to find the plug-ins on the system.

The Scan tab contains a list of directories for Konqueror to scan when looking for Netscape plug-ins. Initially, it contains a list of directories where plug-ins are typically found. You can add or delete entries from the list with the New and Remove buttons, or move around in the list with the Up and Down buttons. Or you can simply click on the entry you want with the first mouse button. Click the Scan for New Plugins button to have Konqueror rescan the directories in the list. You can also check the box labeled "Scan for

new plugins at KDE startup" to run the scan each

time KDE starts.

The Plugins tab contains a list of plug-ins that Konqueror found when it scanned the directories listed on the Scan tab. Check the box labeled "Use artsdsp to pipe plugin sound through

aRts" if you want to use the analog real-time

synthesizer to play sound.

18.3.11.6 Proxy

If your system is behind a

firewall, you may need to use a proxy server for HTTP and FTP services. The Proxy module lets you enable the use of a proxy and configure it. Check the Use Proxy box to enable the use of a proxy server. The remaining options are:

Automatically detected script file

Konqueror will automatically find the setup configuration script file.

Specified script file

Specify the location of the script file, and enter the path to the file in the Location text box.

Preset environment variables

Check this box if your system has preset environment variables such as **$HTTP_PROXY** for graphical and nongraphical applications to share proxy configuration information, and click the Setup button to provide the environment variable names.

Manually specified settings

Provide the proxy server locations manually, and click the Setup button to provide the information.

Prompt as needed

Konqueror will prompt as needed for login information.

Use automatic login

Check this box if you have login information set up for your proxy server in the file

$KDEHOME/share/config/kionetrc.

18.3.11.7 Stylesheets

The Stylesheets module contains settings for the use of CSS

(Cascading Style Sheets) in Konqueror. On the General tab, choose the stylesheet. Check "Use default

stylesheet" to use the Konqueror default; check

"Use user-defined stylesheet" to

specify a different stylesheet, and enter the path to the stylesheet in the text box. Or check "Use accessibility

stylesheet defined in `Customize'

tab" and go to the Customize tab to define the

stylesheet.

On the Customize tab, select a font family and check the box if you want to use that family for all text, overriding custom font settings. Select a base font size and check the box if you want to use that font size for all elements, overriding custom font size settings. Select a color scheme: Black on White, White on Black, or Custom. If you select Custom, also select a background and foreground color by clicking on the Background and Foreground color buttons. You can use the same color for all text, overriding any custom color settings. Check the box "Suppress

images" if you don't want any

images automatically displayed, "Suppress background

images" to prevent display of background images.

Click the Preview button to see the effect of your choices.

18.3.11.8 User Agent

The User Agent module allows you to specify the user-agent string reported by the browser for sites that may not recognize Konqueror, or anything that's not a major commercial web

browser. Check the box "Send browser

identification" to send information about your

browser; it is recommended that you leave this checked for the use of host sites that customize the page they send depending on the browser. The user-agent string contains the name and version number of the client program making the request. The default for Konqueror at the time of this writing is

"Mozilla/5.0(compatible;Konqueror/3)".

Check the boxes below the default identification to customize the information sent; the example is updated to reflect your choices. The options are:

Add operating system name

Adds the name of your operating system to the identification; e.g., Linux.

Add operating system version

If you checked the box to add the operating system name, you can also check this box to add the version.

Add platform name

Adds the name of your platform; e.g., X11.

Add machine (processor) type

Adds information about your processor; e.g., i586.

Add your language setting

Adds your language; e.g., en_US.

You can set the identification string individually for certain hosts or domains by clicking the New button. Enter the host or domain name in the dialog box that opens, and select the identity from the drop-down box. A description appears in the Alias box. Click the OK

button to add the entry to the list. The Update list box updates the list of identities that are shown in the drop-down list. You can also change and delete individual entries in the list, or delete all of them at once with the Delete All button.

Chapter 19. An Alternative Window Manager: fvwm2

Among the most appealing characteristics of a Linux system are its flexibility, its independence from industry-dominating standards, and the degree of control a user has over his own working environment. Most flavors of Linux come with a default desktop environment replete with handy tools and menus and a consistent look and feel. The most widely used of these desktop environments are GNOME and KDE, the customization of which are detailed in Chapter 17 and Chapter 18, respectively. Both of these environments put a PC-like wrapper around what is basically a no-frills Unix-based system suitable for personal computers. For some people this is a good thing. But if you want a basic, functional desktop, and you're comfortable editing configuration files to

customize your environment, you might instead try the **fvwm2** window manager.

fvwm2 is the latest generation of a window manager called **fvwm**.

fvwm predates both GNOME and KDE as a program that can provide multiple virtual screens to expand your desktop real estate. The letters *vwm* stand for virtual window manager, but the original meaning of the *f* has been forgotten and has led to much speculation. In fact, among the latest group of

the program's developers are a number of cat lovers who claim the *f* stands for *feline*.

The first important concept you should understand in order to work with **fvwm2**

is that your desktop can be larger than the area of your screen. In fact, **fvwm2** allows you to have acres of desktop real estate in the form of virtual screens, or pages. In a typical default environment, you might have a single desktop composed of four virtual screens/pages arranged in a two-by-two grid.

You can run applications on any of the screen pages you want and navigate the entire desktop in a variety of ways. If the default environment doesn't suit you, you can specify a grid of any size you like. How about three screens across and two down? No problem.

And if that still isn't enough space for you, you can also have multiple desktops, each composed of multiple pages. You might use separate desktops for different applications or different projects. **fvwm2** provides the tools to navigate whatever space you design.

fvwm2 is also customizable in a vast number of other ways; the most significant are summarized in this chapter. What it all boils down to is maximum workspace and maximum flexibility.

```
xterm -geometry +50+0 &


xterm -geometry -0+0&


exec fvwm2
```

fvwm2 provides GNOME support as a compile-time option. As this is being written, the current stable version of fvwm2 is 2.4.15. The next major release, currently in development as the 2.5 series, will also support KDE compliance. In addition, fvwm2 now supports Xinerama as a compile-time option. Xinerama is a feature of recent versions of XFree86 that permits multiple monitors to be treated logically as a single monitor (of course, each of these monitors needs its own video card).

19.1.1 Configuration Files

The key to how fvwm2 works is the configuration file it reads at startup or restart. The systemwide

configuration file is called *system.fvwm2rc* and usually lives in the directory */etc/X11/fvwm2*. The typical *system.fvwm2rc* file that gets distributed should create a simple but perfectly workable environment. We'll take a look at one in the next section. There's no guarantee that the file on your system will create the same layout, but you'll get the idea.

If you want to customize fvwm2 to suit your needs, you need to make a copy of *system.fvwm2rc* called *.fvwm2rc* and put it in your home directory. This personal configuration file takes precedence over the systemwide file. You can edit your *.fvwm2rc* file to adapt the window manager to your needs.

There are a few simple rules for editing your *.fvwm2rc* file. First, any line that begins with a pound sign (#) is a comment (i.e., is not interpreted as part of the window manager definition). Second, a plus sign (+) at the beginning of a line means to repeat the first terms from the previous line. Section 19.5.2, later in this chapter, illustrates the use of this syntax. The final thing to keep in mind is that life will be simpler if you weave your own definitions into the file, respecting its current contents and their order.

So, for instance, if you decide to define some function keys, put your new lines in the section of the file that already deals with keys.

In terms of fvwm2 customization, there's some good news and some bad news. The good news is that you can make an extraordinary number of changes to the way fvwm2 looks and operates. That's also the bad news. The window manager has dozens of configuration options, many very handy and easy to use, others complex and even arcane. The sum total can make the configuration file syntax daunting to anyone who isn't accustomed to serious tinkering. In fact, you could get dizzy considering the possibilities.

The fvwm2 manpage gives all the configuration options and illustrates their use; you may also want to consult the manpages for the fvwm2 modules, introduced in the next section. The web site http://www.fvwm.org is the definitive source for fvwm2 documentation, news, source code, and updates.

This chapter will help you cut to the chase in performing some of the more basic and useful

customizations, as well as some tricky but handy upgrades.

19.1.2 A Modular Approach

fvwm2 has been designed to allow the interested user to devise new components, known as modules. A typical module is the Pager (FvwmPager), which provides a map of the desktop space and a way to navigate it, as we'll see a little later in the chapter. The Pager is a default module in just about any desktop environment.

FvwmWinList is another useful module. Though not as ubiquitous as the Pager, it is just as useful. The module displays the WinList, which is a small window that provides a list of all the windows running on all pages of all desktops. The WinList is another navigation tool, allowing you to switch the pointer focus to any application you have running and switch the screen view so that you can use that application. We'll talk more about this later.

A module is actually a separate program from fvwm2 but works in concert with it, passing commands to be executed to the window manager. Most

configurations of fvwm2 have a Root menu with an FvwmModules submenu from which you can start certain of these programs. (Naturally, the list of modules on the menu is configurable.) You might also edit your *.fvwm2rc* file to run modules in other ways (when you type certain keys, when other events happen, etc.).

Since a module is a separate program, users can write their own modules without adversely affecting fvwm2. Note, however, that you must configure fvwm2 to start the module's process; you cannot start a module from the command line. Note also that while some modules, like the Pager, are intended to be used for the entire session, others simply perform a function and exit (e.g., RefreshWindow). Since modules are programs in their own right, many of them have their own manpages too.

Style "*" FocusFollowsMouse

Style "FvwmButtons" Icon toolbox.xpm, ClickToFocus

Style "xman" Icon xman.xpm, RandomPlacement, ClickToFocus

Style "xmag" Icon mag_glass.xpm, RandomPlacement, ClickToFocus

Style "XTerm" Icon xterm.xpm, SloppyFocus, IconBox -70 1 -1 -140

Style "rxvt" Icon term.xpm, SloppyFocus, IconBox -70 1 -1 -140

Module FvwmAuto 200

DeskTopSize 2x2

DeskTopSize 3x2

```
DeskTopSize 1x3

DeskTopSize 1x1

*FvwmButtons(2x2 Frame 2 Swallow(UseOld)
"FvwmPager" "Module FvwmPager 0 0")

*FvwmButtons(1x2 Frame 2 Swallow(UseOld)
"FvwmPager" "Module FvwmPager 0 1")

*FvwmButtons(1x2 Frame 2 Swallow(UseOld)
"FvwmPager" "Module FvwmPager 0 3")

*FvwmButtonsGeometry 520x100-1-1

*FvwmButtons(Frame 2 Padding 2 2
Container(Rows 2 Columns 5 Frame 1 Padding

10 0))

*FvwmButtons(1x2 Frame 2 Swallow(UseOld)
"FvwmPager" "Module FvwmPager 0 1")

*FvwmButtons(3x2 Frame 2 Swallow
"FvwmIconMan" "Module FvwmIconMan")
```

*FvwmButtons(1x2 Frame 0 Container(Rows 2 Columns 2 Frame 0))

*FvwmButtons(Frame 2 Swallow(UseOld,NoHints,Respawn) "xbiff" 'Exec exec

xbiff -bg bisque3')

*FvwmButtons(Frame 3 Swallow(UseOld,NoHints,Respawn) "xclock" 'Exec exec

xclock -bg

bisque3 -fg black -hd black -hl black -padding 0 update 1')

*FvwmButtons(2x1 Frame 2 Swallow(UseOld,NoHints,Respawn) "xload" 'Exec exec

xload -bg bisque3 -fg black -update 5 -nolabel')

*FvwmButtons(1x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 1")

*FvwmButtonsGeometry 620x100-1-1

*FvwmButtons(2x2 Frame 2 Swallow "FvwmIconMan" "Module FvwmIconMan")

*FvwmButtons(2x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 1")

*FvwmButtons(1x2 Frame 0 Container(Rows 2 Columns 2 Frame 0))

Style "xbiff" NoTitle, Sticky, WindowListSkip, ClickToFocus

Style "*lock" NoTitle, NoHandles, Sticky, WindowListSkip, ClickToFocus

Style "Fvwm*" NoTitle, Sticky, WindowListSkip

Style "xpostit" Sticky

Style "xpostit" Sticky, WindowListSkip, ClickToFocus

```
xterm -name mailwindow &

Style "mailwindow" Sticky

Style "bigxterm" StartsOnPage 1 0 1

xterm -name bigxterm &

xterm -xrm '*Page: 1 0 1' &

xterm -geometry +1200+1200 &

EdgeResistance 250 10

EdgeResistance 500 10

EdgeResistance 100 10

Full Paging ON EdgeScroll 100 100


All Paging OFF EdgeScroll 0 0


Horizontal Paging Only EdgeScroll 100 0
```

Vertical Paging Only EdgeScroll 0 100

Partial Paging EdgeScroll 50 50

Full Paging & Edge Wrap EdgeScroll 100000 100000

EdgeScroll's two parameters specify the percentage of a page to scroll when you reach the border of the page. The first parameter is for horizontal moves, the second for vertical. If the horizontal and vertical percentages are multiplied by 1000, scrolling will wrap around at the edge of the desktop. EdgeScroll 100000 100000 will wrap for both horizontal and vertical moves.

Rather than rely on menu items like these, you can make any of these options the default behavior by putting the EdgeScroll variable on its own line in your *.fvwm2rc* file. See the fvwm2 manpage and check out the system.fvwm2rc file for guidance.

Note that none of these variations lets you scroll from one desktop to another. The next section shows how to configure some keyboard shortcuts to do just that.

19.3 Adding Keyboard Shortcuts

The *system.fvwmrc* file that we started with offers little in the way of keyboard shortcuts, or accelerators, for window management functions. But if you're one of those users who prefer to keep their hands on the keyboard and off the mouse as much as possible, you can easily define keys in your *.fvwm2rc* file to perform a variety of functions.

19.3.1 Navigating the Desktop

The Pager is a great tool for getting around one or more desktops, but many people hate using the mouse. You can configure certain keys and key combinations to let you move around in various ways.

Add the following lines to your *.fvwm2rc* file to set up key combinations to scroll one page in any direction on the desktop using Ctrl plus an arrow key. The view scrolls in the direction of the arrow.

Each definition uses the Key variable followed by:

1. The name of the key
2. The context (location) in which it must be typed
3. Any modifying keys that must also be held down
4. The action initiated by the key or key combination

Thus, in the following example, the first definition line says that pressing the left arrow key (Left) in any (A) context, while also holding down the Ctrl (C) key scrolls the screen one page to the left on the current desktop:

```
# Press arrow + Control in any context
# to scroll by one page in the direction of the
arrow
Key Left            A            C
Scroll -100 0
Key Right           A            C
Scroll +100 +0
Key Up              A            C
Scroll +0  -100
Key Down            A            C
Scroll +0  +100
```

Table 19-1 summarizes the functionality.

Table 19-1. Key combinations to change the page

Key combination	Moves view
Control, right arrow key	One page to the right
Control, left arrow key	One page to the left
Control, up arrow key	One page up
Control, down arrow key	One page down

The Scroll variable takes the same parameters as EdgeScroll, which was explained in the previous section. See the **fvwm2** manpage for more information. Note that the key combinations

we've defined let you get around a single desktop but won't let you advance to another desktop. We'll deal with that contingency later.

Here's another possible key binding. This one advances the view to every page in the desktop in order, and finally wraps back to the first page. You use the Tab key while holding down Control, again in any context. The definition line looks like: # Press Tab + Control in any context to scroll # by one page with wrap scrolling Key Tab A C Scroll 100000 0

Table 19-2 summarizes another page-changing combination.

Table 19-2. Another key combination to change the page

Key combination	Moves view
Control, Tab	To the next page in the desktop

Since application windows can straddle pages, there may be times when you want the screen to display a screenful other than a page proper. (You might also want to look at windows on two different pages at once.) The following shortcuts scroll the view one-tenth of a page at a time. Instead of Control, these shortcuts use the so-called Meta key. This is a symbolic namethe actual key that serves the Meta function varies from keyboard to keyboard. In many cases, the key labeled Alt serves as the Meta key. Here are the configuration file definition lines:

```
## Press arrow + meta key in any context
## to scroll by 1/10 of a page in the direction of
arrow
Key Left            A           M
```

```
Scroll -10 +0
Key Right           A           M
Scroll +10 +0
Key Up              A           M
Scroll +0   -10
Key Down            A           M
Scroll +0   +10
```

These lines establish the functionality outlined in Table 19-3.

Table 19-3. Key combinations to scroll the page by 1/10

Key combination	Moves view
Meta, left arrow key	One-tenth of a page to the left
Meta, right arrow key	One-tenth of a page to the right
Meta, up arrow key	One-tenth of a page up
Meta, down arrow key	One-tenth of a page down

If you have more than one desktop, you can also create shortcuts to move between desktops. The following two shortcuts let you go back and forth between desktops in a two-desktop environment: ## Press Control + Return in any context ## to scroll forward by 1 desktop Key Return A C Desk 1 1 1 ## Press Shift + Control + Return in any context ## to scroll back by 1 desktop Key Return A SC Desk -1 0 0

Table 19-4 summarizes these shortcuts.

Table 19-4. Key combinations to scroll to the next desktop

Key combination	Moves view
Control, Return	One desktop ahead
Shift, Control, Return	One desktop back

The second and third parameters to the Desk variable constrain the paging so that you can't page beyond the first or second desktops. (Theoretically, you can page outside the view of the Pager!) If you have more than two desktops, you will need to edit these definitions. See the **fvwm2** manpage for more about the Desk variable.

19.3.2 Moving the Pointer

The previous section outlines some keyboard shortcuts you can define to scroll the page view. But you can also define shortcuts to move the position of the pointer on the screen. If you prefer using the keyboard to the mouse, these shortcuts can come in handy. They employ the CursorMove variable, also described on the **fvwm2** manpage.

The keyboard accelerators in the first group move the cursor symbol one-tenth of a screen at a time. The first definition line says that pressing the left arrow key in any (A) context, while also holding

down the Shift (S) and Meta (M) keys, moves the cursor one-tenth of a page in the direction of the arrow:

```
## Press Shift + Meta + arrow in any context
## to move the pointer by 1/10 of a page in
direction of arrow
Key Left            A            SM
CursorMove -10  +0
Key Right           A            SM
CursorMove +10  +0
Key Up              A            SM
CursorMove +0   -10
Key Down            A            SM
CursorMove +0   +10
```

Table 19-5 summarizes the commands.

Table 19-5. Key combinations to move the pointer by 1/10 of the page

Key combination	Moves pointer
Meta, Shift, left arrow key	One-tenth of a page to the left
Meta, Shift, right arrow key	One-tenth of a page to the right
Meta, Shift, up arrow key	One-tenth of a page up
Meta, Shift, down arrow key	One-tenth of a page down

If you want as much control moving the pointer with keystrokes as you do moving it by hand, you can specify shortcuts to move it a mere one percent of a page at a time:

```
## Press Shift + Control + arrow in any context
## to move the pointer by 1% of a page in
direction of arrow
Key Left            A            SC
CursorMove -1 0
Key Right           A            SC
CursorMove +1 +0
Key Up              A            SC
CursorMove +0 -1
Key Down            A            SC
CursorMove +0 +1
```

Table 19-6 summarizes the commands.

Table 19-6. Key combinations to move the pointer by 1 percent of the page

Key combination	Moves pointer
Shift, Control, left arrow key	One percent of a page to the left
Shift, Control, right arrow key	One percent of a page to the right
Shift, Control, up arrow key	One percent of a page up

Key combination	Moves pointer
Shift, Control, down arrow key	One percent of a page down

19.3.3 Menu and Window Manipulation

So far we've limited our keyboard shortcuts to scrolling the view and moving the pointer. But you can create keyboard bindings for any window manager function.

Here are some sample bindings to perform simple window operations and to display a few menus:

```
# Keyboard accelerators
Key F1          A       M       Iconify
Key F2          A       M       Move
Key F3          A       M       Resize
Key F4          A       M       Popup "RootMenu"
Key F5          A       M       Popup "Misc-Ops"
Key F6          A       M       Popup "Utilities"
Key F7          A       M       Popup "Module-
Popup"
Key F10         A       M       Restart fvwm2
Key F12         A       SM      Close
```

These are just sample bindings; you may want to set up your own keyboard shortcuts to do entirely different things. But these bindings illustrate some of the possibilities, as well as potential problems.

In our sample definition lines, the first binding specifies that if you press the F1 function key while holding down the Meta (M) key, with the pointer in any (A) context, the focus window is iconified (or

deiconified). Meta-F2 lets you initiate moving the focus window, while Meta-F3 starts a resize operation.

Note that if you've adopted the keyboard bindings to move the pointer (as described in the previous section), you can perform the move and resize operations entirely with keystrokes. For example, use Meta-F2 to begin a move, drag the window outline by moving the pointer symbol using the appropriate keyboard shortcuts, then press the Return key to complete the operation.

In addition, we've set up function keys to pop up four different menus, the contents of which are predefined in the *.fvwm2rc* file. Once a menu is popped up, you can use the up and down arrow keys to highlight items on the menu, right and left keys to move down and up through submenus (cascading menus), the Return key to select an item, and Esc to pop down the menu without making a selection.

Because we're doing a lot of tinkering with **fvwm2** customization, we have set up Meta-F10 to restart the window manager. This is much faster than bringing up menus.

We've also created a key combination to close the focus window: Shift-Meta-F12. Certainly it's handy to be able to get rid of a window with a keyboard shortcut, but you don't want it to be too easy or you may do it by mistake. Having an extra modifying key (Shift) and using the very last function key (F12) require you to act deliberately in closing a window using this method.

```
AddToMenu RootMenu "Root Menu" Title +
"XTerm" Exec exec xterm

+ "Rxvt" Exec exec rxvt

+ "" Nop

+ "Remote Logins" Popup Remote-Logins + "" Nop

+ "Utilities" Popup Utilities + "" Nop

+ "Fvwm Modules" Popup Module-Popup + "Fvwm
Window Ops" Popup Window-Ops + "Fvwm Simple
Config Ops" Popup Misc-Ops + "" Nop

+ "Refresh Screen" Refresh

+ "Recapture Screen" Recapture + "" Nop

+ "Exit Fvwm" Popup Quit-Verify
```

You use the AddToMenu variable to create a menu. The first parameter that AddToMenu takes is the name of the menu, in this case RootMenu. The menu name is used to reference the menu elsewhere in the configuration file (e.g., to specify a key binding to

pop up the menu). Note that the AddToMenu variable and the menu name are automatically repeated on each line of the menu definition, as indicated by the plus sign.

Each line of the definition creates a line on the menu as seen by the user; types of lines include the menu title, a menu item proper, a blank line, and a separator. The third component of each line specifies the text that appears on that line. The fourth component specifies the window manager function to be performed.

The first line of our example specifies the menu title. Lines with empty text fields ("") together with the Nop ("No operation") function create divider lines on the menu.

The Popup function is worth looking at more closely. Popup specifies that a menu is displayed; the menu name is given as an argument to Popup. When Popup is invoked from another menu, it creates a submenu (or cascading menu). This sample Root menu definition has six submenus, named Remote-Logins, Utilities, Module-Popup, Window-Ops, Misc-Ops, and Quit-Verify. These menus would also be defined

using the AddToMenu command elsewhere in the configuration file.

You can start from the default menus in the system.fvwm2rc file and use the fvwm2 manpage to modify the existing menus or create your own. It is simple to replace definition lines in the template menus, and not much more difficult to write a new entry or even a menu from scratch.

You can also change how the menus are displayed. Perhaps you don't want a bunch of cascading menus off the Root menu. In the previous section we set up some function keys to display certain menus. That's one option. You might instead specify pointer buttons to display various menus. In a typical default, the first pointer button displays the Root menu and the second displays the Window Ops menu. But since most Window Ops functions (e.g., Move, Resize, Iconify) are available using the pointer directly on parts of a window, you may instead choose to have the second pointer button display another menu (e.g., Utilities).

AddToFunc InitFunction "I" Module FvwmButtons + "I" exec xsetroot -mod 2 2 -fg \#554055 -bg \#705070

+ "I" Module FvwmWinList AddToFunc RestartFunction "I" Module FvwmButtons + "I" exec xsetroot -mod 2 2 -fg \#554055 -bg \#705070

+ "I" Module FvwmWinList

Mouse 3 R A Module FvwmWinList Transient

############################FvwmWinList##########################

*FvwmWinListBack #908090

*FvwmWinListFore Black

FvwmWinListFont -adobe-helvetica-bold-r--*-10-*-*-*-*-*-*-*-*

*FvwmWinListAction Click1 Iconify -1,Focus
*FvwmWinListAction Click2 Iconify
*FvwmWinListAction Click3 Module "FvwmIdent" FvwmIdent *FvwmWinListUseSkipList

*FvwmWinListGeometry +0-1

Style "Fvwm*" NoTitle, Sticky, WindowListSkip

This line specifies that all modules (including the WinList) have no titlebars, are sticky, and do not appear on the WinList. In the case of the FvwmWinList module, having it appear as an entry on itself would be more than a little confusing.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal featured on the cover of *Linux in a Nutshell* is an Arabian horse. Known for its grace and intelligence, the Arabian is one of the oldest breeds of horse, with evidence of its existence dating back 5000 years. The Arabian was instrumental as an ancestor to other popular breeds, most notably the Thoroughbred in the 17th and 18th centuries. Possibly one of the more characteristic horse breeds, the typical Arabian has large expressive eyes and nostrils, small ears, and a short, sturdy back. Its stamina suits it particularly well for endurance riding, where the breed dominates the sport. Its wonderful temperament makes the Arabian an all-around favorite riding horse in North America, although it also can be found in more specialized competitions such as dressage, jumping, and reining.

Emily Quill was the production editor and copyeditor for *Linux in a Nutshell, Fourth Edition*. Derek Di Matteo, Claire Cloutier, Genevieve d'Entremont, Mary Brady, and Colleen Gorman provided quality control. Derek Di Matteo and Jamie Peppard provided production assistance. John Bickelhaupt wrote the index.

Edie Freedman designed the cover of this book, using a 19th-century engraving from the Dover Pictorial Archive. Emma Colby produced the cover layout with QuarkXPress 4.1 using Adobe's ITC

Garamond font.

David Futato designed the interior layout. This book was converted by Joe Wizda to FrameMaker 5.5.6 with a format conversion tool

created by Erik Ray, Jason McIntosh, Neil Walls, and Mike Sierra that uses Perl and XML technologies. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed. The illustrations that appear in the book were produced by Robert Romano and Jessamyn Read using Macromedia FreeHand 9 and Adobe Photoshop 6. The tip and warning icons were drawn by Christopher Bing.

The online edition of this book was created by the Safari production group (John Chodacki, Becki Maisch, and Madeleine Newell) using a set of Frame-to-XML conversion and cleanup tools written and maintained by Erik Ray, Benn Salter, John Chodacki, and Jeff Liggett.

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Y] [Z]

[**SYMBOL**]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

& (ampersand)

ex

2nd

logical AND

< > (angle brackets) ex

* (asterisk)

2nd

multiplication operator

@ (at)
ex

tcsh

\ (backslash)

! (bang)

ex

shell invocation, ftp

!~ (bang tilde), gawk

{ } (braces), sed command

[ ] (brackets)

2nd

^ (caret), gawk

: (colon)
ex editor

pattern search, expr

sed

:set command (ex)

tcsh null command

$ (dollar)

execute macro command, ftp

gawk

prompt

sed

. (dot)

?? (double question marks)

" " (double quotes)

= (equals)
ex

sed

\(\) (escaped parentheses)

/ (forward slash)
division operator

sed command

#! (hash bang)

tcsh shell

(hash mark)

2nd

in shell scripts

sed

tcsh shell

- (hyphen) and metamail command

- (minus operator)

% (modulo operator)

: (null command)

( ) (parentheses)

% (percent)

bash job ID argument

prompt

| (pipe)

2nd

as logical OR symbol

+ (plus) operator

fvwm2

? (question mark)

2nd

ftp help command)

> (redirection operator)

>> (redirection operator)

< (redirection operator)

<< (redirection operator)

; (semicolon)

' ' (single quotes)

~ (tilde)
bash

ex

gawk

sed

tcsh

` (backquote)

[SYMBOL]

[**A**]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

a command (sed)

abbrev command (ex)

account command (ftp)

aclocal command

ACPI (Advanced Configuration and Power Interface)

Ada programming language, compiling

add command (CVS)

addr2line command

address command (ex)

Address Resolution Protocol

[See ARP]
addresses (sed)

addsuffix variable (tcsh)

AddToMenu variable (fvwm2)

admin command (CVS)

2nd

Advanced Package Tool (APT)

Advanced Power Management

[See APM]
AFSUSER environment variable (tcsh)

afsuser variable (tcsh)

agetty command

alias command
bash

tcsh

alloc command (tcsh)

Alt-F1 through F12

Alt-F7

Alt-key (Emacs)

ampersand (&)

ex command

2nd

logical AND

ampm variable (tcsh)

and command (gawk)

angle brackets (< >) command (ex)

annotate command (CVS)

ANSI/VT100 terminal emulation, enabling (screen)

Apache, user authentication files, updating (htdigest)

APM (Advanced Power Management)
BIOS driver

BIOS information, displaying (GRUB)

KDE

apmd command

append command
ex

ftp

applets

apropos command

APT (Advanced Package Tool)

apt command

apt-cdrom command (Debian)

apt-config command (Debian)

apt-extracttemplates command (Debian)

apt-ftparchive command (Debian)

apt-get command (Debian)

2nd

apt-sortpkgs command (Debian)

ar command

arbitrary-precision arithmetic, performing with bc

arch command

archives

copying

creating (shar)

creating and restoring (tar)

in CVS

maintenance (ar)

args command (ex)

argv variable (tcsh)

arithmetic operators (expr)

ARP (Address Resolution Protocol)

2nd

3rd

4th

[See also RARP]
ifconfig and

arp command

as command

ascii command (ftp)

asort command (gawk)

ASs (autonomous systems)

assembly language, generation of object files

asterisk (*)

2nd

multiplication operator

at (@) command
ex

tcsh

at command

atan2 command (gawk)

atd command

atq command

atrm command

audio media commands

audiosend command

Author keyword (RCS)

autoconf command

2nd

[See also autoheader command]
aclocal command and

autocorrect variable (tcsh)

autoexpand variable (tcsh)

autoheader command

autolist variable (tcsh)

autologout variable (tcsh)

automake command

autonomous systems (ASs)

autoreconf command

autoscan command

autoupdate command

awk scripting language

[See gawk scripting language]

[SYMBOL]

[A]

[**B**]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

b command (sed)

background module (KDE)

backslash (\)

backslash_quote variable (tcsh)

backspace key

Emacs

badblocks command

bang (!)

ex

shell invocation, ftp

bang tilde (!~), gawk

banner command

basename command

bash shell

2nd

3rd

arithmetic expressions and operators

built-in commands

built-in shell variables

command history

command substitution

fc command

line-edit mode

command-line editing

Emacs mode

vi mode

commands, usage

configuration files

features

filename metacharacters

invocation, options and arguments

job control

pattern matching

prompt

prompt variables

quoting

redirection, usage

special files

syntax

tcsh, compared to

variables

variable substitution

vi mode editing

batch command

[See also at command]2nd

bc program

function keywords

identifiers

input-output keywords

math library functions

operators and symbols

statement keywords

bdelete command (ex)

bell command (ftp)

bg command
bash

tcsh

2nd

biff command

binary command (ftp)

binary files, encoding in ASCII (uuencode)

BIND (Berkeley Internet Name Domain)

bind command (bash built-in)

binding (ypbind)

bindkey command (tcsh)

biod daemon

BIOS information, displaying (displayapm)

bison command

blocklist command (GRUB)

blocklists (GRUB)

Boleyn, Erich

Bonobo library

boot command (GRUB)

boot loaders

chainloaders

GRUB

[See GRUB]
LILO

[See LILO]
Loadlin

boot methods

boot process

boot sector

boot-time kernel options

/boot/grub.conf

[See GRUB, conguration file]
bootable CDs, difficulties of making

bootp command (GRUB)

bootpd command

bootpgw command

bootptest command

Bourne shell

braces { }, sed command

brackets [ ]

2nd

branching (CVS)

break command
bash built-in

gawk

tcsh

breaksw command (tcsh)

broadcast messaging (wall)

browsers

[See web browsers]
buffer command (ex)

buffers

buffers command (ex)

built-in shell commands

built-ins command (tcsh)

builtin command (bash)

bunzip2 command

[See bzip2 command]
bye command
ftp

tcsh

bzcat command

[See bzip2 command]
bzcmp command

bzdiff command

bzgrep command

[See also grep command]2nd

bzip2 command

bzip2recover command

[See bzip2 command]
bzless command

[See also less program]2nd

bzmore command

[See also more program]2nd

[SYMBOL]

[A]

[B]

[**C**]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

c++ command

[See g++]
c++filt command

C++ programming language
c++filt command and function overloading

compiling (gcc)

debugging (gdb)

c command (sed)

C programming language
compiling (gcc)

debugging (gdb)

imake command

m4 command

preprocessor command (cpp)

C shell command interpreter (csh)

C-s command (Emacs)

cal command

Caldera package manager

[See RPM]
cardctl command

cardmgr command

caret (^), gawk

case command
bash

ftp

tcsh

cat command

GRUB

catalog variable (tcsh)

cc command

[See gcc]
cd command
bash built-in

ex

ftp

tcsh

CD-ROMs
bootable, difficulties of making

ejecting

volume names, displaying (volname)

CDDA (Compact Disc Digital Audio), WAV format, converting to

cdda2wav command

cdparanoia command

cdpath variable (tcsh)

cdrdao command

cdrecord command

cdup command (ftp)

center command (ex)

cfdisk command

chage command

chainloader command (GRUB)

2nd

chainloaders

chains

change command (ex)

character encoding conversions (iconv)

chat command

chdir command (tcsh)

checkout command (CVS)

checkpc command

chfn command

chgrp command

chmod command

ftp

chown command

chpasswd

chroot command

chsh command

chvt command

ci command (RCS)

CIDR (Classless Inter-Domain Routing)

cksum command

Classless Inter-Domain Routing (CIDR)

clear command

clock commands

clock, setting (hwclock)

close command
ex

ftp

gawk

cmp command

GRUB

co command (RCS)

col command

colcrt command

colon (:)
:set command

ex editor

pattern search, expr

sed

tcsh null command

color command (GRUB)

color variable (tcsh)

colorcat variable (tcsh)

column command

COLUMNS environment variable (tcsh)

comm command

command command (bash)

command substitution
bash

2nd

gawk

tcsh

2nd

command variable (tcsh)

command-line editing
bash

tcsh

commands
beginnerÕs guide

audio media commands

file management commands

file storage commands

miscellaneous commands

printing commands

program maintenance commands

searching commands

Debian Package Manager

executing as superuser (sudo)

executing with arguments from standard input (xargs)

finding binary, source, and manpages for (whereis)

Linux

options to

listing pathnames of executed files for (which)

running repeatedly (watch)

commit command (CVS)

comp.os.linux newsgroups

comparison commands
bzcmp

bzdiff

cmp

compgen command (bash)

compilers

[See also gcc]
Ada

bc

[See bc program]
link editor command (ld)

compl command (gawk)

complete command
bash

tcsh

complete=enhance variable (tcsh)

compress command

concatenation command

[See cat command]
Concurrent Versions System

[See CVS]
configfile command (GRUB)

configuration files
bash

bootpd

fvwm2

GRUB

inetd.conf

LILO

rpm command

tcsh

vi

2nd

configuration scripts, generating

context menus

continue command
bash

gawk

tcsh

continue variable (tcsh)

continue_args variable (tcsh)

copy command (ex)

copying files

copyleft

CORBA (Common Object Request Broker Architecture)

corner panels (GNOME)

correct variable (tcsh)

cos command (gawk)

country-specific settings, displaying

cp command

cpio command

cpp command

[See also gcc]2nd

cr command (ftp)

CRCs (cyclic redundancy checks), performing

cron command

crontab command

crontab files

csh command

[See also tcsh command]2nd

csplit command

ctags command

ctime format

Ctrl-Alt-F1 through F12

Ctrl-key (Emacs)

commands

Ctrl-Z command
bash

tcsh

CUPS (Common Unix Print System)

cut command

CVS (Concurrent Versions System)

2nd

access methods

admin command

2nd

security risks

administrator reference

CVSROOT environment variable

branching

checkoutlist file

client global options

commands
for administrators

format of

for users

commitinfo file

common client options

config file

configuring

conflicts and merging

cvsignore file

CVSROOT directory

administrative file variables

files

cvswrappers file

environment variables

global options

global server option

gotchas

history file

importing from RCS

importing into CVS

2nd

code snapshots

from PVCS

from SCCS

locking model

loginfo file

merging model

modules file

notify file

passwd file

password setup

rcsinfo file

readers file

repositories

creating

making changes

structure

repository locators

sandboxes

creating

CVS directory files

.cvsignore files

.cvswrappers files

sharing

structure

security

tagging

taginfo file

user reference

users file

verifymsg file

writers file

cvs command

cwd variable (tcsh)

cyclic redundancy checks (CRCs), performing

[SYMBOL]

[A]

[B]

[C]

[**D**]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

d command (sed)

D command (sed)

daemon commands

daemons

NFS daemons

routing daemons

xinetd

DAO (disk-at-once) mode

date command

format

Date keyword (RCS)

dbm files

creating (makedbm)

dd command

deallocvt command

Debian Package Manager

2nd

commands

files

2nd

package flags

package priorities

package/selection states

shell and Perl scripts

debug command
ftp

GRUB

debugfs command

debugging tools
addr2line

gdb command

gprof

patch command

strace command

declare command (bash)

default command (tcsh)

del key

delete command
ex

ftp

gawk

dependencies

dependency files, creating

depmod command

.desktop files (KDE)

devdump command

device command (GRUB)

device images, displaying

devices
hard disks as

swapping, paging
disabling for (swapoff)

enabling for (swapon)

dextract variable (tcsh)

df command

dhcp command (GRUB)

diff command

CVS

diff3 command

dig command

dip command

dir command

[See also ls command]2nd

ftp

dircolors command

directories
creating (mdir)

displaying verbosely (vdir)

listing contents of
dir

ls

dirname command

[See also basename command]2nd

dirs command
bash

tcsh

dirsfile variable (tcsh)

dirstack variable (tcsh)

disconnect command (ftp)

disk buffers, flushing to disk (update)

disk space, measuring

disk usage, displaying
du

MS-DOS

disk-at-once (DAO) mode

disks
ext2 filesystem, formatting for

MS-DOS formatting
mformat

mkdosfs

disown command (bash)

DISPLAY environment variable (tcsh)

displayapm command (GRUB)

displaymem command (GRUB)

distributions

dmesg command

DNS (Domain Name Service)

servers, querying

dnsdomainname command

[See also hostname command]2nd

dnssec-keygen command

dnssec-makekeyset command

dnssec-signkey command

dnssec-signzone command

do command (gawk)

document formatting commands
groff

gs

documentation, accessing
info command

man command

manpath command

whatis command

dollar ($)

execut macro command, ftp

gawk

prompt

sed command

Domain Name Service (DNS)

domain names

researching (whois)

domainname command

2nd

domains

dosfsck command

2nd

[See also fsck command]
dot (.)

double quotes (\\x22)

dpkg command (Debian)

2nd

dpkg-deb command (Debian)

2nd

dpkg-query command (Debian)

dpkg-split command (Debian)

drawer (GNOME)

dselect command (Debian)

2nd

dspmbyte variable (tcsh)

du command

dual booting

Linux and Windows NT/2000/XP

dump command (GRUB)

dumpe2fs command

dumpkeys command

dunique variable (tcsh)

[SYMBOL]

[A]

[B]

[C]

[D]

[**E**]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

e2fsck command

2nd

[See also fsck command]
e2image command

e2label command

echo command

bash

tcsh

echo variable (tcsh)

echo_style variable (tcsh)

echotc command (tcsh)

ed editor

edge panels (GNOME)

EdgeResistance variable (fvwm2)

edit command
CVS

ex

edit variable (tcsh)

EDITOR environment variable
bash

2nd

tcsh

editors command (CVS)

egrep command

[See also grep command]2nd

eject command

ellipsis variable (tcsh)

else command (tcsh)

elvis text editor

elvtune command

Emacs editor

bindings, compared to vi bindings

buffers

command-line editing mode
bash

tcsh

commands

buffer manipulation

capitalization

centering

Ctrl-key commands

cursor movement

deletion

essential commands

file handling

help

2nd

indentation

macros

marking

Meta-key commands

paragraphs and regions

searching

special shell mode characters

stopping and undoing

summary by name

transposition

windows

word abbreviation

common problems

cutting and pasting in

Del or Backspace, fixing in

editing modes

kill and yank

point and mark

windows

embed command (GRUB)

enable command (bash)

encryption
keys, generating (ssh-keygen)

using SSH

end command (tcsh)

2nd

[See also foreach command]3rd

[See also while command]
end-of-file character (EOF)

endif command (tcsh)

endsw command (tcsh)

env command

environment variables

CVS

tcsh

EOF (end-of-file) character

equals (=)
ex

sed

etags command

[See also Emacs text editor]2nd

/etc/at.allow

/etc/at.deny

/etc/exports file

/etc/fstab file

/etc/hosts.deny file

/etc/hosts.allow file

/etc/inetd.conf

/etc/lilo.conf

[See LILO, configuration file]
/etc/rpmrc file

eval command
bash

tcsh

ex editor

2nd

3rd

command syntax

command-line options

commands
alphabetical summary

vi, entering from

:set command

exec command
bash

tcsh

exit command
bash

gawk

tcsh

exp command (gawk)

expand command

export command
bash

CVS

expr command

expressions, testing (test)

ext access method (CVS)

ext2 (Second Extended Filesystem)

debugging

disk formatting

e2fsck command

ext3, converting to

tuning (tune2fs)

ext3 (Third Extended Filesystem)

journaling level, specifying

potential booting issues

Extensible Filesystem (XFS)

eXternal Data Representation (XDR)

exusage command (ex)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[**F**]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

false command

Fast Lexical Analyzer Generator (flex) command

fc command (bash)

fdformat command

fdisk command

[See also cfdisk command]2nd

fetchmail command

fflush command (gawk)

fg command
bash

tcsh

fg job control command (tcsh)

fgconsole command

fgrep command

fignore variable (tcsh)

file archives, copying

file command

ex

file compression

gzexe command

gzip command

file management commands

chgrp

chmod

chown

compress

cp

cpio

csplit

file storage commands

file transfer commands
lftp

lftpget

filename expansion vs. pattern matching

files

access modes, changing (chmod)

archives, copying from or restoring to (tar)

attributes, listing

byte, word, and line counts, displaying (wc)

character encoding conversions (iconv)

checksums, calculating for (sum)

classification of (file command)

columnar output, formatting in

comparison
cmp command

comm command

compressed to uncompressed files (zcmp)

diff command

diff3 command

sdiff

zdiff command

copying

between Unix and MS-DOS partitions

creating or updating (touch)

crontab files

Debian package management

disabling for swapping, paging (swapoff)

display format options (hexdump)

duplicate lines, removing (uniq)

enabling for swapping, paging (swapon)

fgrep, searching with

finding binary, source, and manpages for (whereis)

first lines, displaying (head)

last lines, displaying (tail)

linking

2nd

listing (ls)

locking (lockfile)

merging (merge)

object files, displaying profile data for

overwriting to make unrecoverable

ownership, changing (chown)

printing (lpr)

[See also CUPS; LPRng line printer daemon]2nd

in reverse (tac)

reading compressed files to output (zcat)

renaming zipped files (zforce)

secure copying between networks (scp)

sorting (sort)

splitting
csplit

split

string matches, searching for (look)

text files, manipulating
cut command

text, opening in vi

transfer between networks

[See ftp]
uncompressing
and paging (zless)

and passing to grep (zgrep)

from .Z and recompressing to .gz format (znew)

uncompress command

filesystems

exporting

images, displaying

journaling

managing

mounting

potential booting issues

types

unmounting (umount)

filetest command (tcsh)

find command

GRUB

fingerd command

firewalls

compiling support

FTP transfers (KDE)

ipchains command

iptables command

proxy servers and (KDE)

setting rules for in ipchains

in iptables

flags, Debian Package Manager

flex (Fast Lexical Analyzer Generator) command

2nd

[See also lex command; bison command; yacc command]
floating panels (GNOME)

floppy disks
configuring (setfdprm)

formatting (fdformat)

fmt command

fold command

ex

foldclose command (ex)

foldopen command (ex)

for command
bash

gawk

foreach command (tcsh)

fork access method (CVS)

form command (ftp)

formail command

formatting disks
floppies (fdformat)

MS-DOS (mformat)

Fortran programming language, compiling (gcc)

forward slash (/)
division operator

sed command

free command

free disk space, measuring

free software

Free Software Foundation (FSF)

freenode IRC network, Linux channels

freenode IRC service

fsck command

fsck.ext2 command

[See e2fsck command]
fsck.minix command

fsck.msdos

[See dosfsck command]
FSF (Free Software Foundation)

fstest command (GRUB)

ftp (File Transfer Protocol)

commands

secure transfer using ssh (sftp)

tftp command

ftpd command

ftpd daemon

function command
bash

gawk

fvwm2 window manager

AddToMenu variable

application window management

button bar

cascading menus

configuration changes, implementing

configuration files

customizability

desktop configuration

EdgeResistance variable

features

focus configurations

fvwm2 web site

FvwmPager (fvwm2)

FvwmWinList

2nd

mouse operations

GNOME support

Icon Manager

keyboard shortcuts

menu and window manipulation

for navigation

for pointer movement

manpages

menus, customizing

modules

Pager

paging options

pointer navigation

Popup function

switching from GNOME or KDE

window configuration

StartsOn options

sticky windows

window management

.fvwm2rc file

FvwmWinList (fvwm2)

fwhois command

[See whois command]

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[**G**]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

g command (sed)

G command (sed)

g++ command

[See also gcc]2nd

gated routing daemon

gateways

protocols

gawk editor

2nd

gawk scripting language

array assignments

bang tilde (!~)

BEGIN pattern

boolean operators

caret (^)

command-line syntax

commands

alphabetical summary

listing by group

dollar ($)

END pattern

operators

options

patterns

procedures

PROCINFO array

system variables

tilde (~)

variables

variable assignments

gcc (GNU Compiler Collection)

general options

options
linker

warning

pragma directives

GDB (GNU DeBugger)

gensub command (gawk)

geometry command (GRUB)

get command (ftp)

getkeycodes command

getline command (gawk)

getopts command (bash)

getty

[See agetty command]
GhostScript, gs command

gid variable (tcsh)

glob command (ftp)

glob command (tcsh)

global command (ex)

GNOME desktop

2nd

adding icons

Control Center

accessibility tool

background tool

bounce keys

date and time tool

file associations tool (MIME types)

fonts tool

hardware settings

keyboard shortcuts tool

keyboard tool

look and feel settings

manage software tool

menus button

mouse settings tool

network proxies

password, changing

peripherals

repeat keys

screensavers tool

session management tool

slow keys

sound tool

sticky keys

system settings tools

theme selection

toggle keys

URL handlers tool

GConf (configuration database)

.gnome-desktop directory

launcher

menu panel

menus and panel menus

mouse

arrow keys, using for

middle button features

panels

2nd

applets

application launchers, adding

creating

options

panel types

personal settings

start menu

V. 2 platform

V. 2.0 platform
compared to V. 1.4

GNU C language preprocessor command (cpp)

GNU Network Object Model Environment

GNU Zebra routing daemon

GNU/Linux

goto command (tcsh)

GPL (General Public License)

gpm command

gprof command

graphical desktops

developer web sites

differences

fvwm2

[See fvwm2 window manager]
GNOME

[See GNOME desktop]
KDE

[See KDE desktop]
licensing

startup files

grep command

bzgrep

egrep and

groff command

groffer command

GROUP environment variable (tcsh)

group variable (tcsh)

groupadd command

groupdel command

groupmod command

groups command

grpck command

grpconv command

grpunconv command

GRUB (Grand Unified Bootloader)

blocklists

boot-time kernel options

commands

blocklist

boot

bootp

cat

chainloader

cmp

color

configfile

debug

device

dhcp

displayapm

displaymem

dump

embed

find

fstest

ftpserver

geometry

halt

help

hide

ifconfig

impsprobe

initrd

install

ioprobe

kernel

lock

makeactive

map

md5crypt

module

modulenounzip

pager

partnew

parttype

password

pause

quit

rarp

read

reboot

root

rootnoverify

savedefault

serial

setkey

setup

splashimage

terminal

testload

testvbe

unhide

uppermem

vbeprobe

configuration file

device map

grub command syntax

GRUB shell

initrd option

installing

LILO, compared to

menu interface

naming conventions

stages

gs command

gserver access method (CVS)

gsub command (gawk)

gunzip command

[See gzip command]
gzexe command

gzip command

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[**H**]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

h command (sed)

H command (sed)

halt command

GRUB

hard disks
devices, designation as

naming conventions

under GRUB

parameters, setting (hdparm)

partitioning
cfdisk command

fdisk command

hash bang (#!)

tcsh shell

hash command
bash

ftp

hash mark (#)

2nd

in shell scripts

sed

tcsh shell

hashstat command (tcsh)

hdparm command

head command

Header keyword (RCS)

help
Emacs, invoking in

2nd

help command
bash

CVS

ex

ftp

GRUB

hexdump command

hide command
ex

GRUB

histchars variable (tcsh)

histdup variable (tcsh)

histfile variable (tcsh)

histlit variable (tcsh)

history command
bash

CVS

tcsh

history variable (tcsh)

HOME environment variable (tcsh)

home variable (tcsh)

host command

HOST environment variable (tcsh)

hostid command

hostname command

HOSTTYPE environment variable (tcsh)

HPATH environment variable (tcsh)

htdigest command

hup command (tcsh)

hwclock command

hyphen (-) and metamail command

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

i command (sed)

ICMP (Internet Control Message Protocol)

iconv command

id command

Id keyword (RCS)

IDE drives, designation as devices

ident command (RCS)

identd command

idle command (ftp)

if command
bash

gawk

tcsh

ifconfig command

2nd

GRUB

ignoreeof variable (tcsh)

image command (ftp)

image section options

imake command

[See also cpp command; make command]2nd

IMAP (Interactive Mail Access Protocol) server daemon

imapd command

implicitcd variable (tcsh)

import command (CVS)

impsprobe command (GRUB)

in.fingerd command

in.ftpd command

in.identd command

in.rlogind command

in.tftpd command

[See tftpd command]
index command (gawk)

index keyword (expr)

inetd command

info command

init command

CVS

processes invoked by (agetty)

initrd

initrd command
GRUB

inode contents, displaying (stat)

inputmode variable (tcsh)

insert command (ex)

insmod command

install command

GRUB

installation commands

int command (gawk)

Interactive Mail Access Protocol (IMAP) server daemon

Internet Boot Protocol Gateway

Internet Boot Protocol server

Internet connections, initiating
chat command

dip command

Internet Control Message Protocol (ICMP)

Internet protocols

Internet relay chat (IRC)

internet services daemon

ioprobe command (GRUB)

IP addresses

ipchains

2.2 kernel, compiling support into

ipchains command

ipchains-restore command

ipchains-save command

ipcrm command

ipcs command

iptables

iptables command

2nd

iptables-restore command

iptables-save command

IPv6

IRC (Internet relay chat)

ISO9660 images
contents, displaying

information, displaying

integrity, verifying

size, displaying

isodump command

isoinfo command

isosize command

isovfy command

ispell command

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[**J**]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

Java programming language, compiling (gcc)

JFS (Journaled Filesystem)

job control
bash

tcsh

jobs command
bash

tcsh

join command

ex

journaling filesystems

[See also ext3]2nd

jumps command (ex)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[**K**]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

k command (ex)

kbd_mode command

kbdrate command

KDE desktop

2nd

accessibility options

application launch feedback, setting

application windows

color selection

control bar

Control Center

context-sensitive help

cryptography

CUPS

default environment, setting as

desktop

adding links

background

desktop folder

preferences

email configuration

file management

fonts

icon configuration

keyboard configuration

accessibility options

kwin window manager

laptop power and battery management

look and feel

MIME file types

modules

mouse configuration

accessibility options

MS Windows share configuration

nationality and language configuration

network configuration

OpenSSL

panel

adding links

configuration

passwords

2nd

personalization options

screensaver

session management

shortcuts

SOCKS support

sound configuration

spell checkers

status information

style settings

system component information, accessing

system configuration modules

date and time

fonts

login manager (kdm)

printing manager (KDEPrint)

web browsing

taskbar

taskbar configuration

terminal emulator

2nd

theme manager

web browsers

2nd

window behavior configuration

window decoration preferences

kernel command (GRUB)

kernel image

kernels

boot-time options

compiling firewall support into

examining or modifying (sysctl)

managing

scancode-to-keycode mapping table, viewing

video configuration (vidmode)

kernelversion command

keyboards
character repeat rate, setting

driver translation tables, listing

modes, setting

putting in Unicode mode

keycodes, displaying (showkey)

keymaps, loading

keywords (expr)

kill command

bash

Emacs

tcsh

2nd

killall command

killall5 command

killdup variable (tcsh)

killring variable (tcsh)

klogd command

Konqueror file manager and browser (KDE)

2nd

Konsole terminal emulator (KDE)

2nd

Korn shell

kserver access method (CVS)

kserver command (CVS)

ksyms command

kwin (KDE)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[**L**]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

l command (sed)

LANG environment variable (tcsh)

last command

lastb command

lastlog command

LC_CTYPE environment variable (tcsh)

lcd command (ftp)

ld command

ldconfig command

ldd command

left command (ex)

length command (gawk)

length keyword (expr)

less program

bzless

commands

keybindings, configuring for (lesskey)

options

lesskey command

let command (bash)

lex

[See flex]
lftp command

lftpget command

LILO (Linux Loader)

boot-time kernel options

configuration file

global options

image options

kernel options

GRUB, compared to

initrd option

lilo command

2nd

limit command (tcsh)

LINES environment variable (tcsh)

link command

link editor (ld)

linker options, gcc

Linux

2nd

advantages of

default shell

[See bash shell]2nd

changing

distributions and support

online support

periodicals

shells

[See shells]
Unix, compared to

Usenet newsgroups

web sites devoted to

Linux documentation

Linux User Groups (LUGs)

list command (ex)

listflags variable (tcsh)

listing loaded modules (lsmod)

listjobs variable (tcsh)

listlinks variable (tcsh)

listmax variable (tcsh)

listmaxrows variable (tcsh)

ln command

loadkeys command

Loadlin

initrd option

local access method (CVS)

local command (bash)

locale command

locate command

lock command (GRUB)

Locker keyword (RCS)

lockfile command

locking model

CVS

log command
CVS

gawk

tcsh

Log keyword (RCS)

logger command

logical operators (expr)

login command

2nd

CVS

tcsh

login shells, changing (chsh)

logins
display of
bad login attempts (lastb)

last

lastlog

finding usersÕ login names

KDE, management in

loginsh variable (tcsh)

logname command

LOGNAME environment variable (tcsh)

logout command
bash

CVS

tcsh

logout variable (tcsh)

logrotate command

look command

loop devices, setting up (losetup)

losetup command

lpc command

lpd command

lpd daemon

lpq command

lpr command

lprm command

LPRng line printer daemon

printer capability database, inspection and repair

lpstat command

ls command

[See also dircolors command]2nd

ftp

ls-F command (tcsh)

LS_COLORS environment variable

LS_COLORS environment variable (tcsh)

lsattr command

lshift command (gawk)

lsmod command

LUGs (Linux User Groups)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[**M**]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

m4 command

macdef command (ftp)

machine architecture, identifying

machine uptime status, tracking (ruptime)

MACHTYPE environment variable (tcsh)

mail commands
audiosend

biff

fetchmail

formail

imapd

mail

mailq

mailstats

mailto

metamail

sendmail

MAIL environment variable (tcsh)

mail filtering, formail command

mail variable (tcsh)

mailq command

mailstats command

mailto command

make command

description file lines

functions

internal macros

macro modifiers

macro string substitution

pattern rules

special target names

makeactive command (GRUB)

makedbm command

2nd

makemap command

man command

manpages

groffer, displaying with

manpath command

map command
ex

GRUB

maps (NIS)

mark command (ex)

marks (Emacs)

marks command (ex)

masquerading

master boot record

[See MBR]
match command (gawk)

match keyword (expr)

matchbeep variable (tcsh)

mattrib command

mbadblocks command

MBR (master boot record)

pre-Linux Windows version, restoring

mcat command

mcd command

mcopy command

md5crypt command (GRUB)

mdel command

mdelete command (ftp)

mdeltree command

mdir command

ftp

mdu command

memory information, displaying (vmstat)

memory usage information, accessing (free)

menu panels (GNOME)

merge command

merging model (CVS)

mesg command

Meta-key (Emacs)

commands

metacharacters

2nd

pattern matching vs. filename expansion

programs, listing by

metamail command

metasend command

mformat command

mget command (ftp)

Microsoft

[See MS-DOS Windows]
MIME types
GNOME, setting in

mailto command

metamail command

metasend command

mimencode command

minfo command

minus (-) operator

mkdir command

ftp

mkdosfs command

mke2fs command

mkexrc command (ex)

mkfifo command

mkfs command

mkfs.ext2 command

[See mke2fs command]
mkfs.ext3 command

mkfs.minix command

mkfs.msdos command

[See mkdosfs command]
mkisofs command

mklost+found command

mknod command

mkpasswd command

mkraid command

mkswap command

mktemp command

mktime command (gawk)

mlabel command

mls command (ftp)

mmd command

mmount command

mmove command

mode command (ftp)

modinfo command

modprobe command

depmod and

modtime command (ftp)

module command (GRUB)

modulenounzip command (GRUB)

modules
fvwm2

KDE

loaded, listing of (lsmod)

loading (insmod)

loading with initrd

modulo (%) operator

more command

mount command

2nd

mountd command

mountd daemon

mouse accessibility settings (GNOME)

mouse commands (gpm)

move command (ex)

mpartition command

mpg123 command

mpg321 command

mput command (ftp)

mrd command

mren command

MS-DOS
booting from

changing directories on

disk usage, displaying

disks, formatting for

files
changing attributes of (mattrib)

deleting

filesystems
bad blocks, checking for

directory contents, listing (mdir)

formatting disks in (mformat)

restoring the pre-Linux MBR

mshowfat command

mt command

mtools commands

mtoolstest command

mtype command

Multiboot Specification

mv command

mzip command

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[**N**]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

n command (sed)

N command (sed)

Name keyword (RCS)

name servers

name service

named command

named daemon

2nd

namei command

nameif command

netfilter

netstat command

2nd

network administration

commands
arp

NFS and NIS administration

TCP/IP troubleshooting

Network File System

[See NFS]
Network Information System

[See NIS]
networking commands

networks

file transfers between

[See ftp]
securing remote connections cryptographically (ssh)

status messaging (rwhod)

new command (ex)

newaliases command

newer command (ftp)

newgrp command

tcsh

newusers command

next command
ex

gawk

nextfile command (gawk)

NFS (Network File System)

administration commands

daemons

NFS/NIS commands
domainname

makedbm

rpc.yppasswdd command

showmount

ypbind

ypcat

ypchfn

ypinit

ypmatch

yppasswd

yppoll

yppush

ypserv

ypset

yptest

ypwhich

ypxfr

nfsd command

nfsd daemons

nfsstat command

nice command

tcsh

NIS (Network Information System)

administration

administration commands

clients, setting up

map manipulation utilities

maps

servers, setting up

user accounts

nlist command (ftp)

nm command

nmap command (ftp)

nobeep variable (tcsh)

noclobber variable (tcsh)

noding variable (tcsh)

noglob variable (tcsh)

nohlsearch command (ex)

nohup command

nohup command (tcsh)

nokanji variable (tcsh)

nonomatch variable (tcsh)

NOREBIND environment variable (tcsh)

nostat variable (tcsh)

notify command (tcsh)

notify job control command (tcsh)

notify variable (tcsh)

nroff command, managing output with col, colcrt

nslookup command

[See also dig command]2nd

nsupdate command

ntrans command (ftp)

null command (:)

number command (ex)

numbers, base conversions using bc

nvi text editor

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[**O**]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

objcopy command

objdump command

Objective-C programming language, compiling (gcc)

od command

onintr command (tcsh)

online documentation

open command
ex

ftp

open source software

[See free software]
OpenSSH

scp command

openvt command

option argument syntax

options to Linux commands

or command (gawk)

ORBit library

OSTYPE environment variable (tcsh)

output formatting commands
fmt

fold

owd variable (tcsh)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[**P**]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

p command (sed)

P command (sed)

package flags (Debian)

package managers

[See also Debian package manager; RPM]2nd

pager (KDE)

pager command (GRUB)

paging commands
bzless

bzmore

less

[See less program]
more

zless

zmore

panel (KDE)

parentheses ( )

escaped \(\)

partitioning
fdisk command

hard disks (cfdisk command)

partnew command (GRUB)

parttype command (GRUB)

passwd command

password command (GRUB)

passwords
KDE, setting in

2nd

managing
chage command

chfn command

chpasswd

paste command

patch command

PATH environment variable (tcsh)

path variable (tcsh)

pathchk command

pattern addressing (sed)

pattern matching

vs. filename expansion

foreign languages and

gawk

[See gawk scripting language]
POSIX characters and

pause command (GRUB)

PCMCIA sockets
card daemon

controlling with cardctl command

percent (%)

bash job ID argument

prompt

perl command

Perl scripts for Debian package management

pidof command

ping command

pipe (|)

2nd

plus (+) operator

fvwm2

point (Emacs)

Point-to-Point Protocol (PPP)

2nd

popd command
bash

tcsh

Popup function (fvwm2)

portmap command

portmap daemon

POSIX character lists

poweroff command

PPP (Point-to-Point Protocol)

2nd

pppd command

pppd daemon

pr command

pragma directives (gcc)

praliases command

preserve command (ex)

previous command (ex)

print command
ex

gawk

print queue status, showing

print spool queue, checking (lpq)

printenv command

tcsh

printexitvalue variable (tcsh)

printf command

bash

gawk

printing commands

2nd

banner

checkpc

line printer parameters, controlling (tunelp)

lpc

lpd

lpq

lpr

lpstat

removing jobs from the queue (lprm)

processes
killing

2nd

pending jobs, deleting

pending jobs, listing

setting execution times for

program addresses, translation from hexadecimal to filenames

program maintenance commands

programming languages
gcc compiler

source files, inspection, editing (ctags)

programs, listing required libraries for (ldd)

prompt command (ftp)

prompt variables (tcsh)

formatting substitutions

promptchars variable (tcsh)

prompts

proxies

proxy command (ftp)

ps command

pserver access method (CVS)

pserver command (CVS)

pushd command
bash

tcsh

pushdsilent variable (tcsh)

pushdtohome variable (tcsh)

put command
ex

ftp

pwck command

pwconv command

pwd command

bash

ftp

PWD environment variable (tcsh)

pwunconv command

python command

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[**Q**]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

q command (sed)

qall command (ex)

Qt

question mark (?)

2nd

double question marks

ftp help command

quit command
ex

ftp

GRUB

quota command

quote command (ftp)

quote keyword (expr)

quotes

quoting

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[**R**]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

r command (sed)

raidstart command

raidstop command

RAM disks, configuring support for

ramsize command

rand command (gawk)

ranlib command

rannotate command (CVS)

2nd

RARP (Reverse Address Resolution Protocol)

2nd

[See also ARP]
rarp command (GRUB)

rarpd command

raw command

rcp command

RCS (Revision Control System)

basic operations

commands

alphabetical summary

environment variables

keyword substitution

options

revision numbering

state

timestamps

rcs command (RCS)

2nd

rcsclean command (RCS)

rcsdiff command (RCS)

RCSfile keyword (RCS)

RCSINIT environment variable (RCS)

rcsmerge command (RCS)

rdate command

rdev command

rdiff command (CVS)

rdist command

rdistd command

read command
bash

ex

GRUB

readelf command

readlink command

readonly command (bash)

reboot command

GRUB

recexact variable (tcsh)

recognize_only_executables variable (tcsh)

recover command (ex)

recv command (ftp)

Red Hat Package Manager

[See RPM]
redirection operators

redo command (ex)

reget command (ftp)

regular expressions

rehash command (tcsh)

Reiser FS (Reiser Filesystem)

embed command (GRUB)

relational operators (expr)

release command (CVS)

Remote Procedure Call

[See RPC]
remotehelp command (ftp)

REMOTEHOST environment variable (tcsh)

remotestatus command (ftp)

removable media, ejecting

remove command (CVS)

rename command

ftp

renice command

repeat command (tcsh)

repositories (CVS)

creating

reset command

2nd

[See also tset command]
ftp

resize command (ex)

resize2fs command

resolvers

restart command (ftp)

return command
bash

gawk

Return command (ex)

rev command

Reverse Address Resolution Protocol (RARP)

revision control

2nd

[See also RCS; CVS]
Revision Control System

[See RCS]
Revision keyword (RCS)

rewind command (ex)

rexecd command

richtext command

right command (ex)

ÒrippingÓ audio CDs

rlog command
CVS

RCS

rlogin command

rlogind command

rm command

rmail command

rmdir command

ftp

rmmod command

rmstar variable (tcsh)

root command (GRUB)

rootflags command

rootnoverify command (GRUB)

route command

routed command

routed daemon

routing daemons

routing tables

RPC (Remote Procedure Call)

rpc.mountd command

rpc.nfsd command

rpc.portmap command

rpc.rusersd command

rpc.statd command

[See statd command]
rpcgen command

rpcinfo command

RPM (Red Hat Package Manager)

rpm command

[See rpm command]
rpm command

2nd

configuration files

database rebuild options

downloading packages off the internet

examples

FTP/HTTP options

information selection options

install/upgrade options

package selection options

query options

signature check options

uninstall options

verify options

rpmbuild command

rprompt variable (tcsh)

rsh command

rshd command

rshift command (gawk)

rtag command (CVS)

run-parts command

runique command (ftp)

runlevel command

runlevels

changing (telinit)

rup command

ruptime command

rusers command

rusersd command

rwall command

rwho command

rwhod command

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[**S**]

[T]

[U]

[V]

[W]

[X]

[Y]

[Z]

s command (sed)

sandboxes (CVS)

savedefault command (GRUB)

savedirs variable (tcsh)

savehist variable (tcsh)

sbnext command (ex)

sbuffer command (ex)

sched command (tcsh)

sched variable (tcsh)

scheme command

scp command

screen command

script command

ex

scripts, Debian package management

SCSI drives, designation as devices

sdiff command

search commands

egrep

expr

fgrep (file searching)

find

grep

locate

look

whereis

zgrep

searching

2nd

[See also pattern matching]
and replacing

vi editor, in

Second Extended Filesystem

[See ext2]
security
APM (Advanced Power Management) and

Crypto module (KDE)

CVS and

encryption (ssh)

firewalls

GRUB configuration file and

ipchains command

iptables command

login manager (KDE)

passwords module (KDE)

sed command

[See sed editor]
sed editor

2nd

commands

alphabetical listing

branching commands

editing

input/output processing

line information

multiline input processing

syntax

yanking and putting

invocation

pattern addressing

select command (bash)

semicolon (;)

send command (ftp)

sendmail command

sendport command (ftp)

seq command

serial command (GRUB)

Serial Line IP (SLIP)

serial-line communication

serial-lines, attaching as network interfaces (slattach)

server command (CVS)

set command
bash

ex

2nd

colon in (:set)

tcsh

setenv command (tcsh)

setfdprm command

setkey command (GRUB)

setkeycodes command

setleds command

setmetamode command

setsid command

settc command (tcsh)

setterm command

setty command (tcsh)

setup command (GRUB)

sftp command

sh command

[See also bash shell]2nd

shar command

shell archives
creating (shar)

unpacking

shell command (ex)

shell commands

SHELL environment variable (tcsh)

shell scripts

comment character (#)

Debian package management

shell variables, tcsh shell

shells

bash

[See bash shell]
C shell (tcsh)

common features

creating for user (su)

default, setting in /etc/passwd

shell variables

tcsh

[See tcsh shell]
shift command
bash

tcsh

SHLVL environment variable (tcsh)

shlvl variable (tcsh)

shopt command (bash)

showkey command

showmount command

shred command

shutdown command

Simple Mail Transport Protocol (SMTP)

Simple Network Management Protocol (SNMP)

sin command (gawk)

site command (ftp)

size command

ftp

skill command

slattach command

sleep command

sliding panels (GNOME)

SLIP (Serial Line IP)

SMTP (Simple Mail Transport Protocol)

snext command (ex)

snice command

[See also skill command]2nd

SNMP (Simple Network Management Protocol)

sort command

sound configuration
GNOME

KDE

source code

source command
bash

ex

tcsh

Source keyword (RCS)

spacing and syntax

2nd

special characters

[See also pattern matching]2nd

splashimage command (GRUB)

split command

ex

gawk

sprevious command (ex)

sprintf command (gawk)

sqrt command (gawk)

srand command (gawk)

ssh program

ssh-add command

ssh-agent command

ssh-keygen command

ssh-keyscan command

sshd command

[See also ssh program]2nd

SSL (Secure Sockets Layer), configuration in KDE

Stallman, Richard

stat command

statd command

state

State keyword (RCS)

status command
CVS

ftp

status variable (tcsh)

stop command
ex

tcsh

2nd

storage commands

strace command

stream editor

[See sed editor]
strfile command

strftime command (gawk)

string matches, searching files for (look)

string searches with apropos

strings command

strip command

strtonum command (gawk)

struct command (ftp)

stty command

su command

sub command (gawk)

subdomains

substitute command (ex)

substr command (gawk)

substr keyword (expr)

sudo command

sum command

sunique command (ftp)

SuSE package manager

[See RPM]
suspend command
bash

ex

tcsh

sview command (ex)

swapoff command

swapon command

switch command (tcsh)

symlinks variable (tcsh)

sync command

syntax, spacing and

sysctl command

sysklogd command

syslogd command

[See sysklogd command]
system
information, displaying (uname)

logged-in users, displaying (who)

status, displaying (uptime)

system administration

commands

agetty

apmd

atd

badblocks

cardctl

cardmgr

cfdisk

chat

checkpc

chpasswd

chroot

clocks

cron

daemons

debugfs

depmod

dip

dmesg

dnssec-keygen

dnssec-makekeyset

dnssec-signkey

dnssec-signzone

dosfsck

dumpe2fs

e2fsck

e2image

e2label

elvtune

fdisk

fetchmail

filesystems management

fsck

fsck.minix

gpm

groupadd

groupdel

groupmod

grpck

grpconv and grpunconv

halt

host

hwclock

init

insmod

install

installation

ipchains

ipchains-restore

ipchains-save

ipcrm

ipcs

iptables

iptables-restore

iptables-save

kbdrate

kernel management

klogd

ksysms

lastlog

ldconfig

logrotate

losetup

lpc

lsmod

mailq

mailstats

makemap

mkdosfs

mke2fs

modprobe

networking

printing

rusersd

sendmail

setkeycodes

setsid

shutdown

statd

swapoff

swapon

sync

sysctl

sysklogd

telnit

tune2fs

tunelp

umount

update

user management

useradd

userdel

usermod

vidmode

zdump

zic

system calls, tracing (strace)

system command
ftp

gawk

system status commands
load average, displaying (tload)

top

usage information, displaying (w)

systime command (gawk)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[**T**]

[U]

[V]

[W]

[X]

[Y]

[Z]

t command
ex

sed

Tab

tac command

tag command
CVS

ex

tags command (ex)

tail command

talk command

talkd command

tar command

taskbar (KDE)

tbl command, managing ouput with col, colcrt

TCP/IP (Transmission Control Protocol/Internet Protocol)

commands

arp

bootpd

bootptest

dnsdomainname

fingerd

ftpd command

identd

ifconfig

inetd

logger

lpd

ruptime

rusers

rwall

rwhod

slattach

sshd

talkd

tcpd

telnetd

tftpd

traceroute

xinetd

configuring

gateways and routing

included protocols

IP addresses

name service

troubleshooting commands

tcpd command

tcsh command

tcsh prompt

tcsh shell

2nd

arithmetic operators

assignment operators

bash, compared to

bitwise and logical operators

built-in commands

command completion

command history

command substitution

history modifiers

special aliases

word substitution

command structure

command-line editing

Emacs mode

vi mode

comparison operators

configuration files

environment variables

expressions

operator precedence

operators

file inquiry operators

formatting substitutions, prompt variable

invocation, options and arguments

job control commands

multiple redirection

quoting

characters

redirection

shell variables

special characters

special files

syntax

variables

modifiers

variable substitution

tcsh variable (tcsh)

tee command

telinit command

telltc command (tcsh)

telnet command

telnetd command

tenex command (ftp)

TERM environment variable (tcsh)

term variable (tcsh)

TERMCAP environment variable (tcsh)

terminal command (GRUB)

terminal configuration (setterm)

terminal display, clearing

terminal emulators
ANSI/VT100 emulation, enabling (screen)

Konsole (KDE)

2nd

terminal sessions, recording (script)

terminals
displaying name of (tty)

I/O options setting (stty)

initializing (tset)

test command

bash

testload command (GRUB)

testvbe command (GRUB)

text editors
dual mode vs. modeless editing

Emacs

[See Emacs editor]
ex

[See ex editor]
pattern matching in

sed

[See sed editor]
vi

[See vi editor]
tftp command

tftpd command

tftpserver command (GRUB)

Third Extended Filesystem

[See ext3]
tilde (~)
bash

ex command

gawk

sed command

tcsh

time and date commands
cal

date

time command

time command (tcsh)

time variable (tcsh)

times command (bash)

timezones, displaying data about

tload command

TLS (Transport Layer Security), configuration in KDE

tolower command (gawk)

top command

top-level domains

Torvalds, Linus

touch command

toupper command (gawk)

tperiod variable (tcsh)

tr command

trace command (ftp)

traceroute command

tracing system calls (strace)

trap command (bash)

troff command

[See groff command]
TrollTech, Inc.

true command

[See also false command]2nd

tset command

tty command

tty variable (tcsh)

tune2fs command

tune2fs utility
ext2, converting to ext3

tunelp command

type command
bash

ftp

typeset command (bash)

[See declare command]

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[**U**]

[V]

[W]

[X]

[Y]

[Z]

UDP (User Datagram Protocol)

uid variable (tcsh)

ul command

ulimit command (bash)

umask command
bash

ftp

tcsh

umount command

unabbreviate command (ex)

unalias command
bash

tcsh

uname command

uncomplete command (tcsh)

uncompress command

uncompressing and paging (zmore)

undo command (ex)

unedit command (CVS)

unexpand command

unhash command (tcsh)

unhide command
ex

GRUB

unicode_start command

unicode_stop command

uniq command

Unix shell command (sh)

Unix, compared to Linux

unlimit command (tcsh)

unmap command (ex)

unset command
bash

tcsh

unsetenv command (tcsh)

unshar command

unstr command

[See strfile command]
until command (bash)

update command

CVS

uppermem command (GRUB)

uptime command

Usenet newsgroups

user command (ftp)

USER environment variable (tcsh)

user login times, tracking (ruptime)

user variable (tcsh)

useradd command

userdel command

usermod command

users
finding login names for

management commands

users command

usleep command

uudecode command

uuencode command

uuidgen command

UUIDs (Universal Unique Identifiers), creating (uuidgen)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[**V**]

[W]

[X]

[Y]

[Z]

v command (ex)

vacation command

variables, gawk scripting language

usage in

vbeprobe command (GRUB)

vdir command

VENDOR environment variable (tcsh)

verbose command (ftp)

verbose variable (tcsh)

version command
CVS

ex

version control

2nd

[See also RCS; CVS]
version variable (tcsh)

vi command (ex)

vi editor

2nd

address symbols

bindings, compared to Emacs bindings

command mode

alphabetical key list

command syntax

command-line editing mode (bash)

2nd

command-line options

configuration

~/.exrc file

2nd

edit commands

changing text

deleting text

exiting

multiple files

saving

text insertion

editing operators

enhanced versions

ex

[See ex editor]
files, opening

insert mode

line numbering

macros

marking position

miscellaneous commands

movement commands

character

lines

screens

text

operating modes

searching

:set command (ex)

shell, interacting with

starting a session

status-line commands

tcsh command-line editing mode

user-defined commands, characters for

window commands

vidmode command

view command (ex)

vile text editor

vim command

[See vi editor]
vim editor

virtual consoles
destroying

number of, determining

virtual terminals, switching between

visiblebell variable (tcsh)

visual command (ex)

VISUAL environment variable
bash

tcsh

viusage command (ex)

vmstat command

volname command

vsplit command (ex)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[**W**]

[X]

[Y]

[Z]

w command

sed

wait command
bash

tcsh

wall command

ex

warning options, gcc

watch command

CVS

watch variable (tcsh)

watchers command (CVS)

watchlog command (tcsh)

wc command

web browsers (KDE)

2nd

web sites

whatis command

whatis database, searching with apropos command

where command (tcsh)

whereis command

which command

tcsh

while command
bash

gawk

tcsh

whitespace, converting to tabs (unexpand)

who command

who variable (tcsh)

whoami command

whois command

Window List Applet (GNOME)

window managers

windows (Emacs)

Windows (Microsoft)
dual booting, NT/2000/XP and Linux

restoring the pre-Linux MBR

share configuration (KDE)

wnext command (ex)

wordchars variable (tcsh)

working directories
CVS

pwd command, identifying with

Workspace Switcher Applet (GNOME)

wq command (ex)

wqall command (ex)

write command

ex

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[**X**]

[Y]

[Z]

X command (ex)

x command (sed)

X Window System

xargs command

XDR (eXternal Data Representation)

XFS (Extensible Filesystem)

Ximian Red Carpet

Xinerama

2nd

xinetd command

xit command (ex)

xor command (gawk)

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[**Y**]

[Z]

y command (sed)

yacc command

yank command
Emacs

ex

yes command

ypbind command

ypcat command

ypchfn command

ypinit command

2nd

ypmatch command

yppasswd command

yppasswdd command

yppoll command

yppush command

2nd

ypserv command

ypset command

yptest command

ypwhich command

ypxfr command

[SYMBOL]

[A]

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[I]

[J]

[K]

[L]

[M]

[N]

[O]

[P]

[Q]

[R]

[S]

[T]

[U]

[V]

[W]

[X]

[Y]

[**Z**]

z command (ex)

Z shell

zcat command

zcmp command

zdiff command

zdump command

zforce command

zgrep command

zic command

zless command

zmore command

znew command