

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Introduction](#)

[About the Lead Author](#)

[Part I—Introduction to UNIX](#)

[Chapter 1—The UNIX Operating System](#)

[What Is UNIX?](#)

[Understanding Operating Systems](#)

[Hardware Management, Part 1](#)

[Process Management](#)

[Hardware Management, Part 2](#)

[The UNIX Operating System](#)

[The History of UNIX](#)

[The Early Days](#)

[Berkeley Software Distributions](#)

[UNIX and Standards](#)

[UNIX for Mainframes and Workstations](#)

[UNIX for Intel Platforms](#)

[Source Versions of “UNIX”](#)

[Introduction to the UNIX Philosophy](#)

[Simple, Orthogonal Commands](#)

[Commands Connected Through Pipes](#)

[A \(Mostly\) Common Option Interface Style](#)

[No File Types](#)

[Summary](#)

[Brief](#) [Full](#)

☐ [Advanced](#)

[Search](#)

☐ [Search Tips](#)

Chapter 2—Getting Started: Basic Tutorial

Logging In to the System

User Account Setup

Logging In to the System

After Login Succeeds

Different Privileges for Different Users

Logging Out

Using Commands

What Is a Command?

Redirecting Input and Output

Configuring Your Environment

Viewing and Setting Environment Variables

Using Shell Startup Files

Configuring with rc files

Managing Your Password

Working on the System

Erase

Kill

Stop and Start

eof

Where to Find Information About Using UNIX

UNIX Manual Pages

Manual Page Organization

The Manual Page Command

Web Sites

Book Lists

Frequently Asked Questions (FAQ)

Tutorials

FTP Sites

Newsgroups

UNIX User Groups

Professional Associations

The Electronic Frontier Foundation

The Open Group

USENIX

UniForum

The X Consortium

Publications

UNIX Review

UNIX World

Sys Admin

Sun World

SunExpert

Summary

Chapter 3—The UNIX File System

File System Organization

Naming Files and Directories

File Types

Regular Files

Directory Files

Character and Block Device Files

Sockets

Named Pipes

Symbolic and Hard Links

Naming Files and Directories

Working with Directories

Listing Files and Directories with ls

Creating and Deleting Directories: mkdir and rmdir

Using the find Command

Reviewing Disk Utilization with du and df

Determining the Nature of a File's Contents with file

File and Directory Permissions

The Permission Bits

Default Permissions: umask

Changing Permissions: chmod

Changing Owner and Group: chown and chgrp

Setuid and Setgid

Summary

Chapter 4—General Commands

User-Related Commands

login

rlogin

telnet

passwd

exit

Locating Commands

which

whence

where

Determining Command Usage

man

Administration

install

shutdown

ulimit

umask

Process-Related Commands

kill

nice

ps

jobs

wait

nohup

sleep

Communication

cu

ftp

mailx

talk

vacation

write

File Comparison

cmp

comm

diff

diff3

dircmp

sdiff

File-Manipulation Commands

touch

chmod

chgrp

chown

rm

mv

cp

cat

rcp

ln

Directory-Manipulation Commands

mkdir

rmdir

File Information Commands

ls

find

file

File Content-Related Commands

more

less

tail

head

wc

read

od

pg

tee

vi

File Content Search Commands

[egrep](#)
[fgrep](#)
[grep](#)
[strings](#)

Printing

[cancel](#)
[lp](#)
[pr](#)
[lpstat](#)

Scheduling

[at](#)
[atq](#)
[crontab](#)

Storage

[compress](#)
[cpio](#)
[dd](#)
[pack](#)
[pcat](#)
[tar](#)
[uncompress](#)
[unpack](#)
[zcat](#)

Status Commands

[date](#)
[env](#)
[iostat](#)
[sar](#)
[uname](#)
[uptime](#)
[vmstat](#)

Text Processing

[cut](#)
[ex](#)
[fmt](#)
[fold](#)
[join](#)
[paste](#)
[sort](#)
[tr](#)
[uniq](#)
[sed](#)

Miscellaneous Commands

[banner](#)
[bc](#)
[cal](#)

Examples

calendar

clear

time

xargs

Regular Expression

Character Set

Position Specifier

Metacharacters

Executing Commands

Summary

Chapter 5—Getting Around the Network

What Is a Network?

UUCP—The Basic Networking Utilities

TCP/IP—LAN, WAN, and the Internet

Names and Addresses

Connecting to Other Systems—rlogin, telnet, and cu

Before Using rlogin, rsh, and rcp

Using rlogin

Using telnet

Before Using cu

Transferring Files—rcp, ftp, and uucp

Using rcp

Using ftp

Using uucp, uuto, and uupick

Other Networking Services

archie

gopher

World Wide Web

Troubleshooting TCP/IP

nslookup to Check Address Mapping

Is There Anybody Out There? (ping)

Summary

Chapter 6—Communicating with Others

Electronic Mail (Email)

Components of a Mail Message

Sending Binary Data

Finding Addresses

Mail Programs

Mailing Lists

Automatic Mail Sorting

Usenet

Newsreaders

Finding Your Groups

Talk

Internet Relay Chat (IRC)

[Basic IRC Structure](#)

[Getting IRC Clients](#)

[Connecting to a Server](#)

[Choosing Channels](#)

[Need Help?](#)

[Bad Moves](#)

[Further Info](#)

[Multimedia](#)

[Internet Infrastructure](#)

[Multicast Backbone](#)

[Audio Over the Internet](#)

[Video Over the Internet](#)

[Summary](#)

[Chapter 7—Text Editing with vi and Emacs](#)

[Full-Screen Editors Versus Line Editors](#)

[What Is vi?](#)

[The Relationship of vi and ex](#)

[Why Would I Be Interested in Using vi?](#)

[Starting Up and Exiting vi](#)

[Getting Started with vi: The Big Picture](#)

[vi Has Modes](#)

[Starting vi](#)

[Moving Around and Simple Editing](#)

[Advanced Editing with vi: Tips and Techniques](#)

[Using the Power of ex from vi](#)

[Using Basic ex Commands to Manipulate Blocks of Text](#)

[Search and Replace](#)

[Regular Expressions](#)

[Global Search and Replace with Regular Expressions](#)

[Working with Files](#)

[Using the Power of UNIX from Within vi](#)

[Marking Your Position](#)

[For the Power User: Customizing vi](#)

[Other Advanced Editing Techniques](#)

[What Is Emacs?](#)

[Comparison to vi](#)

[How To Get Emacs](#)

[Why Would I Be Interested in Using Emacs?](#)

[Starting Up and Exiting the Program](#)

[Basic Editing with Emacs: Getting Started](#)

[Moving Around and Simple Editing](#)

[Advanced Editing with Emacs: Tips and Techniques](#)

[Search and Replace](#)

[Using Multiple Buffers](#)

[Formatting for Various Languages](#)

Using Emacs as an Environment

Command Summary vi and Emacs

Summary

Chapter 8—GUIs for End Users

What Is a GUI?

The X Window System

New X Features (Version X11R6.4)

Obtaining X

Starting X

Customizing X Resources

Working with X Clients

Motif Window Manager

Starting Motif Window Manager

Working with Motif Windows

Customizing Motif

Other Window Managers

TWM Window Manager

FVWM Window Manager

AfterStep Window Manager

AmiWm Window Manager

CTWM Window Manager

LessTif Window Manager

OLVWM Window Manager

wm2 Window Manager

Summary

Part II—UNIX Shells

Chapter 9—What Is a Shell?

How the Kernel and the Shell Interact

UNIX Calls the Shell at Login

The Shell and Child Processes

Auto-Execution of the Shell

The Functions and Features of a Shell

Command-Line Interpretation

Reserved Words

Shell Metacharacters (Wildcards)

Program Commands

File Handling: Input/Output Redirection and Pipes

Command Substitution

Maintenance of Variables

Shell Startup—Environment Control

Shell Programming

Summary

Chapter 10—The Bourne Shell

Shell Basics

The Shell Invocation and Environment

Special Characters and Their Meanings

How the Shell Interprets Commands

Entering Simple Commands

Shell Variables

Storing Data or User-Defined Variables

Conditional Variable Substitution

Positional Variables or Shell Arguments

Shell Script Programming

What Is a Program?

A Simple Program

The Shell as a Language

Using Data Variables in Shell Programs

Entering Comments in Shell Programs

Doing Arithmetic on Shell Variables

Passing Arguments to Shell Programs

Decision Making in Shell Programs

Building Repetitions into a Program

Conditional Command Execution with the And/Or
Constructs

Customizing the Shell

Customizing the Shell with Environment Variables

Adding Your Own Commands and Functions

Specialized Topics

Debugging Shell Programs

Grouping Commands

Using the Shell Layer Manager shl

Summary

Chapter 11—The Bourne Again Shell

Features

Definitions

Building and Installing bash

Invocation

Interactive Mode

Noninteractive Shells

Invocation Options

Using bash as the Login Shell

Syntax

Variables

Expansion

Quoting

Commands

Redirection

Flow Control

Loops

Initialization Files

Initialization File Considerations

[Prompt Customization](#)

[Shell Customization](#)

[Aliases](#)

[Functions](#)

[Command Line and History](#)

[The Command Line](#)

[Command Line Editing](#)

[Completion](#)

[History](#)

[Summary](#)

[Chapter 12—The Korn Shell](#)

[Shell Basics](#)

[Wildcard Expressions](#)

[Command Substitution](#)

[An Improved cd Command](#)

[Aliases](#)

[Defining Aliases](#)

[Removing an Alias](#)

[Writing an Alias Definition](#)

[Using Exported Aliases](#)

[Using Tracked Aliases](#)

[Shell Options](#)

[Command History](#)

[Displaying the Command-History List](#)

[Re-executing a Command from the History](#)

[Accessing the History List: fc](#)

[Command Editing](#)

[Activating Command-Edit Mode](#)

[vi Edit Mode](#)

[EMACS Edit Mode](#)

[Variables](#)

[Predefined Variables](#)

[Referencing Variables](#)

[Array Variables](#)

[Variable Arithmetic](#)

[Shell Programming](#)

[Conditional Expressions](#)

[Functions](#)

[Scanning Arguments with getopt](#)

[Using the select Statement](#)

[Using Coprocesses](#)

[Cautionary Tales](#)

[Customizing the Korn Shell](#)

[Setting Control Keys with stty](#)

[Controlling Resources with ulimit](#)

[Setting Local Variables for Shell Control](#)

Defining Aliases

Defining Functions

Setting Shell Options

Executing Commands Every Time You Log On

Executing Your .profile After Changing It

Creating an ENV File

Adding Settings for Other Programs to Your .profile

Controlling Jobs

Summary

Chapter 13—The C Shell

Shell Basics

Executing Commands

Redirecting Input and Output

Filename Substitutions (Globbing)

Executing Commands in a Subshell: ()

Quoting

Working with Directories

Variables

Customizing Your Shell Environment

.cshrc

.login

.logout

Advanced Features

Aliases

Command History

Job Control

Summary

Chapter 14—Shell Comparison

Interactive Usage

Bourne Shell

POSIX Shell

C Shell

Korn Shell

TC Shell

Bourne Again Shell

Z Shell

Interactive Shell Feature Comparison

Shell Scripts for Personal Use

Shell Scripts for Public Consumption

Summary

Part III—Programming

Chapter 15—awk

When to Use awk

When Not to Use awk

Features of awk

awk Fundamentals

Using awk from the Command Line
awk Processing (Patterns and Actions)
Regular Expression Patterns
Comparison Operators and Patterns
Compound Pattern Operators
Range Pattern Operators
Handling Input
Coding Your Program

Actions

Variables
Predefined Variables
Strings
String Constants
String Operators
Special String Constants
Arrays
Array Functions
Multidimensional Arrays
Built-in Numeric Functions
Arithmetic Operators
Conditional Flow
The Null Statement
The Conditional Operator
Looping
The do Statement
Loop Control (break and continue)
The for Statement
The while Statement

Advanced Input and Output

Input
next and exit
getline
Input from a File
Input from a Command
Ending Input from a File or Command
Output
Pretty Formatting (printf)
Output to a File
Output to a Command
Closing an Output File or Pipe

Functions

Function Definition
Function Parameters
Explicit Returns from Functions (return Statement)

Writing Reports

Complex Reports

Extracting Data

Commands On-the-Fly

One Last Built-in Function: system

Summary

Chapter 16—Perl

Overview of Perl

Where Can I Get Perl?

A Simple Sample Program

Using Comments

Reading from Standard Input

Storing Values in Scalar Variables

Assigning a Value to a Scalar Variable

Scalar Variables Inside Character Strings

Writing to Standard Output

Working with Scalar Variables

Understanding Scalar Values

Using Scalar Variable Operators

Using Lists and Array Variables

Introducing Lists

Using List Ranges

Storing Lists in Array Variables

Assigning to Array Variables

Retrieving the Length of a List

Using Array Slices

Other Array Operations

Using Command-Line Arguments

Standard Input and Array Variables

Controlling Program Flow

Conditional Execution: The if Statement

Two-Way Branching Using if and else

Conditional Branching Using unless

Repeating Statements Using while and until

Using Single-Line Conditional Statements

Looping with the for Statement

Looping Through a List—The foreach Statement

Exiting a Loop with the last Statement

Using next to Start the Next Iteration of a Loop

Using Labeled Blocks for Multilevel Jumps

Terminating Execution Using die

Reading from and Writing to Files

Opening a File

Reading from a File

Writing to a File

Closing a File

Determining the Status of a File

Reading from a Sequence of Files

Using Subroutines

Defining a Subroutine

Using a Subroutine

Returning a Value from a Subroutine

Using Local Variables

Passing Values to a Subroutine

Associative Arrays

Defining Associative Arrays

Accessing Associative Arrays

Copying to and from Associative Arrays

Adding and Deleting Array Elements

Listing Array Indexes and Values

Looping with an Associative Array

Formatting Your Output

Defining a Print Format

Displaying a Print Format

Displaying Values in a Print Format

Writing to Other Output Files

Specifying a Page Header

Formatting Long Character Strings

References

Understanding References

References and Arrays

Multidimensional Arrays

References to Subroutines

References to File Handles

Object-Oriented Programming

Packages

Creating a Module

Creating a Class and Its Objects

Methods

Overrides

Inheritance

Using Built-In Functions

The \$_ Variable

Summary

Chapter 17—The C and C++ Programming Languages

Introduction to C

Programming in C: Basic Concepts

Creating, Compiling, and Executing Your First Program

An Overview of the C Language

Elementary C Syntax

Expressions

Statement Controls

Creating a Simple Program

- Writing the Code
- Compiling the Program
- Executing the Program
- Building Large Applications
- Making Libraries with ar
- Debugging Tools
- Introduction to C++
 - Programming in C++: Basic Concepts
 - Scope of Reference in C and C++
 - Overloading Functions and Operators in C++
 - Functions Within C++ Data Structures
 - Classes in C++

- Summary

Chapter 18—The make Utility

- Introduction to make
- Makefiles
- Target Lines
 - Library Targets
 - Rule Targets
 - Built-In Targets
 - Common Targets
- Shell Command Lines
- Macros
 - Macro Syntax
 - Macro Substitution
 - Special Built-In Macros
- make Directives
- Command-Line Arguments
 - Command-Line Options
 - Command-Line Macro Definition
 - Command-Line Target Specification
- Different make Programs
 - GNU make
 - imake
 - nmake
- make Utilities
 - makedepend
 - mkmf

- Summary

Chapter 19—Source Management and Revision Control

- Revision Control Concepts
 - Registering the Initial Revision
 - Creating a New Revision
 - Revision History
 - File Locking
 - Symbolic Names, Baselines, and Releases

RCS

Using RCS—A Simple Example

Manipulating Revisions

For More RCS Information

SCCS

Basic Operation

SCCS Command Summary

Extra SCCS Features

Creating a Revision Branch

CVS

How Is CVS Different from RCS and SCCS?

Starting a Project

Maintaining Source Code Revisions

Branches

Environment Variables

Summary

Part IV—System Administration

Chapter 20—What Is System Administration?

Technical Concepts for New System Administrators

Multiple Users and Multiple Accounts

Network Centricity

UNIX Networking: Sharing Files and Information

Network Security Issues

UNIX Is Heterogeneous

Administration Tools

Graphical Interfaces

Command-Line Interfaces

System Administration Tasks

Supporting Users

Supporting Systems

Administration Resources

The Manual Pages

Internet Information Resources

Tools of the Trade

The Shell

Perl and Other Automation Tools

Summary

Chapter 21—UNIX Installation Basics

What Are the Differences Between the Different Distributions?

What Do I Need to Know from the Start?

Space Requirements

Who Is Going to Use This System?

What Other Systems Are Located on This Segment of the LAN?

Summarizing What You Need to Know Before Starting

Planning for the Installation

From Where Am I Going to Install?

Dataless or Standalone Server System?

Naming the System

Choosing Which Packages to Install Locally

Laying Out the Partitions

Why Multiple File Systems?

The root Partition

The swap Partition

The usr Partition

The var Partition

The home Partition

The tmp Partition

Assigning Partitions to Disk Drives

Assigning IP (Network) Addresses

Do You Have the Needed Network Connections?

Using NIS/NIS+

Performing the Installation

Booting the Installation Media

Installing the Master System

Installing Optional or Additional Packages

Summary

Chapter 22—Starting Up and Shutting Down

Startup

Initialization Process

Configuration File

RC Scripts

Startup Daemons and Programs

Shutdown

HP-UX

Solaris

Linux

Summary

Chapter 23—User Administration

Adding New Users

Password File

Shadow Password File

Group File

Miscellaneous Files

Pseudo Users

User Maintenance Commands

User Monitor Commands

User Limiting Commands

Anonymous ftp

Summary

Chapter 24—File System and Disk Administration

What Is a File System?

inodes

Superblocks

Types of Files

File Permissions

Managing File Permissions, Owners, Groups, and
Special Files

Managing File Systems

Mounting and Unmounting File Systems

Common Commands for File System Management

Security Tools for Your File Systems

Repairing File Systems with fsck

The fsck Utility

What Is a Clean (Stable) File System?

Where Is fsck?

When Should I Run fsck?

How Do I Run fsck?

What Do I Do After fsck Finishes?

Dealing with What Is in lost+found

Creating File Systems

Disk Types

What Are Partitions and Why Do I Need Them?

The Device Entry

Partitioning Disks and Creating File Systems

Summary

Chapter 24—File System and Disk Administration

What Is a File System?

inodes

Superblocks

Types of Files

File Permissions

Managing File Permissions, Owners, Groups, and
Special Files

Managing File Systems

Mounting and Unmounting File Systems

Common Commands for File System Management

Security Tools for Your File Systems

Repairing File Systems with fsck

The fsck Utility

What Is a Clean (Stable) File System?

Where Is fsck?

When Should I Run fsck?

How Do I Run fsck?

What Do I Do After fsck Finishes?

Dealing with What Is in lost+found

Creating File Systems

Disk Types

What Are Partitions and Why Do I Need Them?

The Device Entry

Partitioning Disks and Creating File Systems

Summary

Chapter 25—Kernel Basics and Configuration

What Is a Kernel?

Kernel Services

System Initialization

Kernel Mode

Process Management

Process Structure

Process Run States

Process Scheduler

Memory Management

Input and Output Management

RAM I/O

Hard Drive I/O

File System Management Subsystem

File System Types

Hardware Architecture

File System Concepts and Format

Kernel Configuration Process

When Do You Rebuild the Kernel

HP-UX 10.X

Solaris 2.5.x

SVR4

AIX 4.x

Linux

Summary

Chapter 25—Kernel Basics and Configuration

What Is a Kernel?

Kernel Services

System Initialization

Kernel Mode

Process Management

Process Structure

Process Run States

Process Scheduler

Memory Management

Input and Output Management

RAM I/O

Hard Drive I/O

File System Management Subsystem

File System Types

Hardware Architecture

File System Concepts and Format

Kernel Configuration Process

When Do You Rebuild the Kernel

HP-UX 10.X

Solaris 2.5.x

SVR4

AIX 4.x

Linux

Summary

Chapter 26—Networking

Introduction

Basics of TCP/IP Communications

TCP/IP Protocol Architecture

IP Addresses

Routing

TCP/IP Startup

TCP/IP Startup on SVR4

TCP/IP Startup on Solaris 2.x

TCP/IP Startup on Linux

inetd

Name Services

The /etc/hosts file

The Domain Name System (DNS)

Domain Name Service Implementation

The Network File System

NFS Daemons

Setting Up the NFS Server

Setting Up the NFS Client

Summary

Chapter 27—System Accounting

UNIX System Accounting Basics

Connect Session Statistics

Process Usage

Disk Space Utilization

Printer Usage (AIX 4.x)

Command Definitions

Commands That Run Automatically

System Accounting Commands That Run

Automatically or Manually

Manually Executed Commands

Configuration Procedures

Setting Up the AIX 4.x Accounting System

Setting Up the HP-UX 10.x Accounting System

Disk Accounting Statistics

Setting Up the Solaris 2.5 (or higher) Accounting System

System Accounting Directory Structure

System Accounting High-Level Directory Layout

System Accounting Report Generation

Generation of System Accounting Data Reports

Daily Automated Reports

Summary

Chapter 28—Performance Monitoring

Performance and Its Impact on Your Business

Introduction to UNIX Performance

Tools for Monitoring the Overall System Status

Using uptime

Using perfmeter

Using sar

Process Monitoring

Using ps

Possible Corrective Action

Memory Usage Monitoring

Using vmstat

Using sar

Possible Corrective Action

Disk Subsystem Performance Monitoring

Possible Corrective Actions

Network Performance Monitoring

Using netstat

Using spray

Using nfsstat

Possible Corrective Actions

CPU Performance Monitoring

Using mpstat

Possible Corrective Actions

Kernel Tuning

Displaying Tunable Kernel Parameters

Showing Current Values

Modifying Values

Conclusion of Kernel Tuning

Third-Party Solutions

Summary

Chapter 29—Device Administration

Service Access Facility Under SVR4

Port Services

Port Monitors

Service Access Controller

SAF Administrative Commands and Files

SAF Administration and Management

[The ttymon Port Monitor](#)

[The listen Port Monitor](#)

[Device Administrative Tasks Under SVR4](#)

[Connecting Terminals and Modems](#)

[Connecting Terminals and Modems Under BSD](#)

[Connecting Terminals and Modems Under Linux](#)

[Connecting Printers](#)

[How the LP Print Service Works \(SVR4, Solaris 2.x\)](#)

[Setting Up Local Printing Services \(SVR4, Solaris 2.x\)](#)

[Setting Up Network Print Servers \(SVR4, Solaris 2.x\)](#)

[Managing Printers](#)

[Setting Up Local Printing on BSD Systems](#)

[Setting Up Network Printing Under BSD](#)

[Managing Print Services Under BSD](#)

[Connecting a PC to UNIX Systems](#)

[Connecting the PC Using COM Ports](#)

[Connecting to UNIX Using TCP/IP](#)

[Summary](#)

[Chapter 30—Mail Administration](#)

[Email Overview and Terminology](#)

[Common Mail Front Ends \(MUAs\)](#)

[Using elm](#)

[Using elm](#)

[Using pine](#)

[Using Netscape Mail](#)

[Using Microsoft Mail](#)

[Mail Transfer Agents \(MTAs\)](#)

[Using sendmail](#)

[Setting Up sendmail](#)

[Configuring sendmail](#)

[Conclusion](#)

[Other Mail Transfer Agents](#)

[Summary](#)

[Chapter 31—News Administration](#)

[Additional Sources of Information](#)

[Frequently Asked Questions \(FAQ\) Documents](#)

[News Transport Software Documentation](#)

[Request for Comments \(RFC\) Documents](#)

[Usenet Newsgroups](#)

[News Systems and Software](#)

[News Articles](#)

[Newsgroup Hierarchies](#)

[Where News Articles Live](#)

[The News Overview Database \(NOV\)](#)

[Distributing the News](#)

[Sharing News Over the Network](#)

Transferring News to Other Hosts

Host-to-Host News Transport Protocols

News Transport System Configuration Files

The User Interface—Newsreaders and Posting Programs

GUI Newsreaders

Planning a News System

Do You *Really* Want to Be a Usenet Site?

Shared News Versus One News Spool Per Host

Isolating the News Spool

Configuring Your News Spool's File System

Where Will You Get Your News?

Site Policies

Expiration Policies

Automatic Response to newsgroup/rmgroup Control

Messages

The ABCs of News Transport Software

Getting Your Hands on the Sources

An INN Distribution Roadmap

Learning About INN

Configuring INN—the config.data File

Building INN

Installing INN

Site Configuration

System Startup Scripts and news cron Jobs

Miscellaneous Final Tasks

Checking Your Installation and Problem Solving

Getting Help

Summary

Chapter 32—UUCP Administration

What Is UUCP?

Running Commands Remotely

Under the Hood, uucico

Setting Up UUCP

Receiving UUCP Calls

Initiating UUCP Calls

Administering UUCP Files

UUCP Daemons

Using UUCP

uucp

uux

uuto and uupick

Summary

Chapter 33—Administering FTP Services

Overview of FTP Protocol and Service

FTP Connections

Reliability of FTP

Operational Characteristics of FTP

A Sample FTP Session

FTP Internal Commands

FTP Responses

Unattended Transfers

FTP Proxy Transfers

Administering FTP

Setting Up FTP

Administering FTP Users

Anonymous FTP

Troubleshooting FTP

Connection Problems

Timeouts

File Transfer Problems

Debug Mode

Hash Mode

Server Problems

Summary

Chapter 34—Backing up and Restoring Your System

Generic Backup Commands

The tar Command

The dump Command

Using cpio

Using the backup and restore Commands

Using pax

HP-UX Backup Commands

fbackup

frecover

AIX Backup Systems

Using mksysb

Using sysback

Using savevg and restvg

Using rdump and rrestore

Sun Solaris Backup Commands

Using ufsdump and ufsrestore

Making Backups on SVR4 Systems

IRIX Backup Commands

BSD System Backup Commands

Linux System Backup Commands

Using taper

Using taper

Add-On Solutions

Summary

Part V—UNIX and the Internet

Chapter 35—Introducing Hypertext Transfer Protocol

(HTTP)

What HTTP Does

Protocol Definition

HTTP Example Operation

Messages, Headers, and Return Codes

HTTP Request Messages

Identifying and Overcoming HTTP Server Performance Problems

Connection Establishment—The Backlog Queue

Connection Termination

Communication Protocol Operation—TCP and Congestion Management

Providing Multiple Links Within an HTML Page

Using a Cache to Reduce Downloads

Looking to the Future

Persistent TCP Connections

New Request Methods

Summary

Chapter 36—Programming Web Pages—A Brief

Introduction to HTML

What Are URLs?

What Is Hypertext?

Description of HTML

Using a Web Browser

Coding HTML

A Minimal HTML Document

Font Control

Formatting Text

Lists

Extensions to Lists

Hypertext Tags

A Brief Description of Forms

A Brief Description of Tables

Frames

Tools

CGI Scripts and Java Applets

Special Characters

Tag Summary

Summary

Chapter 37—Monitoring Server Activity

Access Logs

Uses for Access Log Data

Access Log Format

Result Codes

Extended Logs

[Referrer](#)
[User-Agent](#)
[Error Logs](#)
[Basic Analysis](#)
 [General Statistics](#)
 [Periodic Reporting](#)
 [Demographic Reporting](#)
 [Page Reporting](#)
[Advanced Analysis](#)
 [Sessioning](#)
 [Pathing](#)
[Log Accuracy](#)
 [Adjusting for Caching](#)
[Analysis Tools](#)
 [Choosing an Analysis Tool](#)
 [Popular Tools](#)
 [Commercial Analysis Tools](#)
[Summary](#)

[Glossary](#)

[Index](#)

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

Table of Contents

Introduction

by Robin Burk

Welcome to the third edition of *UNIX Unleashed*!

Who Should Read This Book

As with previous editions, this book is written to be useful to a wide range of readers:

- People new to UNIX
- Anyone using UNIX that wants to learn more about the system and its utilities
- Programmers looking for a tutorial or reference guide to C, C++, Perl, GUI environments, and UNIX shells
- Systems administrators concerned about security and performance on their machines
- Anyone who wants to bring his or her UNIX skills and knowledge base up-to-date

UNIX continues to evolve as the user community widens. Expansive growth of the Internet and the World Wide Web has brought UNIX into mainstream corporations, Internet Service Providers, and desktop workstations around the world. If you are a technical professional, chances are that you will either work directly with UNIX or

interact with a UNIX-based system regularly.

What's New in This Edition

Our highly popular first edition brought comprehensive, up-to-date information on UNIX to a wide audience. As use of UNIX exploded along with the Internet, we offered a successful second edition in two volumes, one for Internet programming (server- and client-side) and the other for systems administrators. The second edition offered specific help in using the many variants of UNIX (many of them vendor-specific) that had emerged into use.

This third edition continues our tradition of providing timely information on the evolving world of UNIX. We've updated this edition of UNIX Unleashed to provide you current information regarding:

- The most frequently used UNIX variants and shells, including the updated version of `bash`
- Updated information on security issues and technologies you can use to protect your system
- The most popular Graphical User Interfaces (GUIs) and editors
- How to create and deploy useful programs of your own
- On-line, Internet and journal sources for more information about UNIX

The ongoing evolution of UNIX into the mainstream of computing has brought a certain degree of consolidation regarding the variants, shells and third party utilities which proliferated during the mid-1990's. UNIX lovers still have a wide variety of choices in these matters; however, this consolidation has allowed us to return to our popular, single volume format for *UNIX Unleashed*.

How UNIX Unleashed, Third Edition Is Organized

Part I, "Introduction to UNIX," is designed to get you started using UNIX. It provides you with general information regarding the organization of the UNIX operating system, how and where to find files, the standard commands and general information on networking and communicating with other systems. Part I also covers graphical user interfaces, and using `vi` and `emacs` to edit text.

Part II, "UNIX Shells," provides you with information regarding your choices for a user interface with UNIX. The most popular shells: Bourne, Bourne Again (BASH), Korn and C, are covered, as well as a comparison between them. In addition to providing your basic user interface, shells allow you to customize your computing environment in powerful ways.

Part III, "Programming," introduces the most popular program development languages: awk, Perl, C, C++. This section also discusses the make utility, a powerful rule-based method for controlling program compilation, as well as the standard utilities for managing source code libraries and controlling code revisions.

Part IV, "System Administration," covers the tasks needed to install, administer and manage UNIX on single-user or shared systems. In this section you will find solid

advice from experienced administrators on topics ranging from the duties of a systems administrator through user account management, periodic maintenance tasks, performance monitoring and optimization, security matters and managing UNIX upgrades. UNIX systems administrators have a number of very useful third-party utilities which are discussed here as well.

Part V, “UNIX and the Internet,” introduces the communications protocols, programming tools and concepts used with the World Wide Web. Here you will find information on HTML (used to program Web pages) and HTTP (the protocol used by Web servers). This section also contains information on administering Web servers and monitoring server performance.

Conventions Used In This Volume

This book uses the following typographical conventions:

- Menu names are separated from the individual menu options with a vertical bar (|). For example, “File|Save” means select the File menu and then choose the Save option.
- New terms appear in *italic*.
- All code appears in `monospace`. This includes pseudocode that is used to show a general format rather than a specific example.
- Words that you are instructed to type appear in **bold monospace**.
- Placeholders (words that stand for what you actually type) appear in *italic monospace*.
- Line of code that are too long to fit on one line in this book are broken at a convenient place and continued on the next line. A code continuation character [return] precedes the new line. All code that contains this character should be entered as one long line without a line break.
- An elipsis ... in code indicates that some code has been omitted for the sake of brevity.

Table of Contents

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Table of Contents](#)

About the Lead Author

Robin Burk has been fooling around with computers for more years than she would like to admit. A successful executive in entrepreneurial companies, she had led teams in the development of operating systems, network protocol stacks, language tools, and multimedia applications. She holds an undergraduate degree in physics and an MBA in finance and operations.

For relaxation, Robin breeds, trains, and shows dogs. She founded the Internet discussion list for English Cocker Spaniels and can be reached at robink@wizard.net.

About the Contributing Authors

Bill Ball, author of *Sams Teach Yourself Linux in 24 Hours* and Que's *Using Linux*, is a technical writer, editor, and magazine journalist. A reformed Macophile, he broke down and bought a PC after using Apple computers for nearly 10 years. Instead of joining the Dark Side, he started using Linux! He has published more than a dozen articles in magazines such as *Computer Shopper* and *MacTech Magazine*, and first started editing books for Que in 1986. He is forever grateful to Dale Puckett for lessons in how to use casts and code tight loops in 6800 assembler—and to William Boatman for lessons in how to cast and throw tight loops on the stream. An avid fly fisherman, Bill builds bamboo fly rods and fishes on the nearby Potomac River. He lives in the Shirlington area of Arlington County, Virginia.

Chris Byers is currently a consultant for a major credit card company. As a consultant and former disaster recovery specialist, he has many years of experience in the wide world of UNIX. He lives in South Jersey with his wife, his son (plus another child on the way), and their cat Amelia. He can be reached at southst@voicenet.com.

Lance Cavener is cofounder of Senarius. His function is to provide support to employers in eastern Canada. He is also the president and senior network Administrator of ASCIO Communications, a subsidiary of Senarius. He provides the public and businesses with Internet-related services. Lance has been actively involved in UNIX since 1990 as an administrator for corporate networks at various companies in eastern Canada. His work includes working with BIND/DNS, Sendmail, Usenet setup, Web servers, and UNIX security. He has also written various programs for SunOS, MS-DOS, MS Windows, and VMS.

Matt Coffey performs UNIX systems administration on SunOS, Solaris, IBM AIX, and HP systems. He has been responsible for a Sun Parallel Database Server that supported users nationwide. Currently he is supporting Sun legacy systems and an IBM SP2 Frame comprised of over 30 individual nodes.

David B. Horvath, CCP, is a senior consultant in the Philadelphia, Pennsylvania area. He has been a consultant for over 14 years and is also a part-time adjunct professor at local colleges teaching topics that include C programming, UNIX, and database techniques. He is currently pursuing an M.S. degree in dynamics of organization at the University of Pennsylvania (the degree should be received on December 22, 1998). He has provided seminars and workshops to professional societies and corporations on an international basis. David is the author of *UNIX for the Mainframer* (Prentice-Hall PTR), contributing author to *UNIX Unleashed Second Edition: System Administration Edition* and *Internet Edition* volumes (with cover credit), contributing author to *Red Hat Linux Second Edition*, contributing author to *Using UNIX Second Edition* (Que), and has written numerous magazine articles. When not at the keyboard, he can be found working in the garden or soaking in the hot tub. He has been married for over 11 years and has several dogs and cats. David can be reached at unx3@cobs.com for questions related to this book. No spam please!

For the past 15 years, **Chris Negus** has written or contributed to dozens of books on the UNIX system, computer networking, and the Internet. As a consultant, Chris worked at AT&T Bell Laboratories and UNIX System Laboratories on UNIX System V development teams. Later he worked with Novell's UnixWare system development. He was a contributing author for previous editions of *Using UNIX*, *UNIX Unleashed*, and *Microsoft Office Administrator's Desk Reference* for Macmillan Computer Publishing. Chris lives in Salt Lake City, Utah with Sheree, Caleb, and Seth.

William Pierce is an MIS director at DSI-CSS in Indianapolis, Indiana. His background includes system administration and very basic Oracle database administration on SMP UNIX systems. He has programmed in C, C++, COBOL, Scripting, Awk, and Assembler. He graduated from Ball State University in Muncie, Indiana in 1976. He currently lives in Anderson, Indiana with his wife, Jody, and their son, Jonathan.

Steve Shah is a systems/network administrator at the Center for Environmental

Research and Technology at the University of California, Riverside. In a parallel life he does research in the area of network bandwidth control algorithms and is hoping he can contribute his thesis back into Linux. Sadly, this leaves little time for his better half and personal cheerleader, Heidi, but he is adamant about rectifying the situation as soon as he gets his master's degree next year. Steve is also a contributing author of *Unix Unleashed, System Administrator's Edition* and *Red Hat Linux Unleashed, Second Edition*.

Sriranga Veeraraghavan has worked at Cisco Systems, Inc. in the area of network management since 1996. He enjoys developing software in C, C++, Java, Perl, and Shell on Linux and Solaris. He has contributed to several UNIX and Linux books. He takes pleasure in listening to classical music, reading classical literature, mountain biking, and playing Marathon on the home network with his brother, Srivathsa. Sriranga earned a bachelor's degree in engineering from the University of California at Berkeley in 1997 and is currently working toward a master's degree at Stanford University. He can be reached at ranga@soda.berkeley.edu.

Daniel J. Wilson is a senior principal consultant with Oracle Corporation and works out of the Indianapolis, Indiana practice. His background includes UNIX systems administration and Oracle database administration in both SMP and Clustered environments. He has extensive experience in UNIX systems and Oracle database performance tuning and troubleshooting. He has programmed in C, C++, COBOL, and SQL. He graduated from Ball State University in Muncie, Indiana in 1984. He currently resides just outside of Indianapolis with his wife, Angela, and their two children, Timothy and Emily. Many thanks to Linda Billingsley at PRC and to Ron James at Hewlett-Packard for providing the best support ever!

William D. Wood works at Software Artistry, Inc., as a support specialist on UNIX systems. He supports the Expert Advisor software it runs on SunOS, HP-UX, and IBM AIX. He has specialized in multi-systems and remote systems support since 1985, when he started work at the Pentagon. He has solely supported infrastructures that span the world and others that span just the U.S. He has also supported up to 80 UNIX machines at one time.

About the Technical Editors

Eric C Richardson (eric@rconsult.com) is a professional Webmaster with nine years Internet experience and over two decades of work with computers under his belt. He has worked as a college instructor, writer, consultant, and author in the Internet field. He currently oversees the Web sites and related extranet for Nabisco Incorporated. He is a member of the World Organization of Webmasters and serves on their national accreditation board. He enjoys spending time with his wife, Stacie, and their daughter, Katie. He has written or coauthored eight books about the Internet and has worked as editor on nearly a dozen.

Nalneesh Gaur works for TimeBridge Technologies as a systems engineer. His current work involves UNIX/Windows NT integration, Web design, and Internet/intranet security.

Acknowledgements

I'd like to begin by thanking the staff of Macmillan for the hard work that they contribute to their books. As with past book projects, they have been invaluable in assembling the chapter authors, responding to the inevitable project glitches and getting this book out to the UNIX community. Beginning with Jane Brownlow,—the acquisitions editor for this project—through the development, layout, and distribution staff, it's been a pleasure to work with you all.

I'd also like to acknowledge the special contribution of David Horvath to this edition. David and I worked together on the second edition of *UNIX Unleashed* and I was pleased to have him back on the team for this edition as well.

The support of my husband, Roger, has been invaluable as I've juggled book efforts, work responsibilities, and my dog hobby. Between keyboard and kennel, I'm often distracted in one project or another. Thanks, Love, for putting up with all this and buying groceries in the meanwhile. Yes, we *will* have a chance to take a weekend off together now that the book is finished!

Finally, I'd like to acknowledge the companionship of my dogs, the Laurelwood English Cocker Spaniels. Show dogs, hunters, couch potatoes, and accomplished beggars, they spent hours at my feet while I wrote and edited. Right now I'm surrounded by Haley, an AKC Champion of Record who will have her first litter of puppies soon and who gave up beloved walks by the lake for this book; by Fancy, who gave up field work; and by Bogie, who looks to be our next young Champion and who wants you to know he doesn't get nearly enough treats. <smile>

—Robin Burk

Dedication

To Roger and the Laurelwood English Cocker Spaniels, all of whom were patient and supportive while I took time for this book.

Robin Burk

Tell Us What You Think!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an Executive Editor for the Operating Systems team at Macmillan Computer Publishing, I welcome your comments. You can fax, email, or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author as well as your name and phone or fax number. I will carefully review your comments and share them with the author and editors who worked on the book.

Fax: 317-817-7070
E-mail: opsys@mcp.com
Mail: Jeff Koch
Operating Systems
Macmillan Computer Publishing
201 West 103rd Street
Indianapolis, IN 46290 USA

Table of Contents

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Part I

Introduction to UNIX

In This Part

- The UNIX Operating System
- Getting Started: Basic Tutorial
- The UNIX File System
- General Commands
- Getting Around the Network
- Communicating with Others
- Text Editing with vi and EMACS
- GUIs for End Users

Chapter 1

The UNIX Operating System

by Robin Burk and Rachel and Robert Sartin

In This Chapter

- What Is UNIX?
- Understanding Operating Systems
- The UNIX Operating System
- The History of UNIX

- UNIX and Standards
- Introduction to the UNIX Philosophy

Welcome to the world of UNIX. Once the domain of wizards and gurus, today UNIX has spread beyond the university and laboratory to find a home in global corporations and small Internet servers alike. This capability to scale up or down, to accommodate small installations or complex corporate networks with little or no modification, is only one of the characteristics that have won UNIX its popularity and widespread use.

As you'll see through the course of this book, UNIX is a rich and complex system built on simple, powerful elements. Although many more recent operating systems have borrowed concepts and mechanisms from UNIX, those who are most familiar with legacy mainframe environments, or whose experience is mostly limited to single-user personal computers, may find UNIX to be a bit intimidating at first. The best advice we can give is to take it slowly, but don't give up. As you read through these chapters and begin to use some of the features and utilities described in this book, you'll find that once-foreign ideas have taken clear and concrete shape in your mind.

Note:

One distinctive characteristic of UNIX compared to other operating systems is the fact that there are several flavors, or variants, of the operating system. Because the source code of the early versions was made available to a variety of computer manufacturers and third parties, many slightly different forms of UNIX coexist. Some are specific to a given hardware manufacturer; others differ in the utilities, configuration methods, or user interfaces they offer. This book points out the differences between the most commonly used UNIX variants.

Tip:

Throughout this book you will find specific details regarding how to accomplish tasks in each of the most popular versions of UNIX.

At its base, UNIX is both simple and elegant, with a consistent architecture that, in turn, underlies and guides the design of its many application programs and languages. If you are new to UNIX, we're a bit jealous of the fun you'll have as you begin to explore this fascinating environment for the first time. If you are a more experienced UNIX user, administrator, or programmer, this revised edition of *UNIX Unleashed* contains a wealth of information that can help you extend your UNIX use to Internet and World Wide Web applications, guard against hackers and other unauthorized intruders, and fine-tune your system management skills.

What Is UNIX?

UNIX is:

- A trademark
- A multitasking, multiuser operating system

- The name given to a whole family of related operating systems and their most common application, utility, and compiler programs
- A rich, extensible, and open computing environment

Let's take these one at a time. To begin with, UNIX is a trademark, which means that there is intellectual property associated with UNIX that is not in the public domain. Some versions of UNIX require a paid license for their use.

The term *UNIX* also refers to a powerful multitasking, multiuser operating system.

Once upon a time, not so long ago, everyone knew what an operating system (OS) was. It was the complex software sold by the maker of your computer system, without which no other programs could function on that computer. Application (user) programs asked the operating system to perform various functions; users seldom talked to the OS directly.

Today those boundaries are not quite so clear. The rise of graphical user interfaces, macro and scripting languages, suites of applications that can exchange information seamlessly, and the increased popularity of networks and distributed data all have blurred the traditional distinctions. Today's computing environments consist of layers of hardware and software that interact together to form a nearly organic whole.

At its core (or, as we say in UNIX, in the kernel), however, UNIX does indeed perform the classic role of an operating system. Like the mainframe and minicomputer systems that came before, UNIX enables multiple people to access a computer simultaneously and multiple programs and activities to proceed in parallel with one another.

Unlike most proprietary operating systems, however, UNIX has given birth to a whole family of related, or variant, systems. Some differ in functionality or origin; others are developed by computer vendors and are specific to a given line of machines; still others were developed specifically as shareware or even freeware. Although these various flavors of UNIX differ from one another to some degree, they are fundamentally the same environment. All offer their own version of the most common utilities, application programs, and languages. Those who use *awk*, *grep*, the Bourne shell, or make in one version of UNIX will find their favorites available on other UNIX machines as well.

Those who do not care much for these programs, however, are free to substitute their own approach for getting various computing jobs done. A salient characteristic of UNIX is that it is extensible and open. By extensible, I mean that UNIX allows the easy definition of new commands, which can then be invoked or used by other programs and terminal users. This is practical in the UNIX environment because the architecture of the UNIX kernel specifically defines interfaces, or ways that programs can communicate with one another without having been designed specifically to work together.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Understanding Operating Systems

An operating system is an important part of a computer system. You can view a computer system as being built from three general components: hardware, operating system, and applications (see Figure 1.1). The hardware includes physical pieces such as a central processing unit (CPU), keyboard, hard drive, and printer. Applications are why you use computers; they use the rest of the system to perform the desired task (for example, play a game, edit a memo, send electronic mail). The operating system is the component that on one side manages and controls the hardware and on the other manages the applications.

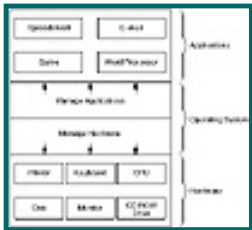


Figure 1.1. Computer system components.

When you purchase a computer system, you must have at least hardware and an operating system. The hardware you purchase can use (or run) one or more different operating systems. You can purchase a bundled computer package, which includes the hardware, operating system, and possibly one or more applications. The operating system is necessary to manage the hardware and the applications.

When you turn on your computer, the operating system performs a series of tasks, presented in chronological order in the next few sections.

Hardware Management, Part 1

One of the first things you do, after successfully plugging together a plethora of cables and components, is turn on your computer. The operating system takes care of all the starting functions that must occur to get your computer to a usable state. Various pieces of hardware need to be initialized. After the startup procedure is complete, the operating system awaits further instructions. If you shut down the computer, the operating system also has a procedure that makes sure that all the hardware is shut down correctly. Before turning off your computer again, you might want to do something useful, which means that one or more applications are executed. Most boot ROMs do some hardware initialization but not much. Initialization of I/O devices is part of the UNIX kernel.

Process Management

- After the operating system completes hardware initialization, you can execute an application. The operating system's job is to manage execution of the application. When you execute a program, the operating system creates a new process or instance of a thread of logic executing within the computer and its main memory. Many processes can exist simultaneously, but only one process actually engages the CPU at a given moment. The operating system switches between your processes so quickly that it can appear that the processes are executing simultaneously. This concept is referred to as *time-sharing* or *multitasking*.

When you exit your program (or it finishes executing), the process terminates, and the operating system manages the termination by reclaiming any resources that were being used.

Most applications perform some tasks between the time the process is created and the time it terminates. To perform these tasks, the program makes requests to the operating system, and the operating system responds to the requests and allocates necessary resources to the program. When an executing process needs to use some hardware, the operating system provides access for the process.

Hardware Management, Part 2

To perform its task, a process may need to access hardware resources. The process may need to read or write to a file, send data to a network card (to communicate with another computer), or send data to a printer. The operating system provides such services for the process. This is referred to as *resource allocation*. A piece of hardware is a resource, and the operating system allocates available resources to the different processes that are running.

Table 1.1 shows a summary of different actions and what the operating system (OS) does to manage them.

Table 1.1.Operating system functions.

Action	OS Does This
You turn on the computer.	Hardware management
You execute an application.	Process management
Application reads a tape.	Hardware management
Application waits for data.	Process management
Process waits while other process runs.	Process management
Process displays data onscreen.	Hardware management
Process writes data to tape.	Hardware management
You quit; the process terminates.	Process management
You turn off the computer.	Hardware management

From the time you turn on your computer until you turn it off, the operating system is coordinating the operations. As hardware is initialized, accessed, or shut down, the operating system manages these resources. As applications execute, request, and receive resources, or terminate, the operating system takes care of these actions. Without an operating system, no application can run, and your computer is just an expensive paperweight.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

The UNIX Operating System

The previous section looked at operating systems in general. This section looks at a specific operating system: UNIX. Traditionally used on minicomputers and workstations in the academic community, UNIX is now available on personal computers, and the business community has started to choose UNIX for its openness. This section looks at how UNIX fits into the operating system model.

UNIX, like other operating systems, is a layer between the hardware and the applications that run on the computer. It has functions that manage the hardware and functions that manage executing applications. What characterizes UNIX is its internal implementation and the interface that is seen and used by users. For the most part, this book ignores the internal implementation. If you want to know these details, many texts cover them. This book describes the interface in detail. The majority of UNIX users need to be familiar with the interface and need not understand the internal workings of UNIX.

The UNIX system is actually more than strictly an operating system. Beyond the traditional operating system components, a standard UNIX system includes a set of libraries and a set of applications. Figure 1.2 shows the components and layers of UNIX. Sitting above the hardware are two components: the file system and process control. Next is the set of libraries. On top are the applications. The user has access to the libraries and to the applications. These two components are what many users think of as UNIX because together they constitute the UNIX interface.



Figure 1.2. The layers of UNIX.

The part of UNIX that manages the hardware and the executing processes is called the kernel. In managing all hardware devices, the UNIX system views each device as a file (called a device file). This allows the same simple method of reading and writing files to be used to access each hardware device. The file system manages read and write access to user data and to devices, such as printers, attached to the system. It implements security controls to protect the safety and privacy of information. In executing processes the UNIX system allocates resources (including use of the CPU) and mediates accesses to the hardware.

One important advantage that results from the UNIX standard interface is the capability of a single application to be executed on various types of computer hardware without being modified. This can be achieved if the application uses the UNIX interface to manage its hardware needs. UNIX's layered design insulates the application from the different types of hardware. UNIX goes beyond the traditional operating system by providing a standard set of libraries and applications that developers and users can use. This standard interface allows application portability and facilitates user familiarity with the interface.

The History of UNIX

How did a system such as UNIX ever come to exist? UNIX has a rather unusual
 - history that has greatly affected its current form.

The Early Days

In the mid-1960s, AT&T Bell Laboratories (among others) was participating in an effort to develop a new operating system called Multics. In 1969, Bell Labs pulled out of the Multics effort, and members Ken Thompson, Dennis Ritchie, and others developed and simulate what later evolved into the UNIX file system.

As the team continued to experiment, they deployed their work to do text processing for the patent department at AT&T. Shortly afterward, the now famous C programming language was developed on and for UNIX, and the UNIX operating system itself was rewritten into C. This then radical implementation decision is one of the factors that enabled UNIX to become the open system it is today.

As a then-regulated telephone company, AT&T was not allowed to market computer systems. Nonetheless, the popularity of UNIX grew through internal use at AT&T and licensing to universities for educational use. By 1977, commercial licenses for UNIX were being granted. Later versions developed at AT&T (or its successor, UNIX System Laboratories) included System III and several releases of System V. The two

most recent releases of System V, Release 3 (SVR3.2) and Release 4 (SVR4) remain popular for computers ranging from PCs to mainframes. All versions of UNIX based on the AT&T work require a license from the current owner, UNIX System Laboratories.

Berkeley Software Distributions

In 1978, the research group turned over distribution of UNIX to the UNIX Support Group (USG), which had distributed an internal version called the Programmer's Workbench. In 1982, USG introduced System III, which incorporated ideas from several different internal versions of and modifications to UNIX, developed by various groups. In 1983, USG released the original UNIX System V, and thanks to the divestiture of AT&T, was able to market it aggressively. A series of later releases continued to introduce new features from other versions of UNIX, including the internal versions from the research group and the Berkeley Software Distribution.

The Computer Science Research Group at the University of California at Berkeley (UCB) developed a series of releases known as the Berkeley Software Distribution, or BSD. The original PDP-11 modifications were called 1BSD and 2BSD. Support for the Digital Equipment Corporation VAX computers was introduced in 3BSD. VAX development continued with 4.0BSD, 4.1BSD, 4.2BSD, and 4.3BSD, all of which (especially 4.2 and 4.3) had many features (and much source code) adopted into commercial products.

UNIX and Standards

Because of the multiple versions of UNIX and frequent cross-pollination between variants, many features have diverged in the different versions of UNIX. With the increasing popularity of UNIX in the commercial and government sector came the desire to standardize the features of UNIX so that a user or developer using UNIX could depend on those features.

The Institute of Electrical and Electronic Engineers (IEEE) created a series of standards committees to create standards for "An Industry-Recognized Operating Systems Interface Standard based on the UNIX Operating System." The POSIX.1 committee standardizes the C library interface used to write programs for UNIX. The POSIX.2 committee standardizes the commands available for the general user.

In Europe, the X/Open Consortium brings together various UNIX-related standards, including the current attempt at a Common Open System Environment (COSE) specification. X/Open publishes a series of specifications called the X/Open Portability. The MOTIF user interface is one popular standard to emerge from this effort.

The United States government has specified a series of standards based on XPG and POSIX. Currently, FIPS 151-2 specifies the open systems requirements for federal purchases.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

UNIX for Mainframes and Workstations

Many mainframe and workstation vendors make a version of UNIX for their machines. We discuss several of these variants (including Solaris from SunSoft, AIX from IBM, and HP-UX from Hewlett Packard) throughout this book.

UNIX for Intel Platforms

Thanks to the great popularity of personal computers, many UNIX versions are available for Intel platforms. Unfortunately, the UNIX industry has not settled on a complete binary standard for the Intel platform.

Source Versions of “UNIX”

Several versions of UNIX and UNIX-like systems have been made that are free or inexpensive and include source code. These versions have become particularly attractive to the modern-day hobbyist, who can now run a UNIX system at home for little investment and with great opportunity to experiment with the operating system or make changes to suit his or her needs.

An early UNIX-like system was MINIX, by Andrew Tanenbaum. His book *Operating Systems: Design and Implementations* describes MINIX and includes a source listing of the original version.

The most popular source version of UNIX is Linux (pronounced *ìlin nucks*). Linux was designed from the ground up by Linus Torvalds to be a free replacement for UNIX, and it aims for POSIX compliance. Linux has emerged as the server platform of choice for small to mid-sized Internet service providers and Web servers.

Introduction to the UNIX Philosophy

UNIX has its roots in a system that was intended to be small and supply orthogonal common pieces. Although most UNIX systems have grown to be fairly large, and monolithic applications are not uncommon, the original philosophy still lives in the core commands available on all UNIX systems. There are several common key items throughout UNIX:

- Simple, orthogonal commands

- Commands connected through pipes
- A (mostly) common option interface style
- No file types

Simple, Orthogonal Commands

The original UNIX systems were very small, and the designers tried to take every advantage of those small machines by writing small commands. Each command attempted to do one thing well. The tools could then be combined (either with a shell script or a C program) to do more complicated tasks. One command, called `wc`, was written solely to count the lines, words, and characters in a file. To count all the words in all the files, you would type `wc *` and get output like that in Listing 1.1.

Listing 1.1. Using a simple command.

```
$ wc *
351      2514      17021 minix-faq
1011      5982      42139 minix-info
1362      8496      59160 total
$
```

Commands Connected Through Pipes

To turn the simple, orthogonal commands into a powerful toolset, UNIX enables the user to use the output of one command as the input to another. This connection is called a *pipe*, and a series of commands connected by pipes is called a *pipeline*. For example, to count the number of lines that reference MINIX in all the files, one would type `grep MINIX * | wc` and get output like that in Listing 1.2.

Listing 1.2. Using a pipeline.

```
$ grep MINIX * | wc
105      982      6895
$
```

A (Mostly) Common Option Interface Style

Each command has actions that can be controlled with options, which are specified by a hyphen followed by a single letter option (for example, `-l`). Some options take option arguments, which are specified by a hyphen followed by a single letter, followed by the argument (for example, `-h Header`). For example, to print on pages with 16 lines each all the lines in the file `minix-info` that mention Tanenbaum, you would enter `wc minix-info | pr -l 16` and get output like that in Listing 1.3.

Listing 1.3. Using options in a pipeline.

```
$ grep Tanenbaum minix-info | pr -l 16

Feb 14 16:02 1994    Page 1

[From Andy Tanenbaum <ast@cs.vu.nl> 28 August 1993]
The author of MINIX, Andrew S. Tanenbaum, has written a book describing
Author:      Andrew S. Tanenbaum
subjects.ast (list of Andy Tanenbaum's
Andy Tanenbaum since 1987 (on tape)
Version 1.0 is the version in Tanenbaum's book, "Operating Systems: Design

$
```

No File Types

UNIX pays no attention to the contents of a file (except when you try to run a file as a command). The meaning of the characters in a file is entirely supplied by the command(s) that uses the file. This concept is

familiar to most PC users but was a significant difference between UNIX and other earlier operating systems.

Summary

UNIX has a long history as an open development environment. More recently, it has become the system of choice for both commercial and some personal uses. UNIX performs the typical operating system tasks, but also includes a standard set of commands and library interfaces. The building-block approach of UNIX makes it an ideal system for creating new applications.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Chapter 2 Getting Started: Basic Tutorial

by Robin Burk, Rachel and Robert Sartin, and Fred Trimble

In This Chapter

- Logging In to the System
- Logging Out
- Using Commands
- Configuring Your Environment
- Managing Your Password
- Working on the System
- Where to Find Information About Using UNIX
- UNIX Manual Pages
- Web Sites
- Newsgroups
- UNIX User Groups
- Professional Associations
- Publications

UNIX is a multiuser, multitasking environment. Unlike personal computers, UNIX systems are inherently designed to allow simultaneous access to multiple users.

Whether you are working with UNIX on a large, multiuser system or have a dedicated

UNIX-based workstation on your desk, the multiuser, multitasking architecture of the operating system influences the way you will work with the system and the requirements it will place on you as a user and a system administrator.

The purpose of this chapter is to acquaint you with the basics of UNIX from the user's point of view. Not all UNIX systems actually support multiple users with keyboards or terminals of their own. Some workstations are dedicated to a single person, and others function as servers that support multiple remote computers rather than end users. In all cases, however, UNIX operates as if it might be called on to furnish a fully multiuser, multitasking capability. For the purpose of this tutorial, we'll assume that you have a dedicated UNIX workstation on your desk.

Logging In to the System

Several people can be using a UNIX-based computer at the same time. For the system to know who you are and what resources you can use, you must identify yourself. In addition, because UNIX expects to communicate with you over a terminal (or a PC running terminal-emulation software), you must establish the ground rules that will govern the transfer of information. The process of establishing the communications session and identifying yourself is known as *logging in*.

Note:

UNIX actually distinguishes between a communications session and a login session, in that it is possible to log in as one user, log out, and log in again as another user without disrupting the communications session. Because an increasing number of people access UNIX systems from a PC, and for purposes of simplicity in this tutorial, we've treated the communications and login sessions as identical in this chapter. As you become more familiar with the UNIX environment and with utilities such as `telnet`, this distinction will become more important.

User Account Setup

After a UNIX system is booted, you cannot simply start using it as you do a PC. Before you can access the computer system, someone—usually the system administrator—must configure the computer for your use. If you are running UNIX on your PC at home, you will most likely need to do these things for yourself. If you are a UNIX novice trying to set up your home computer system, you can refer to Chapter 21, “UNIX Installation Basics.”

No matter who sets up your computer account, you must know two things before you can use the system: your username and your password. If you don't know what these are, you must stop and find out what has been assigned to you. The username is a unique name that identifies you to the system. It is often related to your real name, such as your first name, your last name, or a combination of first initial and last name (for example, “frank,” “brimmer,” or “fbrimmer,” respectively). If you get to request a username, try to choose something that makes others think of you alone and is not vague or common enough to cause confusion with others. The system administrator will verify that no one else on your system has this name before allowing you to have

it. The password that you request or that has been assigned to you is a temporary string that allows you to initially access the computer system. The initial password isn't of any real importance because you should change it to something of your choice the first time you log in to the system (see "Managing Your Password" later in this chapter).

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Logging In to the System

Now that you know your username (say it's "brimmer") and password (say it's "new_user"), you can access the system. When you sit down in front of a UNIX workstation, you are expected to log in to the system. This is true whether you are local to your UNIX system or are accessing a remote machine using a remote service such as `telnet`. The system prompts (asks) you for your username by displaying `login:`. You should then enter your username. Next, UNIX prompts you for your password by displaying `Password:`. Enter your password. As you type your password, don't be alarmed if the characters you type are not displayed on your screen. This is normal and is for your protection. No one else should know your password, and this way no one can look at your screen and see your password when you log in.

```
login: brimmer
Password:
Please wait...checking for disk quotas
```

```
Marine biology word of the day:
Cnidaria (n.) Nigh-DARE-ee-uh (L. a nettle) - a phylum of basically
radially symmetrical marine invertebrates including corals, sea
anemones, jellyfish and hydroids. This phylum was formerly known
as Coelenterata.
$
```

Tip:

Some keyboards have a key labeled Return. Some have a key labeled Enter. If your keyboard has both, Return is probably the correct key to use.

Tip:

On some systems, erase is #, and kill is @. On others, erase is Backspace or Delete, and kill is Control+U or Control+X.

If you typed everything correctly and the system administrator has everything set up correctly, you are now logged in and may use the system. If you get a message saying `Login Incorrect`, you may have typed your username or password incorrectly. If you make a mistake while entering your username, the Backspace key and the Delete key may not undo this mistake for you. The easiest thing to do is start over by pressing Enter twice to get to a new `login:` prompt.

Other error messages you might receive are `No Shell`, `No Directory`, or `Cannot Open Password File`. If you see any of these messages, or if multiple attempts at logging in always produce the `Login Incorrect` message, contact your system administrator for help.

Tip:

The `No Shell` message means that UNIX is not able to start the command interpreter, which was configured when your account was set up. Depending on the UNIX system, your login may complete successfully, and the default shell will be used. If this happens, you can use the `chsh` command, which changes the shell specified in your account. See Part II, “UNIX Shells,” for more information about various shells. The `No Directory` message means that UNIX cannot access your home directory, which was specified when your account was set up. Again, depending on the system, your login may complete successfully, placing you in a default directory. You may need to then enlist the help of the system administrator to create your home directory or change the home directory value for your account. See Chapter 3, “The UNIX File System,” regarding directories and, specifically, your home directory. The `Cannot Open Password File` message means that UNIX is having a problem accessing the system password file, which holds the account information (username, password, user ID, shell, group, and so on) for each user. If there is a problem with this file, no user can log in to the system. Contact your system administrator if you see this message.

If your system is configured to use a graphical user interface (GUI), you probably have a login screen. This screen performs the same function as the command-line prompts but is presented as a graphical display. The display probably has two boxes for you to fill in, each with a label. One box is for your username, and the other is for your password.

After Login Succeeds

After a successful login, several messages appear on your screen. Some of these may be the date and time of your last login, the system’s informative message (called the “Message of the Day”), and a message informing you whether you have (electronic) mail. The Message of the Day can be an important message to watch because it is one way that administrators communicate with the system users. The next scheduled down time (when no one can use the system) is an example of information that you might see here.

After all the messages scroll by, the system is ready and waiting for you to do something. This ready-and-waiting condition is signified by a prompt followed by a cursor. Typical prompts are `$` or `%`. The dollar-sign prompt is commonly used by Bourne and Korn shells, and the percent sign by C shells. The value of this prompt (your primary prompt) can be changed if you want. The person who set up your account may have already configured a different prompt value. To change this prompt, you need to change the value of the environment variable `PS1` (for Bourne and Korn) or `prompt` (for C shell). (See the section “Configuring Your Environment” in this chapter for details on environment variables.) The cursor (the spot on the screen where the next character you type is displayed) is commonly an underline (`_`) or a box, either of which can be blinking. The cursor you see may vary from system to system.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous Table of Contents Next

Different Privileges for Different Users

If you are administering your own personal system, it is still important for you to set up a personal account for yourself, even though your system will come configured with some type of administrative account. This account should be used to do systemwide administrative actions. Be careful when using this account because it has special privileges. UNIX systems have built-in security features. Most users cannot set up a new user account or do other administrative procedures. The user “root” is a special user, sometimes called superuser, which can do anything at all on the system. This high degree of power is necessary to fully administer a UNIX system, but it also allows its user to make a mistake and cause system problems. For this reason, you should set up a personal account for yourself that does not have root privilege. Then, your normal, day-to-day activities will affect only your personal environment, and you will be in no danger of causing systemwide problems. In a multiuser, nonpersonal environment, you will most likely have only user (and not superuser) privileges. This security is even more important when more than one person is involved because one mistake by the root can affect every user and the entire system.

UNIX also has security to help prevent different users from harming each other on a multiuser system. Each user “owns” his or her environment and can selectively let groups or all others have access to this work. If you are doing private work in one area that no one else should be allowed to see, then you should restrict access to the owner

(you). If you and your team members are working on a group project, you can restrict access to the owner (you) and everyone in your group. If this work should be shared with many or all people on the system, then you should allow access to everyone.

Logging Out

When you are done using the system, you should log out. This prevents other people from accidentally or intentionally getting access to your files. It also makes the system available for their use.

The normal way to log out from almost any shell is to type `exit`. This causes your shell to exit, or stop running. When you exit from your login shell, you log out. If you are using `csh`, you can also type `logout`; if you are in a login shell, then `csh` logs you out. Some shells, depending on your configuration, also log you out if you type the end-of-file character (typically Control+D; see “Working on the System” later in this chapter).

If you have a graphical user interface, your logout procedure may be different. Consult your manuals or online help to learn about logging out of your GUI.

Using Commands

During the login process, described in the “Logging In to the System” section, UNIX performs several actions that prepare you and the system for each other. These include performing system accounting, initializing your user environment, and starting a command interpreter (commonly called a *shell*). Commands are how you tell the system to do something. The command interpreter recognizes these commands and passes the information off to where it is needed. UNIX systems originally came with a command interpreter called the Bourne shell (usually referred to as `sh`, though some systems ship Korn or POSIX as `sh`—see the Note that follows). This shell is still available on most UNIX computer systems. A newer shell that is common to most UNIX systems is the C shell (referred to as `csh`). Another commonly used, but not as pervasive, shell is the Korn Shell (referred to as `ksh`). Among different shells, there is some variation of the commands that are available. Refer to Part II for details on these UNIX shells.

Note:

There are a number of different common shells on various UNIX operating systems. The most common are as follows:

- `sh` The Bourne shell is the most common of all the shells. (May be installed as `bsh`.)
- `ksh` The Korn shell is a derivative of the Bourne shell, which adds history and command-line editing. (Sometimes installed as `sh`.)
- `sh` The POSIX shell is much like the Korn shell. The POSIX standard requires it to be installed as `sh`. Some vendors install it as `/bin/sh`. Some put it in a special directory and call it `sh`, leaving the Bourne shell as `/bin/sh`.
- `csh` The C shell is based on the popular C language.

bash The Bourne Again shell is less common.

tcsh This is a version of the C shell with interactive command-line editing.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKKnowledge

[home](#)[account
info](#)[subscribe](#)[login](#)[search](#)[My
ITKnowledge](#)[FAQ/
help](#)[site
map](#)[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)[Table of Contents](#)[Next](#)

What Is a Command?

A UNIX command is a series of characters that you type. These characters consist of words that are separated by whitespace. *Whitespace* is the result of typing one or more Space or Tab keys. The first word is the name of the command. The rest of the words are called the command's *arguments*. The arguments give the command information that it might need, or specify varying behavior of the command. To invoke a command, simply type the command name, followed by arguments (if any); to indicate to the shell that you are finished typing and are ready for the command to be executed, press Enter.

Try it out. Enter the `date` command. The command's name is "date," and it takes no arguments. Therefore, type `date` and press Enter and see what happens. You should see that the computer has printed the current date and time. If the date or time does not match reality, ask the system administrator to fix it. How about trying a command that has arguments? Try the `echo` command. The name of the command is "echo," and it takes a series of arguments. The `echo` command then writes, or echoes, these arguments out to your screen. Try creating a command that writes your first and last names on the screen. Here is what these commands and output look like on our system:

```
$ date
Sat Aug 5 11:11:00 EST 1997
$ echo MyName
MyName
$
```

Note:

Some commands such as `echo` are part of the particular shell you are using. These are called built-ins. In this case, the commands are not standard from one shell to another. Therefore, if you learn one shell and then later have to (or want to) switch to using a different shell, you may have to learn new commands (and unlearn others). Other commands are standard UNIX commands and do not depend on what shell you are using. These should be on every UNIX system. The remaining commands are nonstandard UNIX and may or may not be on a particular UNIX system.

UNIX commands use a special type of argument called an *option*. An option commonly takes the form of a dash (made by using the minus sign key) followed by one or more characters. The options provide information to the command. Most of the time, options are just a single character following a dash. Two of the other lesser used forms are a plus sign rather than a minus sign, and a word following a dash rather than a single character. The following paragraph shows a common command with two of its common options. The `ls` command lists the files in your current directory.

First, try the `ls` command with no arguments. Then, try it with the `-a` option and note that the directory listing contains a few files that start with a period. These hidden files get listed by the `ls` command only if you use the `-a` option. Next, try the `ls` command with the `-l` option. This option changes the format of the directory listing so that each file is displayed along with some relevant details. Finally, try the `ls` command with both of these options, so that your command is as follows: `ls -a -l`.

```
$ ls
visible
$ ls -a
.      ..      .hidden  visible
$ ls -l
total 0
-rw-rw-rw-    1 sartin    uu              0 Mar  5 12:58 visible
$ ls -a -l
total 16
drwxrwxrwx    2 sartin    uu             1024 Mar  5 13:03 .
drwxr-xr-x   37 sartin    uu             3072 Mar  5 13:03 ..
-rw-rw-rw-    1 sartin    uu              0 Mar  5 12:58 .hidden
-rw-rw-rw-    1 sartin    uu              0 Mar  5 12:58 visible
$
```

A command developer often tries to choose option letters that are meaningful. Regarding the `ls` command, you might think of the `-a` as meaning that “all” files should be listed (including the special files starting with period). And you might think of the `-l` option as meaning a “long” directory listing because the format is changed so that each line contains one file along with its details. This makes for a longer listing.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Redirecting Input and Output

One pervasive concept in UNIX is the redirection of commands' input and output. Before looking at redirection, though, it is a good idea to look at input and output without modification. UNIX uses the word *standard* in this subject to mean the default or normal mode. Thus, UNIX has the term *standard input*, which means input coming from the default setting, and the term *standard output*, which means output going to the normal place. When you first log in to the system, and your shell executes, your standard input is set to be what you type at the keyboard, and your standard output is set to be your display screen. With this in mind, follow along with the example.

The `cat` command takes any characters from standard input and then echoes them to standard output. For example, type the `cat` command, with no arguments. Your cursor should be sitting on the next line without a prompt. At this point, the `cat` command is waiting for you to enter characters. You can enter as many as you want, and then you should specify that you are finished. Type a few words and then press Return. Now type the special character, Control+D (hold down the Control key while typing the D key). This is the "eof" control character. (See "Working on the System" later in this chapter for a description of control characters.) The words you typed should be on your screen twice—once caused by you entering them from the keyboard, and next as the `cat` command outputs them to your screen. This first step used standard input (from you typing on the keyboard), and standard output (the command results being printed on the screen).

```
$ cat
```

```
$
A few words
<CTRL><D>
A few words
$ cat > scotty
Meow, whine
meow
<CTRL><D>
$ cat < scotty
Meow, whine
meow
$ cat scotty
Meow, whine
meow
$
```

Although this simple case may not seem terribly useful yet, wait to see its use as you add redirection.

UNIX shells have special characters that signify redirection. Only the basics are covered here. Refer to Part II for details on each shell's redirection syntax. Output redirection is signified by the `>` character, and input redirection is signified by the `<` character. Output is commonly redirected to and input is redirected from a file. Now, continue with the rest of the example.

- Next, try the `cat` command using output redirection, leaving standard input alone. Enter `cat > filename`. The *filename* is a name of your choice. Once again, the `cat` command should be waiting for input (coming from standard input, which is your keyboard) at the beginning of the next line. Enter a few words, as you did before, press Return, and then, at the start of the next line, press Control+D. The words you typed didn't show up on your screen because you redirected the output of the `cat` command. The output was directed to go to the file *filename*. But how do you know it is there? To verify this, use the `cat` command with input redirection—which is the next order of business.

Caution:

`<Ctrl><D>` must be specified as the first character of an input line for it to be seen as “eof.”

To see the contents of the file *filename*, you want the input of the `cat` command to come from that file, and the output to go to the screen so that you can see it. Therefore, you want to redirect standard input and leave the output alone. Enter `cat < filename`. This time, the `cat` command did not wait for you—because you were not supplying the input. The file supplied the input. The `cat` command printed the contents of the file to the screen.

Tip:

Note the subtle distinction between these two commands: `cat >`

`filename` and `cat < filename`. You can remember the difference by verbalizing which way the sign points; does it point into the command or out of the command? Into the command is input redirection, and out of the command is output redirection.

The `cat` command allows you to specify a filename to use as input. Try showing the contents of the file this (more common) way: enter `cat filename`. Many commands are designed similarly—they have an argument that is used to specify a file as the input. Because of this common command design, redirecting input in this way is not nearly as common as redirecting the output.

UNIX was developed with the philosophy of having simple commands that do well-defined, simple things. Then, by combining these simple commands, the user could do powerful things. Pipes are one of the ways UNIX allows users to combine several commands. The pipe is signified by the vertical bar (`|`) symbol. A pipe is a means of taking the output of one command and redirecting it as the input of another command.

Say that you want to know how many files you have in your current directory. Recall that the `ls` command lists all the files in your current directory. You could then count the number of files. But UNIX has a command that counts the number of characters, words, and lines of input and displays these statistics. Therefore, you can combine these two commands to give you the number of files in your directory.

One way you could do this is as follows: `ls -l | wc -l`. Analyzing this command, you can see that the first part is something familiar. The `ls -l` command gives a directory listing in long format. In fact, it prints one file per line. The `wc -l` command gives the number of lines that are in the input. Combining the two commands via a pipe takes the output of the first command (the long directory listing) and gives it to the input of the second command. The output of the second command (which is not redirected—it goes to standard output) is displayed on your screen.

These basic forms of redirection allow you to be versatile as you learn a few commands at a time. Try to learn a command and use it with various options and arguments, then add redirection of input and output. And finally, combine commands with pipes. This approach should help you to feel comfortable with the commands and their varied uses.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

Brief

Full

□ [Advanced](#)

[Search](#)

□ [Search Tips](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Configuring Your Environment

To make using the shell easier and more flexible, UNIX uses the concept of an environment. Your environment is a set of values. You can change these values, add new values, or remove existing ones. These values are called environment variables—environment because they describe or define your environment, and variables because they can change.

Viewing and Setting Environment Variables

Every user’s environment looks a little different. Why don’t you see what your environment looks like? Type the `env` command with no arguments. The output formatting and variable names depend on which shell you are using and how your system is configured. A typical environment might include some of the following:

```
$ env
HOME=/u/sartin
LOGNAME=sartin
MAIL=/usr/mail/sartin
MANPATH=/usr/man:/usr/contrib/man:/usr/local/man
PATH=/bin/posix:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin
SHELL=/bin/sh
TERM=vt100
TZ=CST6CDT
$ echo $HOME
/u/sartin
```


\$

Sometimes the number of variables in your environment grows quite large, so much so that you don't want to see all the values displayed when you are interested in just one. If this is the case, you can use the `echo` command to show an environment variable's current value. To specify that a word you type should be treated differently—as a value of an environment variable—you immediately precede the variable name with a dollar sign (`$`). Be careful not to type any whitespace between the `$` and the word. One of the variables in the example is `HOME`. You probably have this variable in your environment, too. Try to display its value using `echo`.

Note:

If you use `csh`, some environment variables are automatically copied to and from `csh` variables. These include `HOME`, `TERM`, and `PATH`, which `csh` keeps in `home`, `term`, and `path`.

You can create a new environment variable by simply giving it a value. If you give an existing variable a value, the old value is overwritten. One difficulty in setting environment variables is that the way you set them depends on the shell you are using. To see how to set environment variables, look at the details about the shell you are using in Part II.

For your screen to display the output correctly, the environment variable `TERM` needs to have a reasonable value. This variable name comes from the times when terminals were used as displays (before PCs and graphics displays were common). Different terminals supported varying output control. Therefore, UNIX systems have various terminal types that they support. These are not standard, so you need to find out which terminal type to use from your support personnel. If you are using a PC to connect to a UNIX system, your PC is running a terminal emulation tool. Most of these tools have the capability to emulate several types of terminal. The important point here is to make sure that your emulator and your `TERM` variable are the same (or compatible). Start by seeing what your `TERM` variable is set to by entering `echo $TERM`. Refer to your PC terminal emulation manual and ask your system administrator for help to make sure that this is set up correctly.

Tip:

Many terminal emulators (including the Microsoft Windows “Terminal” program) support either “VT100” or ANSI standard terminal control sequences. Try setting `TERM` to `vt100` or `ansi` for this type of terminal emulator.

Using Shell Startup Files

Where do all these environment variables come from? Well, the system sets up various ones for you. And each user commonly sets up others during the login process. Yes, you may be doing this without even knowing it. During the startup, which happens at login, a shell is started. This shell automatically looks in a special place or two for some startup information. One of these places is in your home directory. The startup information in your home directory is found in special files. The specific shell you are using determines the name of the particular file. When the shell starts up, it examines this file and performs whatever actions are specified. One of the common actions is to give values to

environment variables. This action is called *initializing* or setting the values.

One environment variable that is commonly set in a user's shell startup file is the `PATH` variable (or lowercase `path` for C shell users). This variable's value is a list of places (directories) on the system where the shell should look to locate a command. Each command you type is physically located as a file somewhere on your file system. It is possible for the same command name to be located in different places (and to have either the same or different behavior when executed). Say that you have a program called `my_program` that is stored in your home directory, and your friend has a program called `my_program`, which is in her home directory. If you type `my_program` at the prompt, the shell needs to know where to look to find the storage location of `my_program`. The shell looks at the value of the `PATH` variable and uses the list of directories as an ordered directory search list. The first directory that has a `my_program` stops the search, and the shell executes that file. Because all files within a single directory must be unique, this gives a straightforward and sufficient method for finding executables (commands).

You probably want `$HOME/bin` to be toward the beginning of your `PATH` directory list, whereas you may want your friend's binary directory to be toward the end, or not listed at all. This way, when you type `my_program`, you will execute your `my_program` rather than hers. You can do all types of things in shell startup files in addition to setting environment variable values. If you want, you can add an `echo` command that prints out a greeting or reminds you to do something. One common item that is configured inside a shell startup file is the setup of your control characters. (See "Working on the System" later in this chapter.) These startup files are a powerful tool for you, the user of the shell, to configure the behavior of the shell automatically. Shell startup files are covered in more detail in Part II, "UNIX Shells."

Tip:

It is a good idea to create a `bin` directory in your `HOME` and store executables there. Include `$HOME/bin` in your path.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Configuring with rc files

The idea of having a file that is read on startup is not only used by the shells. In fact, many commands have special files containing configuration information that the user can modify. The general class of files is called `rc` files. This comes from the naming convention of these files. Most of these files end with the letters `rc`. Some of the more common files are `.exrc`, `.mailrc`, and `.cshrc`. These are all dot files; that is, they begin with a period (dot). The significance of starting a filename with a dot is that this file is not displayed during normal directory listing. If you want to see these files, use the `-a` option to the `ls` command. The `.exrc` file is used by the `vi` and `ex` editors (see Chapter 7, "Text Editing with `vi` and `Emacs`"). The `.mailrc` file is used by various electronic mail tools (see Chapter 6, "Communicating with Others"). The `.cshrc` file is the C shell startup file just discussed. The `rc` files are normally found in your home directory; that is, the default location for most of these files. Look at which `rc` files you have in your home directory (use the `ls -a` command). Then examine the contents of one of the files (use the `cat filename` command).

Your environment has a great effect on the use of your system. It is initialized during login with a shell startup file, and it grows and changes as you create new variables and change existing ones. Your environment affects every command you execute. It is important to get your environment set up to make your common actions easy. Spend the time to do this now, and you will be glad you did later.

Managing Your Password

During login, UNIX asks you to enter your password. If this is your first time on this computer, your password was configured by the system administrator. One of the first things you should do after logging in is change your password so that no one, not even the system administrator, knows what it is. You can do this via the `passwd` command. But before you do this, you should put some thought into what you want your password to be. Here are some points to consider:

1. It should be easy for you to remember. If you forget what your password is, no one, not even the system administrator, can look it up for you. The only thing the system administrator can do is to reset your password to a value. This wastes the administrator's time as well as yours.
2. It shouldn't be easy for anyone to figure out. Do not make it anyone's name or birth date, your username, or any of these spelled backwards. It is also wise to avoid something that appears in a dictionary. A good idea would be to include at least one nonalphabetic character (for example, a period or a dollar sign).
3. Make it a reasonable length. Some systems impose a minimum number of characters for a password. At least five characters is adequate. There isn't usually a limit as to the maximum number of characters, but only the first eight are significant. The ninth character and after are ignored when checking to see whether you typed your password correctly.
4. Change your password once in a while. Some systems check the last time you changed your password. If a time limit has been reached, you will be notified that your password has expired as you log in. You will be prompted to change it immediately and won't be allowed to log in until you successfully get it changed. This time limit is system imposed. Changing your password every few months is reasonable.
5. Don't write it down or tell it to anyone. Don't write it on scraps of paper. Don't tell your mother. Don't write it in your calendar. Don't write it in your diary. Don't tell your priest. Don't put it in a dialup terminal configuration file. Nowhere. Nobody. Maybe in your safe deposit box.

After you have thought about what you want your password to be, you can change it with the `passwd` command. Try it now; you can change your password as often as you want. Enter `passwd`. First, a prompt asking you to enter your old password is displayed. Type your old password and press Return. Next, you are prompted for your new password. Type it in and press Enter. Finally, you are prompted to re-enter your new password. This confirmation helps avoid changing your password if you made a typing error. If you make a mistake entering your old password, or if the two new password entries are not identical, then no change is made. Your old password is still in effect. Unless you make the same mistake both times that you enter the new password, you are in no danger of erroneously changing your password.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Brief](#) [Full](#)
[Advanced Search](#)
[Search Tips](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

Working on the System

Most keys on the keyboard are fairly obvious. If you type the S key, an s character appears on your screen. If you hold down the Shift key and type the S key, a capital s character (S) appears on your screen. In addition to the letters and digits, the symbols, some of which are above the digits, are familiar—such as the percent sign (%) and the comma (,). There are some UNIX and system-specific special characters in addition to these, with which you should become familiar. They help you manage your work and typing more effectively. The general type of character is called a *control character*. The name comes from the way in which you type them. First, locate the Control key—there should be one or maybe two on your keyboard. It may be labeled Ctrl or some other abbreviation of the word Control. This key is used like the Shift key. You press it but don't release it. While the Control key is pressed, you press another key, often a letter of the alphabet. If you press the Q key while holding down the Control key, this is called Control+Q and is commonly written ^Q (the caret symbol, which is found above the digit 6, followed by the alphabetic character).

Note:
When you see the notation ^Q, this does not mean to hold down Control and Shift while pressing Q. All you do is to hold down the Control key while pressing Q.

UNIX uses these control keys for various common keyboard actions. They can come in very handy. But the hard part is that different systems have different default Control key settings for these actions. Therefore, first you should find out what your current settings are, and then you can change them if you want. To look at what your current settings are, use the stty command. Enter stty -a at your command prompt and look at the results. Refer to the next example for an output of this command.

Tip:
If you're typing and nothing is showing on your screen, a ^S (or stop control character) inadvertently may have been typed. Try typing ^Q (or start control character) and see whether your typed characters now appear.

```
$ stty -a
speed 28800 baud; line = 0; susp <undef>; dsusp <undef>
rows = 44; columns = 120
intr = ^C; quit = ^\; erase = ^H; kill = ^X; swtch <undef>
eof = ^D; eol = ^@; min = 4; time = 0; stop = ^S; start = ^Q
-parenb -parodd cs8 -cstopb hupcl cread -clocal -loblk -crts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
ixon -ixany -ixoff -rtsxoff -ctsxon -ienqak
isig icanon iexten -xcase echo echoe echok -echonl -noflsh
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel -tostop tab3
$
```

Referring to the preceding example of `stty` output, look for the section that has the words `erase`, `kill`, and `eof`. Associated with each word is a control character. Find the similar part of your `stty` output. Keep this handy as you read the next topics.

Erase

Look at the word `erase` in the `stty` output. Next to this word is `^H` (verbalized as Control+H). Therefore, on my system, Erase, which means to back up over the last character typed, is done by typing `^H`. The Erase key is how you can fix your typing errors. Remember to look at your `stty -a` output because your system may be configured differently from this example. Try it out on your system. First, type a character you want to erase, say, an A. Now type your Control, Backspace, or Delete key associated with your Erase. If everything goes right, your cursor should have backed up to be on top of your A, and the next key you type will be where the A was. Try typing a correct series of keys, say `date<Return>`, to make sure that the control character actually worked. If you get a message similar to `A^Hdate not found`, then Erase is not working. To make it work correctly, pick the key you want associated with Erase and input the following (assuming that you have picked the Backspace key):

```
$ stty erase '^H'
$
```

Now, try entering the `date` command again and deleting the A in `dAte` and replacing it with a.

Note:

Depending on your display, erasing characters may not actually make the character disappear. Instead, it may reposition the cursor so that the next keystroke overwrites the character.

The Erase key is one of the most used control keys because typing without mistakes is difficult. Therefore, most keyboards have one or more special keys that are suited to this job. Look for keys labeled “Delete” or “Backspace.” One of these usually works as an erase key. Try typing some characters and seeing what happens when you then press Backspace or Delete. Normally the Backspace key is set up to be `^H`, so, if your `erase` is configured to be `^H`, Backspace most likely will work.

Kill

The Kill control character is similar to the Erase control character, in that it allows you to back up over typing mistakes. Whereas Erase backs up one character at a time, Kill backs up all the way to the prompt. Therefore, if you are typing a really long command and you realize, toward the end, that you forgot to do some other command first, you can start over by typing the control character associated with Kill. If you can't see what your Kill is set to, redo the `stty` command. In the `stty` output example, the system has `kill` set to `^X`. Again, remember that your system can be configured differently from this example. Now, try typing several characters followed by your Kill control character and see what happens. All the characters should be erased, and your cursor should be after the prompt.

Stop and Start

Two other commonly used control characters are Stop and Start. Their normal values are ^S and ^Q, respectively. Stop allows you to temporarily pause what is happening on your screen, and Start allows you to resume activity following a stop. This is useful if text is scrolling on your screen too fast for you to read. The Stop control character pauses the scrolling indefinitely so that you can read at your leisure. You might try this during your next login while the Message of the Day is scrolling by (see the section earlier in this chapter called “Logging In to the System”). But remember to be prepared and be swift because that text can scroll by quite quickly. Try to stop the scrolling, and then don’t forget to continue the scrolling by typing your Start control character.

Note:

On modern GUIs and high-speed connections, Stop and Start give poor control of output. This is because the output is so fast an entire screen may go by before you type the Stop character.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)

All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

•

Register for EarthWeb's
Million Dollar
Sweepstakes!



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

eof

The eof control character is used to signal the end of input. The letters eof come from end of file. The normal value of the eof control character is ^D, but be sure to verify this using the stty command. You can see how the eof character is used in the section called “Redirecting Input and Output” earlier in this chapter.

There are several other control characters that we will not look at here. Refer to the stty command in your system documentation for information. Or better yet, keep reading because we will show you how to find information about commands via the UNIX online help facility.

The stty command is also used to set the value of control characters. You can simply enter `stty erase ^H` to change your Erase character to Backspace. Do not enter a Control+H here; rather, enter `^H`. Some shells, including the original Bourne shell, treat the caret specially, so you may need the quotes. (Double quotation marks would also work in this example.) Try changing the value of your Erase control character and then use the `stty -a` command to make sure that it happened.

Tip:

Remember that typing the end of file character to your shell might log you out

Where to Find Information About Using UNIX

One of the most important things to know about UNIX or any computer system is how to get help when you don't know how to use a command.

Some information is available online. For instance, most UNIX commands give you a usage message if you incorrectly enter the command. This message shows you the correct syntax for the command. This can be a quick reminder of the arguments and their order. For many commands, you can get the usage message by using the option `-?`. The usage message often does not give you any semantic information.

The UNIX command `man` is another powerful tool that gives you complete online access to the UNIX manuals.

- Finally, a number of books, such as this one, and technical publications can be helpful. Also, because the UNIX operating system has had a profound impact on the development of the Internet, many Internet and Web sites exist that provide information on many facets of UNIX. In addition to identifying some important Web sites, this chapter identifies some key newsgroups, user groups, and publications to help you become a UNIX guru.

UNIX Manual Pages

Each UNIX system comes with a set of printed documentation. Most UNIX system administrators configure their systems to make this information available to their users. They are often referred to as *man pages*, because they are accessed with the `man` command. The `man` command is discussed later in this section. If the manual pages are not available on your system, see your system administrator.

Manual Page Organization

The manual pages are divided into eight sections. They are organized as follows:

- | | |
|----------------------|---|
| 1. Commands | This section provides information about user-level commands, such as <code>ps</code> and <code>ls</code> . |
| 2. UNIX System Calls | This section gives information about the library calls that interface with the UNIX operating system, such as <code>open</code> for opening a file, and <code>exec</code> for executing a program file. These are often accessed by C programmers. |
| 3. Libraries | This section contains the library routines that come with the system. An example library that comes with each system is the math library, containing such functions as <code>fabs</code> for absolute value. Like the system call section, this is relevant to programmers. |

- | | |
|-----------------------|---|
| 4. File Formats | This section contains information on the file formats of system files, such as <code>init</code> , <code>group</code> , and <code>passwd</code> . This is useful for system administrators. |
| 5. File Formats | This section contains information on various system characteristics. For example, a manual page exists here to display the complete ASCII character set (<code>ascii</code>). |
| 6. Games | This section usually contains directions for games that came with the system. |
| 7. Device Drivers | This section contains information on UNIX device drivers, such as <code>scsi</code> and <code>floppy</code> . These are usually pertinent to someone implementing a device driver, as well as the system administrator. |
| 8. System Maintenance | This section contains information on commands that are useful for the system administrator, such as how to format a disk. |

At first, knowing which section to search can seem bewildering. After a little practice, however, you should be able to identify the appropriate section. In fact, certain man page options allow you to span sections when conducting a search.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Click Here!

ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

The Manual Page Command

The man command enables you to find information in the online manuals by specifying a keyword. You can use it to do so in the following ways:

- List all entries whose one-line summary contains the keyword.
- Display all one-line descriptions based on the specified keyword.
- Display the complete manual page entry for the specified keyword.
- Search only the specified section, as outlined previously, for the specified keyword.

The simplest way to invoke the man command is to specify a keyword without any options. For example, if you want more information on the finger command, invoke the man finger command. On an HP system running the HP-UX version of UNIX, the following output is displayed:

```
NAME
  finger - user information program

SYNOPSIS
  finger [-options] user names...

DESCRIPTION
  Provides all types of data for each user name in the system:
  - all login names
  - all group names
  - ...

EXAMPLES
  To get information for all users:
  % finger
  To get information for the user's home directory and shell:
  % finger -h

SEE ALSO
  Other man pages at the present file is present.

FILES
  /usr/include /usr/lib

AUTHOR
  R. M. K. K. K. K.
```

Figure 2.1. Output to man finger command on an HP-UX machine.

Notice that the man page is divided into a number of sections, such as NAME, SYNOPSIS, and DESCRIPTION. Depending on the manual page, there are other sections, including DIAGNOSTICS, FILES, and SEE ALSO.

If you have a particular subject that you want to investigate in the online documentation but don't know where to start, try invoking the man command with the -k option. This searches all the descriptions in all eight of the manual page sections and returns all commands where there is a match. For example, suppose that you want to find out information related to terminals, but you aren't sure which command you should specify. In this case, specify the command man -k terminal. The following is sent to your screen:

```
clear (1) - clear terminal screen
ctermid (3s) - generate file name for terminal
getty (8) - set terminal mode
gettytab (5) - terminal configuration database
lock (1) - reserve a terminal
- lta (4) - Local Area Terminal (LAT) service driver
pty (4) - pseudo terminal driver
script (1) - make typescript of terminal session
```

You can then use the `man` command to find more information on a particular command.

Unfortunately, not all systems are configured to use this option. To use this feature, the command `/usr/lib/what is` must be in place. If it is not in place, see your system administrator.

Finally, when you invoke the `man` command, the output is sent through what is known as a pager. This is a command that lets you view the text one page at a time. The default pager for most UNIX systems is the `more` command. You can, however, specify a different one by setting the `PAGER` environment variable. For example, setting it to `pg` allows you to use features of the `pg` command to go back to previous pages.

Web Sites

The World Wide Web has an abundance of sites with useful information on UNIX. This section presents a survey of some helpful ones and provides the sites' URLs (Uniform Resource Locator) for access through a Web browser, along with a brief description of the kinds of information you'll find.

Book Lists

<http://www.amsoft.ru/unixbooks.html>

This site gives a bibliography of UNIX books. It also includes comments for some of the books. Most of the titles came from `misc.books.technical faq`.

http://www.cis.upenn.edu/~lwl/unix_books.html

This site provides a list of UNIX titles, along with a brief review of the book.

<http://wwwhost.cc.utexas.edu/cc/docs/unix20.html>

This site also contains a list of recommended UNIX titles, along with a discussion of the book contents. Its contents include books on introductory UNIX, text editing and processing, networking topics, advanced UNIX programming, and UNIX system administration.

Frequently Asked Questions (FAQ)

<http://www.cis.ohio-state.edu/hypertext/faq/usenet/unix-faq/faq/top.html>

This site contains the Usenet FAQ for questions in the `comp.unix.questions` and `comp.unix.shell` newsgroups. Due to its size, it is divided here into seven sections.

`gopher://manuel.brad.ac.uk/11/.faq/.unix`

This gopher site contains links to the FAQs for many UNIX-related topics, such as `rccs`, `sccs`, shells, and UNIX variants.

Finally, many UNIX FAQs have been reproduced and appear at the end of the *Internet Edition*.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Click Here!



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Tutorials

<http://www.tc.cornell.edu/Edu/Tutor>

This page contains a link called “UNIX Basics.” It includes sections on basic UNIX concepts for beginners, as well as tutorials on vi, emacs, and ftp.

<http://www.cs.indiana.edu/eip/tutorial.html>

This site contains a tutorial for the emacs editor and the elm mail program, along with a brief overview of some UNIX commands.

<http://www.cco.caltech.edu/cco/refguide/unixtutorial.html>

This site contains an extensive tutorial on UNIX, including logging in, manual pages, file and directory structure, mail, and job control. It ends with a summary of useful UNIX file commands.

http://www.eos.ncsu.edu/help/tutorials/brain_tutorials

Here, you will find a wide variety of practical tutorials, covering vi, emacs, email, ftp, tar, remote system access, network newsreader, advanced UNIX commands, and more.

<http://www.cs.indiana.edu/usr/local/www>

From this page, check out the UNIX link. It contains links to scores of other UNIX-related Web pages. Here, you will find information on Usenet FAQs, UNIX shell FAQs, IBM AIX, HP-UX, UNIX for PCs, Sun Systems, X Window, networking, security, Linux, UNIX humor, and much more.

<http://www.physics.orst.edu/tutorial/unix>

This site provides an excellent interactive UNIX tutorial called “Coping With UNIX: An Interactive Survival Kit.” It is sponsored by the Northwest Alliance for Computational Science and Engineering. The

tutorial runs best on a Web browser that supports frames and is Java-enabled.

<http://www.towson.edu/~michele/GUIDES/dirstruc.html>

This page contains an overview of the UNIX directory structure.

<http://www.uwsg.indiana.edu/uhelp/tutorials/toc.html>

This page contains a list of UNIX tutorials that can be found on systems at Indiana University and on outside systems as well. This page contains five links: beginning tutorials, intermediate tutorials, advanced topics and tutorials, quick references, and other references. Each link contains a number of UNIX references.

`gopher://hp.k12.ar.us/11/classes/unixbasics`

This gopher site contains five introductory UNIX lessons. Each lesson can be downloaded to your system.

<http://goophy.physics.orst.edu/~maestri/work/book.html>

This tutorial is named “Coping With UNIX, A Survival Guide.” It covers UNIX basics with a sense of humor.

<http://wsspinfo.cern.ch/file/doc/unixguide/unixguide.html>

This UNIX tutorial is extensive, containing information about many UNIX commands and utilities. It also contains information about the Internet and the World Wide Web.

<http://albrecht.ecn.purdue.edu/~taylor/4ltrwrd/html/unixman.html>

This tutorial, entitled “UNIX is a Four Letter Word, and vi is a Two Letter Abbreviation,” contains a humorous look at some basic UNIX commands and the vi editor.

<http://www.cs.curtin.edu.au/units/cg252-502/src/notes/html/contents.shtml>

This tutorial contains an outline of X Window programming concepts.

<http://www.uwsg.indiana.edu/usail/>

This tutorial is entitled USAIL, which stands for UNIX System Administration Independent Learning. It is designed to be an independent study course for prospective UNIX system administrators. It contains information on typical system administrator tasks, including installation, network administration, maintaining mail, backup and restore, and system performance. It contains a self-evaluating quiz.

<http://www.bsd.org/unixcmds.html>

This page contains a summary of UNIX commands.

<http://www.indiana.edu/~ucspubs/b017>

This site also contains a summary of UNIX commands.

<http://www.bsd.org>

This site contains many interesting links, including FAQ lists for most popular UNIX vendors, a DOS-to-UNIX command information sheet, and a number of links to other interesting sites.

<http://www.nda.com/~jblaine/vault>

This site, named “Jeff’s UNIXVault.” contains many links to interesting UNIX sites. Topics include “unices” (links to sites that focus on different flavors of UNIX), windowing systems on UNIX, shells, security, shell scripting, organizations, publications, UNIX and PCs, and UNIX newsgroups.

<http://www.perl.com>

As the name implies, this site contains a link to “The Perl Language Home Page.” It gives information on how to download the latest version of Perl, as well as documentation, a Perl FAQ, and Perl bug reports. It also contains links to other Perl sites, Perl mailing lists, Perl security information, and “The Perl Journal,” a newsletter dedicated to Perl.

<http://wagner.princeton.edu/foldoc>

While working with UNIX, you are likely to come across terms that you haven’t seen before. This site provides a free online dictionary of computing that will help you discover the meaning of such terms. It even provides a search mechanism for ease of use.

<http://rossi.astro.nwu.edu/lentz/misc/unix/home-unix.html>

This site contains a number of links to other interesting sites. It contains links to sites that cover networking issues, UNIX organizations, and various UNIX utilities, such as Perl, Tcl/Tk, Python, elm, and pine.

<http://alabanza.com/kabacoff/Inter-Links/guides.html>

Here, you will find many links to Internet guides. The site also contains a couple of UNIX-specific links.

<http://www.python.org>

Python is a portable, interpreted, object-oriented language that runs on many UNIX systems. This site contains a wealth of information, including links to other relevant sites.

<http://athos.rutgers.edu/~patra/unix.html>

This site contains many links to a wide variety of sites, including UNIX FAQ, UNIX security, Perl, UNIX system administration, and C and C++ programming.

<http://www.intersource.com/faqs/unix.html>

Here, you find descriptions of many UNIX and Internet commands.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb’s [privacy](#) statement.

▪

Register for EarthWeb's
Million Dollar
Sweepstakes!



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

FTP Sites

<ftp://ftp.gnu.ai.mit.edu>

This site contains all the programs available from the GNU Software Foundation.

<ftp://ftp.x.org>

This site contains a great deal of X Window software.

<ftp://src.doc.ic.ac.uk/computing/systems/unix>

Here, you find many UNIX utilities for programmers and system administrators alike.

Newsgroups

The UNIX operating system has played a major role in the development of the Internet and the World Wide Web. Consequently, a number of newsgroups are dedicated to various aspects of UNIX. For more information on how to participate in a newsgroup on the Internet, see Chapter 6, "Communicating with Others." Here is a listing of various UNIX discussion groups, in alphabetical order:

<code>cern.security.unix</code>	This newsgroup holds discussions on UNIX security at CERN. CERN is the European Particle Physics Laboratory and is where the World Wide Web originated.
<code>comp.lang.c</code>	Discussion about the C programming language.
<code>comp.lang.perl</code>	Discussion of the Perl programming language.
<code>comp.os.linux.advocacy</code>	These groups discuss the benefits of Linux, compared to other operating systems.
<code>comp.os.linux.answers</code>	This is a moderated discussion group that includes FAQs on Linux.
<code>comp.os.linux.hardware</code>	This group discusses Hardware compatibility and Linux.
<code>comp.os.linux.misc</code>	General information about Linux that is not covered in the other group.
<code>comp.os.linux.setup</code>	Linux installation and system administration.
<code>comp.security.unix</code>	Discussion of UNIX security.
<code>comp.sources.unix</code>	This contains postings of complete UNIX-oriented source code (moderated).
<code>comp.std.unix</code>	Discussion for the P1003 UNIX committee (moderated).
<code>comp.unix.admin</code>	This newsgroup discusses any topic related to UNIX system administration.
<code>comp.unix.aix</code>	This group is dedicated to discussions of IBM's flavor of UNIX (AIX).
<code>comp.unix.amiga</code>	Discussion of UNIX on the Commodore Amiga.
<code>comp.unix.aux</code>	Discussion of UNIX on the Apple Macintosh II computer.
<code>comp.unix.internals</code>	Discussions on UNIX internals.
<code>comp.unix.large</code>	UNIX on mainframes and on large networks.
<code>comp.unix.misc</code>	UNIX topics that seem to fit other groups.
<code>comp.unix.programmer</code>	This is a question and answer forum for people who program in a UNIX environment.
<code>comp.unix.questions</code>	This group is appropriate for newcomers to UNIX, with general questions about UNIX commands and system administration. It is one of the most widely used newsgroups on the Internet.
<code>comp.unix.shell</code>	This group discusses using and programming shells, including the Bourne shell (<code>sh</code>), Bourne Again shell (<code>bash</code>), C Shell (<code>csh</code>), Korn shell (<code>ksh</code>), and restricted shell (<code>rsh</code>).
<code>comp.unix.solaris</code>	Discussion of Sun's solaris variant of UNIX.
<code>comp.unix.sys5.r4</code>	Discusses UNIX System V Release 4.
<code>comp.unix.ultrix</code>	This group is dedicated to discussions of DEC's flavor of UNIX (<code>ultrix</code>).
<code>comp.unix.unixware</code>	Discussion about Novell's UnixWare products.

<code>comp.unix.user-friendly</code>	Discussion of UNIX user-friendliness.
<code>comp.unix.xenix.misc</code>	This group discusses general questions about Xenix, not including SCO.
<code>comp.unix.xenix.sco</code>	This group discusses Xenix from SCO (Santa Cruz Operation).
<code>comp.unix.wizards</code>	This is a moderated discussion group for advanced UNIX topics.
<code>comp.windows.x</code>	This is a discussion group for the X Window system.
<code>info.unix-sw</code>	UNIX software that is available via anonymous ftp.

UNIX User Groups

Joining a UNIX user group can be a great way to learn about UNIX. Many groups sponsor meetings, which often include a guest speaker as well as a forum to share ideas and experiences. Literally hundreds of UNIX user groups are in existence worldwide, so to list them here would not be practical. However, an excellent listing of UNIX user groups can be found by visiting http://www.sluug.org/~newton/othr_uug.html.

Professional Associations

Many professional associations are dedicated to the discussion and advancement of UNIX and Open Systems. This section gives information about some of the larger groups.

The Electronic Frontier Foundation

The Electronic Frontier Foundation (EFF) is a nonprofit organization dedicated to privacy and free expression, including social responsibility, for online media. For anyone interested in encryption, the Internet, and legal issues, this site is a must.

For more information, including membership, contact them at <http://www.eff.org>, or at the following:

The Electronic Frontier Foundation
 1550 Bryant Street, Suite 725
 San Francisco CA 94103-4832 USA
 Phone: (415) 436-9333
 Fax: (415) 436-9993
 Email: ask@eff.org

The Open Group

The Open Group, consisting of X/Open and the OSF (Open Software Foundation), is an international consortium of vendors and end users from many disciplines, including

industry, government, and academia. It is dedicated to the advancement of multivendor information systems. The group is actively involved in creating UNIX standards that incorporate widely accepted practices. For more information about The Open Group, including membership information, see its Web site at <http://www.osf.org>.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

USENIX

USENIX is the advanced computing system's technical and professional association. Since 1975, Usenix has supported the presentation and discussion of advanced developments in all aspects of computing. Each year, the association sponsors a number of conferences, symposia, and workshops covering a wide variety of topics. Their conferences are usually well attended and are aimed at the UNIX developer and researcher. USENIX also has a number of programs for colleges and universities, including student research grants, undergraduate software projects, scholarship programs, and student stipends so that students can attend USENIX events. They also provide a discount to students for the yearly dues.

USENIX also sponsors a technical group called SAGE (System Administrator's Guild). SAGE is an organization dedicated to the system administration profession. They publish a number of excellent booklets and pamphlets with practical information on system administration. They also publish a bi-monthly newsletter with useful tips for system administrators.

For more information, you can contact USENIX at <http://www.usenix.org>. In addition to membership information, you will find a number of useful articles from their publication, "i:login:," as well as papers published at previous conferences.

You can also contact USENIX with the following information:

The USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710 USA
Email: office@usenix.org
Phone: (510) 528-8649
Fax: (510) 548-5738

UniForum

UniForum is a professional association aimed at promoting the benefits and practices of open systems. Its members come from a wide variety of backgrounds, including engineers, developers, system administrators, system integrators, MIS directors, and CIOs. One of its stated goals is to provide a vendor-neutral approach to the evaluation and development of open systems. Each year, it sponsors a number a high-quality conferences and seminars. It also provides a number of technical publications that help you to understand open systems technologies. For example, you can access its online newsletter “UniNews Online,” through its Web site (<http://www.uniforum.org>). The association has a number of excellent technical articles on its Web site as well. Perhaps its most popular publication is the Open Systems Products Directory. It contains a description of thousands of open systems products and services and is free of charge to members.

For more information about UniForum, including how to become a member, contact them at <http://www.uniforum.com>, or at the following address or phone number:

UniForum Association
10440 Shaker Drive, Suite 203
Columbia, MD 21046
(410) 715-9500
(800) 255-5620 (US Only)

The X Consortium

This group was recently incorporated into The Open Group. It is dedicated to the X Window desktop environment and its role in the UNIX environment. It is a nonprofit organization for developing user interface standards and graphics technology in an open systems environment. For more information, including membership, visit its Web site at <http://www.x.org>.

Publications

Several useful UNIX-related publications are available. Some are available free of charge for qualified individuals.

UNIX Review

UNIX Review is a monthly magazine that covers the latest in UNIX technologies. It contains useful information for both UNIX developers and administrators. The magazine covers many aspects of UNIX-based systems, including software, hardware, peripherals, and support services. You can subscribe to *UNIX Review* by filling out an

online qualification form at its Web site (<http://www.unixreview.com>). In fact, you may qualify for a free subscription.

UNIX World

UNIX World is a subscription-free Web-based magazine. It provides practical tutorials on a wide variety of subjects. It contains a handy online search facility for searching for articles in its archives. You can find UNIX World at <http://www.unixworld.com/uworld>.

Sys Admin

Sys Admin magazine focuses on UNIX system administration. It provides in-depth coverage of multiple versions of UNIX on a variety of platforms. It covers a number of topics, including system monitoring, system security, backup and recovery, crash recovery, shell scripting, and X Window. You can subscribe to Sys Admin by filling out the subscription form on its Web page at <http://www.samag.com>.

Sun World

This is an online magazine with a focus on Sun products and services. Each month, it contains many practical articles for users and system administrators alike, such as the column “UNIX 101.” It also provides a means to search for back issues by keyword. You can access the magazine at <http://www.sun.com/sunworldonline/index.html>.

SunExpert

This is another magazine dedicated to the Solaris flavor of UNIX. Monthly columns include “Ask Mr. Protocol,” “UNIX Basics,” and “System Administration.” It recently merged with another UNIX publication entitled *RS/magazine*. It is free of charge to qualified readers. For subscription information, visit <http://www.netline.com/sunex/sunex.main.html>.

Summary

This chapter presented a basic tutorial in using UNIX and listed a number of additional resources for learning UNIX.

Because it was designed as a multiuser, multitasking operating system, UNIX provides a rich set of commands and capabilities. One characteristic of UNIX is that commands typically do simple tasks and can be combined in flexible, powerful ways.

The UNIX manual pages are accessible online and are a good place to start in learning how to use UNIX. In addition to books such as this one, a number of other resources are available to help you become a UNIX expert. The Internet contains a great deal of information, including tutorials, newsgroups, UNIX FAQ lists, and online publications. There are also a number of user groups and organizations dedicated to UNIX topics.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Register for EarthWeb's
Million Dollar
Sweepstakes!



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Chapter 3 The UNIX File System

by Robin Burk and Sanjiv Guha

In This Chapter

- File System Organization
- Naming Files and Directories
- File Types
- Naming Files and Directories
- Working with Directories
- File and Directory Permissions

Nearly all computers reference and use information that is stored but changeable. This information, whether it is executable instructions, a document, or other data, is organized into files.

In some older operating systems, it was common for the operating system itself to know something about the internal structure and purpose of the information in a file. UNIX treats every file the same way, as a stream of bytes (also known as characters) encoded in ASCII.

Of course, from the point of view of commands and other programs, the data within a UNIX file can and usually does have an internal structure beyond being a byte stream. A few of these structures, such as executable programs, are native to UNIX; that is, UNIX knows about the internal structure and typically has commands that interpret or manipulate them. However, the information in some files can't be processed by native UNIX commands. For example, a file containing data for a third-party database such as Oracle needs special programs to be processed, viewed, and so on. UNIX still can move, copy, and report on these files but won't be able to manage the information within them.

A file can reside on different media. A file can also be a permanent file on disk, a temporary file in the memory, or a file displaying or accepting data from the terminal. The functions usually performed on a file include the following:

- Opening a file for processing
- Reading data from a file for processing
- Writing data to a file after processing
- Closing a file after all the necessary processing has been done

File System Organization

As with most contemporary operating systems, UNIX supports the capability to organize files to make finding and naming them more feasible.

The most common file system organization, and the one adopted by UNIX, is called a *tree structure*. To follow the metaphor for a moment, a tree has a main trunk, which branches into subtrunks, which branch again, and so on until the final elements are the leaves themselves. Similarly, the file system on a UNIX machine has a root organization, or directory, which may be subdivided into subdirectories, each of which in turn may be subdivided into sub-subdirectories and so on. The final elements are individual files.

Just as the subtrunks of a tree do not necessarily branch and subdivide into exactly the same final pattern, so too the main subdirectories may support different numbers of sub-subdirectories. And, just as even a main trunk can directly grow leaves in many tree species, so too in UNIX a directory can hold files themselves as well as sub-subdirectories.

If you start at the root of a tree and trace it upwards as the trunk branches, and branches again, you will trace the path from the ground to a given leaf. Similarly, we speak of pathnames or paths as a shorthand way to trace the subdivisions of the UNIX file structure, which take us to a given file or files.

Naming Files and Directories

The pathname of a file is given in the following form

/Directory/subdirectory/sub-subdirectory / filename

The *root directory* begins all other pathnames and is denoted by a single / (forward slash). Figure 3.1 shows a typical directory tree structure. In a curious twist on the metaphor, it has become common to draw tree structures with the “root node” at the top of the picture.

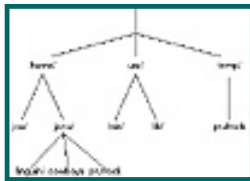


Figure 3.1. A directory tree.

UNIX is flexible regarding the naming of directories and specific files. By convention, however, a number of names are generally used for certain kinds of information, and most distributions of UNIX assume some or all of these names unless configured otherwise. Table 3.1 provides a list of standard directory names in the UNIX file system. This list is not exhaustive. A complete list would depend on the UNIX system you are working with.

Table 3.1.List of standard UNIX directories.

Directory name	Details about the directory
/	Root directory. This is the parent of all the directories and files in the UNIX file system.
/bin	Command-line executable directory. This directory contains all the UNIX native command executables.
/dev	Device directory containing special files for character- and block-oriented devices such as printers and keyboards. A file called <code>null</code> existing in this directory is called the bit bucket and can be used to redirect output to nowhere.
/etc	System configuration files and executable directory. Most of the administrative, command-related files are stored here.
/lib	The library files for various programming languages such as C are stored in this directory.
/lost+found	This directory contains the in-process files if the system shuts down abnormally. The system uses this directory to recover these files. There is one <code>lost+found</code> directory in all disk partitions.
/u (or) /home	Conventionally, all the user home directories are defined under this directory.
/usr	This directory has a number of subdirectories (such as <code>adm</code> , <code>bin</code> , etc, and <code>include</code> . For example, <code>/usr/include</code> has various header files for the C programming language.

Because the full pathname is required to fully specify a file in UNIX, it is acceptable (and often desirable) for subdirectory and filenames to be duplicated in various

portions of the tree. For example, if you kept mail received by month and day, you could create directories named january, february, march, and so on. Under each of these directories, you could create files such as day01, day02, and day03. The same holds true for directories. That is, you can have the same directory name under different directories.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Although the file tree in most UNIX machines is complex, typically you will be working in one local area of it at any given time. UNIX provides a way to identify the local region you're working in by differentiating between the current directory and pathnames relative to that directory, on the one hand, and absolute pathnames on the other hand.

The current directory is that fully specified pathname whose contents (subdirectories or data files) are considered local to you, and which are considered to be the location of any files to which you might (explicitly or implicitly) refer while executing programs or issuing commands.

Absolute pathnames give the full path, beginning at the root. The current directory is a node in the tree whose contents you are examining or manipulating. For example, if you were in the directory named `january` and you executed the command

```
ls -l day01
```

you would get the attributes of the file `day01` under the `january` directory, which means that UNIX looked in the directory you were currently in to find out whether the file you specified was present. All the commands in UNIX use the current directory to resolve the filename if the filename does not have directory information.

The relative pathname is simply a shorthand way of specifying a pathname in terms of its location relative to the current directory you are in.

For example, if you were in the directory `january` and wanted to get the attributes of file `day01` in directory `february`, you could specify the absolute pathname of the file:

```
ls -l /home/username/february/day01
```

However, the current directory might exist at several levels of subdirectory from the root, in which case this would be awkward to remember and type. UNIX uses the special characters `..` (two consecutive periods) as a relative pathname to indicate the parent directory, or the directory of which the current directory is an immediate subdivision. For example, if you were in the directory `/home/username/january`, `..` (two consecutive periods) would be a shorthand way to refer to the `/home/username` directory (which is the parent directory of `/home/username/january`), and `../..` in the relative pathname would indicate the `/u` directory. This convention provides an easy way to refer to nodes that are nearby the current directory.

File Types

This section discusses the various file types available in UNIX. You might be familiar with some of these types of files, such as text documents and source files.

Regular Files

Regular files are permanent in nature and contain data such as source code for a program, mail received from the boss, and a letter you are writing to a friend. These files almost always contain text information. In these files, the data is organized into *records* delimited by the newline character. Most UNIX commands support text processing. However, keep in mind that text files are not the only type of regular files. Some files have a stream of bytes without any newline characters. Although UNIX is built to process text documents, the data in these files cannot be processed by UNIX.

The following are examples of some of the regular files:

- `prog.c` is a file containing a C source program.
- `prog.cbl` is a file containing a COBOL source program.
- `prog.exe` is a file containing executable code for a program.
- `invite.doc` is a file containing an invitation to a party from a coworker.

Note:

The examples provided here follow the usual UNIX file naming conventions. However, these are just conventions, not the rules. So, it is possible for someone to name a file `prog.c`, even though it contains a letter to his or her boss.

Files have attributes or characteristics that are known to UNIX. The attributes of a file can be displayed using the `ls` command:

```
ls -al testfile
```

The result is

```
rw-r--r--  2 username  staff      1012 Oct 30 18:39 testfile
```

UNIX keeps track of the file attributes using a data structure called *inode*. Each inode in the system is identified by a number called the *inode number*. Each file in the system has an associated inode that contains information such as the following:

- Ownership details of a file
- Permission details of a file
- Time stamps of a file (date and time of creation, date and time of modification, and so on)
- Type of file

A number of time stamps are associated with a file. These times are

- Last access time
- Last modification time
- Last inode modification time

The last access time changes whenever you perform any operation on a file. The last modification date changes when the contents of the file are modified. The last inode modification time is when any of the information stored in the inode changes.

Note:

Some UNIX versions, for instance, AIX, do not modify the last access time when you execute them.

Directory Files

A *directory file* is a special file that contains information about the various files stored in the directory, such as file locations, file sizes, times of file creation, and file modifications. This special file can be read only by the UNIX operating system or programs expressly written to do directory processing. You may not view the content of the directory file, but you may use UNIX commands to inquire about these attributes of the directory. When you ask UNIX to process a filename, UNIX consults (and may modify) the specified directory to obtain information about the file. In each directory, you always find two files:

1. . (single period)
2. . . (two consecutive periods)

The single period (.) refers to the current directory, and the two consecutive periods (. .) refer to the directory one level up (sometimes referred to as parent directory).

An example of the directory attributes of a `testdir` are presented here:

```
drwxr-xr-x    2  username      writer           512 Oct 30 18:39 testdir
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
 All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

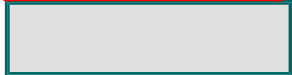


Brief Full

☐ [Advanced](#)

[Search](#)

☐ [Search Tips](#)



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

The `hostname` argument of `telnet` is optional. If you do not use the host computer name as part of the command, you are placed at the `telnet` prompt, usually, `telnet>`. A number of subcommands are available from the `telnet>` prompt. Some of these subcommands are as follows:

Subcommand	Description
<code>exit</code>	Closelose the current connection and return to the <code>telnet></code> prompt if the subcommand <code>open</code> was used to connect to the remote host. If, however, <code>telnet</code> was issued with the <code>host-name</code> argument, the connection is closed, and you return to where you invoked the <code>telnet</code> command.
<code>display</code>	Display operating arguments.
<code>open</code>	Open a connection to a host. The argument can be a host computer name or address.
<code>quit</code>	Exit <code>telnet</code> .
<code>set</code>	Set operating arguments.
<code>status</code>	Print status information.
<code>toggle</code>	Toggle operating arguments (<code>toggle ?</code> for more).

If, for instance, you are currently logged in on `box1`, you can execute the following command to log in to `box2`:

```
telnet box2
```

As a response to this command, `box2` responds with a login screen at which you can enter your user ID and password for `box2`. After completing your work on `box2`, you can come back to `box1`.

passwd

When your user ID is first set up in a computer, the system or security administrator will assign you a temporary password. The first time you try to log in, the system will ask you to change your password; you will use the new password for all subsequent logins.

As a security precaution, it is a good practice to modify your password frequently using the `passwd` command. `passwd` prompts you for your current password, and then prompts you twice more to enter the new password. The new password must follow the rules set up at your installation; these rules typically specify the following:

- The minimum number of alphabetic characters.
- The maximum number of times a single character can be used in a password.
- The minimum number of weeks that must elapse before a password can be changed.
- The maximum number of weeks after which the password must be changed. The system will prompt you to modify the password when this happens.

Passwords control access to your account and your files. Select your password carefully to prevent it from being guessed or from being easily generated by an automated hacker attack.

- Avoid using proper nouns (such as personal names) or strings of characters followed by numbers (such as `xyz01`).
- Use both uppercase and lowercase letters mixed with numbers.
- Use at least 7 characters.
- Don't write down your password.
- Choose a password that you can type quickly.

exit

When you log in to a UNIX system, you are always placed in a shell. The `exit` command allows you to exit the current shell.

You can also exit your current shell by pressing `Ctrl+D`. (Hold down the `Ctrl` key and the `D` key together.) If you are on the command line and press `Ctrl+D`, you will be logged off.

Locating Commands

Most commands are executed by programs stored in executable or script files. Each user has a *search path*—a list of directories in the order to be searched for locating commands.

The default search paths set by the installation usually include standard directories such as `/bin`, `/usr/bin`, and other installation specific directories. You can modify the search path for your environment as follows:

- Modify the `PATH` statement in the `.profile` file (Korn shell and Bourne shell).
- Modify the `set path=(...)` statement in the `.cshrc` or `.login` file (C shell).

Add the directory that contains your commands, or any commands you have modified, to the beginning of the path. They will be found first and executed first.

which

The `which` command can be used to find and report the directory in which a particular command exists. For example, to find out where the `which` command resides, enter this command:

```
which which
```

The system responds with the following message, meaning that the command `which` exists in the directory `/usr/bin`:

```
/usr/bin/which
```

whence

The Korn shell offers the `whence` command, which supports a flag `-v` to produce output in a verbose form.

```
whence -v which
```

The preceding command produces the following output:

```
which is /usr/bin/which
```

However, if you have aliased commands, this command shows the underlying command string behind the alias.

where

The `where` command is used to obtain the full path name of one or more files or directories. No flags are associated with the `where` command. For example, to find

the path name of the current directory, issue the following command:

where

Assuming that you are in the directory `/u/testuser`, the preceding command results in this output:

```
box1:/u/testuser
```

Determining Command Usage

UNIX provides you with online help to learn about various commands and their options and flags. You may be familiar with the most often used commands and their various options, but to find out about less popular commands or command usage, you can use the online help provided by UNIX.

man

The `man` command is used to display the online UNIX manual page for a given command, file, subroutine, and so on. If you do not know the full name of the item for which you want help, you can use the UNIX wildcard to specify the object name. Following are some of the flags and arguments you can use with the `man` command:

Flag	Meaning
<code>-k keyword</code>	Provide a list of summary information about manual sections in the keyword database for the specified keyword.
<code>-f command</code>	Provide details associated with the specified command. The root user must set up the file <code>/usr/man/what is</code> before you can use this option.
<code>-M path</code>	Specify the search path for the <code>man</code> command.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc. All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

The following is a list of sections in the manual page that can be specified:

- 1—Commands
- 2—System calls
- 3—Subroutines
- 4—File formats
- 5—Miscellaneous
- 7—Special files
- 8—Maintenance

For example, if you want to find out about the `find` command, execute the following command:

```
man find
```

To find out about file system related keywords, execute the following command:

```
man -k filesystem
```

Administration

Only the UNIX system administrator can perform certain functions such as starting the system, shutting down the system, setting up new user accounts, monitoring and maintaining various file systems, installing new software on the system, and more. The

following sections discuss the commands necessary to fulfill these duties.

install

The `install` command is used to install new versions of current software programs or brand new software programs. The basic function of the `install` command is to copy the binary executable and any associated files to the appropriate directories.

Note:

For new software, the default permissions are set to 755 with owner and group both set to `bin`. The default directories searched to find whether the command is present are `/usr/bin`, `/etc`, and `/usr/lib`—in that order.

Following are some of the flags that can be used for the `install` command:

Flag	Meaning
<code>-c directory</code>	Install the files in the specified directory, if, and only if, the command does not already exist.
<code>-f directory</code>	Install the files in the specified directory, even if the command already exists in that directory.
<code>-G group</code>	Set the group of installed files to the specified group instead of the default group <code>bin</code> .
<code>-O owner</code>	Set the ownership of the installed files to the specified user instead of the default owner <code>bin</code> .
<code>-o</code>	Save the current version with the prefix <code>OLD</code> ; that is, if the name of the file is <code>sample</code> , it is saved as <code>OLDsample</code> .
<code>-i</code>	Specify that you do not want to search the default directories but want to search the command line specified directories.
<code>-n directory</code>	Specify that you want to install the files in the specified directory if they are not found in the search directories.

You can execute the following command to install the file `sample_program` in the directory `/usr/bin` (default directory):

```
install sample_program
```

Assuming that `sample_program` already exists in the `/u/testuser/exe` directory, you can install the new version by executing the following command:

```
install -f /u/testuser/exe sample_program
```

shutdown

To shut down the machine in an orderly fashion, use the `shutdown` command from the `root` user. By default, this command brings down the system to single-user mode from multiuser mode. Following is a list of some of the flags that can be used with the `shutdown` command:

Flag	Meaning
-h	Halt the operating system completely.
-i	Display interactive prompts to guide the user through the shutdown.
-k	Simulate a system shutdown.
-m	Bring the system down to maintenance (single user) mode.

It is also possible to specify the time when restart is to be done by specifying a future date or relative time. In that case, the system sends messages to the users periodically about the impending shutdown.

ulimit

The `ulimit` command, available in the Korn and Bourne shells, sets limits on certain resources for each process. The corresponding command in the C shell is `limit`. There are two types of limits:

- *Hard limits* are those defined in the resources on a system-wide basis and can be modified only by `root` authority.
- *Soft limits* are the default limits applied to a newly created process. Soft limits can be increased to a system-wide hard limit.

Following are the flags that can be used with the `ulimit` command:

Flag	Meaning
-a	Show the soft limits.
-Ha	Show how the hard limits.
-c <i>size</i>	Set the <code>coredumpsize</code> in blocks.
-t <i>size</i>	Set the CPU time in seconds.
-f <i>size</i>	Set the maximum file size in blocks.
-d <i>size</i>	Set the maximum size of the data block in kilobytes.
-s <i>size</i>	Set the maximum size of the stack in kilobytes.

`-m size`

Set the maximum size of the memory in kilobytes.

To obtain the current setting of the hard limits, execute the following command:

```
ulimit -Ha
time(seconds)      unlimited
file(blocks)       4097151
data(kbytes)       unlimited
stack(kbytes)      unlimited
memory(kbytes)     unlimited
coredump(blocks)   unlimited
```

To obtain the current setting of the soft limits, execute the following command:

```
ulimit -a
time(seconds)      unlimited
file(blocks)       4097151
data(kbytes)       2048576
stack(kbytes)      82768
memory(kbytes)     909600
coredump(blocks)   102400
```

umask

The `umask` command is used by the system administrator to set the default permissions to be assigned to each file created by a user. You, as a user, can modify this default setting.

Three groups of permissions are associated with a file or directory: owner, group, and world (sometimes referred to as others). The permissions for these three groups are assigned using octal numbers—one octal number for each group. The values for each group depend on the following three bits:

- read bit (0 or 1)
- write bit (0 or 1)
- execute bit (0 or 1)

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#) [Table of Contents](#) [Next](#)

Using binary arithmetic, the value can vary from 0 (all bits having a value of zero) to 7 (all bits having a value of 1).

The value associated with `umask` can be used to derive the value of the default permission by subtracting it from 777. For example, if the value of `umask` is 022, then the value of the permission is $777 - 022 = 755$ (read, write, execute for owner; read, execute for group; and read, execute for the world).

To obtain the default value of the `umask`, execute the following command:

```
umask
```

For example, to set the value of the permission to 751, you set `umask` to the value 026 ($777 - 751$), like this:

```
umask 026
```

Process-Related Commands

In UNIX, a *process* is a program that has its own address space. Simple commands such as `umask` have only one process associated with them; a string of commands connected by pipes has multiple processes associated with it.

Processes can be categorized into the following broad groups:

- *Interactive processes* are invoked at the terminal and run either in the foreground or in the background. In a *foreground process*, the input is accepted from standard input, output is displayed to standard output, and error messages are sent to standard error. While executing a process in the background, the terminal is detached from the process so that it can be used for executing other commands. It is possible to move a process from the foreground to the background and vice versa.
- *Batch processes* are not submitted from terminals. They are submitted to job queues to be executed sequentially.
- *Daemons* are never-ending processes that wait to service requests from other processes.

In UNIX, each process has a number of attributes associated with it. Following is a list of some of these attributes:

- *Process ID* is a unique identifier assigned to each process by UNIX.
- *Real User ID* is the user ID of the user who initiated the process.
- *Effective User ID* is the user ID associated with each process and determines the process's access to system resources. Under normal circumstances, the Real User ID and the Effective User ID are one and the same.
- *Real Group ID* is the group ID of the user who initiated the process.
- *Effective Group ID* is the group ID that determines the access rights. The Effective Group ID is similar to the Effective User ID.
- *Priority (Nice Number)* is the priority associated with a process relative to the other processes executing in the system.

kill

- The `kill` command is used to send signals to nonforeground processes. The `kill` command terminates the process unless the process has the appropriate logic to accept the signal and respond differently. Unless you are logged in as the `root` user, you can kill only the processes initiated by you.

The flags associated with the `kill` commands are as follows:

Flag	Meaning
-l	Obtain a list of all the signal numbers and their names that are supported by the system.
-signal number	The signal number to be sent to the process. You can also use a signal name in place of the number. The strongest signal you can send to a process is 9 or <code>kill</code> .

The argument to the `kill` command is the Process ID (PID). You can specify more than one PID as arguments to kill more than one process. The value of the PID can be one of the following:

- PID greater than zero to kill specified processes.
- PID equal to zero to kill all processes whose Process Group ID is the same as the Process Group ID of the user initiating the `kill` command.
- PID equal to -1 to kill all processes owned by the Effective User ID of the user initiating the `kill` command.
- PID equal to a negative number but not -1 to kill processes whose PID is equal to the absolute value of the number specified.

Use the `kill` command carefully because it immediately ends any specified processes.

If you are running a command in the background and think it has gone into a loop, you will want to terminate it. If the process number is, for example, 2060, execute the following command:

```
kill 2060
```

If, for some reason, this command does not kill it, use the stronger version:

```
kill -kill 2060
```

The preceding command is the same as the `kill -9 2060` command.

nice

Each process has a relative priority that governs the resources allocated to it by the operating system. The `nice` command is used to lower priorities, typically for background or batch processes. You can also raise priorities if you have `root` privileges.

A negative number signifies a higher priority than a positive number. The value is usually in the range -20 to 20.

If you want to find, at a lower priority in background, all the C source files in the current directory or its subdirectories, execute the following command:

```
nice find . -name *.c -print &
```

This command sets the process to a default `nice` priority, which may be 10. To run the process at an even lower priority, execute the following command:

```
nice 16 find . -name *.c -print &
```

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

ps

The `ps` command is used to find out which processes are currently running. Depending on the options you specify, you can find all processes or only those initiated by your user ID. When the `ps` command is executed without any flags or arguments, it lists all processes (if any) initiated from the current terminal.

Following is a list of some of the flags that determine what processes are listed by the `ps` command:

Flag	Meaning
-A	List details of all processes running in the system.
-e	List details of all processes, except kernel processes.
-k	List all the UNIX kernel processes.
-p list	List details of all processes specified in the list.
-t list	List details of all processes initiated from the terminals specified in the list.
-U list (-u list)	List details of all processes initiated by the users specified in the list.
a	List details of all processes that have terminals associated with them.
g	List details of all processes.
x	List details of all processes that do not have any terminal associated with them.

Note:
Some of the flags used by the `ps` command are not preceded by a hyphen (-).

Following is a list of some of the flags for the `ps` command that determine which details are displayed for each process listed:

Flag	Meaning
-l	Generate the listing in a long form.
-f	Generate a full listing.
-F <i>format</i>	Generate a formatted output.

The following details are displayed if formatting flags are not used:

- Process ID of the process.
- Terminal ID associated with the process. Hyphen (-) if no terminals are associated.
- CPU time consumed by the process.
- Command being executed as part of the process.

By using the formatting command, some of the details that can be obtained are as follows:

- User ID of the user who initiated the process.
- Process ID of the process.
- Parent Process ID of the process.
- CPU utilization of the process.
- Start time of the process. If the process has been started on the same day, it shows the time; otherwise, it shows only the date.
- Terminal ID associated with the process. It display a hyphen (-) if there are no terminals associated with the process.
- CPU time consumed by the process.
- Commands being processed as part of the process.

To display the processes initiated by the current user at the current terminal, execute the following command:

```
ps
```

The result is displayed as follows:

```
PID      TTY      TIME    CMD
- 66874    2        0:00  -ksh
  71438    2        0:00  ps
```

The two processes displayed here are the login shell (in this case, the Korn shell) and the `ps` command itself.

If you want more details, execute the following command:

```
ps -f
```

This generates the following display:

```
USER      PID     PPID     C    STIME      TTY      TIME    CMD
testuser  66874      1        1  22:52:26    2    0:00  -ksh
testuser  480076   66874     6  00:21:33    2    0:00  ps -f
```

If you want to know all the processes executing at terminal `tty2`, execute the following command:

```
ps -f -t tty2
```

Here is the result:

USER	PID	PPID	C	STIME	TTY	TIME	CMD
testuser	66874		1	1 22:52:26		2 0:00	-ksh
testuser	703277	66874	6	00:24:17		2 0:00	ps -f -t tty2

jobs

In UNIX, there is a subtle difference between a *process* and a *job*. A *job* is typically one command line of commands, which can be a single command, a shell script, or a chain of piped commands. In a chain of piped commands, each command has a unique process ID, but all have the same job ID.

The C shell and some versions of the Korn and Bourne shells offer the `jobs` command. You can use the `jobs` command to find out the details about active jobs. Once you have the job ID, you can start using it to do primitive job controls.

You use the `%` (percent sign) in front of the job number to indicate that the number is a job number rather than a process ID.

For instance, if you want to bring job number 5 from the background to the foreground, execute the following command:

```
fg %5
```

If you have one job called `sample_job` running in the background, execute the following command to get the details of the job:

```
jobs
```

The resulting display looks like this:

```
[1] +   Running                  nohup sample_job > sample_log &
```

If you use the `-l` option, the process number is also displayed as follows:

```
[1] + 270384      Running          nohup sample_job > sample_log &
```

wait

You can use the `wait` command to wait for completion of jobs. This command takes one or more process IDs as arguments. The `wait` command is useful during shell programming when you want one process to finish before the next process is invoked. If you do not specify a process ID, UNIX will wait for termination of all processes spawned by the current environment.

To find out whether the process ID 15060 has completed, execute the following command:

```
wait 15060
```

The return code from the `wait` command is zero if you invoked the `wait` command without any arguments. If you invoked the `wait` command with multiple process IDs, the return code depends on the return code from the last process ID specified.

nohup

Executing processes under UNIX can be running in the foreground or the background. In a foreground process, you are waiting at the terminal for the process to finish. Under such circumstances, you cannot use the terminal until the process is finished. However, you can put the foreground process into the background as follows:

Ctrl-z
bg

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

The processes in UNIX will be terminated when you log out of the system or exit the current shell whether they are running in foreground or background. The only way to ensure that the process currently running is not terminated when you exit is to use the `nohup` command.

The `nohup` command has default redirection for the standard output. It redirects the messages to a file called `nohup.out` under the directory from which the command was executed. That is, if you want to execute a script called `sample_script` in the background from the current directory, use the following command:

```
nohup sample_script &
```

The `&` (ampersand) tells UNIX to execute the command in the background. If you omit the `&`, the command is executed in the foreground. In this case, all the messages will be redirected to `nohup.out` under the current directory. If the `nohup.out` file already exists, the output is appended to it. If the permissions of the `nohup.out` file are set up so that you cannot write to that file, UNIX creates or appends to the `nohup.out` file in your home directory.

When you initiate the `nohup` command in the background (by using `&` at the end of the command line), UNIX displays the process ID associated with it. You can later use this information with commands such as `ps` to find out the status of the execution.

For example, to find the string `sample_string` in all the files in the current

directory, execute the following command:

```
nohup grep sample_string * &
```

UNIX responds with the following message:

```
[2]      160788
Sending output to nohup.out
```

The first line of the display is the process ID; the second line is the informational message about the output being directed to the default `nohup.out` file. You can later go into the file `nohup.out` to find out the result of the `grep` command. To redirect the output to a file called `mygrep.out`, execute the following command instead:

```
nohup grep sample_string * > mygrep.out &
```

sleep

If you want to wait for a certain period of time between execution of commands, use the `sleep` command. Use this command in cases where you want to check for, say, the presence of a file every 15 minutes. The argument is specified in seconds.

Communication

UNIX has several commands used to connect to another host, to transfer files between host computers, and so on without logging into a user session on the remote system.

cu

The `cu` command allows you to connect to another host computer, either directly or indirectly. That is, if you are currently on `host1` and you use `cu` to connect to `host2`, you can connect to `host3` from `host2`, so that you are connected directly to `host2` and indirectly to `host3`.

- Following is a list of some of the flags that can be used with `cu` command:

Flag	Meaning
-d	Print diagnostic messages.
-lLine	Override the default device to be used for communication.
-n	Prompt for the telephone number rather than accepting it as part of the command line.
-sSpeed	Specify the speed (in bauds) of the line to be used for communication between hosts.
-Tseconds	Specify the length of time UNIX will try to connect to the remote host.

The following arguments can be specified with the `cu` command:

- `System Name` is the name of the system to which you want to connect. This name must be defined in the `/etc/uucp/Systems` file. Also defined in this file are parameters, such as telephone number, line speed, and more, which are used to connect to the remote host. If you are using `System Name`, you should not use the `-l` and `-s` option.
- `Telephone Number` is the telephone number to be used to connect to the remote host. The number can be local or long distance.

After making the connection, `cu` runs as two processes: The transmit process reads data from the standard input and—except for lines beginning with `~` (tilde)—passes the data to the remote system. The receive process accepts data from the remote system and—except for lines beginning with `~` (tilde)—passes it to the standard output.

Once you are able to successfully log in to the remote host, you can use several subcommands, listed here. Remember to prefix the `~` with a `\` so that UNIX does not apply special meaning to `~`.

Subcommand	Meaning
<code>~.</code>	Disconnect from the remote host.
<code>~!</code>	Disconnect from the remote host. To activate an interactive shell on the local host, you can toggle between interactive shells of local and remote hosts using <code>~!</code> and <code>Ctrl+D</code> .
<code>~!Command</code>	Execute <i>Command</i> at the local host.
<code>~%cd directory</code>	Change the directory on the local host to the specified directory.
<code>~%put From [To]</code>	Copy an ASCII text file on the local system to a file on the remote system. If the <i>To</i> variable is not specified, the local file is copied to the remote system under the same filename.
<code>~%take From [To]</code>	Copy an ASCII text file from the remote system to a file on the local system. If the <i>To</i> variable is not specified, the remote file is copied to the local system under the same filename.
<code>~\$Command</code>	Run, on the local system, the command denoted by the <i>Command</i> variable and send the command's output to the remote system for execution.

For example, if `remote2` is defined in the `/etc/uucp/Systems` file, you can execute the following command:

```
cu remote2
```

If you want to connect to a specific device, `tty1`, on a remote host at a specified line speed, 2400, execute the following command:

```
cu -s 2400 -l tty1
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

After you are in the remote host, you can execute any of the `cu` subcommands. To the directory `/u/testuser` on the local host, execute this command:

```
\~%cd /u/testuser
```

If you have a file called `local_file` from directory `/u/testuser` on the local host, you can copy it to the current directory in the remote host with the same name by executing the following command:

```
\~%put /u/testuser/local_file
```

To execute the `ls` command on the local system, execute the following command:

```
\~!ls
```

ftp

You can use the `ftp` command to transfer files between two host computers. You cannot copy files from directories recursively using a single `ftp` command. Instead, you must transfer each file or directory individually. You should also be aware that `ftp` can be used between different types of systems, which means that system-dependent file attributes may not be preserved when files are transferred.

While using the `ftp` command, you need to use a login and password to log in to the

remote host. However, it is possible to log in to the remote host without a password if the home directory has a `.netrc` file and that file contains the macro necessary for logging in to the remote host.

If a login ID and password are needed, the `ftp` command prompts for that information. In such a case, you cannot proceed until you are able to successfully log in using the correct user ID and password.

After you log in successfully, the `ftp` command displays an `ftp>` prompt. From the `ftp>` prompt, you can use a number of subcommands to perform file transfers.

You can specify the host name as part of the `ftp` command or open a connection to a host in the `ftp>` prompt. If you specify the host name as part of the command line, you are prompted for the login ID and password before you see the `ftp>` prompt. For example, if you execute the following command, you will be prompted for a user ID and password:

```
ftp box2
```

Some of the flags that can be used with the `ftp` command are as follows:

Flag	Meaning
-i	If you do not want to be prompted with filenames in case more than one file is being transferred using a single command.
-d	For debug information.
-n	To prevent automatic login in case a <code>.netrc</code> file is present.
-v	To display the messages from the remote host on the terminal.

Once you are at the `ftp>` prompt, you can use several subcommands:

Subcommand	Meaning
!	Invoke the interactive shell on the local host. Optionally, you can invoke a command with arguments by specifying the command and arguments after the <code>!</code> sign.
?	Display a list of subcommands available. Optionally, a subcommand can be specified after the <code>?</code> to get information specific to that subcommand.

type	Set the file transfer mode. Valid modes are ASCII for text files, binary for files that may contain special files so that the data is transferred without any translation, and EBCDIC to transfer files in EBCDIC code.
cd	Change to the home directory on the remote host. If a directory is specified, the current directory is changed to the specified directory on the remote host.
pwd	Print the current working directory on the remote host.
ls	Print a list of the files on the remote host. If no directory is specified, files and directories in the specified directory are printed. You can direct the list of files to a file on the local host by specifying a local filename.
mkdir <i>directory</i>	Make a new directory under the current directory on the remote host.
dir	Generate a list of files. Similar to the <code>ls</code> command, but produces a detailed list.
rmdir <i>directory</i>	Remove the specified directory on the remote host, provided that the directory is empty.
rename <i>oldname</i> <i>newname</i>	Rename a file from <i>oldname</i> to <i>newname</i> on the remote host.
delete <i>filename</i>	Delete the specified file on the remote host.
get <i>filename</i>	Transfer a file from the remote host to the local host. The name of the file is not altered. Optionally, you can specify a local <i>filename</i> to which the file from the remote host will be copied.
mget	Transfer multiple files from the remote host to the local host. You can specify a filename using a wildcard, which is expanded by UNIX. If the prompt option is set, you will be prompted for each filename for confirmation. If a prompt option is not set, all files are copied without any confirmation.

<code>put filename</code>	Transfer a file from the local host to the remote host. The name of the file is not altered. Optionally, you can specify a <code>local filename</code> to which the file from the remote host will be copied.
<code>mput</code>	Transfer multiple files from the local host to the remote host. You can specify a filename using a wildcard, which is expanded by UNIX. If the <code>prompt</code> option is set, you will be prompted for each filename for confirmation. If the <code>prompt</code> option is not set, all files are copied without confirmation.
<code>mdelete</code>	Delete multiple files by specifying wildcard filenames.
<code>append filename</code>	Append a local file to the end of a file on the remote host. Optionally, you can specify a filename on the remote host to append at the end of the specified file. If the remote filename is not specified, the local filename is used by default.
<code>open</code>	Open a connection to a remote host by specifying a remote host name.
<code>close</code>	Close the existing connection to the remote host.
<code>bye or quit</code>	Quit the <code>ftp</code> session.
<code>lcd</code>	Change to the home directory on the local host. If a directory is specified, the current directory is changed to the specified directory on the local host.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

Click Here!

ITKnowledge

- home
- account info
- subscribe
- login
- search
- My ITKnowledge
- FAQ/help
- site map
- contact us

- Brief
- Full
- Advanced Search
- Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous Table of Contents Next

Examples

To open an ftp session on a remote host named otherhost, execute one of the following commands:

```
ftp otherhost

ftp
ftp> open otherhost
```

In both cases, you are prompted to enter the user ID and password, if you are not set up in the .netrc file with the user ID and password. The prompts and messages appear as follows:

```
Connected to otherhost.
220 otherhost FTP server (Version 4.14 Fri Oct 10 13:39:22 CDT 1994)
ready.
Name (otherhost:testuser): testuser
331 Password required for testuser.
Password:
230 User testuser logged in.
```

After you are at the ftp> prompt, you can execute the ftp subcommands. If you want to find out which directory you are in on the remote host, execute the following subcommand:

```
ftp> pwd
257 "/home/testuser" is current directory.
```

To copy a file called testfile from otherhost to the local host, execute the get subcommand. Following is the subcommand and its response:

```
ftp> get testfile
200 PORT command successful.
150 Opening data connection for testfile (73 bytes).
226 Transfer complete.
80 bytes received in 0.02583 seconds (3.025 Kbytes/s)
```

Similarly, to copy a file called `testfile` from the local host to the remote host, `otherhost`, use the `put` subcommand. Following is the subcommand and its response:

```
ftp> put testfile
200 PORT command successful.
150 Opening data connection for testfile.
226 Transfer complete.
142 bytes sent in 0.02954 seconds (4.695 Kbytes/s)
```

Following is a series of commands and responses that copy a file called `testfile` in binary mode from the `/u/testuser` directory on the local host to a directory called `/u/testuser/testdir` on the remote host `otherhost`.

```
ftp otherhost
Connected to otherhost.
220 otherhost FTP server (Version 4.14 Fri Aug 5 13:39:22 CDT 1994) ready.
Name (otherhost:testuser): testuser
331 Password required for testuser.
Password:
230 User testuser logged in.
ftp> lcd /u/testuser
Local directory now /u/testuser
ftp> cd /u/testuser/testdir
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> put testfile
200 PORT command successful.
150 Opening data connection for testfile.
226 Transfer complete.
46197 bytes sent in 0.03237 seconds (1394 Kbytes/s)
ftp> quit
221 Goodbye.
```

Here are some more examples of transferring multiple files, listing files, and deleting multiple files:

```
ftp otherhost
Connected to otherhost.
220 otherhost FTP server (Version 4.14 Fri Aug 5 13:39:22 CDT 1994) ready.
Name (otherhost:testuser): testuser
331 Password required for testuser.
Password:
230 User testuser logged in.
ftp> mput file*
mput file1? y
200 PORT command successful.
150 Opening data connection for file1.
226 Transfer complete.
46197 bytes sent in 0.03323 seconds (1358 Kbytes/s)
mput file2? y
200 PORT command successful.
150 Opening data connection for file2.
226 Transfer complete.
44045 bytes sent in 0.01257 seconds (3422 Kbytes/s)
mput file3? y
200 PORT command successful.
150 Opening data connection for file3.
226 Transfer complete.
41817 bytes sent in 0.01172 seconds (3485 Kbytes/s)
ls -l
200 PORT command successful.
```

```
150 Opening data connection for /bin/ls.
total 176
-rw-r----- 1 testuser  author      1115 Dec 15 11:34 file1
-rw-r----- 1 testuser  author     43018 Dec 15 11:34 file2
-rw-r----- 1 testuser  author     40840 Dec 15 11:34 file3
226 Transfer complete.
mdel file*
mdel file1? y
250 DELE command successful.
mdel file2? y
250 DELE command successful.
mdel file3? y
250 DELE command successful.
```

mailx

In UNIX, you can send mail to other users in the system and receive mail from them by using the `mailx` commands. The `mailx` commands provide subcommands to facilitate saving, deleting, and responding to messages. The `mailx` command also provides facilities to compose and edit messages before finally sending it to one or more users.

The mail system on UNIX uses mailboxes to receive mail for a user. Each user has a system mailbox in which all mail for that user is received pending action by the user. The user can read, save, and delete the mail after the mail is received. When the user has read mail, the mail can be moved to a secondary, or personal, mailbox. The default secondary mailbox is called the `mbox`. The `mbox` is usually present in the home directory of the user. However, the user can specify the name of a file as a secondary mailbox. All messages saved in the mailbox `mbox` are save indefinitely until moved to other secondary mailboxes, which are sometimes known as *folders*. You can use folders to organize your mail. For example, you can organize folders by subject matter and save all mail pertaining to a subject in a particular mailbox.

You can send messages to one or more users using the `mailx` command. This command allows you to send mail to users on the same host or on other hosts in the network to which the local host is connected. You do not get a positive acknowledgment if the mail delivery is successful, but if the mail cannot be delivered, you are notified.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous Table of Contents Next

Following is a list of some of the flags that can be used with the mail command:

Flag	Meaning
-d	Display debug information.
-f	Display a list of messages in the default mailbox mbox. Optionally, you can specify the name of a folder in which you have saved your mail previously.
-s <i>subject</i>	Associate a subject for the mail to be created.
-v	Display detailed information by the mailx command.

Each mail has information associated with it. The following is a list of this information:

- status indicates the status of a mail. Following is a list of the various statuses of a mail item:

M	Indicates that the message will be stored in your personal mailbox.
>	Indicates the current message.
N	Indicates that the message is a new message.
P	Indicates that the message is to be preserved in the system mailbox.
R	Indicates that you have read the message.

U	Indicates an unread message. An unread message is one that was a new message at the last invocation of <code>mailx</code> but was not read.
*	Indicates that the message has been saved or written to a file or folder.

A message without a status indicates that the message has been read but has not been deleted or saved.

- *number* indicates a numerical ID of the message by which it can be referred to.
- *sender* indicates the user who sent the message.
- *date* indicates when the mail was received in the mailbox.
- *size* indicates the size of the message in number of lines and number of bytes.
- *subject* indicates the subject matter of the mail if the sender has associated a subject with the mail. This may or may not be present, depending on whether the mail has an associated subject.

Following is a list of subcommands you can use from the `mail>` prompt:

Subcommand	Meaning
q	Apply mailbox commands entered this session.
x	Quit.
! <i>command</i>	Start a shell, run a command, and return to the mailbox.
cd	Place you in the home directory. Optionally, you can specify a directory name to place you in the specified directory.
t	Display the current message. Optionally, you can specify a message list to display messages in that list.
n	Display the next message.
f	Display headings of the current message. Optionally, you can specify a message list and display all headings in that message list.
e	Edit the current message. Optionally, you can specify a message number to modify that message.
d	Delete messages or the current message. Optionally, you can specify a message list to delete the message in the message list.
u	Restore deleted messages.
s <i>file</i>	Append the current message, including its heading, to a file. Optionally, you can specify a message list between s and <i>file</i> to append the specified messages to the file.
w <i>file</i>	Append the current message, excluding its heading, to a file. Optionally, you can specify a message list between w and <i>file</i> to append the specified messages to the file.
pre	Keep messages in the system mailbox. Optionally, you can specify a list of messages to keep them in system mailbox.

m <i>addresslist</i>	Create or send a new message to addresses in the <i>addresslist</i> .
r	Send a reply to senders and recipients of messages. Optionally, you can specify a list of messages to send a reply to senders and recipients of all messages in the list.
R	Send a reply only to senders of messages for the current message. Optionally, you can specify a list of messages to send reply to senders of the messages.
a	Display a list of aliases and their addresses.

Examples

You can invoke the `mailx` command by itself to display the `mail>` prompt, from which you can use the subcommands. You can get to the `mail>` prompt only if you have mail. Otherwise, you will get a message similar to `you have no mail`. If you have mail, prompts similar to the following are displayed:

```
mailx
Mail [5.2 UCB] Type ? for help.
"/usr/spool/mail/testuser": 1 message 1 new
>N 1 testuser      Sat Nov 16 22:49  285/9644
&
```

If you now quit the `mailx` command using the `quit` subcommand (which can be abbreviated as `q`), the mail is saved in your personal mailbox (the `mbox` file in your home directory).

To see the mail, use the `mailx -f` command, which results in the following dialog:

```
mailx -f
Mail [5.2 UCB] Type ? for help.
"/u/testuser/mbox": 1 message
> 1 testuser      Sat Nov 23 00:11  162/5175
&
```

To save mail in a folder from in the `mailx` command, execute the following subcommand:

```
& save 1 /u/testuser/folder1/file1
```

This command creates `file1` from the first message in the directory `/u/testuser/folder1`.

If you invoke the `mailx` command to read the file `/u/testuser/folder1`, it results in the following dialog:

```
mailx -f /u/testuser/folder1
Mail [5.2 UCB] Type ? for help.
"/u/testuser/folder1": 1 message
> 1 testuser      Sat Nov 23 00:11  162/5175
&
```

Once you are in `mailx`, you can execute the subcommand `m` to create and send mail to other users as follows:

```
& m friend1
```

Subject: Testing mailx command
This is test of the mailx command
Here we are trying to send a mail to user friend1 with cc to friend2
Cc: friend2
&

The body of the mail is terminated by pressing Ctrl+D. You can send mail to multiple users using the m subcommand.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

talk

You can converse with another user in real time using the `talk` command if the other user is logged on. Using the `talk` command, you can converse with users on the local host or on a remote host.

The `talk` command takes one mandatory argument: the user name or user and host name. You can optionally provide a second argument that specifies the TTY onto which the user is logged.

The user on the remote host can be specified in one of the following formats:

- `username@host`
- `host!username`
- `host.username`
- `host;username`

When you execute the `talk` command, it opens two windows—one for sending messages and one for receiving messages. The command waits for the other user to respond.

Examples

Suppose that you execute the following command to converse with the user `friend`:

```
talk friend
```

You will get the following screen:

```
[Waiting for your party to respond]
-----
```

One half of the screen is used to send messages, and other half is used to receive messages. If the specified user is not logged on, you get a message similar to `Your party is not logged on.`

The user friend gets the following message:

```
Message from Talk_Daemon@host1 at 0:46 ...
talk: connection requested by testuser@host1.
talk: respond with:  talk testuser@host1
[Waiting for your party to respond]
```

To start the conversation, the user friend has to respond with the following command:

```
talk testuser@host1
```

To quit the talk session, press Ctrl+C.

vacation

If you want to notify a mail sender that you are on vacation, you can use the `vacation` command. The message sent can be a customized message if you create the message in a file called `.vacation.msg` in your home directory. If this file does not exist, a system default message is sent. By default, the system message is sent only once a week to a user who sends mail to you.

The `vacation` command can be used to forward messages you receive during vacation to other users by using the `.forward` file in your home directory. Include the user names of all the users to whom you want the messages to be forwarded. The entry in the `.forward` file is of the following format:

```
testuser, "|/usr/bin/vacation testuser"
```

The `vacation` command also lets you store the names of users who sent messages to you while you were on vacation. These user names are stored in the `.vacation.dir` and `.vacation.pag` files in your home directory.

- The `vacation` command has one option flag, `-i`, which is used to initialize the `.vacation.dir` and `.vacation.pag` files before the start of your vacation.

The `.forward` file in your home directory is used by the system to identify that you are on vacation. When you come back, you should delete or rename the `.forward` file.

Before you go on vacation, use the following command to initialize the `.vacation.dir` and `.vacation.pag` files in your home directory:

```
vacation -I
```

This should be followed by the creation of the `.forward` and `.vacation.msg` files.

write

You can use the `write` command to hold a conversation with another user in the local host or remote host, just as you can with the `talk` command. To hold a conversation with another user, the following must be true:

- The user must be logged on.
- The user must not have denied permission by using the `mesg` command.

A message consists of all the characters you type until you press the Enter key. Both you and the other user can send messages this way. To end the conversation, press Ctrl+D.

Following is a list of some of the flags that can be used with the `write` command:

Flag	Meaning
-h <i>Handle, Reply</i>	Reply to a message sent by a utility or shell script using <code>write</code> with the <code>reply</code> option. The <i>handle</i> is a number generated by the system. The <i>reply</i> can be <code>ok</code> , <code>cancel</code> , or <code>query</code> .
-nHost	Specify a remote host if you want to hold conversation with a user on a remote host. It is also possible to specify user at a remote host as <i>username@host</i> .
-q	Find out about messages awaiting replies from users on a host and display them with their handles.

Examples

If you want to hold a conversation with a user called `friend`, execute the following command:

```
write friend
```

If user `friend` is not logged on, you get a message similar to the following:

```
friend is not logged on.
```

If the user `friend` has used the `mesg` command to turn the permission off for conversations, you will get a message similar to the following:

```
write: permission denied
```

If the `write` command succeeds, the user gets a message similar to the following:

```
Message from testuser on mainhost(pts/3) [Fri Nov 22 19:48:30 1996] ...
```

You can use the UNIX input redirection operator to send long messages from a file called, for example, `long_message`:

```
write friend < long_message
```

To start the conversation, the other user must also use the `write` command as follows:

```
write testuser
```

If you want to hold a conversation with user `friend` on the remote host `otherhost`, execute either of the following two commands:

```
write friend@otherhost
```

```
write -n otherhost friend
```

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)

All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

File Comparison

The following sections describe some of the commands you can use for comparing the contents of files. These commands compare the contents and, depending on various options, generate outputs of the differences between the files. There are also commands that can be used to compare the contents of directories.

cmp

The `cmp` command compares the contents of two files and generates output into standard output. Although it is possible to have one of the files be standard input, both files cannot be. You can use a `-` (hyphen) to indicate that the file is standard input, which is terminated using `Ctrl+D`. Use the `cmp` command for nontext files to find out whether they are identical. For text files, use the `diff` command, discussed later.

Following are the outputs generated by the `cmp` command:

- No output if the files are exactly identical.
- Displays the byte number and line number of the first position where the files are different.

Here are the flags that can be used with `cmp` command:

Flag

Meaning

<code>-l</code>	Display, for each difference, the byte number in decimal and the differing bytes in octal.
<code>-s</code>	Return only an exit value without generating any output. The values of the return code are 0 for identical files, 1 if the files are different, or 2 if the <code>cmp</code> file is not successful in comparing the files.

Examples

If you want to compare the new version of an executable file with the old version of the executable file, execute the following command:

```
cmp new_prog1 old_prog1
```

If the files are identical, no output will be generated. If the files are different, output similar to the following will be generated:

```
new_prog1 old_prog1 differ: byte 6, line 1
```

If you are using the `cmp` command in a shell script, you can use the `-s` flag to determine whether two files are identical. Following is part of a shell script that uses the `-s` command:

```
ret_code='cmp -s new_prog1 old_prog1'
if [[ $ret_code -eq 0 ]] then
    echo "Files are identical ..."
else
    echo "Files are different ..."
fi
```

If the files are identical except that one file has extra bytes at the end, use of the `-l` flag will generate the following output:

```
cmp -l new_prog1 old_prog1
    18  12  63
cmp: EOF on new_prog1
```

comm

If you have files that are sorted and you want to compare them, use the `comm` command. The `comm` command can be used to either exclude or include the common lines between the two files. You can use a `-` (hyphen) to indicate that one of the files is to be accepted from standard input.

The default output is generated on the standard output in three columns, which are as follows:

- Lines that exist only in the first file.
- Lines that exist only in the second file.
- Lines that exist in both files.

Following is a list of flags that can be used with the `comm` command:

Flag	Meaning
-1	Suppress the display of the first column.
-2	Suppress the display of the second column.
-3	Suppress the display of third column.

Examples

Assume that you have two files named `file1` and `file2`. The content of the files has been displayed using the `more` command as follows:

more file1

```
line 1
line 2
line 3
line 5
line 6
line 7
line 8
line 9
```

more file2

```
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 9
```

If you compare the two files `file1` and `file2`, you get the following output:

comm file1 file2

```

line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
```

line 9

The output shows that `file1` has one line that is not in `file2` (in column 1); `file2` has one line that is not in `file1` (in column 2); there are seven lines that exist in both `file1` and `file2` (in column 3).

If you are interested only in the differences, you can drop column 3 as follows:

```
comm -3 file1 file2  
      line 4  
line 8
```

If you are interested only in finding out which lines are identical in `file1` and `file2`, you can drop columns 1 and 2 as follows:

```
comm -12 file1 file2  
line 1  
line 2  
line 3  
line 5  
line 6  
line 7  
line 9
```

diff

Although you can compare text files with the `diff` command, you cannot compare nontext files. You can use `diff` to compare individual files or multiple files with identical names in different directories. To compare individual files, use a `-` (hyphen) to indicate that one of the files is to be accepted from standard input.

The group of output lines of the `diff` command is preceded by an information line containing the following information:

- Lines that are different in the first file. This may a single line number or two line numbers separated by a comma (which refers to the lines within that range).
- Action code, for which valid values are `a` (for lines added), `c` (for lines modified), and `d` (for lines deleted).
- Lines that are different in the second file. This may a single line number or two line numbers separated by a comma (which refers to the lines within that range).

The `diff` command generates `ed` commands. The `ed` commands can be applied to the first file; doing so makes the second file identical to the first file.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

The following three forms of output are generated by the `diff` command:

- `Number1aNumber2 , Number3`, which means that line `Number2` through line `Number3` in the second file must be added to the first file after line `Number1` to make the files identical. This form is followed by the actual lines from the second file preceded by a `>` (greater-than sign).
- `Number1dNumber2`, which means that line `Number1` through line `Number2` must be deleted from the first file to make the files identical. This form is followed by the actual lines from the first file preceded by a `<` (less-than sign).
- `Number1 , Number2cNumber3 , Number4`, which means that line `Number1` through line `Number2` in the first file must be modified by the lines `Number3` through `Number4` in the second file to make the files identical. This form is followed by the actual lines from the first file, each preceded by `<` (less-than sign), and then actual lines from the second file, each preceded by `>` (greater-than sign).

Following is a list of some of the flags that can be used with the `diff` command:

Flag	Meaning
------	---------

- b Ensure that more than one space or tab character is considered as one. However, leading space or tab characters are processed as-is.
- C LINE Produce the output in a format different from the default format. The output first identifies the files being compared along with their creation date and time. Each modified group of lines is separated by a line with an * (asterisk) followed by lines from the first file and lines from the second file. The lines removed from the first file are designated with a - (hyphen). The lines added to the second file are designated with a + (plus sign). Lines that exist in both files but differ are designated with an ! (exclamation point). Changes that lie within the specified context lines of each other are grouped together as output.
- c LINE Produce the output in a format different from the default format. The LINE parameter is optional. The output first identifies the files being compared along with their creation date and time. Each modified group of lines is separated by a line with an * (asterisk) followed by lines from the first file and lines from the second file. The lines removed from the first file are designated with a - (hyphen). The lines added to the second file are designated with a + (plus sign). Lines that exist in both files but differ are designated with an ! (exclamation point). Changes that lie within the specified context lines of each other are grouped together as output.
- D String Create a merged version of the first and second files on the standard output. The C preprocessor controls are included so that a compilation of the results, without defining String, is equivalent to compiling the first file. Defining String compiles the second file.
- e Generate output that can be input to ed to produce the second file from the first file.
- f Create output that is the reverse of what is produced by the -e flag.
- i Compare the two files ignoring case of letters.

<code>-l</code>	Generate formatted output. This flag also generates a summary at the end.
<code>-n</code>	Generate output in the reverse order of the <code>-e</code> flag. Additionally, this flag generates the count of lines added and deleted.
<code>-r</code>	Execute the <code>diff</code> command on all identically named subdirectories of the specified directory.
<code>-s</code>	Generate output to obtain a list of identical files along with differences between identically named files. This flag also generates a list of files that are only in one of the directories.
<code>-S FILE</code>	Ignore files. This option can be used while comparing directories to ignore filenames that collate less than that of the specified <code>FILE</code> .
<code>-t</code>	Preserve the original indentation of the lines. The indentation of lines in the output can be modified by the use of the <code>></code> (greater-than sign) or the <code><</code> (less-than sign) in the output.
<code>-w</code>	Ignore all space and tab characters. That is, treat them identically for comparison purposes.

Examples

Assume that you have two files, `file1` and `file2`. The contents of the files are shown using the `more` command:

more file1

```
This is the first line
This is the second line
This is the fourth line
This is the fifth line
This is the sixth line
This is the seventh          line
This is the eighth line
This is the NINTH line
```

more file2

```
This is the first line
This is the second line
This is the third line
This is the fourth line
This is the sixth line
This is the seventh line
This is the eighth          line
```

This is the ninth line

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

The plain-vanilla `diff` command on `file1` and `file2` generates output as follows:

```
diff file1 file2
2a3
> This is the third line
4d4
< This is the fifth line
6,8c6,8
< This is the seventh line
< This is the eighth line
< This is the NINTH line
---
> This is the seventh line
> This is the eighth line
> This is the ninth line
```

This means add `This is the third line` to `file1`, delete `This is the fifth line` from `file1`, and modify lines 6 and 7 to `This is the seventh line` and `This is the eighth line` in `file1` to make `file1` identical to `file2`.

If you do not care about space and tab characters, use the `-b` flag in the following command:

```
diff -b file1 file2
2a3
> This is the third line
4d4
< This is the fifth line
8c8
< This is the NINTH line
---
> This is the ninth line
```

As you can see, lines 6 and 7 are not displayed because the only reason these lines are different is the existence of extra space and tab characters.

You can display the output in special format using the `-C` or `-c` flag in the following command:

```
diff -C 0 file1 file2
*** file1      Thu Nov 28 22:15:23 1996
--- file2      Thu Nov 28 18:05:59 1996
*****
*** 2 ****
--- 3 ----
+ This is the third line
*****
*** 4 ****
- This is the fifth line
--- 4 ----
*****
*** 6,8 ****
! This is the seventh           line
! This is the eighth line
! This is the NINTH line
--- 6,8 ----
! This is the seventh line
! This is the eighth           line
! This is the ninth line
```

The output contains the filenames being compared and their creation dates and times; the output has a format different from the default `diff` format.

If you want to generate output in a format that can be input to `ed`, use the `-e` flag as in the following command:

```
diff -e file1 file2
6,8c
This is the seventh line
This is the eighth           line
This is the ninth line
.
4d
2a
This is the third line
.
```

This output can then be input to `ed` to be applied to `file1` to make it identical to `file2`. This can be used to maintain a base version and incremental changes so that any version can be created by applying the changes to the base version. Following is an example that shows how we can make `file1` identical to `file2` using the `-e` command.

Let's redirect the output of the `diff` command to a file called `diffout` as in the following command:

```
diff -e file1 file2 > diffout
```

If you execute the `more` command on `diffout`, you will get the following output:

```
more diffout
```

```
6,8c
```

```
This is the seventh line
```

```
This is the eighth line
```

```
This is the ninth line
```

```
.
```

```
4d
```

```
2a
```

```
This is the third line
```

```
.
```

Let's now add an extra command line at the end of the file `diffout`. The command is `w`, which ensures that when the file is input to `ed`, the output is written back to the file on which it is applied.

- If we now execute `more` on the file `diffout`, you can see the extra line at the end:

```
more diffout
```

```
6,8c
```

```
This is the seventh line
```

```
This is the eighth line
```

```
This is the ninth line
```

```
.
```

```
4d
```

```
2a
```

```
This is the third line
```

```
.
```

```
w
```

To update `file1` and see the result, we use the following commands:

```
ed - file1 < diffout
```

```
more file1
```

```
This is the first line
```

```
This is the second line
```

```
This is the third line
```

```
This is the fourth line
```

```
This is the sixth line
```

```
This is the seventh line
```

```
This is the eighth line
```

```
This is the ninth line
```

If you do not want to update the original file, use the command `l , $p` instead of `w`. This command generates the output to the standard output instead. You can also redirect the output to a file as follows:

```
ed - file1 < diffout > file1.new
```

To generate output in reverse order from that specified by `-e`, execute the following command with the `-f` flag:

```
a2
```

```
This is the third line
```

```
.
```

```
d4
```

```
c6 8
```

```
This is the seventh line
```

```
This is the eighth line
```

```
This is the ninth line
```

If you do not care about whether the files have lowercase or uppercase letters, use the `-i` flag to execute a case-insensitive `diff` command as in the following command:

```
diff -i file1 file2
2a3
> This is the third line
4d4
<This is the fifth line
6,7c6,7
< This is the seventh line
< This is the eighth line
---
> This is the seventh line
> This is the eighth
```

Notice that the lines `This is the NINTH line` in `file1` and `This is the ninth line` in `file2` are evaluated to be equal when we use the `-i` flag.

If you want to know the number of lines affected by each insertion or deletion, use the `-n` flag as in the following command:

```
diff -n file1 file2
a2 1
This is the third line
d4 1
d6 3
a8 3
This is the seventh line
This is the eighth line
This is the ninth line
```

The information in this output is with respect to `file1`. It tells you that one line is to be inserted after line 2 and then shows the lines to be inserted, one line is to be deleted at line 4, three lines are to be deleted at line 6, and three lines are to be inserted after line 8 and then lists the lines to be inserted.

To ignore all tab and space characters, use the `-w` flag. The difference between the `-b` flag and the `-w` flag is that `-b` ignores all space and tab characters except leading ones, while `-w` ignores all space and tab characters. Following is an example of the use of the `-w` flag:

```
diff -w file1 file2
2a3
> This is the third line
4d4
< This is the fifth line
8c8
< This is the NINTH line
---
> This is the ninth line
```

So far, we have seen the actions of the `diff` command for comparing two files. Now let's look at some examples of comparing two directories. Assume that the two subdirectories `testdir1` and `testdir2` exist under the current directory. Now let's see what files exist under these directories:

ls -R test*

```
testdir1:
file1      file2      file3      file4      file5      file6      testdir3

testdir1/testdir3:
filea  fileb  filec  filed  filee

testdir2:
file2      file4      file5      file7      file8      testdir3

testdir2/testdir3:
fileb  filed  filee  filef  fileg
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous Table of Contents Next

The simplest form of the diff command (without any flags to compare the two directories) results in the following output:

```
diff testdir1 testdir2
Only in testdir1: file1
Only in testdir1: file3
Only in testdir1: file6
Only in testdir2: file7
Only in testdir2: file8
Common subdirectories: testdir1/testdir3 and testdir2/testdir3
```

In this example, the diff command does not go through the subdirectory testdir3 under the directory testdir1 and testdir2. If you want the diff command to traverse the subdirectory under the specified directories, use the -r flag as in the following command:

```
diff -r testdir1 testdir2
Only in testdir1: file1
Only in testdir1: file3
Only in testdir1: file6
Only in testdir2: file7
Only in testdir2: file8
Only in testdir1/testdir3: filea
Only in testdir1/testdir3: filec
Only in testdir2/testdir3: filef
Only in testdir2/testdir3: fileg
```

If you want a list of all files in the directories that are identical, use the -s command as in the following command:

```
diff -rs testdir1 testdir2
```

```

Only in testdir1: file1
Files testdir1/file2 and testdir2/file2 are identical
Only in testdir1: file3
Files testdir1/file4 and testdir2/file4 are identical
Files testdir1/file5 and testdir2/file5 are identical
Only in testdir1: file6
Only in testdir2: file7
Only in testdir2: file8
Only in testdir1/testdir3: filea
Files testdir1/testdir3/fileb and testdir2/testdir3/fileb are identical
Only in testdir1/testdir3: filec
Files testdir1/testdir3/filed and testdir2/testdir3/filed are identical
Files testdir1/testdir3/filee and testdir2/testdir3/filee are identical
Only in testdir2/testdir3: filef
Only in testdir2/testdir3: fileg

```

If you do not want to process all files whose names collate before the specified filename (in this case file2), use the `-S` flag as in the following command:

```

diff -r -S file2 testdir1 testdir2
Only in testdir1: file3
Only in testdir1: file6
Only in testdir2: file7
Only in testdir2: file8
Only in testdir1/testdir3: filea
Only in testdir1/testdir3: filec
Only in testdir2/testdir3: filef
Only in testdir2/testdir3: fileg

```

diff3

The `diff` command compares two files. If you want to compare three files at the same time, use the `diff3` command. The `diff3` command writes output to the standard output that contains the following notations to identify the differences:

- `====` means all three files differ.
- `====1` means the first file differs.
- `====2` means the second file differs.
- `====3` means the third file differs.

Following is a list of flags that can be used with the `diff3` command:

Flag	Meaning
<code>-3</code>	Produce an edit script that contains only lines containing the differences from the third file.
<code>-E, -X</code>	Produce an edit script in which the overlapping lines from both files are inserted by the edit script, bracketed by <code><<<<<<</code> and <code>>>>>>></code> lines.
<code>-e</code>	Create an edit script that can be input to the <code>ed</code> command to update the first file with differences that exist between the second and third files (that is, the changes that normally would be flagged with <code>====</code> and <code>====3</code>).
<code>-x</code>	Produce an edit script to incorporate only changes flagged with <code>====</code> .

The format of the generated output is as follows:

- `File Id;Number1 a` means that lines are to be added after line `Number1` in the file `File Id`. The `File Id` can be 1, 2, or 3, depending on the file it is referring to. This is followed by the lines to be added.
- `File Id;Number1[,Number2]c` means that lines in the range `Number1` through `Number2` are to be modified. This is followed by the lines to be modified.

Examples

Assume that we have three files: `file1`, `file2`, and `file3`. The contents of these three files are shown using the `more` command:

more file1

```
This is the first line in first file
This is the second line
This is the third line
This is the fourth line
This is the fifth line
This is the sixth line
This is the seventh line
This is the eighth line
This is the ninth line
```

more file2

```
This is the first line
This is the second line
This is the third line
This is the 3.5th line
This is the fourth line
This is the sixth line in second file
This is the seventh line
This is the eighth line
This is the ninth line
```

more file3

```
This is the first line
This is the second line
This is the third line
This is the fourth line
This is the sixth line in third file
This is the seventh line
This is the eighth line
This is the ninth line
This is the tenth line
This is the eleventh line
```

Now execute `diff3` on these three files without using any flag, as in this command:

diff3 file1 file2 file3

```
===1
1:1c
    This is the first line in first file
2:1c
3:1c
    This is the first line
===2
1:3a
2:4c
    This is the 3.5th line
```

```

3:3a
====
1:5,6c
    This is the fifth line
    This is the sixth line
2:6c
    This is the sixth line in second file
3:5c
    This is the sixth line in third file
====3
1:9a
2:9a
3:9,10c
    This is the tenth line
    This is the eleventh line

```

The first group of lines starting with ====1 show that line 1 in file1 is different from line 1 in file2 and file3. The lines starting with ====2 show that line 4 in file2 should be inserted after line 3 of file1 and file3 to make all three files identical. The lines starting with ==== show that lines 5 and 6 of file1, line 6 of file2, and line 5 of file3 are all different. The lines starting with ====3 show that lines 9 and 10 of file3 should be inserted after line 9 of file1 and file2 to make all three files identical.

If you are interested in finding only the differences in file3, use the -3 flag as in the following command:

```

diff3 -3 file1 file2 file3
9a
This is the tenth line
This is the eleventh line
.
w
q

```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
 All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

This output tells you that there are two lines (lines 9 and 10) that are present in `file3` but are not in `file1` or `file2`.

If you want to apply changes between `file2` and `file3` to `file1`, use the `-e` flag to create an edit script as in the following command:

```
diff3 -e file1 file2 file3
9a
This is the tenth line
This is the eleventh line
.
5,6c
This is the sixth line in third file
.
w
q
```

This output means that `file3` has two extra lines at line 9 and that line 6 of `file2` has been replaced by line 5 of `file3`. If, however, you are interested in changes, use the `-x` flag as in the following command:

```
diff3 -x file1 file3 file2
5,6c
This is the sixth line in second file
.
w
q
```

dircmp

If you want to compare the contents of two directories, use the `dircmp` command. This command compares the names of the files in each directory and generates a list of filenames that exist in only one of the directories followed by filenames that exist in both and whether they are identical or not.

Following is a list of flags that can be used with `dircmp` command:

Flag	Meaning
-d	Generate a list of files that exist in either of the directories followed by a list of files that exist in both directories; specify whether they are identical or different. This information is further followed by the output of the <code>diff</code> command on pairs of files that are different.
-s	Generate a list of files that exist in either of the directories followed by a list of files that are different.

Examples

Assume that there are two subdirectories, `testdir1` and `testdir2`, in the current directory. The list of files in these directories are as follows:

```
ls testdir1
file1 file2 file3 file4 file5 file6
```

```
ls testdir2
file2 file3 file5 file6 file7 file8
```

If you want to do a plain-vanilla `dircmp` between these two directories, execute the following command:

```
dircmp testdir1 testdir2
```

```
Fri Nov 29 22:51:34 1996 testdir1 only and testdir2 only Page 1
```

```
./file1      ./file7
./file4      ./file8
```

```
Fri Nov 29 22:51:34 1996 Comparison of testdir1 and testdir2 Page 1
```

```
directory    .
different    ./file2
same         ./file3
same         ./file5
same         ./file6
```

The first part of this report lists on the left the files present only in `testdir1`; on the right, it lists the files found only in `testdir2`. The second part of the report shows a comparison of the directories and also shows which files are identical and which are different. If you want further information about the differences between the files in these directories, use the `-d` flag as in the following command:

```
testdir -d testdir1 testdir2
```

```
Fri Nov 29 22:56:01 1996 testdir1 only and testdir2 only Page 1
```

```
./file1      ./file7
./file4      ./file8
```

```
Fri Nov 29 22:56:01 1996 Comparison of testdir1 and testdir2 Page 1
```

```
directory    .
different    ./file2
same         ./file3
same         ./file5
```

```
same                ./file6
```

```
Fri Nov 29 22:56:01 1996 diff of ./file2 in testdir1 and testdir2 Page 1
```

```
1c1
< This file is in testdir1
---
> This file is in testdir2
```

If you only want a list of the files that are unique to each directory and the files that are different, use the `-s` flag as in the following command:

```
dircmp -s testdir1 testdir2
```

```
Fri Nov 29 23:39:59 1996 testdir1 only and testdir2 only Page 1
```

```
./file1            ./file7
./file4            ./file8
```

```
Fri Nov 29 23:39:59 1996 Comparison of testdir1 and testdir2 Page 1
```

```
different          ./file2
```

If you want to suppress the display of identical files, but you want a list of the files that are different and the difference between these files, execute the `dircmp` command with both `-s` and `-d` flags.

sdiff

The command `sdiff` compares two files and displays output on the standard output in a side-by-side format. Following is the detail of the display format:

- If the two lines are identical, the command displays each line of the two files with a series of spaces between them.
- If the line exists in only the first file, a `<` (less-than sign) is displayed at the end of the line.
- If the line exists in only the second file, a `>` (greater-than sign) is displayed at the beginning of the line.
- If the lines from the two files are different, a `|` (vertical bar) is displayed between the lines.

The flags that can be used with `sdiff` command are as follows:

Flag	Meaning
<code>-s</code>	If you do not want to display the identical lines.
<code>-w number</code>	Set the width of the display to <i>number</i> .
<code>-l</code>	Display only the line from the first file if the lines from the two files are identical.
<code>-o file</code>	Create a merged file from the first and second file depending on a number of subcommands you can specify.

Examples

Assume that we have two files, `file1` and `file2`, whose contents are displayed using the `more` command:

```
more file1
This is the first line in first file
This is the second line
```


This is the third line
This is the fourth line
This is the fifth line
This is the sixth line
This is the seventh line
This is the eighth line
This is the ninth line

more file2

This is the first line
This is the second line
This is the third line
This is the 3.5th line
This is the fourth line
This is the sixth line in second file
This is the seventh line
This is the eighth line
This is the ninth line

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 [EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



home	account info	subscribe	login	search	My ITKnowledge	FAQ/help	site map	contact us
----------------------	------------------------------	---------------------------	-----------------------	------------------------	--------------------------------	--------------------------	--------------------------	----------------------------

- [Brief](#) [Full](#)
- [Advanced Search](#)
- [Search Tips](#)

[Previous](#) [Table of Contents](#) [Next](#)

If, however, you do not want to display the identical lines, use the `-s` flag as in the following command:

```
sdiff -s file1 file2
This is the first line in first file      | This is the f
first line                                > This is the 3
```

```
.5th line
This is the fifth line          | This is the s
ixth line in second file
This is the sixth line         <
```

You can use the `-l` flag to display only the line from the first file if the lines are identical so that the other lines stand out:

```
sdiff -l file1 file2
This is the first line in first file      | This is the f
irst line
This is the second line
This is the third line
                                     > This is the 3

.5th line
This is the fourth line
This is the fifth line          | This is the s
ixth line in second file
This is the sixth line          <
This is the seventh line
This is the eighth line
This is the ninth line
```

File-Manipulation Commands

The following sections discuss several commands you can use to manipulate various attributes of one or more files, as well as to copy and move files from one location to another. The various attributes that can be manipulated include modification time, permission, and more.

touch

The `touch` command can be used for a number of purposes depending on whether or not a file already exists. If a file does not exist, the `touch` command creates it (if you have write access to the directory). If a file is already present, the `touch` command modifies the last modification time of the file.

Examples

To create a file called `testfile` in the current directory, execute the following command:

```
touch testfile
```

To create `testfile` in the `/u/testuser/testdir`, execute the following command:

```
touch /u/testuser/testdir/testfile
```

chmod

You may have to modify the permission of a directory or files to either secure them or to make them accessible to others. You can use the `chmod` command to modify the permission for files and directories. The permission in UNIX is specified in an octal number (0 thorough 7). Permission for a file or directory can be specified for the following:

- Owner: The user who created the file.
- Group: The group to which the owner belongs.
- World or others: Users other than the owner and users in the group to which the owner belongs.

For each of these entities, one octal number is specified to designate the permission granted. The permission granted to the owner, group, and world is derived from three bits associated with read, write, and execute authority for the file. That is, the bit for read has a value of 1 if read permission is to be granted; the bit for write has a value of 1 if write permission is to be granted, and the bit for execute has a value of 1 if execute permission is to be granted.

You should be aware that the execute bit functions differently for directories than it does for files. The execute permission for a directory is used to designate whether you can access that directory.

The combination of these three bits is expressed as an octal number and is used to designate the permission. The weight associated with the read bit is 4, the weight associated with the write bit is 2, and the weight associated with the execute bit is 1. The value of the permission is derived as follows:

$$(4 * \text{value of read bit}) + (2 * \text{value of write bit}) +$$

③ $(1 * \text{value of execute bit})$

The value of the permission can vary from 0 (no read, write, or execute permission) to 7 (read, write, and execute permission).

For example, if you want to provide read and write permission but no execute permission, the value to be used is calculated as follows:

$$(4 * 1) + (2 * 1) + (1 * 0) = 6$$

Remember that execute permission for a directory means that the directory can be accessed. That is, operations can be performed on files residing in that directory. If you provide write permissions to a directory, the user can read, write, delete, and execute all files in that directory, regardless of the permissions associated with the individual files.

With the `chmod` command, you can specify the new permissions you want on the file or directory. The new permission can be specified in one the following ways:

- As a three-digit numeric octal code
- As symbolic mode

Examples

Suppose that you want the file `testfile` to have these permissions:

- Owner with read, write, and execute permission
- Group with read only permission
- Others with execute only permission

To assign these permissions, you must execute the following command:

```
chmod 741 testfile
```

If you are using the symbolic mode, the following must be specified:

- Whose permission (owner, group, or others) you want to change.
- What operation—+ (add), – (subtract), or = (equals)—you want to perform on the permission.
- The permission (r, w, x, and so on).

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

If you want to set up the permission for `testfile` owned by you in the current directory so that only you and users in your group can read and write the file, execute the following command using absolute permissions:

```
chmod 660 testfile
```

If you want to add write permission for the group for `testfile` in the current directory (assuming that currently `testfile` has 741 permission), execute the following command:

```
chmod g+w testfile
```

Similarly, if you want to revoke the read permission for others for `testfile` in the current directory, execute the following command:

```
chmod o-r testfile
```

If you want to grant the same permissions to the world (others) as the group for `testfile` in the current directory has, execute the following command:

```
chmod o=g testfile
```

Note:
Modifying the permissions does not have any effect on the `root` user. The `root` user has access to all files and directories regardless of the permissions you may have granted.

chgrp

If you want to change the group to which the file belongs, use the `chgrp` command. The group must be one of the groups to which the owner belongs. That is, the group must be either the primary group or one of the secondary groups of the owner.

Examples

Assume that user `testuser` owns the file `testfile`, and the group of the file is `staff`. Also assume that `testuser` belongs to the groups `staff` and `devt`. To change the owner of `testfile` from `staff` to `devt`, execute the following command:

```
chgrp devt testfile
```

chown

If you want to change the owner of a file or directory, use the `chown` command.

Caution:

On UNIX systems with disk quotas, only the `root` user can change the owner of a file or directory.

Examples

If the file `testfile` is owned by the user called `testuser`, you can change the ownership of the file to user `friend` by executing the following command:

```
chown friend testfile
```

rm

When you are done with a file and do not want to use it any more, you may want to remove the file to regain the disk space used by the file. The `rm` command lets you do this by removing files permanently from the disk. If an entry is the last link to a file, the file is deleted. To remove a file from a directory, you do not need either read or write permission to the file, but you do need write permission to the directory containing the file. The `rm` command is usually used to remove files, but it provides a special flag `-r` to remove files in a directory recursively, including the directory and its subdirectories.

Following is a list of some of the flags that can be used with the `rm` command:

	Flag	Meaning
-	<code>-i</code>	Interactively remove the files.
	<code>-f</code>	Remove the files without any messages. This flag does not generate any messages for cases in which a file does not exist or you do not have permission to remove one or more files.
	<code>-r</code>	Remove files within a directory and directories themselves recursively.

The native version of the `rm` command does not ask for confirmation before removing files. You should be careful when using wildcards with the `rm` command.

Examples

If you want to remove all files starting with `test` in the current directory, execute the following command:

```
rm test*
```

However, if you make a typing mistake and type `rm test *`, you will remove all the files because of the asterisk (*).

Caution:

Be careful when using wildcards with the `rm` command. Be sure that you have typed the command correctly before you remove a file; once a file is removed, it cannot be recovered.

To avoid disastrous mistakes, use the `-i` flag to indicate that you want to execute the `rm` command in interactive mode. In this mode, the system asks you for confirmation before removing files. Only if you confirm the prompt by pressing `Y` will the system remove the file. Following is the dialog you can have with the system if you want to remove two files in the current directory: `testfile1` and `testfile2`, using the `-i` flag with the `rm` command:

```
rm -i testfile*
Remove file testfile1? y
Remove file testfile2? y
```

You can use the `-f` flag with the `rm` command if you do not want to see any messages from the command. Usually, `rm` displays a message if a file is not present (for example, if you provide an incorrect filename). However, using the `-f` flag forces `rm` to not display any messages. Suppose that you execute the following command:

```
rm -f testfile
```

The file `testfile` is deleted if it is present, and no action is taken if `testfile` is not present. In either case, you will not see any message from the `rm` command. Also, the `rm -f` command always has a return code of 0 (zero).

You can use the `-r` flag to remove files in directories recursively, including directories and subdirectories. If there is a directory called `testdir` under the current directory that, in turn, has the files `testfile1` and `testfile2`, you can execute the following command to remove the files `testfile1` and `testfile2` as well as the directory `testdir`:

```
rm -r testdir
```

It is advisable that you set up an alias for the `rm` command as `rm -i` in your environment so that you have to confirm before deleting any files.

The `rm` command processes a hard-linked file in a different way. If you have a `testfile1` file in your current directory, execute the following command to create a file called `testfile2` that is hard-linked to the file `testfile1`:


```
ln testfile1 testfile2
```

This command, in effect, creates two identical files: `testfile1` and `testfile2`. Suppose that you now execute the following command:

```
ls -l testfile*
```

You get the following result:

```
-rw-r--r--    2 testuser    staff          10 Nov  3 14:28 testfile1
-rw-r--r--    2 testuser    staff          10 Nov  3 14:28 testfile2
```

In this example, both `testfile1` and `testfile2` show the number of links as 2 because they are linked using a hard link. Now suppose that you remove the file `testfile1` using the `rm` command as follows:

```
rm testfile1
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

The system must take two actions—to remove the file `testfile1` and to decrease the link count of the file `testfile2` from 2 to 1. If you repeat the `ls` command, you get the following display:

```
-rw-r--r--    1 testuser      staff              10 Nov  3 15:38 testfile2
```

Notice that the number of links for `testfile2` is 1.

mv

If you are not satisfied with a filename, you may want to name the file differently. The `mv` command lets you do that. In addition, the `mv` command allows you to move files from one directory to another, retaining the original filename. This action is equivalent to copying the files from the source directory to the destination directory and then removing the file from the source directory. You may choose to move files if you are reorganizing your files. When you are moving files or directories, the target file or directory gets the permission of the source file or directory, regardless of whether or not the target file or directory already exists.

Following is a list of some of the flags that may be used with the `mv` command:

Flag	Meaning
-i	Move or rename files interactively.

-f

Move or rename files without any messages. Use of this flag suppresses messages when you are trying to rename a nonexistent file or you do not have permission to rename a file.

The `mv` command takes two arguments. The first argument is the source file or directory name; the second argument is the destination file or directory. However, the behavior of the `mv` command depends on whether the destination file or directory name exists.

If you move files within the same file system, all links to other files are retained. But if you move the files across file systems, the links are not retained.

Examples

To rename a file in the current directory, use the following command:

```
mv source_file dest_file
```

If the file `dest_file` does not exist, a new `dest_file` is created by copying `source_file` into it; the original `source_file` is removed. If `dest_file` exists and you have write permission to it, `source_file` is copied to `dest_file` and removed. On the other hand, if you do not have permission, the `mv` command does not take any action.

To move `source_file` from the current directory to the `/u/testuser/target_dir` directory, retaining the name, execute one of the following commands:

```
mv source_file /u/testuser/target_dir
```

```
mv source_file /u/testuser/target_dir/.
```

If the file already exists in `/u/testuser/target_dir`, the existing file is overwritten.

To move `source_file` from the current directory to the `/u/testuser/target_dir` directory with the name `target_file`, execute the following command:

```
mv source_file target_dir/target_file
```

-

If you are not sure whether the file `target_file` exists, use the `-i` flag as follows:

```
mv -i source_file target_dir/target_file
```

If the file `target_file` exists, the system prompts you to confirm whether you want to move the file.

The `.` (period) as the target filename indicates that the source filename is to be retained. This convention is especially useful if you want to move multiple files to another directory. If you want to move all files with names beginning with `test` to the `/u/testuser/target_dir` directory, execute the following command:

```
mv test* /u/testuser/target_dir/.
```

To rename the directory `source_dir` to the `/u/testuser/target_dir` directory, execute the following command:

```
mv /u/guahs/source_dir /u/testuser/dest_dir
```

If the directory `dest_dir` does not exist, the directory `/u/testuser/source_dir` is renamed to `/u/testuser/dest_dir`. If `/u/testuser/dest_dir` exists and you have write permissions to it, all the files and subdirectories under `/u/testuser/source_dir` are moved to `/u/testuser/dest_dir`.

cp

The `cp` command can be used to make a copy of the contents of one or more source files as specified target files. If the target file already exists, it is overwritten with the contents of the source file. The behavior of the `cp` command varies depending on whether the source and the target are files or directories.

Following is a list of some of the flags that can be used with the `cp` command:

Flag	Meaning
-p	Retain the modification date and time as well as permission modes of the source file.
-i	Execute the <code>copy</code> command in interactive mode so that it asks for confirmation if the target file already exists.
-h	Follow the symbolic links.
-r	Copy files under the specified directories and their subdirectories. Treat special files, such as linked files, the same way as regular files.

Examples

You can execute the following `cp` command (its simplest form) to copy `source_file` to `target_file` under the current directory:

```
cp source_file target_file
```

If you want to copy `source_file` to the `/u/testuser/target_dir` directory retaining the filename, execute the following command:

```
cp source_file /u/testuser/target_dir/.
```

To copy all files in `/u/testuser/source_dir` to the `/u/testuser/target_dir` directory while retaining the filenames, the last modification date and time, and permissions, execute the following command:

```
cp -p /u/testuser/source_dir/* /u/testuser/target/dir/.
```

This command does not copy any subdirectories or files under those subdirectories. To copy all the files in a directory, as well as its subdirectories and files in those subdirectories, while retaining the last modification date and time and permissions for all files and subdirectories, use the following command:

```
cp -r /u/testuser/source_dir /u/testuser/target_dir/.
```

If you are not sure whether the target file already exists, use the `-i` flag. Following is the dialog for copying `testfile` from the current directory to the `/u/testuser/testdir` directory assuming that `testfile` already exists in the `/u/testuser/testdir` directory:

```
cp -i testfile /u/testuser/testdir/.  
overwrite /u/testuser/testdir/testfile? y
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

cat

You have seen that the `cp` command allows you to copy one file into another file. It does not allow you, however, to copy multiple files into the same file. To concatenate multiple files into a single file, use the `cat` command. By default, the `cat` command generates output into the standard output and accepts input from standard input. The `cat` command takes in one or more filenames as its arguments. The files are concatenated in the order they appear in the argument list.

Following is a list of some of the flags that can be used with the `cat` command:

Flag	Meaning
-b	Eliminate line numbers from blank lines when used with the <code>-n</code> flag.
-e	Display a \$ (dollar sign) at the end of each line, when specified with the <code>-v</code> flag.
-n	Display output lines preceded by line numbers, numbered sequentially from 1.
-q	Suppress messages if the <code>cat</code> command cannot find one or more of the input files.
-v	Display nonprintable characters in the file as printable characters.

Caution:

If you are using the output redirection operator (>) to redirect the standard output of the `cat` command, be careful not to use one of the input filenames as the output filename. If you do that, the input filename is overwritten. Some UNIX versions may give you an error message when you try to do that but will overwrite the file anyway.

If you are accepting input from the standard input, you should press `Ctrl+D` to indicate the end of the input.

Examples

In its most simple form, you can just type the command `cat`, which puts you in entry mode. In this mode, you can type multiple lines; press `Ctrl+D` to signal the end of the input. The `cat` command will display the lines you have just entered:

```
cat
This is test line 1
This is test line 1
This is test line 2
This is test line 2
Ctrl+D
```

You should be aware that the `cat` command does not provide any prompt, as is shown in the preceding example.

If you want display a file called `testfile` in the current directory on your terminal, execute the following command:

```
cat testfile
```

This command produces output such as the following:

```
This is a test file
This does not contain anything meaningful
This is for demo only
```

Be careful if the file is big. A large file will scroll by on your terminal, and you will see only the last few lines. You can get around this by piping the output to either the `more` or `pg` command as follows:

```
cat testfile | more
```

- To concatenate multiple files for display on the terminal, use the following command:

```
cat testfile1 testfile2 testfile3
```

If you want to concatenate these files into a file called `testfile`, use the redirection operator `>` as follows:

```
cat testfile1 testfile2 testfile2 > testfile
```

If the file `testfile` already exists, it is overwritten with the concatenated files `testfile1`, `testfile2`, and `testfile3`. If `testfile` already exists and you want to concatenate at the end of the existing file, instead of using the redirection operator `>`, use the `>>` operator (two consecutive greater-than signs) as follows:

```
cat testfile1 testfile2 testfile2 >> testfile
```

If you try to concatenate a file or a number of files but one or more files do not exist, `cat` concatenates all the available files and, at the end, generates a message about the nonexistent files. Suppose that you want to concatenate two files, `testfile1` and `testfile2`, into the file `testfile` in the current directory, but you mistype `testfile2` as `testtile2` while executing the following command:

```
cat testfile1 testtile2 > testfile
```

In this example, you will get a message similar to the following, and `testfile` will have only the contents of `testfile1`:

```
cat: cannot open testtile2
```

If you use the `-q` flag, you will not see the error message.

If you have `testfile` in the current directory containing the following lines (note that the last line contains special characters), `cat` shows the following:

```
This is a test file
```

```
This file does not contain anything meaningful  
This file is for demo only  
^F^F^F^F^F
```

If you execute the `cat` command with the `-n` flag, `cat` will display lines with line numbers, but the last line with special characters will be displayed as a blank line:

```
cat -n testfile  
  1 This is a test file  
  2  
  3 This file does not contain anything meaningful  
  4 This is for demo only  
  5
```

If you want to be sure that the blank lines displayed actually do not contain any characters other than nonprintable ones, use the `-v` flag with the `cat` command. This flag ensures that the nonprintable characters are displayed as printable characters:


```
cat -v testfile
```

```
This is a test file
```

```
This file does not contain anything meaningful
```

```
This is for demo only
```

```
^F^F^F^F^F^F
```

rcp

So far, we have seen a number of commands that move or copy files between directories within the local host. If you need to copy files from one host to another, you can use the `rcp` command to copy files between the same or different hosts. You can execute the `rcp` command on a local host to copy files between the local host and a remote host or between two remote hosts.

The filename on the remote host is preceded by the remote host ID as `hostname:dirname/filename`. The colon (:) is used as a delimiter between the host name and the filename.

It is also possible to specify the user name at the remote host as `username@hostname:dirname/filename`. The @ (at sign) is used as a delimiter between the user name and the host name. The user name, however, is optional. If not specified, the user name at the remote host is the same as user name at the local host.

If neither the source nor the target file specifies the host name, the `rcp` command behaves the same way as the `cp` command.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

If the filename on the remote host is not qualified fully, starting with the root directory, the filename or directory name is assumed to start with the home directory of the remote user.

If the files do not already exist on the remote host, they are created with the default permission of the remote user. If the files already exist on the remote host, the permissions of the target files are preserved.

As you can with the `cp` command, you can use the `rcp` command to copy directories and files within directories.

Following is a list of some of the flags that can be used with the `rcp` command:

Flag	Meaning
-p	Create the target file with the modification date and time of the source file as well as the permission of the source file.
-r	Copy files recursively while copying directories.

Note:
To use the `rcp` command to transfer files from or to a remote host, you must have the local host name defined in the `/etc/hosts.equiv` file at the remote host, or the local host name and the user name defined in the `.rhosts` file in the user's home directory on the remote host.

Examples

If you want to copy `testfile` from the current directory to `testfile` in the directory `testdir` under the home directory on the remote host called `otherhost`, execute the following command:

```
rcp testfile otherhost:testdir/testfile
```

If the user name on the local host is `testuser`, this command will assume that the user name on the remote host is `testuser`. If the user name `testuser` does not exist on the remote host and you must use the user name `newtestuser` on the remote host, execute the following command:

```
rcp testfile newtestuser@otherhost:testdir/testfile
```

If you must transfer `testfile` from a remote host `otherhost1` to another remote host `otherhost2`, and you want to preserve the modification date and time as well as the permissions, execute the following command:

```
rcp -p testuser1@otherhost1:testfile testuser2@otherhost2:testfile
```

This command copies `testfile` from the home directory of user `testuser1` on the remote host `otherhost1` to `testfile` in the home directory of `testuser2` on remote host `otherhost2`.

If you want to copy all the files in the directory `/u/testuser/testdir` from the remote host `otherhost` to the current directory on the local host, execute the following command:

```
rcp testuser@otherhost:/u/testuser/testdir/* .
```

- This command does not copy any subdirectories you may have in `testdir` or any files in those subdirectories. To copy all the subdirectories and files in those subdirectories, use the following command:

```
rcp -r testuser@otherhost:/u/testuser/testdir/* .
```

ln

Sometimes, you have to provide alternate names for a file. This task can be achieved by linking a filename to another filename using the `ln` command. It is possible to link a file to another name in the same directory or to the same name in another directory.

When linking a filename to another filename, you can specify only two arguments: the source filename and the target filename. When linking a filename to a directory, you can specify multiple filenames to be linked to the same directory.

If you are linking using hard links, you cannot link to a file in another file system. If you are using soft links, however, you can link filenames across file systems.

The flags that can be used with the `ln` command are as follows:

Flag	Meaning
------	---------

<code>-s</code>	Create a soft link to another file or directory. In a soft link, the linked file contains the name of the original file. When an operation on the linked filename is done, the name of the original file in the link is used to reference the original file.
<code>-f</code>	Ensure that the destination filename is replaced by the linked filename if the file already exists.

By default, the `ln` command creates a hard link.

Examples

If you want to link `testfile1` to `testfile2` in the current directory, execute the following command:

```
ln testfile1 testfile2
```

This command creates a hard-linked `testfile2`, linking it to `testfile1`. In this case, if one of the files is removed, the other will remain unaltered.

If `testfile` is in the current directory and is to be linked to `testfile` in the directory `/u/testuser/testdir`, execute the following command:

```
ln testfile /u/testuser/testdir
```

To create a symbolic link of `testfile1` in the current directory, execute the following command:

```
ln -s testfile1 testfile2
```

This command creates a linked `testfile2`, which contains the name of `testfile1`. If you remove `testfile1`, you are left with an orphan `testfile2`, which points nowhere.

If you want to link all the files in the current directory to another directory, `/u/testuser/testdir`, execute the following command:

```
ln * /u/testuser/testdir/.
```

Directory-Manipulation Commands

When you are set up as a user in a UNIX operating system, you usually are set up with the directory `/u/username` as your home directory. To organize your files, you must set up directories of your liking. The following sections present the commands to create and remove directories.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

mkdir

To create a directory, use the `mkdir` command. The `mkdir` command accepts multiple directory names for creation at the same time. As you did with files, you can use a relative path name or an absolute path name to create a directory. To create a directory, you must have write permission for its parent directory. UNIX uses the current permission settings (refer to the `umask` command) to set the permission for the directory.

Following is a list of the flags that can be used with the `mkdir` command:

Flag	Meaning
-p	Create all the directories in the part name of the specified directory if they do not exist.
-m	Permission to specify permission for the directory to be created.

Examples

If your current directory is `/u/testuser`, you can use the following command to

create a directory called `temp` under the directory `/u/testuser`, whose absolute path name is `/u/testuser/temp`:

```
mkdir temp
```

You can also use the following command to have the same effect as the previous one:

```
mkdir /u/testuser/temp
```

Use this command to create the `/u/temp` directory:

```
mkdir ../temp
```

In this example, we used `..` (two consecutive periods) as part of the relative path name to indicate that the directory `temp` will be created under the directory one level up, that is, `/u`.

To create `testdir1` and `testdir2` in the current directory, use the following command:

```
mkdir testdir1 /u/testuser/temp/testdir2
```

This command creates `testdir1` in the current directory and creates `testdir2` in `/u/testuser/temp` (assuming that it exists). In this example, `testdir1` uses a relative path name and `/u/testuser/temp/testdir2` uses an absolute path name.

If the directory `testdir` is already present and you try to create the directory again, you get a message similar to the following:

```
mkdir: cannot create testdir.  
testdir: File exists
```

If you want to create the directory `testdir` under the current directory and grant the access 770 to it, execute the following command:

```
mkdir -m 770 testdir
```

If you want to create the directory `testdir` under the current directory and create the subdirectory `temp` under `testdir`, create both of them using a single command as follows:

```
mkdir -p testdir/temp
```

rmmdir

When you are done with a directory or you run out of space and want to remove a directory, use the `rmmdir` command. You can remove a directory only if it is empty, that is, if all the files and directories in it have been removed. You can specify multiple

directory names as arguments to `rmdir` command. To remove a directory, you must have write permission to the parent directory.

You can use the `-p` flag with the `rmdir` command. The `-p` flag removes all the directories in the specified path name.

Examples

If your current directory is `/u/testuser`, and it contains the `temp` subdirectory, you can remove `temp` with this command:

```
rmdir temp
```

If the directory `temp` is not empty, you will get a message similar to the following:

```
rmdir: Directory temp is not empty.
```

Assume that you are in the directory `/u/testuser` and that it contains a subdirectory `testdir`; also assume that the subdirectory `testdir` contains a subdirectory `temp`. To remove the directory `testdir` in the current directory and the subdirectory `temp` under `testdir`, execute the following command (assuming that all the files and directories under them have been removed):

```
rmdir -p testdir/temp
```

File Information Commands

Each file and directory in UNIX has several attributes associated with it. UNIX provides several commands to inquire about and process these attributes.

ls

The `ls` command can be used to inquire about the various attributes of one or more files or directories. You must have read permission to a directory to be able to use the `ls` command on that directory and the files under that directory. The `ls` command generates output to standard output, which can be redirected to a file using the UNIX redirection operator `>`.

You can provide the names of one or more filenames or directories to the `ls` command. The file and directory names are optional. If you do not provide them, UNIX processes the current directory.

By default, the list of files within a directory is sorted by filename. You can modify the sort order by using some of the flags discussed later.

You should be aware that files starting with a `.` (period) are not processed unless you use the `-a` flag with the `ls` command. This means that entries starting with `.` (single period) and `..` (two consecutive periods) are not processed by default.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



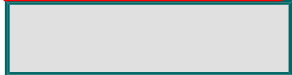
Brief

Full

Advanced

Search

Search Tips



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Following is a list of some of the flags that can be used with the `ls` command:

Flag	Meaning
-A	List all entries in a directory except <code>.</code> (single period) and <code>..</code> (two consecutive periods).
-a	List all entries in a directory including hidden files (filenames starting with a <code>.</code> (period)).
-b	Display nonprintable characters. The characters are displayed in an octal (<code>\nnn</code>) notation.
-c	Use the time of the last modification of the <code>i</code> -node. When used with the <code>-t</code> flag, the output is sorted by the time of last modification of the <code>i</code> -node. When used with the <code>-l</code> flag, the time displayed is the last modification time of the <code>i</code> -node. This flag must be used with the <code>-t</code> or <code>-l</code> flag.
-C	Sort output vertically in a multiple column format. This is the default method when output is to a terminal.

- d Restrict the information displayed to that of only the directory specified. By default, the information about the files or subdirectories under a directory is also displayed.
- e Display the following information for each specified file or directory:
 - Permission associated with the files and directories
 - Number of links
 - Owner
 - Group
 - Size (in bytes)
 - Time of last modification
 - Name of each file. For a special file, the size field contains the major and minor device numbers. If the file is a symbolic link, the path name of the linked-to file is printed preceded by a -> (a minus sign followed by a greater-than sign). The attributes of the symbolic link are displayed.
- f List the name in each slot for each directory specified in the directory parameter. This flag turns off the -l, -t, -s, and -r flags and turns on the -a flag. The order of the listing is the order in which entries appear in the directory.
- F Put special characters before different file types as follows:
 - / (slash) after each directory
 - * (asterisk) if the file is executable
 - = (equal sign) if the file is a socket
 - | (pipe sign) if the file is a FIFO
 - @ (at sign) for a symbolic link
- g Display the following information for files and directories:
 - Permission
 - Number of links
 - Group
 - Size (in bytes)
 - Time of last modification
- i Display the i-node number in the first column of the report for each file.
- l Display the following information about specified files and directories:
 - Permission
 - Number of links

	<ul style="list-style-type: none"> • Owner • Group • Size (in bytes) • Time of last modification
-m	Display the output in a comma-separated format.
-n	Display the following information for specified files and directories: <ul style="list-style-type: none"> • Permission • Number of links • Owner ID • Group ID • Size (in bytes) • Time of last modification
-o	Display the following information about specified files and directories: <ul style="list-style-type: none"> • Permission • Number of links • Owner ID • Size (in bytes) • Time of last modification
-p	Put a slash (/) after each directory name.
-q	Display nondisplayable characters in filenames as a ? (question mark).
-r	Reverse the order of the sort. If the list is to be displayed in name order, it will be displayed in reverse name order. If the list is to be displayed in descending time order (using the -t flag, that is, the latest one first), the list will be displayed in ascending time order (oldest one first).
-R	List all subdirectories recursively under the specified directory.
-s	Provide the size of files and directories in kilobytes.
-t	Sort the list of entries by time of last modification (latest first) instead of by name.
-u	Use the time of last access instead of the time of last modification. If used with -l, the time of last access is displayed instead of the time of last modification. If used with -t, the output is sorted by the time of last access instead of last modification. This flag must be used with the -l and -t flags.
-x	Sort the output horizontally in a multiple-column format.
-1	Display the output as one entry per line.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

The permission details displayed by the `ls` command when certain flags are used (such as `-l`) consists of 10 characters, details of which are as follows:

- **Byte 1:** `d` designates a directory, `b` designates a block special file, `c` designates a character special file, `l` designates a symbolic link, `p` designates a first-in, first-out (FIFO) special file, `s` designates a local socket, `-` designates an ordinary file (for example, one that contains text).
- **Byte 2:** `r` if read permission for owner has been granted, `-` (hyphen) if read permission for owner has not been granted.
- **Byte 3:** `w` if write permission for owner has been granted, `-` (hyphen) if write permission for owner has not been granted.
- **Byte 4:** `x` if execute permission for owner has been granted, `-` (hyphen) if execute permission for owner has not been granted, `s` if the file has set-user-ID mode.
- **Byte 5:** `r` if read permission for group has been granted. `-` (hyphen) if read permission for group has not been granted.
- **Byte 6:** `w` if write permission for group has been granted, `-` (hyphen) if write permission for group has not been granted.
- **Byte 7:** `x` if execute permission for group has been granted, `-` (hyphen) if execute permission for group has not been granted, `s` if the file has setgroup-ID mode.
- **Byte 8:** `r` if read permission for others has been granted, `-` (hyphen) if read permission for others has not been granted.
- **Byte 9:** `w` if write permission for others has been granted, `-` (hyphen) if write permission for others has not been granted.
- **Byte 10:** `x` if execute permission for others has been granted, `-` (hyphen) if

execute permission for others has not been granted.

The execute permission for a file means that the file is an executable file. But the execute permission for a directory means that you can execute searches on the specified directory to locate one or more files.

Examples

Assume that the following files and directories are present in the current directory: `.`, `dot1`, `test1`, `test2`, `test3`, and `test4`. Also assume that `test2` is a directory.

The simplest form of the `ls` command can be used to get the list of files and directories in the current directory as follows:

```
ls
test1  test2 test3  test4 test5
```

In this list, the entry `.dot1` is not displayed because the file `.dot1` is a hidden file. To display all the entries including the hidden files, execute the following command:

```
ls -a
.  .. .dot1 test1  test2 test3  test4 test5
```

From this output, you cannot see details about any entry. To get a detailed list of all the files and directories, execute the following command:

```
ls -la
total 56
drwxrwx---  3 testuser  author    3072 Nov 24 17:35 .
-rw-r--r--  1 testuser  author         0 Nov 24 14:54 .dot1
drwxr-xr-x 36 root      system    2048 Nov 23 19:51 ..
-rw-r--r--  1 testuser  author        10 Nov 24 17:36 test1
drwxr-xr-x  2 testuser  author     512 Nov 24 17:32 test2
-rw-r--r--  1 testuser  author         0 Nov 24 14:58 test3
-rw-r--r--  1 testuser  author         0 Nov 24 17:33 test4
-rw-r--r--  1 testuser  author    11885 Nov 24 11:50 test5
```

Use of the `-a` flag displays the two special entries that are present in all directories: `.` (a single period) to identify the specified directory and `..` (two consecutive periods) to identify the parent directory of the specified directory. In the preceding example, `.` (a single period) identifies current directory; `..` (two consecutive periods) identify the parent directory.

If you just want to have a list of directories, execute the following command with the `-d` flag:

```
ls -ald
drwxrwx---  3 testuser  author    3072 Nov 24 17:15 .
```

As you have seen in these examples, the list of files and directories are ordered by name. If you want to get a list of the entries by time of last modification so that you know which

files you worked on last, execute the following command:

```
ls -lat
total 56
drwxrwx---    3 testuser    author    3072 Nov 24 17:37 .
-rw-r--r--    1 testuser    author      10 Nov 24 17:36 test1
-rw-r--r--    1 testuser    author       0 Nov 24 17:33 test4
drwxr-xr-x    2 testuser    author    512 Nov 24 17:32 test2
-rw-r--r--    1 testuser    author       0 Nov 24 14:58 test3
-rw-r--r--    1 testuser    author       0 Nov 24 14:54 .dot1
-rw-r--r--    1 testuser    author   11885 Nov 24 11:50 test5
drwxr-xr-x   36 root        system    2048 Nov 23 19:51 ..
```

Until now, we have not specified any file or directory name in the `ls` command. If you want to search for all entries that start with `test`, specify `test*` as the entry name, as in this example:

```
ls -la test*
-rw-r--r--    1 testuser    author      10 Nov 24 17:36 test1
-rw-r--r--    1 testuser    author       0 Nov 24 14:58 test3
-rw-r--r--    1 testuser    author       0 Nov 24 17:33 test4
-rw-r--r--    1 testuser    author   11885 Nov 24 11:50 test5

test2:
total 16
drwxr-xr-x    2 testuser    author    512 Nov 24 17:32 .
drwxrwx---    3 testuser    author   3072 Nov 24 17:41 ..
-rw-r--r--    1 testuser    author       0 Nov 24 17:45 test21
-rw-r--r--    1 testuser    author       0 Nov 24 14:58 test22
```

Notice that the entries `.` (single period), `..` (two consecutive periods), and `.dot1` are not displayed because the wildcard `*` (asterisk) does not match the `.` (period) character.

If you want to obtain a comma-separated list of file and directory names in the current directory, execute the following command with the `-m` flag:

```
ls -am
., .., .dot1, test1, test2, test3, test4, test5
```

If you want to obtain a list of entries while being able to identify the directories with `/` (slash), execute the following command with the `-p` flag:

```
ls -ap
./          test1          test4
../         test2/        test5
.dot1       test3
```

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

A similar output can be obtained using the `-F` flag. `-F` is more versatile than the `-p` flag because `-F` can also identify executable files, symbolic links, and so on.

If you want to get the list of entries in the reverse order of name, execute the following command with the `-r` flag:

```
ls -rla
total 56
-rw-r--r--  1 testuser  author  11885 Nov 24 11:50 test5
-rw-r--r--  1 testuser  author    0 Nov 24 17:33 test4
-rw-r--r--  1 testuser  author    0 Nov 24 14:58 test3
drwxr-xr-x  2 testuser  author   512 Nov 24 17:32 test2
-rw-r--r--  1 testuser  author   10 Nov 24 17:36 test1
-rw-r--r--  1 testuser  author    0 Nov 24 14:54 .dot1
drwxr-xr-x 36 root      system  2048 Nov 23 19:51 ..
drwxrwx---  3 testuser  author  3072 Nov 24 18:00 .
```

To obtain a list of all files in the current directory as well as all files under all the subdirectories, execute the following command with the `-R` flag:

```
ls -lR
total 40
-rw-r--r--  1 testuser  author   10 Nov 24 17:36 test1
drwxr-xr-x  2 testuser  author  512 Nov 24 17:32 test2
-rw-r--r--  1 testuser  author    0 Nov 24 14:58 test3
-rw-r--r--  1 testuser  author    0 Nov 24 17:33 test4
```

```
-rw-r--r--    1 testuser      author    11885 Nov 24 11:50 test5

./test2:
total 0
-rw-r--r--    1 testuser      author          0 Nov 24 17:45 test21
-rw-r--r--    1 testuser      author          0 Nov 24 14:58 test22
```

Following are examples of the `ls` command with and without the `-u` flag. The list without the `-u` flag displays the time of last modification; the list displayed with the `-u` flag displays the time of last access:

```
ls -lu
total 40
-rw-r--r--    1 testuser      author          10 Nov 24 17:34 test1
drwxr-xr-x    2 testuser      author        512 Nov 24 18:19 test2
-rw-r--r--    1 testuser      author          0 Nov 24 14:58 test3
-rw-r--r--    1 testuser      author          0 Nov 24 17:33 test4
-rw-r--r--    1 testuser      author    11885 Nov 24 17:56 test5

ls -l
total 40
-rw-r--r--    1 testuser      author          10 Nov 24 17:36 test1
drwxr-xr-x    2 testuser      author        512 Nov 24 17:32 test2
-rw-r--r--    1 testuser      author          0 Nov 24 14:58 test3
-rw-r--r--    1 testuser      author          0 Nov 24 17:33 test4
-rw-r--r--    1 testuser      author    11885 Nov 24 11:50 test5
```

find

If you are not sure where a particular file is stored, use the `find` command to search for the particular file. The `find` command gives you the flexibility to search for a file by various attributes, such as name, size, permission, and so on. Additionally, the `find` command allows you to execute commands on the files that are found as a result of the search.

The format of the `find` command is as follows:

```
find directory-name search-expression
```

The *directory-name* can be a full path name or a `.` (single period) for the current directory.

Following is a list of terms that can be used with the `find` command:

- `-name filename` to specify the name of the file (including wildcards) to be used for searching. You can also use a range as part of the wildcards. If you want to use wildcard characters, you must specify them within quotes. For example, `"test*"` finds all files starting with the letters `test`. If you specify, `"test[1-2]"`, you will find files that start with `test` and have 1 or 2 as the last characters such as `test1` and `test2`.
- `-size Number` to specify the size of the file to be used for searching. The file size specified is in blocks. To specify that you want to match the size of files less than the specified size, use a `-` (minus sign) in front of the size; to match the size of

files greater than the specified size, use a + (plus sign) in front of the size. Note that the size of a file while matching is always rounded up to the next nearest block. For example, `-size 5` matches files that have size of 5 blocks, `-size -5` matches files that have size of less than or equal to 5 blocks, and `-size +5` matches files that have size of more than 5 blocks.

- `-size Numberc` to specify the size of the file to be used for searching. That is, specify a `c` at the end of the number. The file size is then taken to be specified in number of bytes. To specify that you want to match the size of files less than the specified size, use a - (minus sign) in front of the size; to match the size of files greater than the specified size, use a + (plus sign) in front of the size. For example, `-size 50c` matches files that have a size of 50 bytes, `-size -50c` matches files that have a size of less than or equal to 50 bytes, and `-size +50c` matches files that have a size of more than 50 bytes.
- `-prune` to restrict the `find` command not to process directories recursively. By default, `find` recursively processes all the directories and subdirectories under the specified directory.
- `-atime number` to search for files that have been accessed in the specified number of 24-hour periods. The number of 24-hour periods is computed by adding 1 to the number specified. 0 means the last 24 hours.
- `-mtime number` to search for files that have been modified in the specified number of 24-hour periods. The number of 24-hour periods is computed by adding 1 to the number specified. 0 means the last 24 hours.
- `-ctime number` to search for files whose i-node has been modified in the specified number of 24-hour periods. The number of 24-hour periods is computed by adding 1 to the number specified. 0 means the last 24 hours.
- `-type filetype` to search for a specific type of file. Following is a list of types that can be used:

b	Block special file
c	Character special file
d	Directory
f	Regular file
l	Symbolic link
p	FIFO (a named pipe)
s	Socket

- `-user` to search for files whose owner matches the specified user name.
- `-perm permission` to search for files with a specified permission. The *permission* is specified as an octal number of up to three digits. If the permission is not preceded by a - (hyphen), an exact match of the permission specified is made with the file permissions. If permission is preceded by a - (hyphen), file permission is ANDed with the specified permission. For example, if you want to search for files with owner read permission, use `-perm -400`.
- `-newer filename` to search for files that have a time of modification later than that of the specified filename.
- `-group groupname` to search for files that belong to the specified group.
- `-inum Number` to search for files whose i-node number matches the specified i-node number
- `-links Number` to search for files with a specified number of links. To specify that you want to match the number of links less than the specified number

of links, use a – (minus sign) in front of the number of links; to match a number of links greater than the specified number of links, use a + (plus sign) in front of the number of links.

- `-ls` to print the current path name along with the following attributes:
 - i-node number
 - Size in kilobytes (1024 bytes)
 - Protection mode
 - Number of hard links
 - User
 - Group
 - Size in bytes
 - Modification time
- `-exec command` to execute the command. To execute the command on the list of files found by the `find` command, use `{ }` followed by `\;` (a backslash followed by a semicolon).
- `-ok command` to execute the command. To execute the command on the list of files found by the `find` command, use `{ }` followed by `\;` (a backslash followed by a semicolon). UNIX asks for confirmation before executing the command.
- `-print` to print the output generated as a result of the search.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Click Here!

ITKnowledge

- home
- account info
- subscribe
- login
- search
- My ITKnowledge
- FAQ/help
- site map
- contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

These operators can be specified in conjunction with each other to form complex criteria for searches. You can combine several operators as follows:

- *operator -a operator* to search for files that satisfy both the specified conditions.
- *operator -o operator* to search for files that satisfy either of the specified conditions.
- *!operator* to search for files that do not satisfy the specified condition.

Examples

Assume that the following files exist in the current directory:

```
ls -al
total 64
drwxrwx---  3 testuser  author      3072 Nov 25 00:41 .
drwxr-xr-x 36 root      system      2048 Nov 23 19:51 ..
-rw-r--r--  1 testuser  author         0 Nov 24 14:54 .dot1
-rw-----  1 testuser  author        10 Nov 24 17:36 test1
drwxr-xr-x  2 testuser  author       512 Nov 24 17:32 test2
-r-x-----  1 testuser  author         0 Nov 24 14:58 test3
-rw-r--r--  1 testuser  author         0 Nov 24 17:33 test4
-rw-r--r--  1 testuser  author     15647 Nov 24 18:32 test5
```

You can execute the following `find` command (in its simplest form) to get a list of all the files in the current directory and its subdirectories:

```
find . -print
.
./test5
./test1
./test3
./test4
./test2
```

```
./test2/test21
./test2/test22
./dot1
```

If you want to search for all the files in the current directory that have been modified in the last 24 hours, use the `-mtime` operator as follows:

```
find . -mtime 0 -print
.
./test5
./test1
./test3
./test4
./test2
./test2/test21
./test2/test22
./dot1
```

To search for a file whose permission is 600 (only owner has read and write permissions), execute the following command using the `-perm` operator:

```
find . -perm 600 -print
./test1
```

In this case, only the file that has permission of exactly 600 is displayed. However, if you want to search for a file with owner read and write permission, execute the following command using a `-` (hyphen) in front of 600:

```
find . -perm -600 -print
.
./test5
./test1
./test4
./test2
./test2/test21
./test2/test22
./dot1
```

If you are interested in searching only for directories, use the `-type` operator and execute the following command:

```
find . -type d -print
.
./test2
```

To get more information about the files that are found as a result of the search, use the `-ls` operator and execute the following command:

```
find . -ls
  2    4 drwxrwx---  3 settlea  eod          3072 Nov 25 01:11 .
 16   16 -rw-r--r--  1 testuser  author        647 Nov 24 18:32
./test5
 18    4 -rw-----  1 testuser  author         10 Nov 24 17:36
./test1
 19    0 -r-x-----  1 testuser  author          0 Nov 24 14:58
- ./test3
 20    0 -rw-r--r--  1 testuser  author          0 Nov 24 17:33
./test4
67584  4 drwxr-xr-x  2 testuser  author        512 Nov 24 17:32
```

```

./test2
67585      0 -rw-r--r--  1 testuser      author      0 Nov 24 17:45
./test2/test21
67586      0 -rw-r--r--  1 testuser      author      0 Nov 24 14:58
./test2/test22
    22      0 -rw-r--r--  1 testuser      author      0 Nov 24 14:54
./dot1

```

To search for all filenames that start with `test`, use the `-name` operator and execute the following command:

```

find . -name "test*" -print
./test5
./test1
./test3
./test4
./test2
./test2/test21
./test2/test22

```

As you can see, the `find` command traversed the subdirectory `test2` to obtain the filenames in that directory as well. If you want to restrict the search to only the current directory and leave out the subdirectories, use the operator `-prune` and execute the following command:

```

find . -name "test*" -prune -print
./test5
./test1
./test3
./test4
./test2

```

To find a list of files in the current directory that are newer than the file `test1`, use the operator `-newer` and execute the following command:

```

find . -newer test1 -print
./test5
./test2/test21

```

On the other hand, if you want to find a list of files older than the file `test1`, use the negation operator `!` in conjunction with the operator `-newer`, as in the following command:

```

find . ! -newer test1 -print
.
./test3
./test4
./test2
./test2/test22
./dot1

```

If you want a list of all files that are exactly 10 bytes in size, use the `-size` operator and execute the following command:

```

find . -size 10c -print
./test1

```

If you want to create a list of all files that are less than 10 bytes in size, execute the following command (the command is exactly the same as the preceding one except for the hyphen in front of the 10):


```
find . -size -10c -print
./test3
./test4
./test2/test21
./test2/test22
./dot1
```

If you want a list of all the files that have zero size, execute the `find` command with the `-exec` parameter as follows:

```
find . -size 0c -exec ls -l {} \;
-r-x----- 1 testuser  author          0 Nov 24 14:58 ./test3
-rw-r--r-- 1 testuser  author          0 Nov 24 17:33 ./test4
-rw-r--r-- 1 testuser  author          0 Nov 24 17:45 ./test2/test21
-rw-r--r-- 1 testuser  author          0 Nov 24 14:58 ./test2/test22
-rw-r--r-- 1 testuser  author          0 Nov 24 14:54 ./dot1
```

If you want to remove all the files with zero size but want to confirm the delete before you actually remove them, execute the following command with the `-ok` operator:

```
find . -size 0c -ok rm {} \;
< rm ... ./test3 > (yes)?  y
< rm ... ./test4 > (yes)?  n
< rm ... ./test2/test21 > (yes)?  y
< rm ... ./test2/test22 > (yes)?  y
< rm ... ./dot1 > (yes)?  y
```

In this example, we decided not to remove the file `test4`.

All the examples we have seen so far have used one operator at a time. It is possible to execute the `find` command with complex conditions by combining multiple operators and by using `or` or `and` conditions. If you want to find all the files that start with `test` and have a size of zero, execute the following command:

```
find . -name 'test*' -size 0c -print
./test3
./test4
./test2/test21
./test2/test22
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
 All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

In this example, we combined two different operators. It is possible to use the same operator multiple times and combine it with and or or operators. If you want to search for all files in the current directory that have a size of more than zero bytes *and* less than 50 bytes and whose names start with test, use the following command:

```
find . -size +0c -a -size -50c -name 'test*' -exec ls -l {} \;  
-rw----- 1 testuser      author          10 Nov 24 17:36 ./test1
```

file

The file command can be used to determine the type of the specified file. The file command actually reads through the file and performs a series of tests to determine the type of the file. The command then displays the output as standard output.

If a file appears to be ASCII, the file command examines the first 512 bytes and tries to determine its language. If a file does not appear to be ASCII, the file command further attempts to distinguish a binary data file from a text file that contains extended characters.

If the File argument specifies an executable or object module file and the version number is greater than 0, the file command displays the version stamp.

The file command uses the /etc/magic file to identify files that have some sort of a magic number; that is, any file containing a numeric or string constant that indicates type.

Examples

If you have a file called `letter` in your current directory that contains a letter to your friend, you can execute the following command:

```
file letter
```

This command displays the following result:

```
letter:  commands text
```

If you have an executable program in a file called `prog`, and you are working on IBM RISC 6000 AIX version 3.1, you can execute this command:

```
file prog
```

This command displays the following result (if you are on a RISC 6000 system):

```
prog:      executable (RISC System/6000 V3.1)
```

If you are in the `/dev` directory, which contains all the special files, you can execute this command for a file called `hd1` (which is a disk on which a file system has been defined):

```
file hd1
```

This command displays the following result:

```
hd1:      block special
```

File Content-Related Commands

The following sections discuss some of the commands you can use to look at the contents of a file (or parts of the file). You can use these commands to look at the top or bottom of a file, search for strings in the file, and so on.

more

The `more` command can be used to display the contents of a file one screen at a time. By default, the `more` command displays one screen's worth of data at a time. However, the number of lines displayed can be modified. The `more` command pauses at the end of each page of display. Press the spacebar to display the next page; press the Return or Enter key to display the next line. The `more` command is typically used when output from other commands is piped to the `more` command for display.

Following is a list of flags that can be used with the `more` command:

Flag	Meaning
-d	Prompt to quit, continue, or get help.
-f	Count logical lines in the file.
- <i>number</i>	Set the size of the display window to <i>number</i> .

-p	Disable scrolling. This flag results in the <code>more</code> command clearing the screen at the start of each page.
-s	Display only one blank line when multiple contiguous blank lines are present in the file.
-u	Display lines without special attributes if lines in the file have special attributes.
-v	Prevent display of nondisplayable characters graphically.
-w	Allow you to go back in a file after reaching end-of-file. The default for <code>more</code> is to exit when end-of-file is reached.
+ <i>number</i>	Start display at line <i>number</i> in the file.
+g	Start at the end of the file and be able to go backwards.
+/ <i>pattern</i>	Start in the file at the line number where <i>pattern</i> occurs first.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

As has already been stated, the `more` command pauses at the end of each page of display. There are several subcommands you can use when `more` pauses to control further behavior of the `more` command. These subcommands are as follows:

Subcommand	Action
<code>numberspacebar</code>	Page forward by <i>number</i> and by one screen if <i>number</i> is not specified.
<code>numberd</code>	Page forward by a default number of lines (usually 11) if <i>number</i> is not specified and by <i>number</i> of lines if <i>number</i> is specified.
<code>numberz</code>	Page forward by the specified number of lines if <i>number</i> is specified; otherwise, page forward by one screen page.
<code>numbers</code>	Skip a specified number of lines and display one screen page. If <i>number</i> is not specified, the next line is displayed.

<i>numberf</i>	Skip forward the specified <i>number</i> of screens. If <i>number</i> is not specified, the next screen is displayed.
<i>numberb</i>	Skip backward the specified number of screens. If <i>number</i> is not specified, the previous screen is displayed.
<i>numberCtrl-B</i> or <i>numberCtrl-b</i>	Skip backward the specified <i>number</i> of screens. If <i>number</i> is not specified, the previous screen is displayed.
q	Quit the more command.
v	Invoke the vi editor.
<i>number/expression</i>	Search for the expression and its position for the number of occurrences specified by <i>number</i> . If the file has less than the specified number of occurrences of the expression, the display remains unaltered.
<i>numbern</i>	Search forward for the specified occurrence of the last expression entered. If <i>number</i> is not specified, the next occurrence is searched for and that screen is displayed.
!command	Start <i>command</i> with the filename as an argument if command is specified. If command is not specified, you return to the shell prompt. You can then use the <code>exit</code> command to get back to the more command.
<i>number;n</i>	Skip to the specified file following the current file if you have invoked the more command with multiple files. If the specified relative file <i>number</i> is invalid, more skips to the last file.

`number ; p`

Skip to the specified file before the current file if you have invoked the `more` command with multiple files. If the specified relative file number does not exist, `more` skips to the first file.

`: f` (followed by Return key)

Display the filename of the current file being displayed and the current line number being displayed at the top of the screen.

`: q` or `: Q` (followed by the Return key)

Quit the `more` command.

`.` (single period)

Repeat the last command executed.

Examples

Assume that we have a file called `file1` in the current directory. The content of the file is shown here:

```
This is the line 1
This is the line 2
This is the line 3
This is the line 4
This is the line 5
This is the line 6
This is the line 7
This is the line 8
This is the line 9
This is the line 10
This is the line 11
This is the line 13
This is the line 14
This is the line 15
This is the line 16
This is the line 17
This is the line 18
This is the line 19
This is the line 20
This is the line 21
This is the line 22
This is the line 23
This is the line 24
This is the line 25
```

If you want to display `file1`, use the following command:

`more file1`

```
This is the line 1
This is the line 2
This is the line 3
This is the line 4
This is the line 5
This is the line 6
This is the line 7
This is the line 8
This is the line 9
This is the line 10
This is the line 11
This is the line 13
This is the line 14
This is the line 15
This is the line 16
This is the line 17
This is the line 18
This is the line 19
This is the line 20
This is the line 21
This is the line 22
This is the line 23
--More--(91%)
```

This command has a disadvantage because after the end-of-file is reached, the more command is exited. If do not want to exit from the more command even when the end-of-file is reached, use the `-w` flag. This flag is especially useful if you are looking at a file that is in the process of being created. The following command shows the use of the `-w` flag:

```
more -w file1
```

If you want to start from the bottom of the file rather than at the top of the file and go backwards, use the `+g` flag as in the following command:

```
more +g file1
This is the line 3
This is the line 4
This is the line 5
This is the line 6
This is the line 7
This is the line 8
This is the line 9
This is the line 10
This is the line 11
This is the line 12
This is the line 13
This is the line 14
This is the line 15
This is the line 16
This is the line 17
```



```
This is the line 18
This is the line 19
This is the line 20
This is the line 21
This is the line 22
This is the line 23
This is the line 24
This is the line 25
--More--(EOF)
```

If you want to start the display of the file at line number 20 of file1, use the following command:

```
more +20 file1
```

```
This is the line 20
This is the line 21
This is the line 22
This is the line 23
This is the line 24
This is the line 25
```

If you want to display the five files file1, file2, file3, file4, and file5, execute the following command:

```
more file1 file2 file3 file4 file5
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

Register for EarthWeb's
Million Dollar
Sweepstakes!



ITKnowledge

- home
- account
info
- subscribe
- login
- search
- My
ITKnowledge
- FAQ/
help
- site
map
- contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:


[Previous](#) [Table of Contents](#) [Next](#)

less

The `less` command is one more in the family of commands to view the contents of a file. This command may not be available by default on all UNIX systems. The `less` command behaves similarly to the `more` command, except that the `less` command allows you to go backward as well as forward in the file by default.

Following is a list of subcommands that can be used from the `less` command:

Subcommand	Action
h	Display a list of the subcommands that can be used.
spacebar or Ctrl+v or Ctrl+f or f	Go forward in the file one screen. If preceded by a number, move forward by the specified number of lines.
Return key or Ctrl+N or Ctrl+E or e or j	Move forward by one line. If preceded by a number, move forward by the specified number of lines.

Ctrl+B or b	Go backward in the file by one screen. If preceded by a number, move backward by the specified number of lines.
Ctrl+D or d	Go forward by a half screen (the default of 11 lines). If preceded by a number, move forward by the specified number of lines. The new number is then registered as the new default for all subsequent d or u commands.
g	Go the top of the file by default. If preceded by a number, the file is positioned at the line specified by the number.
G	Go the bottom of the file by default. If preceded by a number, the file is positioned at the line specified by the number.
p or %	Go to the top of the file by default. If preceded by a number between 0 and 100, positions the screen at the percentage of the file specified by the number.
Ctrl+P or Ctrl+K or Ctrl+Y or k or y	Go backward by one line. If preceded by a number, move backward by the specified number of lines.
Ctrl+U or u	Go backward by a half screen (the default of 11 lines). If preceded by a number, move backward by the specified number of lines. The new number is registered as the new default for all subsequent d or u commands.
 Ctrl+L or Ctrl+R or r	Redraw the current screen.
m <i>lowercaseletter</i>	Mark the position with the <i>lowercaseletter</i> marker for use later.
` <i>lowercaseletter</i>	Go to the position marked with <i>lowercaseletter</i> using the m subcommand. If ` is followed by ^ (caret), less displays the top of the file. If ` is followed by \$ (dollar sign), less displays the bottom of the file.
<i>number/pattern</i>	Position the file in the specified occurrence of the <i>pattern</i> in the file. You can use the following special characters to indicate special actions: <ul style="list-style-type: none"> • ! to search for lines that do not contain the specified <i>pattern</i>.

- * to search multiple files if invoked with multiple files.
- @ to start the search at the top of the first file if invoked with multiple files.
- n to repeat the last search executed. If the subcommand is preceded by a number, the file is positioned to the specified occurrence of the *pattern* previously specified. You can use the N subcommand to search in the reverse direction.

:e filename or :E filename	Execute the <code>less</code> command for a specified filename and add it to the list of files for subsequent use.
:n	Execute the <code>less</code> command on the next file on the list of files. The list of files can be specified by invoking <code>less</code> with multiple files, or added to the list by using the <code>:e</code> subcommand. Similarly, you can use the <code>:p</code> subcommand for the previous file in the list.
: <i>number</i> x	Execute the <code>less</code> command on the first file in the list if <i>number</i> is not specified. If <i>number</i> is specified, the <code>less</code> command is executed on the file in that position in the list.
= or :f	Get information about the file and the current position in the file.
q or Q	Exit the <code>less</code> command.
v	Invoke the <code>vi</code> editor with the current filename.
!command	Execute a shell command. If command is omitted, you are put into a shell prompt. You can exit the shell by using the <code>exit</code> command to go back to the <code>less</code> command.

Example

To invoke the `less` command for a file named `file1`, use the following command:

```
less file1
```

tail

You can use the `tail` command to display a file, on standard output, starting at a specified point from the top or bottom of the file. Whether the `tail` command starts at the top of the file or at the end of the file depends on the parameter and flags used.

One of the flags, `-f`, can be used to look at the bottom of a file continuously as it grows in size. By default, `tail` displays the last 10 lines of the file.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



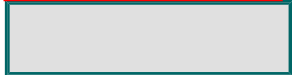
Brief

Full

Advanced

Search

Search Tips



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Following is a list of flags that can be used with the `tail` command:

Flag	Meaning
<code>-c number</code>	Start from the specified character position <i>number</i> .
<code>-b number</code>	Start from the specified 512-byte block position <i>number</i> .
<code>-k number</code>	Start from the specified 1024-byte block position <i>number</i> .
<code>-n number</code>	Start display of the file at the specified line <i>number</i> .
<code>-r number</code>	Display lines from the file in reverse order.
<code>-f</code>	Display the end of the file continuously as it grows in size.

With all these flags, the *number* you specify can be a number prefixed by a + (plus sign) or a – (minus sign). If you specify a +, the `tail` command starts processing from the top of the file. If you specify a – or do not specify any sign, `tail` starts processing from the bottom of the file.

Examples

Assume that we have a file called `file1` that contains 30 lines. The contents of the file are displayed here:

```
This is the line 1
This is the line 2
This is the line 3
This is the line 4
This is the line 5
This is the line 6
This is the line 7
This is the line 8
This is the line 9
This is the line 10
This is the line 11
This is the line 12
This is the line 13
This is the line 14
This is the line 15
This is the line 16
This is the line 17
This is the line 18
This is the line 19
This is the line 20
This is the line 21
This is the line 22
This is the line 23
This is the line 24
This is the line 25
This is the line 26
This is the line 27
This is the line 28
This is the line 29
This is the line 30
```

If you want to see the last 10 lines of the file, execute the `tail` command without any flags as follows:

```
tail file1
This is the line 21
This is the line 22
This is the line 23
This is the line 24
This is the line 25
This is the line 26
This is the line 27
This is the line 28
This is the line 29
This is the line 30
```

In the preceding example, the last 10 lines of `file1` are displayed. If you want to skip 27 lines from the start of the file, execute the following example:

```
tail +27 file1
```

```
This is the line 28  
This is the line 29  
This is the line 30
```

In this example, the display starts at the 28th line from the top of the file. If you want to start from a specified byte position in the file instead of at a particular line position, use the `-c` flag as follows:

```
tail -c +500 file1
```

```
the line 27  
This is the line 28  
This is the line 29  
This is the line 30
```

In this example, the display starts at the 500th byte from the top of the file. If you want to specify an absolute line number from which to display the file, use the `-n` flag as in the following example:

```
tail -n -5 file1
```

```
This is the line 26  
This is the line 27  
This is the line 28  
This is the line 29  
This is the line 30
```

In this example, the display starts at the 5th line from the bottom. If you want to display the lines of `file1` in reverse order, use the `-r` flag as in the following example:

```
tail -r -n -5 file1
```

```
This is the line 30  
This is the line 29  
This is the line 28  
This is the line 27  
This is the line 26
```

In this example, the last five lines are displayed in reverse order, with the last line first.

head

The `head` command displays a file on the standard output. The `head` command starts at the top of the file and displays the specified number of bytes or lines from the top of the file. By default, `head` displays 10 lines.

Following are the flags that can be used with the `head` command:

Flag	Meaning
------	---------

<code>-c <i>number</i></code>	Display the <i>number</i> of bytes from the top of the file.
<code>-n <i>number</i></code>	Display the <i>number</i> of lines from the top of the file.

The *number* can be specified without any sign or can be preceded by a – (minus sign), both of which mean the same thing.

Examples

Assume that we have a file `file1` whose contents are the same as the one shown in the examples for the `tail` command, in the preceding section.

If you want to display a specified number of lines from the top, use the `-n` flag as in the following example:

```
head -3 file1
This is the line 1
This is the line 2
This is the line 3
```

In this example, the first three lines of `file1` are displayed. If you want to display the first specified number of bytes from the top of the file, use the `-c` flag as in the following example:

```
head -c 29 file1
This is the line 1
This is th
```

In this example, the first 29 bytes of `file1` are displayed.

WC

The `wc` command counts the number of bytes, words, and lines in specified files. A *word* is a number of characters stringed together and delimited by either a space or a newline character.

Following is a list of flags that can be used with the `wc` command:

Flag	Meaning
<code>-l</code>	Count only the number of lines in the file.
<code>-w</code>	Count only the number of words in the file.
<code>-c</code>	Count only the number of bytes in the file.

You can use multiple filenames as arguments to the `wc` command.

Examples

If you want to know the number of bytes, words, and lines in `file1`, execute the following command:

```
wc file1
    25      125      491 file1
```

This example shows that `file1` has 25 lines, 125 words, and 491 bytes. If you want to find only the number of words in `file1`, use the `-w` flag as in the following example:

```
wc -w file1
    125 file1
```

If you want to get the word count on `file1` and `file2`, execute the following command:

```
wc -w file1 file2
    125 file1
    463 file2
    588 total
```

Notice that if you use multiple files, the `wc` command displays an extra line in the output that lists the total of all files.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

read

The `read` command is used in shell scripts to read each field from a file and to assign it to a shell variable. A *field* is a string of bytes separated by spaces or newline characters. If the number of fields read is less than the number of variables specified, the rest of the fields are unassigned.

You can use the `-r` flag with the `read` command. Use `-r` to treat a `\`(backslash) as part of the input record and not as a control character.

Examples

Following is a piece of shell script code that reads the first name and last name from the `namefile` file and prints them:

```
while read -r lname fname
do
    echo $lname", "$fname
done < namefile
```

od

The `od` command can be used to display the contents of a file in a specified format. Usually, this command is used to look at executable files or other files that are not text, which most UNIX commands cannot process. You can also specify the offset from which you want the display of the file to start.

Following is a list of the flags that can be used with the `od` command:

Flag	Meaning
-d	Display the output as a signed decimal number.
-i	Display the output as an unsigned decimal number.
-f	Display the output as a floating number.

-b	Display the output as an octal value.
-h	Display the output as a hexadecimal value.
-c	Display the output as ASCII characters.

After the filename, you can specify the offset of the byte at which you want to start the display. If the offset is preceded by 0x, the offset is interpreted as a hexadecimal number. If the offset is preceded by 0, the offset is interpreted as an octal number. The offset can be suffixed by b for bytes, k for kilobytes (1024 bytes), or m for megabytes (1024 × 1024 bytes).

Examples

To display the contents of `file1`, execute the following command:

```
od file1 | more
0000000 000737 000007 031147 104407 000000 000000 000000 000000
0000020 000110 010007 000413 000001 000002 024250 000001 056674
0000040 000012 030504 000001 052320 000000 001000 000000 000000
0000060 000001 055020 000004 000002 000004 000004 000007 000005
```

This command displays `file1` in decimal format. If you want to display the file in hexadecimal format, use the `-h` flag as in the following example:

```
od -h file1 | more
0000000 01df 0007 3267 8907 0000 0000 0000 0000
0000020 0048 1007 010b 0001 0002 28a8 0001 5dbc
0000040 000a 3144 0001 54d0 0000 0200 0000 0000
0000060 0001 5a10 0004 0002 0004 0004 0007 0005
```

If you want to start the display at byte position 40 and display it in ASCII format, use the following command:

```
od -c file1 +40 | more
0000040 \0 \n 1 D \0 001 T 320 \0 \0 002 \0 \0 \0 \0 \0
0000060 \0 001 z 020 \0 004 \0 002 \0 004 \0 004 \0 007 \0 005
```

It is possible to display the contents of `file1` in octal, ASCII, and hexadecimal format all at once, using the following command:

```
od -bch file1 | more
0000000 001 337 000 007 062 147 211 007 000 000 000 000 000 000 000
          001 337 \0 007 2 g 211 007 \0 \0 \0 \0 \0 \0 \0
          01df 0007 3267 8907 0000 0000 0000 0000
0000020 000 110 020 007 001 013 000 001 000 002 050 250 000 001 135 274
          \0 H 020 007 001 013 \0 001 \0 002 ( 250 \0 001 ] 274
          0048 1007 010b 0001 0002 28a8 0001 5dbc
0000040 000 012 061 104 000 001 124 320 000 000 002 000 000 000 000
          \0 \n 1 D \0 001 T 320 \0 \0 002 \0 \0 \0 \0 \0
          000a 3144 0001 54d0 0000 0200 0000 0000
0000060 000 001 132 020 000 004 000 002 000 004 000 004 000 007 000 005
          \0 001 z 020 \0 004 \0 002 \0 004 \0 004 \0 007 \0 005
          0001 5a10 0004 0002 0004 0004 0007 0005.
```

pg

The `pg` command displays the contents of a file one page at a time, just like the `more` and `less` commands do. The `pg` command pauses at the end of each screen display so that you can enter a number of subcommands that can search for a string in the file, go backward or forward in the file, and so on.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#) [Table of Contents](#) [Next](#)

Following is a list of flags that can be used with the `pg` command:

Flag	Meaning
-c	Clear the screen at the end of each page of display and start the display at the top of the screen.
-e	Continue to the next file at the end of one file, if the <code>pg</code> command is invoked with multiple files. Usually, <code>pg</code> pauses at the end of each file.
-f	Truncate lines that are longer than the width of the screen display.
-p <i>string</i>	Display the <i>string</i> as the <code>pg</code> command prompt. The default prompt is <code>:</code> (colon). If the <i>string</i> specified is <code>%d</code> , the page number is displayed at the prompt.
-s	Highlight all messages and prompts issued by the <code>pg</code> command.

<i>+number</i>	Start the display at the specified line number in the file.
<i>-number</i>	Set the size of the display screen to the specified number of lines.
<i>+/pattern/</i>	Search for the <i>pattern</i> in the file and start the display at that line.

A number of subcommands can be used with the `pg` command when it pauses at the end of each screen of display. You must press the Return key after entering each subcommand. Following is a list of some of these subcommands:

Subcommand	Action
<i>-number</i>	Go backward the number of pages specified by <i>number</i> .
<i>+number</i> <i>number</i> .	Go forward the number of pages specified by <i>number</i> .
<i>l</i>	Go forward in the file by one line.
<i>numberl</i>	Start the display in the file at the line specified by <i>number</i> .
<i>+numberl</i>	Go forward in the file the number of lines specified by <i>number</i> .
<i>-numberl</i>	Go backward in the file the number of lines specified by <i>number</i> .
<i>d</i>	Go forward by a half screen.
<i>-d</i>	Go backward by a half screen.
<i>-n</i>	Indicate to <code>pg</code> that it should interpret and execute the subcommands as they are entered without waiting for the newline character to be entered.
<i>Ctrl+L</i>	Redraw the current screen.
<i>\$</i>	Go to the last page of the file.
<i>number/</i> <i>pattern/</i>	Search forward for the <i>pattern</i> in the file starting at the beginning of the next page. If <i>number</i> is specified, <code>pg</code> searches for the specified occurrence number of <i>pattern</i> . The search does not wrap around. If you want to search backward, use <i>?</i> (question mark) instead of <i>/</i> (slash).
<i>numberp</i>	Start executing the <code>pg</code> command on the previous file if <i>number</i> is not specified. If <i>number</i> is specified, start at the file whose position in the list of files is <i>number</i> before the current file.

<i>number</i>	Start executing the <code>pg</code> command on the next file if <i>number</i> is not specified. If <i>number</i> is specified, start at the file whose position in the list of files is <i>number</i> after the current file.
<i>sfilename</i> filename.	Save the current file being processed in the specified
q or Q	Quit the <code>pg</code> command.

Examples

Assume that we have a file `file1` whose content is the same as that shown in the `tail` command example, earlier in this chapter.

To change the number of lines to be displayed by the `pg` command, prefix the size by a `-` (minus sign) as in the following example:

```
pg -7 file1
This is the line 1
This is the line 2
This is the line 3
This is the line 4
This is the line 5
This is the line 6
This is the line 7
:
```

In this example, the number of lines displayed is modified to 7. If you want to start the display at the 7th line, prefix the number with a `+` (plus sign) as in the following example:

```
pg +7 file1
```

If you want to modify the default prompt of `:` (colon) with a personalized prompt, use the `-p` flag as in the following example:

```
pg -7 -s -p "Enter Subcommand -> " file1
This is the line 1
This is the line 2
This is the line 3
This is the line 4
This is the line 5
This is the line 6
This is the line 7
Enter Subcommand ->
```

In this example, the default prompt has been replaced by the `Enter Subcommand ->` prompt. If you want to start the file with the line on which the pattern `line 5` appears, execute the following command:


```
pg +/"line 5"/ file1
```

tee

If you want to execute a command and want its output redirected to multiple files in addition to the standard output, use the `tee` command. The `tee` command accepts input from the standard input, so it is possible to pipe another command to the `tee` command.

You can use the `-a` flag with the `tee` command. Use `-a` to append to the end of the specified file. The default of the `tee` command is to overwrite the specified file.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Examples

If you want to use the `cat` command on `file1` to display it on the screen, but you want to make a copy of `file2`, use the `tee` command as follows:

```
cat file1 | tee file2 | more
```

If you want to append `file1` to the end of an already existing `file2`, use the flag `-a` as in the following example:

```
cat file1 | tee -a file2 | more
```

vi

Caution:

The `vi` command does not lock a file while editing it. It is possible that more than one user can edit the file at the same time. The version of the file saved last is the one that is retained.

The `vi` command can be used to edit one or more files using full-screen mode. If a filename is not provided, UNIX creates an empty work file without any name. If a filename is provided but the file does not exist, UNIX creates an empty work file with

the specified name. The `vi` command does not modify existing files until the changes are saved.

Following is a list of some of the flags that can be used with the `vi` command:

Flag	Meaning
<code>-c subcommand</code>	Execute the specified <i>subcommand</i> before placing the specified file in editing mode.
<code>-r filename</code>	Recover the specified <i>filename</i> .
<code>-R</code>	Place the specified file in editing mode with the read-only option so that any modifications made cannot be saved.
<code>-y number</code>	Set the editing window to a size with <i>number</i> of lines.

Here is a list of the modes the `vi` editor has:

- *Command mode* is the default mode when you enter `vi`. In this mode, you can enter various subcommands to manipulate the lines, such as deleting lines, pasting lines, moving to a different word, moving to a different line, and so on.
- *Text input mode* is the mode in which you can modify the text in the lines or enter new lines. You enter this mode from command mode by using the subcommand `a`, `i`, or `c`. To return to command mode, press the Esc key.
- *Command entry mode* is the mode in which you can enter certain subcommands that require additional parameters. Two of these subcommands are `w` (which requires a filename) and `/` (which requires entry of a pattern). You can use the Esc key to return to command mode.

Following is a quick reference of subcommands that can be used in command mode for *moving within the same line*:

Subcommand	Action
<code>h</code>	Move the cursor left to the previous character in the same line.
<code>l</code>	Move the cursor right to the next character in the same line.
<code>j</code>	Move the cursor down to the next line in the same column.
<code>k</code>	Move the cursor up to the previous line in the same column.
<code>w</code>	Move the cursor to the start of next small word in the same line.
<code>W</code>	Move the cursor to the start of the next big word in the same line.

b	Move the cursor to the start of the previous small word in the same line.
B	Move the cursor to the start of the previous big word in the same line.
e	Move the cursor to the end of the next small word in the same line.
E	Move the cursor to the end of the previous big word in the same line.
f c	Move the cursor to the next character c in the same line.
F c	Move the cursor to the previous character c in the same line.
t c	Move the cursor to one column before the next character c in the same line.
T c	Move the cursor to one column after the previous character c in the same line.
number	Move the cursor to the specified column <i>number</i> .

Following is a quick reference of subcommands that can be used in command mode for *moving across the lines*:

Subcommand	Action
+ or Enter	Move the cursor to the next line's first nonblank character.
-	Move the cursor to the previous line's first nonblank character.
0	Move the cursor to the first character of the current line.
\$	Move the cursor to the last character of the current line.
H	Move the cursor to the top line of the screen.
L	Move the cursor to the last line of the screen.
M	Move the cursor to the middle of the screen.

Following is a quick reference of subcommands that can be used in command mode for *redrawing the screen*:

Subcommand	Action
z-	Make the current line the last line of the screen and redraw the screen.
z.	Make the current line the middle line of the screen and redraw the screen.

Ctrl+L

Redraw the screen.

/*pattern*/z-

Find the next occurrence of *pattern* and make that the last line of the screen.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

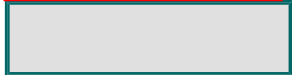
▪



Brief Full

☐ [Advanced Search](#)

☐ [Search Tips](#)



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Following is a quick reference of subcommands that can be used in command mode for *scrolling across pages*:

Subcommand	Action
Ctrl+F	Move forward by one screen.
Ctrl+D	Move forward by one-half screen.
Ctrl+B	Move backward by one screen.
Ctrl+U	Move backward by one-half screen.
Ctrl+E	Scroll the window down by one line.
Ctrl+Y	Scroll the window up by one line.

Following is a quick reference of subcommands that can be used in command mode for *searching for patterns in the file*:

Subcommand	Action
/ <i>pattern</i>	Search for the specified <i>pattern</i> in the forward direction. If end-of-file is reached, the search wraps around.

? <i>pattern</i>	Search for the specified <i>pattern</i> in the backward direction. If the top-of-file is reached, the search wraps around.
n	Repeat the last search in the same direction as was specified in the last search.
N	Repeat the last search in the opposite direction of what was specified in the last search.
/ <i>pattern</i> / + <i>number</i>	Position the cursor the specified <i>number</i> of lines after the line in which the <i>pattern</i> has been found.
/ <i>pattern</i> / - <i>number</i>	Position the cursor the specified <i>number</i> of lines before the line in which the <i>pattern</i> has been found.
%	Find the matching braces or parenthesis.

Following is a quick reference of subcommands that can be used to ***enter text in text-entry mode***. You can terminate text entry at any time by pressing the Esc key.

Subcommand	Action
a	Start entering text after the cursor position.
A	Start entering text at the end of the line.
i	Start entering text before the cursor position.
I	Start entering text before the first nonblank character in the line.
o	Insert an empty line after the line in which the cursor is positioned.
O	Insert an empty line before the line in which the cursor is positioned.

Following is a quick reference of subcommands that can be used to ***modify text from command mode***. You can terminate text entry at any time by pressing the Esc key.

Subcommand	Action
cc or S	Change a complete line.
C	Change the contents of a line after the cursor position.
cw	Change the word at which the cursor is positioned.
dd	Delete the current line.
D	Delete the rest of the line beyond where the cursor is positioned.
dw	Delete part of the word in which the cursor is positioned.

J	Join the contents of the next line to the end of the current line.
r <i>c</i>	Replace the character at the cursor position with the character <i>c</i> .
R	Overwrite the contents of the current line.
u	Undo the last modification.
x	Delete the character at the cursor position.
X	Delete the character to the left of the cursor position.
~ (tilde)	Change uppercase letter to lowercase letter or vice versa.
.	Repeat the last change.
<<	Shift the current line to the left.
>>	Shift the current line to the right.

Following is a quick reference of subcommands that can be used to *move or copy text from one part of the file to another*:

Subcommand	Action
p	Paste the contents of the undo buffer (as a result of deleting or yanking) after the cursor position.
P	Paste the contents of the undo buffer (as a result of deleting or yanking) before the cursor position.
" <i>bd</i>	Delete text into the named buffer <i>b</i> .
" <i>bp</i>	Paste the contents of the named buffer <i>b</i> .
YY	Yank the current line into the undo buffer.
Y	Yank the current line into the undo buffer.
yw	Yank the word from the current cursor position into the undo buffer.

Following is a quick reference of subcommands that can be used to *save a file*:

Subcommand	Action
:w	Save the changes to the original file.
:w <i>filename</i>	Save the changes to the specified <i>filename</i> if the file <i>filename</i> does not exist. If you try to save an already existing file using this subcommand, you will get an error.
!w <i>filename</i>	Save the changes to the specified <i>filename</i> if the file <i>filename</i> already exists.

Following is a quick reference of subcommands that can be used to *move between various files* if you have invoked `vi` with multiple files:

Subcommand	Action
<code>:n</code>	Start editing the next file in the list of files specified when <code>vi</code> was invoked.
<code>:n filenames</code>	Specify a new list of files to be edited.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Following is a quick reference of subcommands that can be used to *move between the current file and the alternate file*:

Subcommand	Action
:e <i>filename</i>	Invoke <i>vi</i> and make <i>filename</i> the alternate file.
:e!	Load the current file again. If changes have been made to the current file, those changes are discarded.
:e + <i>filename</i>	Invoke <i>vi</i> with <i>filename</i> and start editing at the end of the file rather than at the beginning.
:e + <i>number filename</i>	Invoke <i>vi</i> with <i>filename</i> and start editing at the specified line number.
:e #	Start editing the alternate file.

Following is a quick reference of subcommands that can be used to *add lines to the current file from other sources*:

Subcommand	Action
------------	--------

<code>:r filename</code>	Read the complete <i>filename</i> and add it after the current line.
<code>:r !command</code>	Execute the specified <i>command</i> and add the output after the current line.

Following is a quick reference of some of the *miscellaneous subcommands* you can use with `vi`:

Subcommand	Action
Ctrl+G	Get information about the current file being edited.
<code>:sh</code>	Start the shell so that commands can be executed. You can return by using the <code>exit</code> command or by pressing Ctrl+D.
<code>:!command</code>	Execute the specified <i>command</i> .
<code>!!</code>	Reexecute the last <code>:!command</code> .
<code>:q</code>	Quit <code>vi</code> . If you try to quit using this subcommand and you have made modifications to the file, UNIX does not allow you to quit.
<code>:q!</code>	Quit <code>vi</code> regardless of any changes made to the file.
ZZ or <code>:wq</code>	Save changes to the original file and exit <code>vi</code> .

You can use a special file called `.exrc` in which you can specify *special vi subcommands*. To use these subcommands in a `vi` session, use a `:` (colon) in front of the command. Some of these subcommands are as follows:

Subcommand	Action
<code>Ab abb ph</code>	Abbreviate <code>ph</code> to <code>abb</code> .
<code>unab abbreviation</code>	Turn the abbreviation off.
<code>map m seq</code>	Map a sequence of <code>vi</code> commands to a character or key.

File Content Search Commands

We have seen that we can use the `find` command to search for filenames in a directory. To search for a pattern in one or more files, use the `grep` series of commands. The `grep` commands search for a string within the specified files and display the output on standard output.

egrep

The `egrep` command is an extended version of `grep` command. This command

searches for a specified pattern in one or more files and displays the output to standard output. The pattern can be a regular expression in which you can specify special characters to have special meaning, some of which are as follows:

Character	Meaning
.	Match any single character.
*	Match one or more single characters that precede the asterisk.
^	Match the regular expression at the beginning of a line.
\$	Match the regular expression at the end of a line.
+	Match one or more occurrences of a preceding regular expression.
?	Match zero or more occurrences of a preceding expression.
[]	Match any of the characters specified within the brackets.

Following is a list of flags that can be used with the `egrep` command:

Flag	Meaning
-b	Display the block number at the start of each line found.
-c	Display the count of lines in which the pattern was found without displaying the lines.
-f <i>filename</i>	Specify a <i>filename</i> that contains the patterns to be matched.
-h	Suppress the filenames as part of the display if more than one file is being searched.
-i	Search, ignoring the case of the letter.
-l	List just the filenames in which the specified pattern has been found.
-n	Display the relative line number before each line in the output.
-q	Suppress all output.
-s	Display an error message if an error occurs.
-v	Find lines not matching the specified pattern.
-w	Search for specified patterns as words.
-x	Match the patterns exactly to a line.

The `egrep` command has some special features for the patterns you can specify. The features are as follows:

- You can specify a + (plus sign) at the end of a pattern that matches one or more occurrences of the pattern.
- You can specify a ? (question mark) at the end of a pattern that matches zero or one occurrence of the pattern.
- You can specify a | (vertical bar or pipe) between two patterns to match either one or both patterns (or operator).
- You can specify a pattern within a left and a right parentheses to group the patterns.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

- home
- account
info
- subscribe
- login
- search
- My
ITKnowledge
- FAQ/
help
- site
map
- contact us

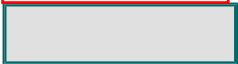


Brief Full

☐ [Advanced](#)

[Search](#)

☐ [Search Tips](#)



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Examples

Assume that we have a file called `file1` whose contents are shown here using the `more` command:

```
more file1
***** This file is a dummy file *****
which has been created
to run a test for egrep
grep series of commands are used by the following types of people
    programmers
    end users
Believe it or not, grep series of commands are used by pros and novices alike
***** THIS FILE IS A DUMMY FILE *****
```

If you want to find all occurrences of `dummy`, use the following command:

```
egrep dummy file1
***** This file is a dummy file *****
```

If you want to find all occurrences of `dummy`, regardless of the case of the letters, use the `-i` flag as in the following example:

```
egrep -i dummy file1
***** This file is a dummy file *****
***** THIS FILE IS A DUMMY FILE *****
```

If you want to display the relative line number of the line that contains the pattern being searched, use the `-n` flag as in the following example:

```
egrep -i -n dummy file1
1:***** This file is a dummy file *****
8:***** THIS FILE IS A DUMMY FILE *****
```

If you are just interested in finding the number of lines in which the specified pattern occurs, use the `-c` flag as in

the following example:

```
egrep -i -c dummy file1
2
```

If you want to get a list of all lines that do not contain the specified pattern, use the `-v` flag as in the following example:

```
egrep -i -v dummy file1
which has been created
to run a test for egrep
grep series of commands are used by the following types of people
    programmers
    end users
Believe it or not, grep series of commands are used by pros and novices alike
```

If you are interested in searching for a pattern that you want to search as a word, use the `-w` flag as in the following example:

```
egrep -w grep file1
grep series of commands are used by the following types of people
Believe it or not, grep series of commands are used by pros and novices alike
```

Notice that the search did not find the pattern `egrep` because that word contains `e` before the pattern `grep`. The use of the `-w` flag forced `egrep` to search for the pattern `grep` delimited by spaces or newline characters.

If you want to search for a pattern that is the only string in a line, use the `-x` command as in the following example:

```
egrep -x "    end users" file1
    end users
```

Now, let's examine some of the special features of `egrep`. If you want to find out where either of two specified patterns occur, use the `|` (vertical bar) to separate the two patterns:

```
egrep "(dummy|pro)" file1
***** This file is a dummy file *****
    programmers
Believe it or not, grep series of commands are used by pros and novices alike
```

In this example, the lines that contain either the pattern `dummy` or the pattern `pro` are displayed. In case you are interested in searching for either `pros` or `programmers`, use the `?` (question mark) at the end of the pattern as in the following example:

```
egrep "pro(grammer)?s" file1
    programmers
Believe it or not, grep series of commands are used by pros and novices alike
```

In this example, the pattern matches both `pros` and `programmers` because `(grammar)?` matches zero or one occurrence of `grammar` with the zero occurrence (for the `pros` line) and with the one occurrence (for the `programmers` line).

To search for lines containing only capital letters `C`, `D`, `E`, or `F`, use regular expressions as follows:

```
egrep [C-F] file1
***** THIS FILE IS A DUMMY FILE *****
```

fgrep

As do `egrep` and `grep`, `fgrep` also searches one or more files for a specified string and displays output on standard output. The `fgrep` command is supposed to be the faster version of the `grep` command, but in reality may not be. Please notice that the `fgrep` command is used to search for a specified *string* and not a pattern (a

string is a regular expression in which special characters can be used to indicate special meaning).

Following is a list of flags that can be used with the `fgrep` command:

Flag	Meaning
-b	Display the block number at the start of each line found.
-c	Display the count of lines in which the pattern was found without displaying the lines.
-f <i>filename</i>	Specify the <i>filename</i> that contains the patterns to be matched.
-h	Suppress the filenames as part of the display if more than one file is being searched.
-i	Search, ignoring the case of the letter.
-l	List just the filenames in which the specified pattern has been found.
-n	Display the relative line number before each line in the output.
-q	Suppress all output.
-s	Display an error message if an error occurs.
-v	Find lines not matching the specified pattern.
-w	Search for specified patterns as words.
-x	Match the patterns exactly with a line.

Examples

Assume that we have a file called `file1` whose contents are shown here using the `more` command:

```
more file1
***** This file is a dummy file *****
which has been created
to run a test for egrep
grep series of commands are used by the following types of people
    programmers
    end users
Believe it or not, grep series of commands are used by pros and novices alike
***** THIS FILE IS A DUMMY FILE *****
```

If you want to find all occurrences of `dummy`, use the following command:

```
fgrep dummy file1
***** This file is a dummy file *****
```

If you want to find all occurrences of `dummy` regardless of the case of the letters, use the `-i` flag as in the following example:

```
fgrep -i dummy file1
***** This file is a dummy file *****
***** THIS FILE IS A DUMMY FILE *****
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc. All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

If you want to display the relative line number of the line that contains the pattern being searched, use the `-n` flag as in the following example:

```
fgrep -i -n dummy file1
1:***** This file is a dummy file *****
8:***** THIS FILE IS A DUMMY FILE *****
```

If you are just interested in finding the number of lines in which the specified pattern occurs, use the `-c` flag as in the following example:

```
fgrep -i -c dummy file1
2
```

If you want to get a list of all lines that do not contain the specified pattern, use the `-v` flag as in the following example:

```
fgrep -i -v dummy file1
which has been created
to run a test for egrep
grep series of commands are used by the following types of people
    programmers
    end users
Believe it or not, grep series of commands are used by pros and novices alike
```

If you are interested in searching for a pattern that you want to search as a word, use the `-w` flag as in the following example:

```
fgrep -w grep file1
grep series of commands are used by the following types of people
Believe it or not, grep series of commands are used by pros and novices alike
```

Notice that the search did not find the pattern `egrep` because that word contains `e` before the pattern `grep`. The use of the `-w` flag forced `fgrep` to search for `grep` delimited by spaces or newline characters.

If you want to search for a pattern that is the only string in a line, use the `-x` command as in the following example:

```
fgrep -x "    end users" file1
    end users
```

grep

The `grep` command can be used to search for a specified pattern in one or more files; it displays the matching output on standard output.

Following is a list of flags that can be used with `grep` command:

Flag	Meaning
-b	Display the block number at the start of each line found.
-c	Display the count of lines in which the pattern was found without displaying the lines.
-E	Indicate that the <code>grep</code> command behaves as the <code>egrep</code> command.
-F	Indicate that the <code>grep</code> command behaves as the <code>fgrep</code> command.
-f <i>filename</i>	Specify <i>filename</i> that contains the patterns to be matched.
-h	Suppress the filenames as part of the display if more than one file is being searched.
-i	Search, ignoring the case of the letter.
-l	List just the filenames in which the specified pattern has been found.
-n	Display the relative line number before each line in the output.
-q	Suppress all output.
-s	Display an error message if an error occurs.
-v	Find lines not matching the specified pattern.
-w	Search for specified patterns as words.
-x	Match the patterns exactly with a line.

- Examples

Assume that we have a file called `file1` whose contents are shown here using the `more` command:

```
more file1
***** This file is a dummy file *****
which has been created
to run a test for egrep
grep series of commands are used by the following types of people
    programmers
    end users
Believe it or not, grep series of commands are used by pros and novices alike
***** THIS FILE IS A DUMMY FILE *****
```

If you want to find all occurrences of `dummy`, use the following command:

```
grep dummy file1
***** This file is a dummy file *****
```

If you want to find all occurrences of `dummy` regardless of the case of the letters, use the `-i` flag as in the following example:

```
grep -i dummy file1
***** This file is a dummy file *****
***** THIS FILE IS A DUMMY FILE *****
```

If you want to display the relative line number of the line that contains the pattern being searched, use the `-n` flag as in the following example:

```
grep -i -n dummy file1
1:***** This file is a dummy file *****
8:***** THIS FILE IS A DUMMY FILE *****
```

If you are just interested in finding the number of lines in which the specified pattern occurs, use the `-c` flag as in the following example:

```
grep -i -c dummy file1
2
```

If you want to get a list of all lines that do not contain the specified pattern, use the `-v` flag as in the following example:

```
grep -i -v dummy file1
which has been created
to run a test for egrep
grep series of commands are used by the following types of people
    programmers
    end users
Believe it or not, grep series of commands are used by pros and novices alike
```

If you are interested in searching for a pattern that you want to search as a word, use the `-w` flag as in the following command:

```
grep -w grep file1
grep series of commands are used by the following types of people
Believe it or not, grep series of commands are used by pros and novices alike
```

Notice that the search did not find the pattern `egrep` because that word contains `e` before the pattern `grep`. The use of the `-w` flag forced `grep` to search for the pattern `grep` delimited by spaces or newline characters.

If you want to search for a pattern that is the only string in a line, use the `-x` command as in the following example:

```
grep -x "    end users" file1
    end users
```

Now, let's examine some of the special features of `grep`. If you want to find out which lines start with a capital letter A through C, use the following command:

```
grep "^[A-C]" file1
Believe it or not, grep series of commands are used by pros and novices alike
```

In this example, the `^` (caret) indicates that the following character is searched for at the beginning of each line. If you are interested in searching for all lines that do *not* start with capital letters A through F, use the following command:

```
grep "^[^A-F]" file1
*** This file is a dummy file *****
which has been created
to run a test for egrep
grep series of commands are used by the following types of people
    programmers
    end users
***** THIS FILE IS A DUMMY FILE *****
```

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous Table of Contents Next

In this example, the ^ (caret) outside the [] (square brackets) searches for the following character at the beginning of the line; the ^ (caret) inside the [] indicates that the match should be made where the character does not match A through F (meaning that all lines that do not have A through F at the beginning of the line are matched).

To search for lines containing only the capital letters C, D, E, or F, use regular expression as follows:

```
grep [C-F] file1
***** THIS FILE IS A DUMMY FILE *****
```

strings

The strings command can be used to search for strings in executable files. A string consists of four or more printable characters terminated by a null or newline.

Following is a list of some of the flags that can be used with the strings command:

Flag	Meaning
-a or -	Search the entire file, not just the data section.
-o	List each string preceded by its offset in the file (in octal).
-Number	Specify a minimum string length other than the default of 4.

Examples

To see the strings that exist in the strings command executable file, execute the following command

in a directory that contains the command:

```
strings strings
|@(#)56
1.17  com/cmd/scan/strings.c, cmdscan, bos320, 9227320b 5/7/92 10:21:20
Standard input
strings.cat
/usr/sbin/strings
Usage: strings [ -a ] [ -o ] [ -# ] [ file ... ]
%7o
%7o
```

If you also want the offset of the strings in the executable file for the `strings` command, use the `-o` flag as follows:

```
strings -o strings
6017 |@(#)56
6027 1.17  com/cmd/scan/strings.c, cmdscan, bos320, 9227320b 5/7/92
10:21:20
6140 Standard input
6164 strings.cat
6200 /usr/sbin/strings
6224 Usage: strings [ -a ] [ -o ] [ -# ] [ file ... ]
6314 %7o
6330 %7o
```

If you want to limit your search to only, say, 15 characters or more in size in the `strings` command executable, execute the following command:

```
strings -o -15 strings
6027 1.17  com/cmd/scan/strings.c, cmdscan, bos320, 9227320b 5/7/92
10:21:20
6200 /usr/sbin/strings
6224 Usage: strings [ -a ] [ -o ] [ -# ] [ file ... ]
```

Printing

- You may have several documents that you want to print, and you may have several printers attached to your computer with which you can print. The following sections discuss some of the commands that direct the printing of specified documents to specified printers and determine the status of the printers. We will also discuss commands to cancel specified printing jobs.

In a UNIX system, you can have multiple printers but only one of the printers can be set up as the default printer to which all the print requests are sent if a printer name is not specified.

cancel

If you have queued up requests to print one or more documents using the `lp` command, and you want to cancel those requests, use the `cancel` command. By using the `cancel` command, you can either cancel a specified job or cancel all queued requests to a specified printer queue. If you are an ordinary user, you can cancel only the jobs that have your user ID.

With the `cancel` command, you can specify either one or more job IDs or a printer name.

Examples

To cancel a print job with ID 734, use the following command:

```
cancel 734
```

To cancel all queued requests that you have queued up in the printer `our_printer`, use the following command:

```
cancel our_printer
```

lp

To print one or more files to a specified printer, use the `lp` command. By default, the `lp` command accepts input from the standard input. If more than one file is specified, the files are printed in the order of their appearance in the command. The files you are printing should exist until they are printed because the `lp` command does not make copies of the file while printing (unless you use the `-c` flag).

Following is a list of flags that can be used with the `lp` command:

Flag	Meaning
<code>-c</code>	Make a copy of the file so that the file can be deleted or modified while the printing is still going on.
<code>-dprintqueue</code>	Specify the <i>printqueue</i> where the print request is to be directed.
<code>-m</code>	Notify the requesting user at successful completion of the print request by mail.
<code>-ncopies</code>	Specify the number of copies to be printed.
<code>-ttitle</code>	Print the specified title on the banner page.

Examples

To print the file `file1`, execute the following command:

```
lp file1
```

In this example, `file1` will be printed on the default line printer. However, if you want to print on the specific printer `our_printer`, use the `-d` flag as in the following example:

```
lp -dour_printer file1
```

If `file1` is big and you want to get notification after the print job is successfully completed, use the `-m` flag as in the following example:

```
lp -m -dmain_printer file1
```

If you want to print multiple copies of `file1` for distribution to your colleagues, use the `-n` flag as in the following example:

```
lp -n15 -dour_printer file1
```

This example prints 15 copies of `file1` on the printer called `our_printer`. If you want to print a title `urgent memo` in the banner page, use the `-t` flag as in the following example:

```
lp -n15 -t"urgent memo" -dour_printer file1
```

This example prints 15 copies of `file1` on `our_printer` with the title `urgent memo` printed on the banner page.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

pr

The `pr` command accepts input from the standard input and generates output on the standard output by default. This command formats the output into pages with the name of the file, date, time, and page numbers. If the line length is longer than the page width, the line is truncated. As the `pr` command formats and paginates the output, you can pipe the output of the `pr` command to a print command such as `lp` to print the output.

Following is a list of some of the flags that can be used with the `pr` command:

Flag	Meaning
-d	Generate the output with double spacing.
-f or -F	Use a form-feed to a new page instead of a sequence of line-feed characters.
-h <i>"heading"</i>	Print <i>heading</i> instead of the filename as header on each page.
-l <i>pagelength</i>	Set the number of lines to be printed on each page to <i>pagelength</i> instead of the default of 66.
-n	Specify the width of the line number to be printed in front of each line. Optionally, you can specify a character to be printed between the line number and the contents of the line.
-o <i>indent</i>	Indent each line by <i>indent</i> columns.
-p	Pause after each page if the output is being directed to standard output. To continue, press the Enter key.
-r	Suppress diagnostic messages.
-t	Suppress printing of page headers and page footers.

<code>-wwidth</code>	Set the width of each page to <i>width</i> instead of the default of 72.
<code>+pagenumber</code>	Specify that the display should start at page number <i>pagenumber</i> instead of 1.

Examples

Assume that we have a file `file1` in the current directory, the contents of which are shown here using the `more` command:

```
more file1
```

```
This is a test file for pr command
```

```
We will use it to show the usage of various flags of pr command
```

The plain-vanilla use of the `pr` command is as follows:

```
pr file1
```

```
Wed Dec  4 00:40:14 1996 file1 Page 1
```

```
This is a test file for pr command
```

```
We will use it to show the usage of various flags of pr command
```

If you want to display the output in double spacing, use the `-d` flag as in the following example:

```
pr -d file1
```

```
Wed Dec  4 00:40:14 1996 file1 Page 1
```

```
This is a test file for pr command
```

```
We will use it to show the usage of various flags of pr command
```

If you want to print a title other than the filename, use the `-h` flag as in the following example:

```
pr -h "TEST FILE FOR pr COMMAND" file1
```

```
Wed Dec  4 00:40:14 1996 TEST FILE FOR pr COMMAND Page 1
```

```
This is a test file for pr command
```

```
We will use it to show the usage of various flags of pr command
```

- If you do not want to print the headers, use the `-t` flag as in the following example:

```
pr -t file1
```

```
This is a test file for pr command
```

```
We will use it to show the usage of various flags of pr command
```

If you want to print the line numbers in front of each line and you want to print a `-` (hyphen) between the line number and the line, use the `-n` flag as in the following example:

```
pr -n-5 file1
```

```
Wed Dec  4 00:40:14 1996 file1 Page 1
```

```
1-This is a test file for pr command
```

```
2-We will use it to show the usage of various flags of pr command
```

lpstat

You can use the `lpstat` command to display the current status of all line printers. If the `lpstat` command is executed without any flags, it displays the status of each printer with the entries queued by the `lp` command.

Following is a list of some of the flags that can be used with the `lpstat` command:

Flag	Meaning
<code>-aqueue</code> or <code>-cqueue</code> or <code>-pqueue</code>	Display status as well as information on jobs in the specified list of <i>queue</i> .
<code>-d</code>	Display the default line printer information.
<code>-oqueue</code> or <code>-ojobnumber</code>	Display the status of the specified <i>queue</i> or to display the status of the specified <i>jobnumber</i> .
<code>-r</code>	Display status and job information for all queues.
<code>-s</code>	Display summary information about all queues.
<code>-t</code>	Display detailed status information for all queues.
<code>-uusername</code>	Display the status of print requests started by the specified <i>username</i> .
<code>-vprintername</code>	Display a list for the specified <i>printername</i> .

Examples

If you want to find out about all the printers in your system, use the `lpstat` command without any flags as in the following example:

```
lpstat | more
Queue   Dev    Status   Job Files           User        PP %    Blks   Cp Rnk
-----
m_prt   lp0     READY
prt_01  bshde  READY
prt_02  lp0     READY
```

If you want to get information about the default line printer, use the `-d` flag as in the following example:

```
lpstat -d
Queue   Dev    Status   Job Files           User        PP %    Blks   Cp Rnk
-----
m_prt   lp0     READY
```

If you are printing `file1` on `printer_01` and want to find out about the status of the printer and the job, use the `-a` flag as in the following example:

```
lpstat -aprinter_01
Queue   Dev    Status   Job Files           User        PP %    Blks   Cp Rnk
-----
systems lpprt  READY
prt_01: prt_01 is ready and printing
prt_01: Rank   Owner      Job  Files           Total Size
prt_01: active testuser    735  file1           156486
bytes
```

Scheduling

UNIX gives you the ability to schedule scripts and commands for execution at some later time. You can specify the exact time when the command should be run. UNIX also provides a way of reporting on the scheduled jobs and removing them if you do not want to execute them.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

at

The `at` command allows you to do the following:

- Schedule a command for execution at a specified time.
- Display a list of scheduled jobs.
- Remove jobs from the scheduled jobs list.

You can schedule jobs by specifying either the absolute time or a time relative to the current time.

Following is a list of some of the flags that can be used with the `at` command:

Flag	Meaning
-l	Display a list of jobs scheduled by you.
-m	Mail a report of successful execution of the job.
-t <i>date</i>	Schedule a job to be executed at the specified date and time.
-r <i>joblist</i>	Remove the jobs specified in the job list from the queue.

You are allowed to execute the `at` command provided that at least one of the following is true:

- The system has an `at.allow` file and your user name appears in the `at.allow` file.
- The system has an `at.deny` file and your name does not appear in the `at.deny` file.

The exact location of the `at.allow` and `at.deny` files depends on the UNIX system you are working with.

The `at` command accepts the time, day, and relative increments in a variety of formats. Some of the formats are as follows:

- 1830 December 4
- 6:30 pm December 4
- 6:30 P December 4
- now + 2 hours
- tomorrow 1830
- 1830 next week
- 1830 Tuesday next week

Examples

If you want to schedule a job called `my_job` at 11:00 p.m. today assuming that the current time is 9:30 p.m., execute any one of the following commands:

```
at 2300 my_job
```

```
at 23:00 my_job
```

```
at 11:00 pm my_job
```

```
at 11:00 P my_job
```

```
at 2300 today my_job
```

If the time currently is 11:30 p.m., the job will be scheduled at 11:00 p.m. the next day.

To schedule `my_job` 6 hours from now, use the following command:

```
at now + 6 hours my_job
```

To schedule `my_job` at 6:30 p.m. next week, use the following command:

```
at 6:30 pm next week my_job
```

In the preceding example, if today is Thursday and the current time is 5:30 p.m., `my_job` will be scheduled for 5:30 p.m. next Thursday. If the current time is 7:30 p.m., `my_job` will be scheduled for 6:30 p.m. next Friday.

To list the jobs scheduled, use the `-l` flag as in the following example:

```
at -l
testuser.850519800.a      Fri Dec 13 18:30:00 1996
testuser.849858400.a      Fri Dec  6 02:46:40 1996
```

To remove a scheduled job, use the `-r` command as in the following example:

```
at -r testuser.850519800.a
at file: testuser.850519800.a deleted
```

atq

The `atq` command can be used to list the jobs scheduled at a later time. The jobs are displayed in the order of the time scheduled with the earlier-scheduled jobs displayed first.

Following is list of flags that can be used with the `atq` command:

Flag	Meaning
-c	Display a list of jobs in order of time at which the <code>at</code> command was executed to schedule the jobs.
-n	Display the number of scheduled jobs.

Examples

To list all the jobs scheduled using the `at` command, use the following command:

```
atq
testuser.849915000.a      Fri Dec  6 18:30:00 1996
testuser.850519800.a      Fri Dec 13 18:30:00 1996
```

If you want to list all the jobs scheduled according to the time the corresponding `at` command was run rather than according to when the scheduled jobs are supposed to run, use the `-c` flag as in the following example:

```
atq -c
testuser.850519800.a      Fri Dec 13 18:30:00 1996
testuser.849915000.a      Fri Dec  6 18:30:00 1996
```

If you want to find out the number of jobs currently scheduled, use the `-n` flag as in the following example:

```
atq -n
2 files in the queue
```

crontab

UNIX systems have a daemon running all the time that can run jobs at regularly scheduled intervals. You can specify the jobs that the `crontab` command will execute in a file, and the `cron` daemon will check it when the `cron` daemon is initialized or when additions or modifications are made to the file.

The entries you can make in the `crontab` file consist of the following fields (separated by spaces or tab characters):

- minute
- hour
- day (of the month)
- year
- day of the week
- command

Each of these fields can have more than one discrete value (separated by commas), a range of values, or an `*` (an asterisk, meaning that all values are to be matched).

Following is a list of flags that can be used with the `crontab` command:

Flag	Meaning
-l	List your <code>crontab</code> file.
-e	Edit or create the <code>crontab</code> file.
-r	Remove your <code>crontab</code> file.
-v	List the status of the <code>crontab</code> jobs.

Examples

If you want to display the string `Time to go for lunch` at 12:30 p.m. every day, set up the following:

```
30 12 * * * echo "Time to go for lunch"
```

If you want to execute `my_job` on Friday at 4:00 p.m. every week, set up the following:

```
0 16 * * 5 my_job
```

Storage

The following sections discuss a number of commands that can be used for file management. There are commands to back up files to different media, to restore files from different media, to compress files to save disk space, to uncompress files to restore them, and so on.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

compress

You can use the `compress` command to reduce the size of a file. A file created by the `compress` command has a `.Z` appended to its name. The compressed file retains the permission and time attributes of the original file.

Following is a list of flags that can be used with the `compress` command:

Flag	Meaning
-d	Force the <code>compress</code> command to act as an <code>uncompress</code> command.
-c	Compress the file to standard output (which can be redirected to another file) so that the original file is intact.
-f or -F	Compress the file and overwrite the compressed file if it already exists.
-v	Display the compression percentage.
-V	Display the current version and compile options.

Examples

To compress `file1`, execute the following command:

```
compress file1
```

If you want the compression statistics, use the `-v` flag as in the following example:

```
compress -v file1
```

```
file1: Compression: 50.85%  -- replaced with file1.Z
```

cpio

You can use the `cpio` command to copy files to archival medium from disk or to restore files from archival medium to disk. There are three major forms of the `cpio` command:

- `cpio -o` to read standard input for path names and copy them to standard output.
- `cpio -i` to read from standard input archival files and create disk files.
- `cpio -p` to read standard input for the path name and copy to the specified directory.

Following is a list of some of the flags that can be used with the `cpio` command:

Flag	Meaning
a	Modify the access time of the copied files to that of the current file.
B	Indicate that <code>cpio</code> should do block I/O.
d	Create a directory if the specified directory does not exist.
f	Copy files that do not match the specified pattern.
r	Copy files interactively with the option of modifying the filename.
t	Create a list of files without actually copying a file.
u	Overwrite a file if it already exists.
v	List the filenames being copied.

Examples

If you have a list of files that you want to copy to a diskette, execute the following command:

```
-ls *.txt | cpio -ov > /dev/rfd0  
file1.txt  
file2.txt  
55 blocks
```

This example copies all files that have an extension of `.txt` and displays the filenames being copied.

If you want to list the files on the diskette, use the `t` and `v` flags as in the following example:

```
cpio -itv < /dev/rfd0
100644 testuser      13771 Dec  07 00:13:38 1996 file1.txt
100644 testuser      13947 Dec  07 00:13:30 1996 file2.txt
55 blocks
```

If you want to copy the files from the diskette and rename them while copying, use the `r` flag as in the following command:

```
cpio -ir "*.txt" < y
Rename <file1.txt>
file3.txt
Rename <file2.txt>
file4.txt
55 blocks
```

In the preceding example, `file1.txt` is renamed to `file3.txt` and `file2.txt` is renamed to `file4.txt`.

If you want to copy all files from the current directory as well as all the files in its subdirectories, use the `-p` flag. Additionally, you can use the `d` flag so that all the needed directories are created. You can execute the commands as follows:

```
find . -print | cpio -pd /u/testuser/cpiodir
```

dd

The `dd` command can be used to read data from the standard input and copy it to the standard output after converting the data according to specified conversion parameters. Along with the data conversion, the physical attributes, such as block size, can also be modified by specifying appropriate parameters.

Following is a list of flags that can be used with the `dd` command:

Flag	Meaning
<code>bs=blocksize</code>	Specify the block size of the input and output file. This flag overrides the <code>ibs</code> and <code>obs</code> flags.
<code>if=filename</code>	Specify the input <i>filename</i> to be copied.
<code>ibs=blocksize</code>	Specify the block size of the input file.

<code>fkskip=numberofeof</code>	Specify the number of end-of-file markers to be skipped in the input file before starting the copy.
<code>files=numberoffiles</code>	Specify the number of files to be copied (such as from a tape containing multiple files).
<code>count=numberofblocks</code>	Specify the number of blocks to copy from the input file.
<code>skip=nummberofblocks</code>	Skip the specified number of blocks in the input file before starting the copy.
<code>of=filename</code>	Specify the output <i>filename</i> to be created.
<code>obs=blocksize</code>	Specify the block size of the output file.
<code>seek=recordnumber</code>	Specify the record number in the output file at which to start copying the input file.
<code>conv=conversionparameter</code>	Specify the type of conversion to be used. Some of the values of this parameter can be ASCII, EBCDIC, block, unblock, lcase, and ucase.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Examples

If you have a file from a system that stores data in EBCDIC format and you want to convert the data to ASCII, use the following command:

```
dd if=file1 of=file2 conv=ascii
```

This command reads `file1` and converts each character of this file to ASCII and copies the characters to `file2`.

If you want to copy `file1` on disk to a tape with a block size of 1024, use the following command:

```
dd if=file1 of=/dev/rmt0 bs=1024 conv=sync
```

If you want to copy the third file on a tape to a file called `file1`, use the following command:

```
dd if=/dev/rmt0 fskip=2 of=file1
```

If you want to print a memo in capital letters, use the following command to convert `file1` to `file2` and then print `file2`:

```
dd if=file1 of=file2 conv=ucase
```

```
lp -dmain_printer file2
```

pack

If you want to save disk space, use the `pack` command to compress a file in a way similar to the `compress` command. The `pack` command compresses a file and generates a new file with `.z` appended to the filename. The original file is removed. The amount of space saved depends on the contents of the file. Usually, you can get from 30 to 50 percent compression for text files. By default, the `pack` command does not compress if it cannot reduce the size of the file.

Following is a list of flags that can be used with the `pack` command:

Flag	Meaning
-	Display statistics about compression.
-f	Force packing.

Examples

If you have a file called `file1` that you want to compress, use the following command:

```
pack file1  
pack: file1: 41.7% Compression
```

If you want more information about the compression, use the `-` (hyphen) flag as in the following example:

```
pack - file1  
pack: file1: 41.7% Compression  
      from 28160 to 16404 bytes  
      Huffman tree has 15 levels below root  
      102 distinct bytes in input  
      dictionary overhead = 124 bytes  
      effective entropy   = 4.66 bits/byte  
      asymptotic entropy  = 4.62 bits/byte
```

In some cases, `pack` may not compress the file and will give you an error as it does in the following example:

```
pack file1  
pack: file1: no saving  
      - file unchanged
```

In such a case, you can force compression by using the `-f` flag as in the following example:

```
pack -f file1
```

```
pack: file1: 40.8% Compression
```

pcat

The `pcat` command can be used to uncompress a file to the standard output. This command does not have any flags.

Examples

If you want to uncompress a file called `file1.z` that you have earlier created by using the `pack` command on `file1`, use one of the following commands:

```
pcat file1
```

```
pcat file1.z
```

tar

The `tar` command is used to copy files from disk to an archival medium (usually tape) or vice versa. The `tar` command does not provide any recovery from tape errors.

Following is a list of some of the flags that can be used with the `tar` command:

Flag	Meaning
<code>-c</code>	Create a new archive and write the file details at the beginning of the archive.
<code>-t</code>	Generate a list of files in the archive.
<code>-x</code>	Obtain one or more files from an archive. If you specify a directory name, all the files in the directory are extracted. If no file or directory is specified, all the files in the specified archive are extracted. If one or more files extracted do not exist, they are created with the original user ID if you have <code>root</code> authority; otherwise, they are created with your user ID.
<code>-m</code>	Use the time of extraction from the archive as the modification time of the extracted file.
<code>-p</code>	Restore the files with their original permission, ignoring the current setting of the <code>umask</code> .
<code>-f archive</code>	Use the specified <i>archive</i> as the archive name instead of the system default.
<code>-v</code>	Display the name of each file as it is processed.

Examples

If you want to extract all the files in the `/u/testuser` directory from the archive file on the `/dev/rmt1` tape device, use the following command:

```
tar --xvf /dev/rmt1 /u/testuser
```

If you want to archive a file to an archive on the default tape drive, use the following command:

```
tar -c file1
```

uncompress

The `uncompress` command can be used to uncompress a file that has earlier been compressed using the `compress` command. By default, the `uncompress` command uncompresses a file in place; that is, the compressed file is deleted and the uncompressed file—without the `.Z` suffix—is created in its place. The uncompressed file retains the permission and modification time attributes of the compressed file, but the user and the group of the file are changed to that of the user who is uncompressing the file.

Following is a list of some of the flags that can be used with the `uncompress` command:

Flag	Meaning
<code>-f</code> or <code>-F</code>	Force the uncompress even though a file by the name of the uncompressed file may already exist.
<code>-c</code>	Uncompress the specified by file to the standard output, retaining the original compressed file.
<code>-v</code>	Display a message with the uncompressed filename.
<code>-q</code>	Suppress display of compression statistics from the uncompress command.

Examples

If you want to uncompress `file1.Z`, use either of the following commands:

```
uncompress file1
```

```
uncompress file1.Z
```

If you want to uncompress `file1.Z` to standard output and retain the original compressed file, use the `-c` flag as in the following example:

```
uncompress -c file1
```

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

unpack

The `unpack` command can be used to uncompress files that have the `.z` extension and that have been compressed using the `pack` command. The uncompressed file is created at the same place as the compressed file, and the compressed file is removed. The uncompressed file retains the attributes (such as the user, group, permission, access, and modification time) of the compressed file. The `unpack` command does not uncompress the file if a file by the name of the uncompressed file already exists.

If you want to uncompress `file1.z`, use either of the following two commands:

```
unpack file1
```

```
unpack file1.z
```

zcat

The `zcat` command can be used to uncompress a file that has been compressed using the `compress` command to the standard output, retaining the compressed file. You can redirect the standard output to another file to get an expanded version of the compressed file. This command works the same way as the `uncompress` command with the `-c` flag.

Examples

If you want to create a copy of the uncompressed version of a file without destroying the compressed file, use the following command:

```
zcat file1.Z > file2
```

This example creates `file2`, which is an uncompressed version of `file1.Z`, and at the same time retains `file1.Z`.

Status Commands

The following sections discuss several commands that display the status of various parts of the system. These commands can be used to monitor the system status at any point in time.

date

You can use the `date` command to display the current date and time in a specified format. If you are the `root` user, use the `date` command to set the system date.

To display the date and time, you must specify a `+` (plus) sign followed by the desired format. The format can be as follows:

Format	Appearance of Date
%A	Display the date complete with weekday name.
%b or %h	Display a short month name.
%B	Display the complete month name.
%c	Display default date and time representation.
%d	Display the day of the month as a number from 1 through 31.
%D	Display the date in mm/dd/yy format.
%H	Display the hour as a number from 00 through 23.
%I	Display the hour as a number from 00 through 12.
%j	Display the day of the year as a number from 1 through 366.
%m	Display the month as a number from 1 through 12.
%M	Display minutes as a number from 0 through 59.
%p	Display AM or PM appropriately.
%r	Display 12-hour clock time (01-12) using the AM/PM notation.
%S	Display seconds as a number from 0 through 59.
%T	Display the time in hh:mm:ss format for a 24-hour clock.

-	%U	Display the week number of the year as a number from 1 through 53 (counting Sunday as first day of the week).
	%w	Display the day of the week as a number from 0 through 6 (with Sunday counted as 0).
	%W	Display the week number of the year as a number from 1 through 53 (counting Monday as first day of the week).
	%x	Display the default date format.
	%X	Display the time format.
	%y	Display the last two digits of the year from 00 through 99.
	%Y	Display the year with the century as a decimal number.
	%Z	Display the time-zone name, if it is available.

Examples

If you want to display the date without formatting, use `date` without any formatting descriptor as follows:

date

```
Sat Dec 7 11:50:59 EST 1996
```

If you want to display only the date in `mm/dd/yy` format, use the following command:

date +%m/%d/%y

```
12/07/96
```

If you want to format the date in `yy/mm/dd` format and the time in `hh:mm:ss` format, use the following command:

date "+%y/%m/%d %H:%M:%S"

```
96/12/07 11:57:27
```

Following is another way of formatting the date:

date +%A", "%B" "%d", "%Y

```
Sunday, December 15, 1996
```

If you want the Julian date, use the following command:

date +%j

```
350
```

If you want to find the week number for the current week, you have two options: the `W` and `U` flags, as shown in the following commands:

date +%W

49

date +%U

50

env

The `env` command can be used to display the current environment or to change one or more of the environment variables and run a specified command. The changes are effective only during the execution of the command.

You can use the `-i` flag with the `env` command. Use `-i` to indicate that only the variables set up as part of the `env` command are used for the specified command; all the current variable setups are ignored.

Examples

If you want to display the current environment, use the following command:

```
env
```

Assume that we have a script called `my_job` that displays the current setting of the variable called `LANG`. If you execute the script `my_job` as part of the `env` command without modifying the `LANG` variable, you get the following result:

```
env PATH=/u/testuser/jobs:$PATH my_job
LANG = C
```

If you modify the `LANG` variable as part of the `env` command, you get the following result:

```
env LANG=C++ PATH=/u/testuser/jobs:$PATH my_job
LANG = C++
```

If you use the `-i` flag and do not modify `LANG` as part of the `env` command, that variable is not available to `my_job`; you get the following result:

```
env -i PATH=/u/testuser/jobs:$PATH my_job
LANG =
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

Click Here!

ITKnowledge

- home
- account info
- subscribe
- login
- search
- My ITKnowledge
- FAQ/help
- site map
- contact us

Brief

Full

Advanced Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

- Previous
- Table of Contents
- Next

iostat

The `iostat` command can be used to obtain statistics about the CPU, disks, and TTY of a system. The first time you run `iostat` after you boot the system, `iostat` provides the statistics since that boot. Subsequent executions of this command provide statistics since the last execution of the `iostat` command.

The `iostat` command displays the following details:

- TTY and CPU header
- TTY and CPU statistics detail
- Physical volume header
- One line for each physical volume

Following is a list of data items displayed for TTY and CPU statistics:

- `tin` displays the number of characters read by the system for all TTYs.
- `tout` displays the number of characters written by the system for all TTYs.
- `%user` displays the utilization percentage of the CPU at the application level.
- `%system` displays the utilization percentage of the CPU at the system level.
- `%idle` displays the utilization percentage of the CPU while it was idling (this represents the unused utilization of the CPU).
- `%iowait` displays the idling percentage of the CPU while waiting for the I/O request.

Following is a list of data items displayed as part of the physical volume utilization:

- `%tm_act` displays the active utilization percentage of the physical volume.
- `Kbps` displays the number of kilobytes transferred per second to or from the physical volume.
- `tps` displays the number of physical I/O requests to the physical volume.
- `msps` displays average number of milliseconds required for each seek of the physical volume.
- `Kb_read` displays the number of kilobytes read from the physical volume.
- `Kb_wrtn` displays the number of kilobytes written to the physical volume.

Following is a list of flags that can be used with the `iostat` command:

Flag	Meaning
-d	Display only the physical volume utilization report. This flag cannot be used with -t flag.
-t	Display only the TTY and CPU utilization report. This flag cannot be used with the -d flag.

Examples

If you want to display only the TTY and CPU utilization, use the `-t` flag as in the following example:

```
iostat -t
```

```
tty:      tin      tout      cpu:   % user   % sys   % idle   % iowait
          0.5      78.7                32.6    25.2    35.7     6.4
```

If you want to display only the utilization of the physical volume of `disk1`, use the `-d` flag as in the following example:

```
iostat -d disk1
```

```
Disks:      % tm_act   Kbps      tps      Kb_read  Kb_wrtn
disk1        6.7       4.3       5.0      2339721  4048758
```

sar

You can use the `sar` command to get a report about system information. The `sar` command allows you to save the information report. By default, the `sar` command generates the CPU utilization reports, but you can use various flags to collect information about other system activities.

Following is a list of some of the flags that can be used with the `sar` command:

Flag	Meaning
-A	Report data on all system activities.
-a	Report data on the use of the file system access routine.
-b	Report buffer activities.
-c	Report system calls such as forks, execs, and so on.
-e optionally followed by time in hh:mm:ss format	Specify the time at which data accumulation should be terminated.
-f <i>file</i>	Extract data from the specified <i>file</i> .
-i <i>seconds</i>	Extract data from the file for the closest time closest in <i>seconds</i> .
-k	Report on kernel activity.
-m	Report on semaphore and message activities.
-o <i>file</i>	Save the activity data in the specified <i>file</i> .
-r	Report on paging statistics.
-s optionally followed by time in hh:mm:ss format	Specify the time at which to start the data accumulation.

-v	Report on process and i-node activity.
-Y	Report on TTY activity.

uname

The `uname` command displays details about the operating system and the computer system on the standard output. You can use certain flags to set the system name.

Following is a list of some of the flags that can be used with the `uname` command:

Flag	Meaning
-m	Display the machine ID.
-r	Display the release number of the operating system.
-s	Display the system name.
-v	Display the operating system version.
-S <i>name</i>	Modify the system name.
-a	Display the machine ID, the release number of the operating system, and the system name.

Examples

If you want to display details about the hardware and operating system, you can use the `-a` flag as in the following example:

```
uname -a
AIX main_system 2 3 000010000526
```

In this example, the information displayed is as follows:

Operating system name	AIX
Machine name	main_system
Operating system release number	2
Operating system version number	3
Machine ID	000010000526

uptime

The `uptime` command displays the following information:

- The current time
- The length of time the system has been up
- The number of users currently logged on
- The number of jobs executing in the system

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.



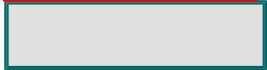
Brief

Full

Advanced

Search

Search Tips



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

vmstat

The `vmstat` command can be used to get information about the processes, virtual memory, physical volumes, and CPU activity. The information includes the CPU utilization, virtual memory, and physical volume. This information can be used to monitor the load on the system.

The first invocation of `vmstat` displays the statistics since the system startup; subsequent invocations display statistics since the last invocation. You can specify a count and an interval parameter to control the number of reports generated and an interval between the reports.

The details displayed by `vmstat` are as follows:

- Processes
- Virtual memory
- Page
- Faults
- CPU

The details displayed for the *processes* are as follows:

- `r` displays the number of processes placed in the queue and ready to execute.
- `b` displays the number of processes placed in the queue and waiting for execution.

The details displayed for the *memory* are as follows:

- `avm` displays the number of pages being consumed (the pages are from the page space).
- `fre` displays the number of pages in the free list.

The details displayed for *page* are as follows:

- **re** displays the number of page reclaims per second observed in the specified interval.
- **pi** displays number of pages brought in from the page space in the specified interval.
- **po** displays the number of pages swapped out to page space in the specified interval.
- **fr** displays the number of pages freed in the specified interval.
- **sr** displays the number of page examined to determine whether they can be freed in the specified interval.
- **cy** displays the number of clock revolutions per second.

The details displayed for *faults* are as follows:

- **in** displays the number of interrupts per second in the specified interval.
- **sy** displays the number of system calls per second in the specified interval.
- **cs** displays the number of context switches per second in the specified interval.

The details displayed for *CPU* are as follows:

- **us** displays the percentage utilization of CPU at the application level during the specified interval.
- **sy** displays the percentage utilization of CPU at the system level during the specified interval.
- **id** displays the percentage utilization of CPU idling during the specified interval without any I/O wait.
- **wa** displays the percentage utilization of CPU idling during the specified interval caused by disk I/O requests.

You can specify up to four physical volume names to get the number of transfers that occurred in those disks in the specified interval.

You can use the **-s** flag with the **vmstat** command. Use **-s** to display the statistics since the system initialization.

Examples

- If you want to display the statistics five times, at intervals of five seconds, execute the following command:

```
vmstat 5 5
```

procs		memory		page						faults			cpu			
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa
1	0	44036	120	0	0	0	125	275	0	366	1458	391	33	25	36	6
1	0	44036	120	0	0	0	542	938	0	446	4932	246	65	24	0	12
1	0	44036	121	0	0	0	624	1116	0	453	5848	259	64	25	0	11
1	0	44037	124	0	0	0	512	1010	0	434	4812	266	59	25	0	16
0	0	44037	121	0	0	0	564	1109	0	426	4838	265	64	24	0	11

Text Processing

UNIX provides several commands to process the contents of a text file.

cut

You can use the **cut** command to extract data from each line of a text file. This command can be used for a file that contains data records so that each line consists of one or more fields separated

by tab characters.

Following is a list of some of the flags that can be used with the `cut` command:

Flag	Meaning
<code>-ccharacterlist</code>	Specify a list of characters to be cut from each line.
<code>-ffieldlist</code>	Specify a list of fields to be cut from each line. You can additionally specify the flag <code>-dcharacter</code> to override the character to be interpreted as the field delimiter. You can also specify the flag <code>-s</code> to suppress lines that do not have the specified delimiter character.

Examples

Assume that we have a file called `file1` whose contents are as follows:

```
more file1
Misty      Ghosh
Saptarsi   Guha
Sanjiv     Guha
```

In this file, the fields are separated by tab characters.

If you want to extract the first field, use the following command:

```
cut -f1 file1
Misty
Saptarsi
Sanjiv
```

If you want to cut the characters 2 to 6, use the following command:

```
cut -c2-5 file1
isty
apta
anji
```

If you want to cut all characters in the first field up to the first `s` character, use the following command:

```
cut -d"s" -f1 file1
Mi
Saptar
Sanjiv Guha
```

You will notice that the third line is cut completely. To suppress lines that do not contain the `s` character, use the `-s` flag as in the following example:

```
cut -d"s" -s -f1 file1
Mi
Saptar
```

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

ex

The `ex` command invokes the `ex` editor to edit one or more files.

Following is a list of some of the flags that can be used with the `ex` command:

Flag	Meaning
<code>-c subcommand</code>	Perform the specified <i>subcommand</i> on the specified file before invoking the <code>ex</code> command.
<code>-R</code>	Disallow updating the file.
<code>-wsize</code>	Set the window to the number of lines equal to <i>size</i> .
<code>-v</code>	Invoke the <code>vi</code> editor.
<code>-r file</code>	Do the recovery on the specified <i>file</i> .

Once you are in the `ex` editor, you can use the following subcommands to move around in the file and edit the file:

Subcommand	Action
<code>z</code>	Invoke full-screen mode.
<code>u</code>	Undo the last change made.
<code>n</code>	Move to the next file if you have invoked the <code>ex</code> editor with multiple files.
<code>/pattern/</code>	Find a <i>pattern</i> in the file.
<code>d</code>	Delete one or more lines.
<code>a</code>	Append.

The `ex` operates in the following modes:

- Command mode*: When the `ex` editor starts, it is in command mode, which displays a `:` (colon)

prompt at which you can enter a subcommand.

- *Text input mode:* In this mode, you can add or change text in the file. You can enter text using the `a`, `i`, or `c` subcommand. The use of these subcommands allows you to enter text in the buffer. You can return to command mode by entering a `.` (a single period) as the first character of the text buffer.

fmt

The `fmt` command can be used to format files to a 72-character line by default. The `fmt` command preserves the blank lines in the input file as well as the spacing between words. You can modify the line length using the `-Width` flag.

Examples

Assume that we have `file1` whose contents are shown here:

```
more file1
```

```
This is a test file for fmt
```

```
The fmt command      formats a file
for mail command
```

Notice that we have a blank line in the file and that the spacing between `command` and `formats` on the third line is more than one character. Let's format `file1` using the `fmt` command to create `file2` as in the following command:

```
fmt file1 > file2
```

Now let's see the contents of `file2` using the `more` command as follows:

```
more file2
```

```
This is a test file for fmt
```

```
- The fmt command      formats a file for mail command
```

In this example, the blank line and interword spacing have been preserved.

fold

The `fold` command can be used to generate multiple lines from a single line by splitting the line at the specified position. By default, the line length is 80 bytes. A newline character is inserted at the end of the line.

Following is list of flags that can be used with the `fold` command:

Flag	Meaning
-b	Specify the position in bytes.
-s	Split a line after the last space at a position that is less than or equal to the specified width.
-w <i>width</i>	Specify the line width.

Examples

Assume that we have `file1` containing one line of 129 characters which is shown here:

```
more file1
```

```
The fold command can be used on files which have line lengths more than 80
bytes
, it breaks the line into multiple 80 byte lines
```

If you want to split the line at byte position 40, use the following command:

```
fold -w 40 file1 > file2; more file2
```

The `fold` command can be used on files which have line lengths more than 80 bytes, it breaks the line into multiple 80 byte lines

In this example, the split happens in the middle of words. If you do not want to split words, use the `-s` flag as in the following example:

```
fold -w 40 -s file1 > file2; more file2
```

The `fold` command can be used on files which have line lengths more than 80 bytes, it breaks the line into multiple 80 byte lines

join

The `join` command can be used to merge two files (one can be standard input) to create a third file (which can be standard output). Each line in the file is merged on the basis of a field that has the same value in both input files to create one line in the output file. The fields in each file are separated by either a space or the tab character.

Following is a list of flags that can be used with the `join` command:

Flag	Meaning
<code>-1 field</code> or <code>-j1 field</code>	Specify that the join should be made on the basis of the <i>field</i> in the first file.
<code>-2 field</code> or <code>-j2 field</code>	Specify that the join should be made on the basis of the <i>field</i> in the second file.
<code>-e string</code>	Specify that blank fields in the output file be replaced by the specified <i>string</i> .
<code>-o fileid.fieldnumber</code>	Specify that the output should consist of the specified fields. You can specify multiple fields by separating them with commas.
<code>-t character</code>	Modify the field separator <i>character</i> from the default value of the space.
<code>-a fileid</code>	Generate an output line for each line in the file specified by the <i>fileid</i> parameter for lines that cannot be matched to the lines in the other file using the join field. The output lines are produced in addition to the default output.
<code>-v fileid</code>	Generate an output line for each line in the file specified by the <i>fileid</i> parameter for lines that cannot be matched to the lines in the other file using the join field. The default output is not produced.

Use of this site is subject to certain [Terms & Conditions](#). [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Examples

Assume that we have two files, file1 and file2, whose contents are shown as follows:

more file1
computer1 16MB 1.2GB 17inch CDROM
computer2 8MB 840MB 14inch
computer3 12MB 1.6GB 17inch
computer4 4MB 270MB 14inch

more file2
computer1 1stfloor office5
computer3 2ndfloor office9A
computer4 1stfloor office2
computer5 3rdfloor office1

If you want to join the two files and display only the matching lines, execute the following command:

join file1 file2
computer1 16MB 1.2GB 17inch CDROM 1stfloor office5
computer3 12MB 1.6GB 17inch 2ndfloor office9A
computer4 4MB 270MB 14inch CDROM 1stfloor office2

If you want to join the two files and display the matching lines as well as the nonmatching lines from the specified file, use the -a flag as in the following example:

join -a1 file1 file2
computer1 16MB 1.2GB 17inch CDROM 1stfloor office5
computer2 8MB 840MB 14inch
computer3 12MB 1.6GB 17inch 2ndfloor office9A
computer4 4MB 270MB 14inch CDROM 1stfloor office2

This example displays the line with `computer2` from `file1` because it does not have a matching line in `file2`. If you want to display only the lines that do not match lines from the specified file, use the `-v` flag as in the following example:

```
join -v2 file1 file2
computer5 3rdfloor office1
```

This example displays the line with `computer5` from `file2` because it does not have a matching line in `file1`.

If you want to display only certain fields from the input files to the output file, use the `-o` flag as in the following example:

```
join -o 1.1 2.2 2.3 1.5 file1 file2
computer1 1stfloor office5 CDROM
computer3 2ndfloor office9A
computer4 1stfloor office2 CDROM
```

In this example, the line with `computer3` is displayed with one field short because that field is not present in the input file. You can insert a fixed legend in the empty field in the output by using the `-e` flag as in the following example:

```
join -o 1.1 2.2 2.3 1.5 -e"NO CDROM" file1 file2
computer1 1stfloor office5 CDROM
computer3 2ndfloor office9A NO CDROM
computer4 1stfloor office2 NO CDROM
```

paste

The `paste` command can be used to paste lines from one or more files (one of them can be the standard input) to the standard output, which can be redirected to a file. The `paste` command concatenates the line from each input file to the output file, separating them by default with the tab character.

Following is a list of flags that can be used with the `paste` command:

Flag	Meaning
<code>-dlist</code>	Specify characters that will be used to separate corresponding lines from the input files in the output file. You can specify multiple characters if you have multiple input files.
<code>-s</code>	Merge subsequent lines from the input file for each input file, one at a time, separated by the specified delimiter character.

-

Examples

Assume that we have two files, `file1` and `file2`, whose contents are shown here:

```
more file1
computer1 16MB 1.2GB 17inch CDROM
computer2 8MB 840MB 14inch
computer3 12MB 1.6GB 17inch
computer4 4MB 270MB 14inch
```

more file2

```
computer1 1stfloor office5
computer3 2ndfloor office9A
computer4 1stfloor office2
computer5 3rdfloor officel
```

If you want to merge file1 and file2, use the following command:

paste file1 file2

```
computer1 16MB 1.2GB 17inch CDROM      computer1 1stfloor office5
computer2 8MB 840MB 14inch              computer3 2ndfloor office9A
computer3 12MB 1.6GB 17inch             computer4 1stfloor office2
computer4 4MB 270MB 14inch              computer5 3rdfloor officel
```

The lines from file1 and file2 are separated by tab characters.

If you want to modify the default separator from the tab character to, for example, the / (slash), use the -d flag as in the following example:

paste -d"/" file1 file2

```
computer1 16MB 1.2GB 17inch CDROM/computer1 1stfloor office5
computer2 8MB 840MB 14inch/computer3 2ndfloor office9A
computer3 12MB 1.6GB 17inch/computer4 1stfloor office2
computer4 4MB 270MB 14inch /computer5 3rdfloor officel
```

If you want to merge the lines from within each input file, use the -s flag as in the following example:

paste -d"/" -s file1 file2

```
computer1 16MB 1.2GB 17inch CDROM/computer2 8MB 840MB 14inch/computer3
[special character]12MB 1.6G
B 17inch/computer4 4MB 270MB 14inch
computer1 1stfloor office5/computer3 2ndfloor office9A/computer4
[special character]1stfloor office
2/computer5 3rdfloor officel
```

sort

The sort command is used to sort one or more files in the specified order by the specified key. It can also be used to merge files that have already been sorted. When more than one file is used, the sort command concatenates these files before sorting according to the specifications.

Following is a list of some of the flags that can be used with the sort command:

Flag	Meaning
-kkey	Specify the <i>key</i> on which to sort. The specification for the key includes the starting field and column position and the end field and column position.
-A	Specify that sorting be done according to ASCII sequence.
-c	Check whether the specified files are sorted according to the specified key and order.
-d	Sort according to dictionary order.
-f	Change all letters to uppercase before the sort.
-i	Ignore nondisplayable characters for comparison.
-m	Merge presorted input files.

-n	Sort according to numeric value.
-o <i>file</i>	Redirect the output to the specified <i>file</i> instead of to the standard output.
-r	Sort the output in the reverse order of the specified order.
-u	Create only one line in the output for lines that sort identically.

Examples

Assume that we have a file called `file1` whose contents are shown here:

```
more file1
disk drive
memory
video memory
monitor
[tape drive]
CD-ROM
3.5inch diskette
modem
monitor
sound blaster
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc. All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

If you want to sort `file1`, use the following command:

sort file1

3.5inch diskette

CD-ROM

[tape drive]

disk drive

memory

modem

monitor

monitor

sound blaster

video memory

If you want to sort in the reverse order, use the `-r` flag as in the following example:

sort -r file1

video memory

sound blaster

monitor

monitor

modem

memory

disk drive

[tape drive]

CD-ROM

3.5inch diskette

If you want to sort according to alphabetic order, use the `-d` flag as in the following example:

sort -d file1

```
3.5inch diskette
CD-ROM
disk drive
memory
modem
monitor
monitor
sound blaster
[tape drive]
video memory
```

In this example, the line `[tape drive]` is sorted as `tape drive` because the `[` and `]` are ignored by the `-d` flag.

If you want only one line to be retained in case more than one line is sorted equally, use the `-u` flag as in the following example:

```
sort -u file1
3.5inch diskette
CD-ROM
[tape drive]
disk drive
memory
modem
monitor
monitor
sound blaster
video memory
```

In this example, the line `monitor` appears only once, even though there are two such entries in the file.

If you want to sort `file1` according to the uppercase-letter sort order, use the `-f` flag as in the following example:

```
sort -f file1
3.5inch diskette
CD-ROM
disk drive
memory
modem
monitor
monitor
sound blaster
video memory
[tape drive]
```

tr

You can use the `tr` command to translate or delete characters from standard input to generate standard output. Following are the details of the main functions of the `tr` command:

- Translate specified characters in the input to new specified characters in the output.
- Delete specified characters in the input from the input to generate the output.
- Delete all but the first occurrence of the specified characters.

Following is a list of some of the flags that can be used with the `tr` command:

Flag	Meaning
-c	Translate all but the specified characters using the specified new character.
-d	Delete the specified characters.
-s	Delete all but the first occurrence of the specified characters.

You can specify the input and output sequence of characters in certain special ways as follows:

- [*character1-character2*] to specify a range of characters including *character1* and *character2*.
- [*character*number*] to specify *number* occurrences of *character*.
- [*character**] to specify the use of as many occurrences as are needed of *character* so that the input string of characters to be translated matches the output characters.
- [*:characterlist:*] to specify a list of characters as the input or output string. The *characterlist* can be upper, lower, alpha, space, digit, and so on.

Examples

Assume that we have `file1` whose contents are shown as follows:

```
more file1
"this      is a test file
for tr command"
"it has 4 lines
but should be 1 line"
```

If you want to change the double quotes to spaces, use the following command:

```
tr '\"' ' ' < file1
this      is a test file
for tr command
it has 4 lines
but should be 1 line
```

If you want to change all lowercase letters to uppercase letter, use the following command:

```
tr [:lower:] [:upper:] < file1
"THIS      IS A TEST FILE
FOR TR COMMAND"
"IT HAS 4 LINES
BUT SHOULD BE 1 LINE"
```

If you want to delete all the newline characters from this file, use the `-d` flag as in the following example:

```
tr -d '\n' < file1
"this      is a test file for tr command"
"it has 4 lines but should be 1 line"
```

If you want to delete all but the first occurrence of a space and replace the space with a - (hyphen), use the `-s` flag as in the following example:


```
tr -s ' ' '-' < file1
"this-is-a-test-file-
for-tr-command"
"it-has-4-lines-
but-should-be-1-line"
```

uniq

The `uniq` command can be used to eliminate duplicate adjacent lines from a file or from standard input to generate standard output or another file. This is the default operation of the `uniq` command. However, it is possible to compare only part of a line for comparison by using certain flags.

Following is a list of some of the flags that can be used with the `uniq` command:

Flag	Meaning
-c	Precede each line with a number while displaying the output (the number specifies the number of occurrences of the line in the input file).
-d	Display only the lines that occur multiple times adjacent to each other in the input file.
-u	Display only the lines that appear only once in the input file.
-s <i>numberofcharacters</i> or <i>+numberofcharacters</i>	Specify the number of characters from the start of a line that will be ignored while comparing adjacent lines.
-numberoffields or -f <i>numberoffields</i>	Specify the number of fields from the start of a line that will be ignored while comparing adjacent lines.

Examples

Assume that we have `file1` whose contents are displayed as shown here:

```
more file1
This is line 1
This is line 1
This is line 2
This is line 3
THIS IS line 3
This is line 4
```

If you want to find the unique lines in `file1`, use the following command:

```
uniq file1
This is line 1
This is line 2
This is line 3
THIS IS line 3
This is line 4
```

In this example, the first line has been dropped because it is identical to the second line. If you want to display only the duplicate lines, use the `-d` flag as in the following example:

```
uniq -d file1  
This is line 1
```

If you want to display the lines that appear only once in `file1`, use the `-u` flag as in the following example:

```
uniq -u file1  
This is line 2  
This is line 3  
THIS IS line 3  
This is line 4
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

In this example, the first two lines are not displayed because they are identical. If you want to skip the first two fields while comparing adjacent lines, use the `-f` flag as in the following example:

```
uniq -f 2 file1
This is line 1
This is line 2
This is line 3
This is line 4
```

sed

You can use the `sed` command to edit a file using a script. In the script, you can specify commands to edit one or more lines according to rules specified as part of one or more commands.

Following is a list of some of the flags that can be used with the `sed` command:

Flag	Meaning
-e <i>command</i>	Use the specified <code>sed</code> <i>command</i> to edit the file.
-f <i>filename</i>	Use the <i>filename</i> as the editing script to edit the file.
-n	Suppress messages from <code>sed</code> .

The `sed` command uses two different areas while performing editing:

- The *pattern area* holds selected lines for editing.
- The *hold area* temporarily holds the lines.

The `sed` subcommands can affect either all of the lines or only the specified lines.

Following is a list of some of the subcommands that affect the *pattern area* used by the sed command:

Subcommand	Action
#	Specify the start of comments. Everything in a line following the # is treated as a comment.
: <i>label</i>	Specify an addressable <i>label</i> that can be used in the script.
[<i>/pattern/</i>]=	Write to output the line number of each line that contains the specified <i>pattern</i> .
[<i>address</i>]a\textstring	Append <i>textstring</i> to each line specified by the <i>address</i> .
[<i>address1</i>][<i>,address2</i>]c\textstring	Replace the lines in the specified address the <i>textstring</i> .
[<i>address1</i>][<i>,address2</i>]d	Delete the lines in the specified address range.
[<i>address</i>]i\textstring	Insert <i>textstring</i> before each specified line.
[<i>address1</i>][<i>,address2</i>]p	Print the lines in the specified address range.
[<i>address1</i>][<i>,address2</i>]n	Specify that the current line be displayed and that the next line be made the current line.
[<i>address1</i>][<i>,address</i>]N	Specify that the current line be appended to the contents of the pattern area separated by a newline character.
[<i>address</i>]q	Exit when the specified address is encountered.
[<i>address1</i>][<i>,address2</i>]s/ <i>old pattern/new pattern/</i> [<i>flag</i>]	Change the <i>old pattern</i> to <i>new pattern</i> in the specified range. The behavior of the replacement can be modified by specified <i>flags</i> .
[<i>address1</i>][<i>,address2</i>]w <i>file</i>	Write the contents of the specified range to the specified <i>file</i> .
[<i>address1</i>][<i>,address2</i>]y/ <i>old character list/new character list/</i>	Modify each character in the <i>old character list</i> by the corresponding character in the <i>new character list</i> .

Following is a list of some of the subcommands that affect the *hold area* used by the sed command:

Subcommand	Action
[<i>address1</i>][<i>,address2</i>]g	Copy the contents of the hold area to the pattern area, which then becomes the new content of the pattern area.
[<i>address1</i>][<i>,address2</i>]G	Append the contents of the hold area to the pattern area following the specified address.
[<i>address1</i>][<i>,address2</i>]h	Copy the contents of the pattern area to the hold area, which then becomes the new contents of the hold area.

[<i>address1</i>][<i>,address2</i>]	Append the contents of the pattern area to the
H	hold area following the specified address.
[<i>address1</i>][<i>,address2</i>]	Exchange the contents of the pattern and hold
x	areas.

Examples

Assume that we have `file1` whose contents are displayed as shown here:

more file1

This file is a test file for sed command

The sed command is used for stream editing files

The sed command a *number* of sub-commands which may be used to do the

editing in specified line

If you want to print the line numbers of the line in which a specified pattern is found, use the following command:

sed -e "/sed/=" file1

1

This file is a test file for sed command

3

The sed command is used for stream editing files

5

The sed command a *number* of sub-commands which may be used to do the

editing in specified line

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)

All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

In this example, the line numbers are displayed for the lines containing the pattern sed. If you want to add a specified text after each specified line, use the following command:

```
sed -f sfile file1
This file is a test file for sed command
+++++
-----
The sed command is used for stream editing files
+++++
-----
The sed command a number of sub-commands which may be used to do the
+++++
-----
editing in specified line
-----
```

In this example, the file sfile contains the following line:

```
/sed/a\
+++++
```

In this example, a string of + (plus signs) is printed after each line containing the string sed. If you want to delete lines containing a specified string, use the following command:

```
sed -f sfile file1
This file is a test file for sed command
The sed command is used for stream editing files
The sed command a number of sub-commands which may be used to do the
editing in specified line
```

In this example, sfile contains the following:

```
/---/d
```

In this example, all lines that contain the string --- are deleted. If you want to change all occurrences of a particular string to another string, use the following command:

```
sed -f sfile file1
```

```
This file is a test file for sed command
```

```
+++++
```

```
The sed command is used for stream editing files
```

```
-----
```

```
The sed command a number of sub-commands which may be used to do the
```

```
-----
```

```
editing in specified line
```

```
-----
```

In this example, sfile contains the following:

```
1,3s/---/++++/g
```

- In this example, all occurrences of --- are replaced by ++++ for lines 1 through 3. If you want to insert a specified string before each line containing a specified string, use the following command:

```
sed -f sfile file1
```

```
++++
```

```
This file is a test file for sed command
```

```
-----
```

```
++++
```

```
The sed command is used for stream editing files
```

```
-----
```

```
++++
```

```
The sed command a number of sub-commands which may be used to do the
```

```
-----
```

```
editing in specified line
```

```
-----
```

In this example, sfile contains the following:

```
/sed/i\
```

```
++++
```

In this example, a string ++++ is printed before each line in which the string sed appears. If you want to change each occurrence of a character to another character, use the following command:

```
sed -f sfile file1
```

```
This file is A test file for sed commAnd
```

```
+++++
```

```
The sed commAnd is used for streAm editing files
```

```
-----
```

```
The sed command a number of sub-commands which may be used to do the
```

```
-----
```

```
editing in specified line
```

```
-----
```

In this example, sfile contains the following:

1,3s/-/+/g

In this example, each occurrence of a - (hyphen) is modified to a + (plus sign), and each occurrence of a is modified to A for lines 1 through 3, inclusive. If you want to delete all lines but the ones in which the specified pattern occurs, use the following command:

sed -f sfile file1

This file is a test file for sed command

The sed command is used for stream editing files

The sed command a *number* of sub-commands which may be used to do the

In this example, sfile contains the following:

/sed/!d

In this example, the ! (exclamation mark) is used to denote that all lines except those that contain the string sed are to be processed.

Miscellaneous Commands

The following sections discuss some of the commands available to do miscellaneous operations in UNIX.

banner

You can use the banner command to print one or more characters in a large size.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

BriefFull

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Example

If you want to print the word `banner` in a large size on the standard output, use the following command:

banner banner

```
#####      ##      #      #      #      #      #####      #####
#      #      #      #      ##      #      ##      #      #      #
#####      #      #      #      #      #      #      #####      #      #
#      #      #####      #      #      #      #      #      #      #####
#      #      #      #      #      ##      #      ##      #      #      #
#####      #      #      #      #      #      #      #####      #      #
```

bc

If you want to perform simple arithmetic expressions in UNIX, use the `bc` command. By default, all the numbers are assumed to be decimal numbers, but you can also perform operations on octal or hexadecimal numbers. You can also scale decimal numbers. The `bc` command accepts input first from the specified file and then from standard input. You can, however, use input redirection to accept input only from a file.

The arguments that can be used with the `bc` commands are as follows:

- Variable name (one letter)
- Variable array name (letter[expression])
- A literal such as `scale`

Some of the other operands that can be used with the `bc` command are as follows:

- `+` for adding
- `-` for subtracting
- `/` for division
- `*` for multiplication
- `%` for percentage
- `++` for adding 1 to the preceding variable
- `--` for subtracting 1 from the preceding variable
- `=` to assign a value
- `sqrt` for square root computation
- `length` for getting length of a number
- `scale` for specifying the number of digits after the decimal

You can also use C program-like statements, expressions, and functions. You can use some special arithmetic functions with `bc`; these functions include the following:

- `s(x)` for sine of x
- `c(x)` for cosine of x
- `l(x)` for log of x

Following is a list of flags that can be used with the `bc` command:

Flag	Meaning
<code>-c</code>	Compile the <code>bc</code> program parameters but do not execute them.
<code>-l</code>	Include the library of math functions.

Examples

Assume that we have `file1` which contains the following `bc` command parameters:

```
more file1
b=5
c=10
a=b+c
a
```

If you want to compile the contents of `file1` without executing them, use the `-c` flag as in the following example:

```
bc -c < file1
5sb
10sc
```

```
lblc+sa
laps.
q
```

If you want to execute the contents of `file1`, use the following command:

```
bc < file1
15
```

- Now assume that we have `file1` whose contents are displayed as shown here:

```
a=0
j=50
for (i=1; i<=j; i++) a=i+a;
a
```

If we execute the `bc` command with this file as input, `bc` will add all the numbers from 1 through 50 and display the total as follows:

```
bc < file1
1275
```

cal

You can use the `cal` command to display the calendar for one or more months on standard output. If you do not specify any arguments, `cal` displays the calendar for the current month. You can specify the month and year for which you want to display the calendar. If you specify only one argument, `cal` displays a calendar for all 12 months of the specified year.

Examples

If you want to display the calendar of the current month, execute the following command:

```
cal
          December 1996
Sun Mon Tue Wed Thu Fri Sat
  1   2   3   4   5   6   7
  8   9  10  11  12  13  14
 15  16  17  18  19  20  21
 22  23  24  25  26  27  28
 29  30  31
```

If you want to display the calendar for January 1995, use the following command:

```
cal 1 1995
          January 1995
Sun Mon Tue Wed Thu Fri Sat
  1   2   3   4   5   6   7
```

8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

If you want to obtain calendars for all 12 months of 1997, use the following command:

cal 1997

```

                                1997

      January                      February
Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat
      1  2  3  4                1  2  3  4  5  6  7  8
  5  6  7  8  9 10 11      9 10 11 12 13 14 15
12 13 14 15 16 17 18      16 17 18 19 20 21 22
19 20 21 22 23 24 25      23 24 25 26 27 28
26 27 28 29 30 31

      March                      April
Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat
      1  2  3  4                1  2  3  4  5
  6  7  8  9 10 11 12      6  7  8  9 10 11 12
13 14 15 16 17 18 19      13 14 15 16 17 18 19
20 21 22 23 24 25 26      20 21 22 23 24 25 26
27 28 29 30 31            27 28 29 30

      May                      June
Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat
      1  2  3                1  2  3  4  5  6  7
  4  5  6  7  8  9 10      8  9 10 11 12 13 14
11 12 13 14 15 16 17      15 16 17 18 19 20 21
18 19 20 21 22 23 24      22 23 24 25 26 27 28
25 26 27 28 29 30 31      29 30

      July                      August
Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat
      1  2  3  4  5                1  2
  6  7  8  9 10 11 12      3  4  5  6  7  8  9
13 14 15 16 17 18 19      10 11 12 13 14 15 16
20 21 22 23 24 25 26      17 18 19 20 21 22 23
27 28 29 30 31            24 25 26 27 28 29 30
                          31

      September                  October
Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat
      1  2  3  4  5  6                1  2  3  4
  7  8  9 10 11 12 13      5  6  7  8  9 10 11
14 15 16 17 18 19 20      12 13 14 15 16 17 18
21 22 23 24 25 26 27      19 20 21 22 23 24 25
28 29 30                26 27 28 29 30 31

      November                  December
Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat

```

						1		1	2	3	4	5	6
2	3	4	5	6	7	8	7	8	9	10	11	12	13
9	10	11	12	13	14	15	14	15	16	17	18	19	20
16	17	18	19	20	21	22	21	22	23	24	25	26	27
23	24	25	26	27	28	29	28	29	30	31			
30													

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
 All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

calendar

You can use the `calendar` command to get reminders from messages stored in a special file named `calendar` in the current directory. The messages are stored in the following format:

- date message
- message date

In these formats, date can be in a variety of formats such as these:

- March 7
- Mar 7
- mar 7
- march 7th
- 3/7
- */7 (7th of each month)

On Friday, the `calendar` command displays the messages for four days: Friday, Saturday, Sunday, and Monday.

clear

You can use the `clear` command to clear the screen of your workstation. This

command checks the terminal type to determine how to clear the screen.

Examples

To clear the screen on your terminal, use the following command:

```
clear
```

time

You can use the `time` command to obtain the execution time of a script, command, or program. The execution time is displayed with the following three times:

- Real
- User
- System

Examples

If you want to find out the execution time of the script `sample`, use the following command:

```
time sample
real    0m6.49s
user    0m0.02s
sys     0m0.03s
```

xargs

You can use the `xargs` command to group multiple arguments and then input them to a command. `xargs` passes as many arguments to the command as necessary to ensure that the maximum size limit for command-line arguments is not exceeded.

Following is a list of some of the flags that can be used with the `xargs` command:

Flag	Meaning
<code>-e</code> <i>endoffilecharacter</i>	Specify the character to be used to terminate the argument string. The default is the <code>_</code> (underline) character
<code>-i</code> <i>string</i>	Use each line as a single parameter in place of the <i>string</i> variable specified as part of the command line. The default <i>string</i> is <code>{ }</code> .

<code>-lnumber</code>	Specify the number of nonempty lines to be used as arguments to the command for each invocation. The last invocation can use fewer than the specified <i>number</i> .
<code>-nnumber</code>	Specify the number of arguments to be used in each invocation. The last invocation can use fewer than the specified <i>number</i> .
<code>-p</code>	Ask for confirmation before executing the command.
<code>-ssize</code>	Set the maximum size of the argument list for each invocation.
<code>-t</code>	Echo the constructed command to the standard error.

Examples

Assume that we have `xfile` whose contents are shown here:

```
more xfile
file1 file2 file3
file4 file5 file6
file7 file8 file9
```

If you want to pass only two arguments to the `ls` command at a time, use the `-n` flag as in the following example:

```
xargs -n2 ls < xfile
file1  file2
file3  file4
file5  file6
file7  file8
file9
```

If you want to pass two lines at a time to the `ls` command, use the `-l` flag as in the following example:

```
xargs -l2 ls < xfile
```

Caution:

The regular expression looks similar to the file-matching pattern used by some commands such as the `find` command. But the regular expression is not the same as the file-matching pattern.

```
file1  file2  file3  file4  file5  file6
file7  file8  file9
```


If you want to confirm the command to be executed before actually executing the command, use the `-p` flag as in the following example:

```
xargs -l2 -p ls < xfile
sfile1file2 file3 file4 file5 file6 ?...y
file1  file2  file3  file4  file5  file6
ls file7 file8 file9 ?...y
file7  file8  file9
```

In this example, you have to press the `y` key to confirm that the command should be executed. If you want to rename all the files that start with the name `file` (`file1` through `file9`), use the `-i` flag as in the following example:

```
ls file* | xargs -t -i cp {} {}.old
cp file1 file1.old
cp file2 file2.old
cp file3 file3.old
cp file4 file4.old
cp file5 file5.old
cp file6 file6.old
cp file7 file7.old
cp file8 file8.old
cp file9 file9.old
```

In this example, the `-t` flag forces the display of the constructed command to the standard error.

Regular Expression

A *regular expression* in UNIX is a string of one or more characters and metacharacters. The commands that accept regular expressions must first expand the expression to get the specified pattern before matching it to the input. The matching is done on a character-by-character basis.

A regular expression contains the following:

- *Character set*, which matches one or more characters at the specified position.
- *Count*, which specifies the number of the previous character to be repeated. This can be an `*` (asterisk) to specify that zero or more of the previous characters should be repeated.
- *Position specifier*, which is a set of special characters to indicate certain fixed positions such as the start of a line, the end of a line, and so on.
- *Metacharacters* to specify special meaning.

Character Set

A *character set* is a list of one or more specified characters. The character set can be specified as follows:

- *Range of characters*, which can be specified as two characters separated by a hyphen enclosed within square brackets. This set matches one occurrence of a character within the specified range. If you specify a ^ (caret) in front of the range, the matching is reversed, that is, all characters *except* those in the specified range will be matched.
- *List of characters*, which can be a list of individual characters enclosed within square brackets. This list matches one occurrence of one of the characters in the list. You can specify a ^ (caret) in front of one or more characters to match all characters *except* those.

Position Specifier

UNIX allows the use of a number of special characters to specify certain special positions in a line. Following is a list of these special characters:

- ^ at the start of a regular expression to specify the beginning of a line.
- \$ at the end of the regular expression to specify the end of a line.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc. All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Metacharacters

A *metacharacter* is a character that, when used as part of a regular expression, has a special meaning. Following is a list of metacharacters:

- `.` to match all characters except the newline character.
- `*` to match zero or more of the preceding characters or regular expressions.
- `^` to match the regular expression following this metacharacter at the beginning of the line (for this to work, you must specify `^` as the first character of the regular expression).
- `$` to match the regular expression preceding this metacharacter at the end of the line (for this to work, you must specify `$` as the last character of the regular expression).
- `[]` to match exactly one of the enclosed characters. The characters enclosed can be a range or a list of individual characters.
- `\{n1, n2\}` to match a minimum of `n1` and a maximum of `n2` occurrences of the preceding character or regular expression.
- `\` to interpret the following character as a regular character rather than as a metacharacter.
- `\(\)` to save the enclosed regular expression for later use. This expression can then be reused by using `\1` through `\9`.
- `\<` to match the following regular expression at the beginning of a word.
- `\>` to match the preceding regular expression at the end of a word.
- `?` to match zero or one instance of the preceding regular expression.
- `+` to match one or more instances of the preceding regular expression.

Examples

Assume that we have a file called `file1` whose contents are shown here:

```
more file1
```

```
This is a test  
THIS IS A TEST  
This is really a test  
Believe it, this is really a test  
This is a test, better believe it
```

You can specify a string of characters as a regular expression. If you want to find the string `really` in `file1`, use the following command:

```
grep really file1
```

```
This is really a test  
Believe it, this is really a test
```

If you want to find the string `THIS` at the beginning of a line, use the `^` (caret) at the beginning of a regular expression as in the following example:

```
grep ^THIS file1
```

```
THIS IS A TEST
```

If you want to find the string `it` at the end of a line, use the `$` (dollar sign) at the end of a regular expression as in the following example:

```
grep it$ file1
```

```
This is a test, better believe it
```

If you want to find both `believe` and `Believe`, use the following command:

```
grep [Bb]elieve file1
```

```
Believe it, this is really a test  
This is a test, better believe it
```

In this example, `[Bb]` matches the single character `B` or `b`. If you want to find characters other than the specified one, use the following command:

```
grep [T][^h] file1
```

```
THIS IS A TEST
```

In this example, the `[^h]` matches anything *other than* `h`. Hence, `[T][^h]` matches anything that starts with `T` and is followed by any character *other than* `h`. If you want to match any six-character string preceded and followed by a space, use the following command:

```
grep " ..... " file1
```

```
This is really a test
```

```
Believe it, this is really a test
This is a test, better believe it
```

In this example, the " " matches a string such as `really` or `better` preceded and followed by a space. If you want to modify all strings that start with a `t`, that have a `t` at the end, and that have any two characters in the middle, use the following command:

```
sed "s/\(t\)..\1/—/g" file1
This is a ----
THIS IS A TEST
This is really a ----
Believe i----his is really a ----
This is a ----, better believe it
```

In this example, `\(t\)` saves the character `t`; `\1` uses the `t` at the specified position. If you want to find one or more instances of a regular expression, use the following command:

```
egrep it+ file1
Believe it, this is really a test
This is a test, better believe it
```

In this example, `it+` tells `egrep` to find one or more instances of the string `it`. If you want to find whether a regular expression is repeated a specified number of times, use the following command:

```
egrep tt\{1,4\} file1
This is a test, better believe it
```

In this example, at least one—and a maximum of four—repetitions of the expression `it` are matched. If you want to modify all characters other than letters, use the following command:

```
sed "s/[^a-zA-Z ]/:/g" file1
This is a test
THIS IS A TEST
This is really a test
Believe it: this is really a test
This is a test: better believe it
```

In this example, all characters—other than `a` through `z`, `A` through `Z`, and spaces—are replaced by a `:` (colon).

Executing Commands

There are several ways to execute the commands you have learned about in this chapter. In this section, you learn about some of the ways you can execute command in isolation and in conjunction with other commands.

By default, UNIX accepts input from *standard input* (which is the keyboard) and displays output on *standard output* (which is the terminal). You can, however, use the UNIX redirection facility to redirect the input from a file or redirect the output to a file.

You can execute a command in the foreground or in the background. When you invoke a command, by default it executes in the foreground. You can force a command to the background by using the & (ampersand) sign. You can start a command in the foreground and then force it into the background. To achieve this, press Ctrl+Z to suspend the command and then use the bg command to put it in the background.

Because all UNIX commands accept input from standard input and generate output to standard output, there is a convenient way of passing output from one command to the next command: the | (pipe) character. You can have a string of commands, each connected to the next using a pipe.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Summary

In this chapter, you learned about various UNIX commands. Most of these commands should work on different UNIX systems as described here, but you may find that some of the commands or flags behave differently. Following is a list of some of the activities you can do using various UNIX commands:

- Log in to related activities using commands such as `login`, `rlogin`, and `passwd`.
- Create, rename, delete, and copy files and directories using commands such as `cp`, `rm`, `rmdir`, and `mkdir`.
- Search for text in files using commands such as `grep`.
- Grant access to files and directories for users using commands such as `chmod` and `chgrp`.
- Modify contents of files using commands such as `vi` and `sed`.
- Display contents of files using commands such as `more`, `tail`, `head`, and `pg`.

[Previous](#)

[Table of Contents](#)

[Next](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Chapter 5 Getting Around the Network

by David B. Horvath, CCP, and Rachel and Robert Sartin

In This Chapter

- What Is a Network?
- Connecting to Other Systems—`rlogin`, `telnet`, and `cu`
- Transferring Files—`rcp`, `ftp`, and `uucp`
- Other Networking Services
- Troubleshooting TCP/IP

The “information superhighway” has received a lot of attention recently. Much of this “network of the future” is with us today. This chapter introduces you to the basic UNIX software used today to connect hundreds of thousands of machines together on the Internet and locally within organizations.

Connecting machines in a network gives you even more computing and information resources than you can get from simply having a computer at your desk or in your computing center. A network of machines connected together enables you to share data files with coworkers, send electronic mail, play multiuser games with people from all over the world, read Usenet news articles, contribute to worldwide discussions, perform searches for software or information you need, and much more. In this chapter, you learn about the two most common ways to connect UNIX machines

together in a network: UUCP and TCP/IP. On this simple base exists a worldwide network of machines and services that has the potential to greatly increase your productivity. By learning to use these services effectively, you open the door to new possibilities using your computer. This chapter only begins to probe the extent of available software and resources. Refer to the Sams Publishing book *Internet Unleashed* for even more information on this topic.

What Is a Network?

A *network* is a system of two or more computers connected to one another. In this chapter, you learn about some common ways to network UNIX machines together. At the simplest end of the scale, a network can be two UNIX machines connected to each other using a serial line (typically through a modem) and running *UUCP*, the UNIX-to-UNIX Copy Program. More complicated network configurations run *TCP/IP*, the Transmission Control Protocol/Internet Protocol, the common name for the protocol family used on the Internet, a collection of networks that allows you to connect your computer to hundreds of thousands of other computers.

UUCP—The Basic Networking Utilities

Early in the history of UNIX, it became apparent that connecting UNIX machines would be advantageous so that they could share resources. One attempt to connect machines together resulted in the UUCP protocol, which allows you to connect two UNIX machines to each other using a serial line (often with a modem attached). The primary focus of UUCP is to allow files to be copied between two UNIX machines, but services built on top of UUCP allow execution of certain commands, such as news and mail commands, thus enabling more sophisticated processing. You can use UUCP to send electronic mail between two UNIX machines and to transmit and receive Usenet news articles. The most common release of UUCP available now is often called either *BNU*, the Basic Networking Utilities—the System V version of UUCP—or HoneyDanBer (HDB). There are other freely available and commercial implementations of UUCP. Although UUCP originated on UNIX and was designed specifically for copying between UNIX machines, there are now versions of UUCP that run on MS-DOS and other platforms.



Just in case your UNIX machine does not include UUCP, there is a freely available version of UUCP (1.06.1) on the CD-ROM. You can build this version on your UNIX machine, and it will interoperate with HDB UUCP.

Note:

Although UUCP comes with most UNIX versions and is also available for most other systems, its use is rapidly declining in favor of TCP/IP.

At one time, there was a National Network in the United States (and in some other countries) based in UUCP where one system would call another, which would call another, and so on. Systems would store and forward data (email, Usenet news, data files) as it moved around from the source machine to the destination machine over what could be many different hops.

`ftp` is used to transfer files over TCP/IP (more about this later in this chapter)

today.

TCP/IP—LAN, WAN, and the Internet

In the 1970s, the United States Department of Defense began a research program called ARPA, the Advanced Research Projects Administration. One of the efforts of ARPA was to create an *Internet*, an interconnected set of networks, that would allow research labs across the country to interact. This network was called the ARPAnet, and the protocol that ran the interconnections was and is called *IP*, or Internet Protocol. Since the original ARPAnet, internetworking has grown incredibly, and there is now a huge and difficult-to-define thing called the Internet that allows interconnections between computers all over the world. The Internet includes hundreds of thousands of machines (because of the amorphous nature of the Internet, it is difficult even to get an accurate count) connected through a series of public and private networks.

Note:

At some point in the mid-1980s, the “Defense” was added on to the name (Defense Advanced Research Projects Agency) resulting in DARPA.

The Internet Protocol allows the sending of packets between any two computers that are connected to the Internet. IP supplies only a primitive service, and further levels of protocol exist that use IP to perform useful functions. Two common protocols are *TCP/IP* and *UDP/IP*. TCP/IP connects two programs in much the same way a serial line connects two computers. UDP/IP, the User Datagram Protocol/IP, supplies a simple way of sending short messages between two programs. Most interesting user programs that use IP networking use TCP to create a connection, so “TCP/IP” is often used to refer to the interconnection protocol on the Internet.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Names and Addresses

To use machines and resources on the network, you need to locate them. Hostnames use a hierarchical naming space that allows each hostname to be unique, without forcing it to be obscure or unpronounceable. For example, `ftp.uu.net` is the name of one host on the Internet. IP itself uses *Internet addresses*, unique identifiers of Internet hosts, which are usually written in *dot notation*, four numbers (each between 0 and 255), separated by periods. For example, `192.48.96.9` is the address (as of this writing) of the host `ftp.uu.net`, which is covered in the section “Transferring Files—`rcp`, `ftp`, and `uucp`.”

What’s in a Name?

Hostnames on the Internet are a series of “words” separated by periods, or *dots*. The dots separate different parts of the name.

The naming system used is called the *domain naming system* (DNS) because it separates responsibility for unique names into administrative domains. The administrator of each domain is responsible for managing and assigning unique names within that domain. The management of the *top-level* or *root* domain, the extreme right word in a hostname, is responsible for the naming conventions. The best way to understand hostnames is to start out by reading them right to left, one word at a time.

Look at the hostname `ftp.uu.net`. Reading right to left, the first word is `net`,

which means that the hostname is a network service provider; see Table 5.1 for explanations of this and other top-level names. The next word is `uu`. Within `.net`, `uu` belongs to UUNET Communications, a company that supplies networking services. Elsewhere in the domain naming space, the name `uu` may mean something else.

Table 5.1.Top-level domains.

Domain	Meaning
EDU	Educational. Colleges and Universities.
ORG	Organizations. Nonprofit and not-for-profit.
NET	Networks. Networking services providers (some under COM).
COM	Commercial. Businesses.
GOV	Government. United States Federal government offices.
MIL	Military. The U.S. Armed Forces.
<i>cc</i>	Countries. <i>cc</i> is an ISO country code.
- US	An example of a country code. The United States.

Note:

Due in part to the history of the ARPAnet, most hosts in the United States (and some international organizations and businesses) are under `EDU`, `ORG`, `NET`, `COM`, `GOV`, or `MIL`. Many hosts in other countries are under a top-level domain that is the two-character ISO country code for the country. To further confuse things, the United States has a `U.S.` zone that includes local organizations, primary and secondary schools, and local governments.

This all may change in the future; various groups are working to change the top-level domain structure to remove the special status of domains in the United States.

Look at the hostnames `conch.newcastle.org` and `conch.austin.tx.us`. The `org` means that the name belongs to an organization. The `newcastle` means that Newcastle Associates is the owner. Finally, `conch` is a particular host in Newcastle's network. In the second name, `us` means the United States, `tx` means Texas, `austin` means the city Austin, and `conch` is a particular hostname. Note that the two machines are completely different machines with different owners. They happen to share one component of their name, but that is not a problem because of the hierarchical namespace presented by DNS.

In fact, there are many repetitions of names. Many machines on the Internet have `ftp` as the first part of their domain names, and many have `www` as the first part of their names. The advantage of using the DNS is that these repetitions are not in conflict. It has been said about names that "all the good ones are taken," but the DNS allows you

to reuse some of the good ones in a different context.

Note:

In addition to having an official name, some hosts have aliases as well. The alias is simply another name for the host. For example, `ftp.x.org` is actually an alias for the current machine being used for `ftp` by `x.org`.

Using Shorter Names

Usually, the DNS is configured to use a *search path* for hostnames that don't end in a dot. This lets you use shorter names for hosts in your search path. Typically, your DNS is configured to search your domain and then search progressively up to the root domain. Check your system documentation to see whether you can change the DNS search path. If you were on `cnidaria.newcastle.org` and used the name `newcastle.net`, it would try the following names, matching the first one that exists:

1. `newcastle.net.newcastle.org`
2. `newcastle.net.org`
3. `newcastle.net`

Tip:

Because of the search algorithm, you may see faster network access if you use full names ending in a dot for machines outside your local domain.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Decoding Addresses and Ports

Although DNS names are a reasonably convenient way for humans to refer to hosts, the Internet Protocol needs to use a 32-bit Internet address to find a host on the network. For example, as of this writing the host `ftp.uu.net` has the Internet Address `192.48.96.9`. Internet addresses are usually written using *dot names*, with four numbers between 0 and 255, separated by dots. Note that each of the four numbers is 8 bits, so you end up with a 32-bit Internet address.

It is not enough just to connect to the correct machine. You also need to connect to the correct program. TCP/IP and UDP/IP use *ports* to specify where a connection goes. To make a connection to a remote computer, there has to be some program listening on the correct port. If you think of IP addresses as being like phone numbers, a *port number* is like an extension. When your IP message reaches the correct machine, the port number enables it to be delivered to the correct program.

When a new protocol is adopted as a standard, it is assigned a port number that always is used for that protocol. For example, the `login` protocol used to implement `rlogin` is assigned port 513, and `telnet` is assigned port 23. You can examine the assignments of ports to protocols by looking at the file `/etc/services` on your machine. If you are running NIS (the Network Information System, formerly called the Yellow Pages), you can run the command `ypcat services` to look at the map.

Look at what happens when you run the command `rlogin remotehost`. If

remotehost is willing to accept *rlogin* requests, there is a program waiting for connections on port 513 on *remotehost*; this program (called *inetd*) handles all the work on *remotehost* that needs to be performed to allow you to use *rlogin* (*inetd* does this by handing the incoming connection to a program called *rlogind*, which implements the protocol). The *rlogin* program on your host attempts to open a connection to port 513 on the *remotehost*. The program monitoring port 513 on *remotehost* accepts the connection and lets your *rlogin* program perform the setup necessary to perform a login.

Converting Names to Addresses

You have seen what hostnames look like and what the low-level Internet address and port numbers are, but you still need to learn how names get converted to addresses.

Hostname conversion is usually handled by the domain naming system, which, in addition to specifying what hostnames look like, specifies a protocol for translating hostnames to addresses. First look at a hostname conversion of the name *ftp.x.org*. When your local host tries to convert the name *ftp.x.org* to an IP address, it contacts a *name server*, a machine that has DNS mappings loaded and is prepared to answer questions about them. Your name server is also configured with information about how to contact other name servers so that it can look up names that it doesn't already know.

A Brief Introduction to NIS

When implementing a network, one common problem that arises is management of *passwd* and *group* files. Some organizations want to have a common user and group list for all or most hosts in a network. The Network Information Service, introduced by Sun Microsystems, is one way to solve this problem. NIS allows sharing of *passwd*, *group*, and other information between hosts that share administrative control. Other (commercial and freely available) solutions to this problem exist, but none have yet become as widespread as NIS.

If you are running NIS, use the command `ypcat passwd` to examine the *passwd* information on your system. The actual `/etc/passwd` file does not list all the users who can log in to a machine running NIS. If you are using NIS to manage *passwd* files, your password is the same on any machine in your network that runs NIS. NIS may also be used to create an environment where you can share files transparently between systems. This is done using the network file system, NFS, which enables you to mount a file system from a mount computer and access it as if it were local. Some computing environments configure NIS so that your *HOME* is always the same directory, no matter what machine you use. This means that your files will be accessible no matter what machine in the network you are using. Check with your system administrators to find out whether NIS is running and whether it is being used to handle automounting of home (and other) directories.

Connecting to Other Systems—*rlogin*, *telnet*, and *cu*

With the three services *rlogin*, *telnet*, and *cu*, you can connect to a remote computer over the network. *rlogin* uses the login service to connect using the TCP/

IP protocol over the network, telnet uses the telnet service to connect using the TCP/IP protocol over the network, and cu connects over a phone line.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Before Using rlogin, rsh, and rcp

Before you use `rlogin`, some user configuration may be needed. The same configuration is used for `rsh` and `rcp`. Refer to these details when reading the next section as well. For reference, `loc-host` is used as the local machine name and `rem-host` is the name of the remote machine.

Two files on the remote machine affect your remote access capability: `/etc/hosts.equiv` and `.rhosts` in the remote user's home directory. The `hosts.equiv` file contains a list of hostnames. Each machine in this list is considered to be a trusted host. Any user who has an account on both `loc-host` and `rem-host` is allowed to access the remote machine from the local machine without question. The "without question" is important and means that the user does not have to supply a password for access.

Tip:

System administrators should seriously consider disabling the `rlogin` and `rexec` protocols on machines directly connected to the Internet because the authentication used on these protocols is very weak. At the very least, be extremely careful about entries in `/etc/hosts.equiv` and any `.rhosts` files.

The `.rhosts` file in the remote user's home directory contains a list of trusted host

and user pairs. This is similar to the trusted hosts of the `hosts.equiv` file but gives a finer grain of control. Each entry grants trusted access to one particular user on a particular host rather than to all common users on a particular host. Lines in `.rhosts` that name only a machine grant access to a user with the same login name. The user on *loc-host* can access *rem-host* without question (that is, without specifying a password). The user authentication is done by the protocol.

Usually only the system administrator can change the values in the `/etc/hosts.equiv` file. Because file allows many users access, this is a system configuration file. But each user can set up his or her own `.rhosts` file. This file must live in the user's home directory and be owned by the user (or by `root`). The ownership restrictions are security measures preventing a user from gaining access to another user's account.

Listing 5.1 and Listing 5.2 show examples of the `hosts.equiv` and `.rhosts` files. These two files are located on the machine called `flounder`, and the `.rhosts` file is owned by user `rob` and is located in his home directory. The two hosts listed in the `/etc/hosts.equiv` file, `manatee` and `dolphin`, are trusted hosts to `flounder`. Any user with an account on `manatee` and `flounder` may remotely access `flounder` from `manatee` without specifying a password. Likewise, any user with an account on `dolphin` and `flounder` may remotely access `flounder` from `dolphin` without specifying a password.

Listing 5.1. `/ETC/HOSTS.EQUIV` AND `$HOME/.RHOSTS` FILES.

```
manatee
dolphin
```

Listing 5.2. `/USERS/ROB/.RHOSTS` ON MACHINE FLOUNDER.

```
french-angel
rob yellowtail
rob dolphin
rob dolphin
root dolphin
diane stingray
rob stingray
root flying-gurnard
root
```

Caution:

Check your local documentation for the exact format of these files; some systems require a wildcard symbol when the username is specified without a remote system name.

The `.rhosts` file of the user `rob` contains a list of users on a remote machine who may access `flounder` as user `rob` without specifying a password. That sentence packed several important points together that need expanding:

- The `.rhosts` file of user `rob`—This implies that the machine `flounder`

has a user account, with `rob` as the username. The home directory of user `rob` (the example implies it is `/users/rob`) has a file named `.rhosts` that is owned by `rob`.

- Users on a remote machine who may access `flounder`—Each entry in the list is a pair of names—the machine name and the associated username. This pair of names describes one particular user who may access `flounder`. That user must be accessing `flounder` from the specified machine. It is not enough for the user to simply have an account on the machine; the remote access must be initiated from that machine (by that user).
- As user `rob`—This is probably the most subtle of all the points, so be careful here. Any of the users who are in the list may access `rob`'s account on `flounder`, as `rob`. They “become” `rob` on `flounder` even if they were a different user on the initiating machine. This is effectively the same as giving `rob`'s password on machine `flounder` to this user. Because of this, be extremely selective about entries in your `.rhosts` files.
- Without specifying a password—Some services (`rlogin`) allow for the possibility of a password prompt. If the user authentication was not successful via the equivalence files, the service can fall back on the prompt method of authentication. So the capability to access a remote host without specifying a password may not be needed. Other services (`rsh` and `rcp`) do not have a way to prompt for a password. To use these services, the access must be configured so that specifying a password is unnecessary.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Using Listing 5.2, for each of the following scenarios, decide whether the user would be able to access flounder—as rob—without a password. Assume that each user has an account on the local machine in the question, as well as on flounder.

1. User root on machine stingray?
2. User root on machine manatee?
3. User root on machine french-angel?
4. User frank on machine dolphin?
5. User frank on machine stingray?
6. User frank on machine tarpon?
7. User diane on machine manatee?
8. User diane on machine dolphin?
9. User diane on machine flying-gurnard?
10. User rob on machine yellowtail?
11. User rob on machine dolphin?
12. User rob on machine manatee?
13. User rob on machine flying-gurnard?

The answers are as follows:

1. Yes; rob's .rhosts file has an entry stingray root.
2. No; rob's .rhosts file does not have an entry manatee root. However, root from manatee could access flounder—as root—without a password because manatee is listed in /etc/hosts.equiv.

3. No; rob's `.rhosts` file does not have an entry `french-angel root`.
4. No; rob's `.rhosts` file does not have an entry `dolphin frank`. However, frank from dolphin could access flounder—as frank—without a password because dolphin is listed in `/etc/hosts.equiv`.
5. No; rob's `.rhosts` file does not have an entry `stingray frank`.
6. No; rob's `.rhosts` file does not have an entry `tarpon frank`.
7. No; rob's `.rhosts` file does not have an entry `manatee diane`. However, diane from manatee could access flounder—as diane—without a password because manatee is listed in `/etc/hosts.equiv`.
8. Yes; rob's `.rhosts` file has an entry `stingray diane`.
9. No; rob's `.rhosts` file does not have an entry `flying-gurnard diane`.
10. Yes; rob's `.rhosts` file has an entry `yellowtail rob`.
11. Yes; the `/etc/hosts.equiv` file has an entry `dolphin`. Note that if the system administrator removed this entry, this answer would still be yes because of the `dolphin rob` entry in his `.rhosts` file.
12. Yes; the `/etc/hosts.equiv` file has an entry `manatee rob`.
13. No; the `/etc/hosts.equiv` file does not have an entry `flying-gurnard` nor does rob's `.rhosts` file have an entry `flying-gurnard rob`.

- Using rlogin

If you need or want to be logged in to a computer that is away from your current location, `rlogin` can help you. The `rlogin` application establishes a remote login session from your machine to another machine that is connected via the network. This machine could be next door, next to you on your desk, or even on a different continent. When you successfully execute an `rlogin` from your screen, whether it is a terminal, or one window of your graphical display, the shell that prompts you and the commands you enter are executing on the remote machine just as if you sat down in front of the machine and entered login.

Establishing a rlogin Connection

The `rlogin` command takes a mandatory argument that specifies the remote host. Both the local and remote host must have `rlogin` available for a connection to be established. If this is the case, the local `rlogin` connects to the specified remote machine and starts a login session.

During a nonremote login, the login process prompts you for two things: your username and your password. Your username identifies you to the computer, and your password authenticates that the requester is really you. During an `rlogin`, the `rlogin` protocol takes care of some (or even all) of this identification/authorization procedure for you. The `rlogin` protocol initiates the login session on the remote host for a particular user. By default, this user is the same as the local user (that is, you). In this case, you never have to type in your username. However, if you want to log in to the remote host as a different user, you may override the default username by using the `-l` option to specify a username.

The `rlogin` protocol may even take care of the authentication for you. If you (or your system administrator) have made the proper entry in the `/etc/hosts.equiv` or your `$HOME/.rhosts` file, no authentication is necessary (that is, you are not prompted for your password). If these files do not have entries for your host and username, a password prompt is printed just like in a local login attempt.

A few examples follow. Assume that your username is `rachel`, and the local machine to which you're logged in is called `moray-eel`.

To log in as yourself on machine `flounder`, you would enter:

```
$ rlogin flounder
```

The connection to `flounder` would take place, and a login session would be initiated for user `rachel` (and fail if user `rachel` doesn't exist on `flounder`). Next, the `rlogin` protocol checks the special files to see whether authentication is necessary. If `moray-eel` is listed in the file `/etc/hosts.equiv` or in `~rachel/.rhosts`, no authentication is needed.

To log in to `flounder` as user `arnie`, you would enter:

```
$ rlogin -l arnie flounder
```

Here the login session is initiated with the username `arnie`. If user `arnie` exists on `flounder`, the special files are checked for authentication. Because the username for the remote login is different from the local username, the `/etc/hosts.equiv` file does not provide authentication. If the file `~arnie/.rhosts` has an entry `moray-eel rachel`, no authentication is necessary (that is, login succeeds without password). If this entry does not exist, the password prompt appears, and you must enter the password associated with user `arnie`. This is not a prompt for your password.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#) [Table of Contents](#) [Next](#)

Failed Connect

Several things might go wrong when you try to connect to a remote machine via `rlogin`. Some of these are problems that are out of your control. In these instances, you should contact a system administrator to help you solve the problem.

In cases where authentication is necessary, you might enter the password incorrectly. If this happens, the result is the same as in a local login attempt. The login process lets you try again by prompting first for your username and then your password. Note that this is the only situation in which you must supply your username if you're trying to `rlogin` as yourself.

For most other problems, you need your system administrator's help. See the section "Troubleshooting TCP/IP" for ways to identify the cause of the problem. Any details about the problem symptoms will help the person responsible for fixing the problem. Some of the problems you might see are the following:

- The user account does not exist on the remote.
- Your local host is not connected to the remote via the network.
- The remote host is down.
- The remote host does not support `rlogin`.
- The network between the local and remote hosts is having problems.

Using the Remote Login Session

After a successful remote login, the `rlogin` protocol initiates your session using some information from your local session. This saves you the trouble of having to initialize your environment totally from scratch. Your terminal type (the value of the `TERM` environment variable) is propagated. Other information, such as baud rate and your screen (window) size, may also be propagated, depending on what the local and remote hosts support.

Then the login process proceeds as if you were actually directly connected to this machine. All the information and files are taken from the remote. The remote password file contains the user account information, including the login shell to be executed and the starting (`HOME`) directory. All shell startup files (found on the remote) execute, which further initializes your environment. When the startup completes, the shell prompt you see is the shell running on the remote host.

Note:

In some LAN environments, the network is configured such that your `HOME` directory is on a remote file server that is mounted on each machine you access. In this case, you actually have just one physical `HOME` directory, and thus just one set of dot files (for example, `.login`). This results in the same login environment for you on each machine. However, this makes writing your dot files a little more complicated because you need to take into account all the different machines to accommodate.

Tip:

Because the remote prompt and local prompt might look alike, you might want to include hostname in your prompt variable (`PS1`). If you're ever in doubt about what host the shell prompt is coming from, use the `hostname` command.

When you see the remote prompt, you can enter any commands you would in a local environment. The `rlogin` protocol transfers input and output between the local and remote hosts. This transfer is transparent to you. Sometimes you might notice slow performance, depending on the network speed and load.

During your remote session, you might want to access your local machine. You could just exit your remote session, at which point you would be back at your local prompt. But if you aren't finished using the remote, using `exit` followed by another `rlogin`, possibly multiple times, is tedious. There is a better way—using the escape character.

Using the Escape Character

The `rlogin` protocol provides an escape character that, when typed as the first character on the command line, is treated specially. The default escape character is the tilde (`~`) character, but you may change this on the `rlogin` command line via the `-e` option. If the character immediately following the escape character is one that the local `rlogin` process recognizes, it performs the function associated with this character. Otherwise, the escape character (and the remaining characters) are executed on the

remote.

The `~.` character sequence is the command to disconnect from remote. This is not a graceful disconnect, as in an `exit`. It immediately disconnects from the remote. This should only be used when, for some reason, you are unable to execute the `exit` command.

If the local `rlogin` was executed by a job-control shell (C shell or Korn shell), then you can suspend the `rlogin` by the escape sequence `~susp`, where `susp` is your suspend control character, usually `Ctrl+Z`. This is very convenient. It saves the multiple `exit` command followed by another `rlogin` sequence you would otherwise need for accessing the local machine. In a graphical user interface environment, having two windows—one for the `rlogin` and one locally—solves this problem as well.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

It is possible to `rlogin` to one machine and then `rlogin` from there to another machine. You can use multiple escape characters to denote any one of the machines in this chain. For example, say that you are locally logged in to Host A. You are using a job-control shell with `suspend` set to `Ctrl+Z`. From Host A, you `rlogin` to Host B. From there, you log in to Host C. And from there, you `rlogin` to Host D. At this point, everything you enter is going all the way to D to execute. To reach any host in the chain, just associate one escape character with each host. You must start with your local host and then go in the same order as the `rlogins`. In this example, a single `~` refers to Host A, `~~` refers to Host B, and `~~~` refers to Host C.

To suspend the `rlogin` from Host B to Host C, type `~~^Z`. This leaves you in your original shell on Host B. To return to `rlogin`, use the `fg` command as with any suspended process.

To disconnect the `rlogin` from Host C to Host D, type `~~~..`.

One common escape sequence, which is not supported on all platforms, is the shell escape, `~!`. Typing this sequence causes the `rlogin` to give you a subshell on the machine that is referred to by `~`. You can use multiple escape characters to denote any host within a chain of `rlogins`. To return to the `rlogin`, simply exit the subshell.

Note:

There is a difference between `~suspend` and `~!`. The `suspend` command puts

`rlogin` in the background and lets you interact with your original shell (the one from which you ran `rlogin`). The shell escape starts a new shell as a child of `rlogin`.

Tip:

`telnet` does not require configuration on the remote system (`.rhosts` and `/etc/hosts.equiv`) like `rlogin` does.

Using telnet

The `telnet` service is used to communicate with a remote host via the `telnet` protocol. Invoking `telnet` with the remote host as an argument causes `telnet` to connect to that host. The remote `telnet` server usually initiates a login just as you would get on a terminal connected to the machine. After your login name and password are entered and verified, you see the shell prompt on the remote machine. All commands and input you enter go to the remote; all output you receive comes from the remote.

If you want to enter `telnet` command mode while you are connected to a remote, type the escape character. The default escape character is `Ctrl+]`, but this can be changed via the `set` command. To return to the remote connection, simply execute a command. A `set` command will do this. If you have nothing you want to send or set, do a `send nop`. The `nop` argument stands for *no operation*.

If you enter `telnet` without any arguments, you start up the `telnet` service in command mode. You see a special `telnet` prompt (`telnet>`). You can enter any of the `telnet` commands. Table 5.2 provides a list of some of the most common `telnet` commands you might use. Refer to your system's manual for `telnet`, for a complete list.

Table 5.2.Common `telnet` commands.

Command	Description
<code>open</code>	Connects to specified host.
<code>close</code>	Disconnects from host and returns to command mode.
<code>quit</code>	Closes the connection (if one exists) and exits <code>telnet</code> .
<code>set</code>	Changes the value for a given argument.
<code>send</code>	Sends a command to the remote and returns to remote connection.
<code>display</code>	Shows current setting of <code>telnet</code> configuration.
<code>status</code>	Shows current status of <code>telnet</code> connection.
<code>?</code>	Gives help.

The following sections look at some of these commands in a bit more detail.

open

The `open` command takes two parameters, `host` and `port`. The `host`, which is mandatory, can be a hostname or an IP address. This specifies the remote host to which a connection is to be established. This remote host must be reachable via the network and must support the `telnet` service. The `port`, which is optional, specifies the port number to use in connecting to the remote host. By default, the port to which `telnet` connects is the well-known `telnet` port (23). When a connection on the remote comes in on the `telnet` port, the remote's `telnet` service handles the connection. The remote `telnet` service assumes that the local connector wants to log in and invokes the login process on this connection. You can use this feature to do certain kinds of debugging and troubleshooting. For example, to connect to the mail server on a machine, you could enter `telnet hostname smtp` (or replace `smtp` with `25` if the first doesn't work). This connects you directly to the Simple Mail Transfer Protocol on `hostname`, and you can use this connection to troubleshoot mail problems. Sometimes network services are offered by `telnet` to a specific port number. For example, many `gopher` and `WWW` providers offer a special port for `telnet` access to the service.

In this default mode, a `telnet open` command is somewhat like an `rlogin`. A remote login is initiated on the remote host. But the `telnet` protocol, unlike `rlogin`, does not perform any conveniences for you. It does not propagate any of your local environment. It does not perform any part of the login procedure (user identification and authentication).

If the first thing you use `telnet` for is an `open` command, you do not need to enter `telnet` command mode at all. On the `telnet` command line, you can enter a host followed optionally by a port number. This causes `telnet` to immediately do an `open` with the command-line arguments.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc. All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

close

The `close` command terminates the open connection (if one exists). On some versions of `telnet`, this does not exit `telnet` command mode. So if you are connected to Host B but decide you really want to be connected to Host C, enter `close` and then enter an `open B` command.

quit

The `quit` command should be used when you are finished using `telnet`. This command performs a `close` on the open connection (if one exists). Then it terminates the `telnet` service, returning you to your local shell prompt.

set

`telnet` has several internal variables used for configuration. You can use the `set` command to change these values. To see the current variable values, use the `display` command. The `telnet` escape character can be changed via `set`.

Tip:

You can set certain special characters (such as `erase`) with `telnet`, but these settings may only work if you run `telnet` in line mode. Line mode is often used for connecting to remote machines that have line-oriented user

interfaces and allows you to compose an entire line of text input before sending it to the remote (when you press Enter). You should probably not use line mode when connecting to a UNIX machine because interactive commands (such as `vi`), job control, and some shell history (`ksh` interactive command editing) rely on receiving characters as they are typed.

?—The Question Mark

The question mark (?) is a `telnet` command that, without arguments, gives a list of all the `telnet` commands. This is useful if you've forgotten the name of a command. To get help about a specific command, use ? with the command as an argument. The ? can also be used as an argument to the `set`, `send`, and `toggle` commands to list the valid arguments of the command.

Before Using cu

Before you can use `cu`, your system administrator needs to configure the appropriate devices and machines for UUCP access. Check your system's UUCP documentation for information on how to do this.

Using cu

The `cu` service calls up another system. This service is used only to connect two computers via phone lines. Your local host must have an outgoing modem, and the remote host must have a modem that supports incoming calls.

Your system administrator may have configured the parameters necessary to call up certain systems. This configuration is kept in the file `/etc/uucp/Systems`.

Note:

The actual file depends on which version of UUCP you have. This is correct for SVR4. Because the location may vary, consider this the "systems" file.

You can enter `cu system-name` to dial the remote host. If the remote host has not been configured in the `/etc/uucp/Systems` file, you can specify the necessary parameters on the command line. The `cu phone-number` command calls up the specified phone number. For example, `cu 9=14085551212` calls using the `ad` device and gives it the phone number 914085551212. The equals sign specifies that a pause is desired before the next digit is dialed.

You can also call up using a local device by specifying it with the `-l` option. You can use the `-l` option to specify the device to use for making the connection. This is generally used only for hard-wired connections: `cu -l dev dir` connects directly to the line named `dev`.

Transferring Files—rcp, ftp, and uucp

Files are the basis for everything you do in UNIX. When you execute a command (aside from shell built-ins), the associated file contains the executing instructions. When you store or retrieve information, the data is kept in one or more files. The UNIX interface to hardware devices is through device files. Files are pervasive. Therefore, having the necessary files within your reach is extremely important.

Sometimes files you need are not stored on your local machine. Client-server environments are designed to provide a means of sharing files among many machines. When machines on a LAN are configured to share files (via the network), many more files become reachable to you. If you are using NFS, some directories on your system are mounted from remote machines. These directories and files are available as part of the normal UNIX file system, and you need no special techniques to access them.

Not all UNIX environments are configured this way. Even those that are might not share all file systems of all machines. Many files exist outside a local LAN environment. In these cases, you might want to obtain a copy of a file from somewhere other than your local environment. You could use the tools in I'm on the wire to remotely log in and access them. But if you need to execute the file locally, or want to have your own copy of the file, you need to copy the remote file to your local system.

The next section presents several tools to do remote copies. Your local configuration, the remote configuration, the way the remote and local configurations are connected, as well as your personal preference will determine which tool you choose.

Using rcp

You might want to review the section “Before Using rlogin, rsh, and rcp” before reading this section. For rcp to work, you must configure the remote machine(s) so that user authentication is not necessary. For each remote you access via rcp, an entry in one or both of `/etc/hosts.equiv` and `$HOME/.rhosts` is mandatory. This is because rcp does not have a mechanism for in-process authentication (unlike rlogin).

After the configuration is complete, you can use rcp in much the same way you use the cp command. Each command basically says to “copy File A to Location B.” The rcp command adds some syntax that enables you to specify remote machines and users.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Specifying a Remote File

You can specify a remote file in several different ways. In general, unless a hostname is specified, the file is considered local. If the character string has a colon (:) before any slashes (/), the string before the colon specifies the remote host, and the string after the colon specifies the file path. Here are three forms of the complete remote file specification:

- `hostname:filepath`
- `user@hostname:filepath`
- `user@hostname.domain:filepath`

The file path in each can be an *absolute* path, a *relative* path, or blank. If it is relative, it is relative to the remote user's HOME directory. The remote user is considered the same as the local user unless explicitly included in the remote specification. In the preceding second and third forms, the remote user is explicitly included.

If the file path is absolute, this is an absolute path on the remote system. If the file path is blank, the user's HOME directory is assumed.

The hostname can be a simple name or an alias of the remote machine, or it can be a host domain name as in the third form shown previously.

If you want to use a different user account on the remote machine, you can specify the

remote file, including the username. The username must refer to an account on the remote machine, and the user's `$HOME/.rhosts` file must contain the proper entry for your local machine.

Understanding the rcp Command-Line Syntax

- The `rcp` command line is flexible; to support this flexibility, there are a few variations of the command line:

- `rcp single-file dest`—In this variation, the first argument, *single-file*, is a single file. This file is copied to the destination *dest*. If *dest* is an existing directory, the file *dest/single-file* is created. If *dest* is an existing file, *dest* is overwritten with *single-file*. Otherwise, the file *dest* is created by copying *single-file*.
- `rcp sources dest`—In this variation, the first argument, *sources*, is one or more files or directories. *dest* must be a directory. Only the members of *sources* that are files are copied to the destination *dest*. If *dest* is an existing directory, the files are copied under directory *dest*. It is unwise to specify a *dest* directory that does not exist with this form of the `rcp` command. The results vary from system to system. See the next form for copying a single directory.
- `rcp -r sources dest`. By adding the option `-r`, the files in *sources* as well as the directories (and all their subdirectories) are copied to *dest*.
- If *sources* is a single directory, it is okay to specify a destination *dest* that doesn't exist. The directory is created for you. This is probably what you want. Beware of this situation because if *dest* *does* exist, the copied directory is placed as a subdirectory of *dest*.
- If *sources* is multiple directories or files, *dest* must be an existing directory. If it doesn't exist, the results are not specified and differ from one UNIX system to another.
- Each version of the `rcp` command line supports an additional option, `-p`. This option causes `rcp` to preserve the modification times as well as the modes when the copy is made.

Using ftp

The `ftp` service is the interface to the file transfer protocol. This service provides a connection service to a remote computer along with file manipulation functions including sending and receiving files. It also provides user authentication, unlike `rcp`. It supports different file types.

To connect with a remote host, you can simply type `ftp hostname`. The *hostname* can either be a hostname or an Internet address. If you do not specify a remote host on the command line, you enter `ftp` command mode. Then you can use the `open` command to initiate a connection.

By default, when a connection is initiated via `ftp`, the remote `ftp` server starts up the login process. You must enter a valid username and password to access the remote system. After you have been authenticated, you are connected to the remote `ftp` server, and it awaits your commands.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

The `ftp` service has many commands. Table 5.3 lists several common commands. For complete details, refer to your system’s manual for `ftp`.

Table 5.3.COMMON FTPSERVICE COMMANDS.

Command	Description
<i>Connection-Related Commands</i>	
<code>open</code>	Open a connection to specified host.
<code>close</code>	Close current open connection.
<code>quit</code>	Close current open connection and exit <code>ftp</code> .
<i>File Transfer-Related Commands</i>	
<code>binary</code>	Change the file representation type to binary.
<code>ascii</code>	Change the file representation type to <code>ascii</code> .
<code>put</code>	Transfer a single file from the local to the remote host.
<code>mput</code>	Transfer multiple files from the local to the remote host.
<code>get</code>	Transfer a single file from the remote to the local host.
<code>mget</code>	Transfer multiple files from the remote to the local host.
<i>File and Directory Management Commands</i>	
<code>cd</code>	Change remote’s current working directory (UNIX <code>cd</code>).
<code>lcd</code>	Change the local’s current working directory (UNIX <code>cd</code>).

<code>cdup</code>	Change remote's current working directory to be the parent directory (UNIX <code>cd ..</code>).
<code>dir</code>	List the remote's current working directory (UNIX <code>ls</code>).
<code>pwd</code>	Print the remote's current working directory (UNIX <code>pwd</code>).
<code>mkdir</code>	Make a new directory on the remote (UNIX <code>mkdir</code>).
<code>rmdir</code>	Delete a directory on the remote (UNIX <code>rmdir</code>).
<code>rename</code>	Change the name of a remote file or directory (UNIX <code>mv</code>).
<code>delete</code>	Delete a remote file (UNIX <code>rm</code> , with one file specified).
<code>mdelete</code>	Delete multiple remote files (UNIX <code>rm</code> , with multiple files).

Miscellaneous Commands

<code>?</code>	Obtain help about <code>ftp</code> .
<code>!</code>	Escape shell.

Connection-Related Commands

The `ftp` connection-related commands are fairly straightforward. The `open` command tries to connect to the `ftp` server on the specified remote host. The `close` command terminates the open connection (if one exists) and then returns to command mode. This is usually used when you want to connect to a different host, so you will commonly follow it with an `open` command. The `quit` command closes the connection and then exits `ftp`.

File Transfer-Related Commands

The `ftp` service defines several file representation types for transfer. The two most common are `ascii` and `binary`. By default, the type is set to `ascii`. Any file that is plain ASCII text can be transferred using `ascii` type. Binary files, like a compiled and linked executable file, must be transferred using `binary` type. Be sure to set the correct type before transferring any files.

Tip:

Transferring ASCII text files between UNIX machines is slightly faster with `binary` type, but using `binary` type to transfer an ASCII text file between a UNIX and a non-UNIX machine might corrupt the file.

Tip:

If you are having trouble decoding or executing a binary file you got elsewhere, check to make sure that you used `binary` type transfer.

The `get` and `mget` commands transfer files from the remote to the local host. The `put` and `mput` commands transfer files from the local to the remote host. Both `get` and `put` transfer one file per command. On both of these commands, you may specify

the destination for the file copy. If the destination is not specified, the file is placed in the current working directory. Both `mget` and `mput` transfer multiple files per command. The files are placed in the current working directory.

File and Directory Management Commands

The file and directory management commands are analogous to UNIX file and directory commands. In Table 5.3, the UNIX command that is analogous to the `ftp` command is given in parentheses. Remember that all these commands, except `lcd`, operate on the remote file system. If you need to perform more in-depth local file management, use the shell escape command (`!`) to escape to a local shell prompt.

Miscellaneous Commands

The `?` command provides help about `ftp` commands. If you want help about a specific command, you can specify this command as the first argument to the `?`. The shell escape command (`!`) is used to start a subshell on the local machine. This is useful if you need to perform some operations on your local host while you are connected to a remote `ftp` server. After you are finished working on the local host, simply exit the (sub)shell, and you will return to `ftp`.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Configuring with .netrc

The `ftp` command can automatically perform the login to remote `ftp` servers and initialize your connection. It does this by reading in the `.netrc` file in your home directory. You can configure the login, password, and account (some `ftp` servers allow or require an extra account specification at authentication time) to use for a particular machine. In the following example from the `.netrc` file, automatic login is included as anonymous for several popular servers:

```
machine dg-rtp.rtp.dg.com login anonymous password sartin@pencom.com
machine town.hall.org login anonymous password sartin@pencom.com
machine ftp.uu.net login anonymous password sartin@pencom.com
machine rtfm.mit.edu login anonymous password sartin@pencom.com
machine ftp.x.org login anonymous password sartin@pencom.com
machine prep.ai.mit.edu login anonymous password sartin@pencom.com
machine ftp.ncsa.uiuc.edu login anonymous password sartin@pencom.com
machine emx.cc.utexas.edu login anonymous password sartin@pencom.com
machine boombox.micro.umn.edu login anonymous password sartin@pencom.com
machine rs.internic.net login anonymous password guest
```

Tip:

Most versions of `ftp` use your `.netrc` for password information only if the file is readable by you only. For password security, this file should be unreadable by others or, better yet, should contain no sensitive passwords.

Anonymous ftp

A special login for `ftp` allows you to anonymously access files on part of a remote machine. Anonymous access is not entirely anonymous because some machines log the connection, the password used, and all files retrieved. To use anonymous `ftp`, you use the login `anonymous` (on some machines, the login `ftp` works) and supply any nonempty string for the password.

Tip:

Some machines do password validation on anonymous logins. Most require that you supply a valid email address.

After you have successfully logged in as anonymous, you are granted limited access to the anonymous ftp subtree on the remote machine. All the commands described in this section can be used. Some sites have a directory called `/incoming` (or a directory named `incoming` somewhere in the ftp tree) where you can put files. Many sites put the publicly accessible files under `/pub`.

Using uucp, uuto, and uupick

The file copying tools `uucp`, `uuto`, and `uupick` are part of the Basic Networking Utilities software release. These might not be on your UNIX system. Even if they are, more recent networking services (for example, `ftp` and `rcp`) are preferred. If you are interested in using the uu tools, check your system documentation to see whether they are supported.

Following, for the sake of completeness, is a brief summary of these tools. For details, check your system's manual entry for each command.

uucp

The `uucp` service copies one or more files from one UNIX machine to another UNIX machine. Use the `uname` command to see what remote machines you can reach via `uucp`. `uucp` uses an older host naming scheme in the form `hostname!filepath`. To copy a local file, `myfile`, to remote machine `rem-host` to directory `/tmp`, enter the command `uucp myfile rem-host!/tmp/`.

uuto and uupick

The `uuto` tool sends a file to a specific user on a remote UNIX host. The file is deposited in a special place on the specified remote host. For the remote user to receive this file, she must use the `uupick` tool. The remote host and user are specified by the syntax `rem-host!username`. To send the local file `myfile` to user `arnie` on machine `sturgeon`, enter the command `uuto myfile sturgeon!arnie`.

Then user `arnie` must use the `uupick` tool to receive the file.

When you are ready to receive files that were sent via `uuto`, simply enter the `uuto` command without any arguments. Each file that has been sent to you is displayed, one at a time. As each is displayed, you have the choice of skipping it, moving it, deleting it, or printing it.

Other Networking Services

This section gives an abbreviated introduction to some other services currently available on the Internet. These services give you access to the wealth of information available on the Internet, including source code, current weather information, financial data, computer conferences on a wide variety of topics, and some more frivolous programs, including a computerized tarot reader.

Caution:

These programs are useful to you only if you are connected to the Internet and have a gateway that allows you to make outgoing connections. Check your local network configuration to be sure.

Caution:

These programs can be addictive. Make sure that you get enough sleep and social activity between your net surfing excursions.

archie

The `archie` program offers access to a large list of files available via anonymous `ftp`. When you run an `archie` string search, the server searches for a name that is an exact match for the string in its list of archives and returns the matches to you. You can modify the search behavior by specifying one of the following:

- `-c`—Case-sensitive substring search
- `-r`—Regular expression search
- `-s`—Case-insensitive substring match

For example, if you were looking for the source to `xmosaic`, you could enter `archie -s xmosaic`. The output lists many sites that have `xmosaic` available via anonymous `ftp`. Here is part of the response from that command:

```
Host ftp.engr.ucf.edu
  Location: /pub/linux-mirrors/tsx11/binaries/usr.bin.X11.nomirror
          FILE -rw-r--r--      497473  Dec 26 18:06  xmosaic-
1.2.term.tar.z
```

For each host that had a match of the string, there is a list of locations that had matches. The best way to use `archie` output is to look for a host “near” you (for example, your service provider, someone in the same city/state as your service provider, someone in the same country) and use `ftp` to retrieve the desired files.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Click Here!

ITKnowledge

- home
- account info
- subscribe
- login
- search
- My ITKnowledge
- FAQ/help
- site map
- contact us

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous Table of Contents Next

gopher

The University of Minnesota has developed a program called gopher, that you can use to retrieve information over the Internet. They report (in the 00README file available by anonymous ftp from boombox.umn.edu in /pub/gopher/00README):

The internet Gopher uses a simple client/server protocol that can be used to publish and search for information held on a distributed network of hosts. Gopher clients have a seamless view of the information in the gopher world even though the information is distributed over many different hosts. Clients can either navigate through a hierarchy of directories and documents -or- ask an index server to return a list of all documents that contain one or more words. Since the index server does full-text searches every word in every document is a keyword.

If you want to test a gopher client without setting up your own gopher server you should configure the client to talk to "gopher.micro.umn.edu" at port 70. This will allow you to explore the distributed network of gopher servers at the University of Minnesota. You can try the Unix client by telneting to consultant.micro.umn.edu and logging in as "gopher".

World Wide Web

In 1991, the European Laboratory for Particle Physics began a project that turned into the World Wide Web, also known as WWW or W3. WWW is fairly difficult to pigeonhole, and the best way to become familiar with it is to explore it. WWW is a set of software, conventions, servers, and a protocol (HTTP) for organizing information in a hypertext structure. It allows linking of pictures (both still and moving), sounds, and text of various kinds into a web of knowledge. You can start at any place (a particularly good place to start is the default home page at NCSA or a copy of it, using xmosaic) and choose the links that interest you. Information is located using a Uniform Resource Locator (URL), which generally looks like this:

protocol://hostname/path

The *protocol* tells how to access the data (and is often `http`, which indicates the Hypertext transfer protocol). The *hostname* tells the name of the host to access. The *path* gives a host-specific location for the resource; paths often look like normal UNIX filenames. A big difference between an URL path and a filename is that an URL path often points to information that is generated on-the-fly (for example, a current weather report), and the actual data returned might depend on what features your WWW client supports. By exploring the Web, you can find information ranging from your personal biorhythms to common questions about the PowerPC to an archive of scuba diving destination reports.

The National Center for Supercomputing Applications at the University of Illinois has developed World Wide Web interfaces called Mosaic. The UNIX version runs with Motif widgets using X11 and is called `xmosaic`.

Troubleshooting TCP/IP

Sometimes you might find that your attempts at making network connection are not working. Some common errors for each command were covered in the sections “Connecting to Other Systems—`rlogin`, `telnet`, and `cu`” and “Transferring Files—`rcp`, `ftp`, and `uucp`.” This section covers some system-level troubleshooting you might want to try if you are having trouble making network connections using TCP/IP (`rlogin`, `telnet`, `rcp`, `ftp`, and the commands mentioned in the section “Other Services”). The suggestions here help solve simple problems and help classify problems. See Chapter 26, “Networking,” for more information on troubleshooting network problems.

nslookup to Check Address Mapping

One common failure in trying to make network connections is either having the wrong hostname or encountering an error or delay in the name service. One way to check the validity of the hostname is to try using the `nslookup` command. The simplest way to run the `nslookup` command is `nslookup hostname`:

```
$ nslookup ftp.uu.net.  
Name Server:  lazarus.pencom.com  
Address:  198.3.201.57
```

```
Name:      ftp.uu.net  
Address:   192.48.96.9
```

```
$ nslookup no.such.name.org  
Name Server:  lazarus.pencom.com  
Address:  198.3.201.57
```

```
*** lazarus.pencom.com can't find no.such.name.org: Non-existent domain  
$
```

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

•



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

This queries the DNS for the name of hostname (`ftp.uu.net` in the first example, `no.such.name.org` in the second).

Tip:

When a machine is newly added to the DNS, it might take a while before the name servers learn about it. During that time, you may get "unknown host" errors. The person who adds a new host to the DNS should be able to give an estimate of how long to wait before a DNS failure should be considered an error.

Is There Anybody Out There? (ping)

If you can find the address of a host but your connections are failing, it might be because the host is unreachable or down. Sometimes you might get a "host unreachable" or "network unreachable" error. You get these messages when the software that manages interconnections determines that it could not send a packet to the remote host. The network routing software has internal tables that tell it how to reach other networks and hosts, and these error messages indicate that there is no table entry that lets you reach the desired network or host.

When a host is simply down, you might get connection timeouts. You might want to try using the `ping` command to test whether a host is running. The `ping` command sends a special kind of message called an *Internet control echo request message* or *ICMP echo request* (ICMP is the Internet control message protocol). This message asks the remote computer to send back an echo reply that duplicates the data of the echo request

message. The low-level networking software of the remote computer handles responding to an echo request, so a machine should be able to respond to a ping as long as the network software is running.

The following example uses ping to check the status of two hosts:

```
- $ /etc/ping conch 100 10
PING conch.pencom.com: 100 byte packets
100 bytes from 198.3.200.86: icmp_seq=0. time=3. ms
100 bytes from 198.3.200.86: icmp_seq=1. time=4. ms
100 bytes from 198.3.200.86: icmp_seq=2. time=3. ms
100 bytes from 198.3.200.86: icmp_seq=3. time=5. ms
100 bytes from 198.3.200.86: icmp_seq=4. time=4. ms
100 bytes from 198.3.200.86: icmp_seq=5. time=8. ms
100 bytes from 198.3.200.86: icmp_seq=6. time=3. ms
100 bytes from 198.3.200.86: icmp_seq=7. time=3. ms
100 bytes from 198.3.200.86: icmp_seq=8. time=3. ms
100 bytes from 198.3.200.86: icmp_seq=9. time=3. ms

conch.pencom.com PING Statistics--
10 packets transmitted, 10 packets received, 0% packet loss

round-trip (ms)  min/avg/max = 3/3/8

$ /etc/ping brat 100 10
PING brat.pencom.com: 100 byte packets

--brat.pencom.com PING Statistics--
10 packets transmitted, 0 packets received, 100% packet loss
$
```

In the first example, the 100 says to use 100 bytes of data in each message, and the 10 says to use 10 messages. All 10 messages were returned. The second example shows what happens when you attempt to ping a host that is not up.

After you determine that the remote host is not responding, you can either attempt to get the machine back up or wait until later to use it. If the machine is on your LAN, it should be fairly easy to go to it and start it running or talk to a local administrator. If the machine is somewhere remote, you might need to phone or email someone to get assistance. If the machine is a resource on the Internet that is offered by some other school or company, you should probably just wait until it is running again unless your need is urgent (for both you and the remote administrator).

Summary

In this chapter, you learned how UNIX machines are networked and how to take advantage of that networking. You have learned to log in to remote machines, copy files, begin to surf the Internet, and troubleshoot minor problems. By using these network services, you can perform useful work on networked systems and explore the “information superhighway.”

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Chapter 6 Communicating with Others

by Chris Byers, Ron Dippold, and Fred Trimble

In This Chapter

- Electronic Mail (Email)
- Usenet
- Talk
- Internet Relay Chat (IRC)
- Multimedia

From its inception, the purpose of the Internet has been to facilitate communication among people. It was originally developed by the military to provide a vast distributed communications network capable of continued operation in case of a nuclear attack. Its designers wanted a distributed network to eliminate the possibility of a “vulnerable central node.” They also wanted a communications protocol that would be independent of any particular physical media. Despite its military roots, it has become characterized by the general public as the “Infobahn,” “Information Superhighway,” and “Cyberspace.” Today, some 20 years later, the benefits of the Internet are being realized by many groups of people, including schools, home users, and private industry. The Internet infrastructure was originally designed to support applications such as electronic mail and file transfer. Although electronic mail is still the most popular application on the Internet, other networking hardware and protocols continue

to evolve so that they can support other types of communication, including real-time audio and video.

Throughout the history of the Internet, UNIX has certainly played a major role. Most early UNIX systems provided built-in support for the Internet's main protocol: TCP/IP (Transmission Control Protocol). Therefore, this chapter covers the following topics, with an emphasis on UNIX facilities where appropriate:

Email	Electronic mail allows you to exchange messages with other people all over the world. Many electronic mail programs have extended features, such as the capability to attach binary files.
Usenet	Usenet is the world's largest electronic discussion forum. One of the most popular features of the Internet, it allows people all over the world to discuss topics and exchange ideas on a wide variety of subjects.
Talk	The <code>talk</code> command allows two people to exchange text messages in real-time.
IRC	Internet Relay Chat extends the capabilities of the <code>talk</code> command. It provides a real-time multiple-person discussion forum, much like a CB radio channel.
Multimedia	The Internet allows real-time audio and video to be transmitted.
The future	This section provides a glimpse into the future of the Internet.

Electronic Mail (Email)

Electronic mail is the most widely used application on the Internet. It is known as an asynchronous type of communication system because after a mail message has been sent, it resides on the recipient's computer until the recipient logs on and retrieves the message. This section focuses on many facets of email, including the structure of a mail message, sending binary data (such as a graphics file) with a mail message, email addressing, how messages are sent over the Internet, and common end-user mail programs.

Components of a Mail Message

A mail message consists of two main sections: a message header and a message body. The header contains information such as who sent the message and when it was sent. The body contains the actual message text. Some people finish their messages with an optional third part known as a "signature." Each of these mail message sections is described in detail in the following sections.

Message Headers

The message header consists of several lines at the top, formatted as "*keyword: value*"

pairs. Messages sent to a user who is located on the same local UNIX host using the mail or mailx program have a simple structure. For example:

From smithj Thu Apr 24 00:42 EDT 1997
To: jonest
Subject: Code Review Meeting
Status: R

Please plan on attending the code review meeting tomorrow at 10:00am.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#) [Table of Contents](#) [Next](#)

The message header of a mail message that ends up going over the Internet, however, is much more complex. For example:

```
From nihil@eniac.seas.void.edu Thu Apr 24 08:15:01 1997
Flags: 000000000015
Received: from phones.com (phones.com [229.46.62.22]) by
happy.phones.com (8.6.5/QC-BSD-2.1) via ESMTP;
id IAA13973 Thu, 24 Apr 1997 08:14:59 -0800 for
<rdippold@happy.phones.com>
Received: from linc.cis.void.edu (root@LINC.CIS.VOID.EDU
[230.91.6.8]) by phones.com (8.6.5/QC-main-2.3) via ESMTP;
id IAA14773 Thu, 24 Apr 1997 08:14:56 -0800 for
<rdippold@phones.com>
Received: from eniac.seas.void.edu (nihil@ENIAC.SEAS.VOID.EDU
[230.91.4.1]) by linc.cis.void.edu (8.6.5/VOID 1.4) with
ESMTP id LAA17163 for <rdippold@phones.com>
Thu, 24 Apr 1997 11:14:45 -0500
Received: from localhost by eniac.seas.void.edu
id LAA24236; Thu, 24 Apr 1997 11:14:44 -0500
From: nihil@eniac.seas.void.edu [B Johnson]
Sender: nihil@ocean.void.edu
Reply-To: nihil@void.edu,nihil@freenet.com
Cc: group-stuff@uunet.UU.NET
Cc: james@foobar.com
Message-Id: <199302011614.LAA24236@eniac.seas.void.edu>
Subject: Re: Apple IIe/IIgs Software and books for SALE...
```

To: rdippold@phones.com (Ron Dippold)
Date: Thu, 24 Apr 97 11:14:44 EST
In-Reply-To: <CMM.342342.rdippold@happy.phones.com>;
from "Ron Dippold" at Apr 24, 97 1:00 am
X-Mailer: ELM [version 2.3 PL11-void1.13]
Mime-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
Content-Length: 10234

Message headers are constructed for you automatically by mail software known as *mail user agents* (MUA) and *mail transport agents* (MTA). In fact, the presence of certain items in the header, such as carbon copies and receipt notification, depend on the sophistication of the mail software itself. These components of an electronic mail system are discussed in detail in a later section. Some header information is intuitive. Other sections require some explanation.

Here's the first line from the previous example:

From nihil@eniac.seas.void.edu Thu Apr 24 08:15:01 1997

This line was added by the MTA on the local system (*sendmail*). It is used as a quick message summary, noting who sent the message and when. Because many mail systems store all of a user's mail messages in a single text file, such summary lines are also used to separate messages within the file. This provides a way to tell the end of one message from the start of the next. For most mail programs, this is the text *From* at the start of a line. This also means that if you try to place a *From* at the start of a line of text in your actual message, your mail program should place a *>* or some other character before it, so that it doesn't falsely indicate the start of a new message.

Flags: 000000000015

The *Flags* field, which is specific to Berkeley mail and *mailx*, was also added by the local mail program. Each message can have several different statuses, such as deleted, unread, and flagged for further attention. This varies with the sophistication of the mail program.

Received: from phones.com (phones.com [229.46.62.22]) by happy.phones.com (8.6.5/QC-BSD-2.1) via ESMTP; id IAA13973 Thu, 24 Apr 1997 08:14:59 -0800 for <rdippold@happy.phones.com>

Each system that receives mail adds its own received header on top of the message. Because this is the first such header in the message, it must indicate the last mail transfer. The machine *happy.phones.com* (where my mail is located) received the message from *phones.com* (our company gateway) on April 24, 1997. The transfer was done using *sendmail* 8.6.5 (although you can't tell from this header that it was *sendmail*), and the protocol used was ESMTP. The intended recipient is listed last. This can change as the message goes through gateways, so it's helpful for tracking mail problems.

Received: from linc.cis.void.edu (root@LINC.CIS.VOID.EDU [230.91.6.8]) by phones.com (8.6.5/QC-main-2.3) via ESMTP;

id IAA14773 Thu, 24 Apr 1997 08:14:56 -0800 for
<rdippold@phones.com>

RFC 822, which documents the standard format for Internet text messages, contains information about the header format.

An alternative to `sendmail` is UUCP. RFC 976 explains the UUCP mail protocol, which is a store-and-forward mechanism. Instead of making a direct connection like a phone circuit, host A makes a temporary connection to send mail to host B, which stores it temporarily and forwards it to host C, the final recipient. UUCP is a pain in the neck, not to mention a serious security hole; avoid it if you can.

Note:

Throughout this chapter, reference will be made to RFCs. RFC stands for Request For Comments and is the means by which the research and development community has documented the standards that form the basis of the Internet. For example, RFC 821 documents the SMTP protocol for sending mail.

The Message Body

The message body is separated from the message header by a single blank line. The message body contains the actual text of the message. Here, you are free to type whatever you want to the recipient.

Signatures

Some email messages conclude with an optional signature. A signature is a brief description of who sent the message, such as full name, telephone and fax numbers, and email address. This is stored in the user's `.sig` file and can be modified with `vi`. Some signatures try to embellish this information with a picture drawn with ASCII characters. It is considered good practice to limit your signature to five lines or less. Most modern mail programs can be configured to automatically append your signature to the end of your message.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Register for EarthWeb's
Million Dollar
Sweepstakes!



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Sending Binary Data

The protocol for sending email over the Internet (SMTP) allows only for the transmission of ASCII text characters. Therefore, binary files, such as audio or video files, are not directly supported. The preferred method for sending binary data is to use a mail program that supports Multipurpose Internet Mail Extensions (MIME). This is discussed in a later section. Before the advent of MIME, a technique used to circumvent this restriction is to encode such data as ASCII text before sending it with a mailer program, such as `elm` or `mailx`. On UNIX systems, use the `uuencode` program to convert a binary file to ASCII text. On the receiving end, use the `uudecode` program to convert the data back to binary.

Mail programs such as `mailx` are also discussed in more detail later in the chapter.

Addressing Remote Systems

To send a message over the Internet, you need to specify a specially formatted Internet address. It is composed of two major sections separated by an @ sign. The part of the address to the left of the @ sign is the Internet account that will receive the mail message. This is usually the login name of the mail recipient. The part of the address to the right of the @ sign is known as the domain name. It uniquely identifies a host on the Internet. All domain names on the Internet comprise the Domain Name System,

which is a hierarchy that divides the Internet into logical groups (domains). The domain is read from right to left and specifies a series of progressively smaller logical domain names. Each part of the domain name is separated with a period. For example, note the following Internet address:

`ccarter@minn.com`

The rightmost portion of the domain, `.com`, indicates that this is a commercial site. The following list shows the most popular domains for the United States:

<code>com</code>	commercial
<code>edu</code>	education
<code>gov</code>	government
<code>mil</code>	military
<code>net</code>	network
<code>org</code>	organization

Outside of the United States, sites can be registered to `.com`, `.net`, and `.org`. In addition, the two-letter ISO country code can also be used. For example, “ca” for Canada, “uk” for the United Kingdom, and so on.

To the left of the highest level domain name (`.edu`, `.org`, and so on) can appear any number of logical subdomains. These are used to specify, in greater detail, the name of the host where the mail recipient can be found. By Internet convention, capitalization in the domain name is ignored. Therefore, the following Internet addresses are equivalent: `ccarter@Minn.com`, `ccarter@MINN.com`, and `ccarter@MINN.COM`. Most modern mail software ignores case in the username portion of the address for consistency. However, this is not a requirement. Therefore, it is considered good practice to preserve case for the username, just in case the recipient’s system is using older mail software.

An older type of addressing scheme is known as a UUCP bang-path address (*bang* is computer jargon for an exclamation point). It is unlikely that you will see an address in this format, though, and is mentioned here for historical reasons. In this type of scheme, you must indicate each system you want the mail to pass through. For example, note the following address for user `katherine`:

`comp01!comp02!comp03!katherine`

This indicates that you want the mail to pass through systems named `comp01`, `comp02`, and `comp03`. After the message has been delivered to `comp03`, it will be delivered to `katherine`.

How Messages Are Routed Over the Internet

Before an Internet address in *username@domain* format can be used for transmission, it must be converted into an IP address. An IP address consists of four numbers, separated by dots, which uniquely identify a host on the Internet. For

example, “128.254.17.7” is an example of an IP address. Translating an Internet address to an IP address is the province of systems on the Internet known as name servers.

When a mail message is sent over the Internet, it is sent as a stream of packets, each containing a portion of the message. Each packet also contains the IP address of the destination. The packets are sent over the Internet using the IP protocol. Specialized networking systems on the Internet, known as routers, examine the IP address in each packet, and route it to the appropriate host. Many factors, such as network traffic volume, on various Internet backbones are taken into consideration to determine the best possible path. In fact, packets from the same mail message may take different routes. All packets are combined in the correct order on the receiving host using the TCP protocol.

Sending Mail to Other Networks

In addition to sending email over the Internet, it is possible to send mail to other networks, such as online services.

Internet Email Gateways

In theory, the Internet is a competitor with all the existing services such as AT&T Mail, CompuServe, and the rest. In practice, it’s a neutral competitor. It’s not some guided, malevolent entity that is trying to do away with any of the other services. Rather, it competes just by its existence; it offers more information and more connectivity than most of the services can ever hope to offer. Smart information services finally realized that this could be put to their advantage. Anyone who cares to can join the Internet, and a service that joins the Internet has advantages over its competitors.

One huge advantage is connectivity. As soon as a mail service adds a computer (known as a *gateway*) that can transfer from its system to the Internet and vice versa, its users can exchange mail with anyone on the service or with anyone on the Internet. That’s a lot of people. So, many services are now offering some sort of mail gateway. Even Prodigy, which was somewhat late to grasp the possibilities, has one now.

Instead of GEnie needing to install a special gateway to talk to Prodigy, and one to CompuServe, and one to SprintMail, and one to BubbaNet, it can set up and maintain just one gateway to the Internet, through which everything flows. Given the glacial speed with which most of the online services implement upgrades like this, requiring only a single gateway is a good thing.

So now anyone can send email anywhere! Well, not exactly.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Addressing Issues

It turns out that the services that connect to the Internet keep their same old account names and horrible mail systems. CompuServe's octal account addresses are as much an anachronism as punch cards, but because of the company's current investment, it isn't going to change them. And you can't just send a mail message to a CompuServe account using an Internet-style address. A CompuServe ID looks something like this:

112233,44

In Internet addressing, a comma separates members of a list so you can't use the comma in the CompuServe address. There's a way around that (use a period instead of a comma), but you have to know that in advance. Someone trying to send mail to a system has to deal with those quirks. Hence this section, which details the translation that has to be done between the major networks.

Again, an Internet email address looks something like this:

`user@machine.site.domain`

Any address to a mail gateway is going to be some variation (minor or major) on this theme.

X.400 Addressing

The Internet uses what is formally known as RFC-822 addressing. Many large commercial services specializing in electronic mail use something known as an X.400 gateway to talk to the Internet. Those addresses look something like this:

/A=value/B=value/C=value

This style is usable from the Internet because RFC 822 allows slashes and equals signs. In fact, there's the opposite problem: RFC 822 allows many characters to be used in addressing that cause an X.400 gateway to go into convulsions, including the @ sign. Because this appears in all Internet-style mail addresses, there's an obvious problem.

Whenever the Internet address has a special character, you need to use the following translation table:

Internet	X.400
@	(a)
%	(p)
!	(b)
"	(q)
_	(u)
((l)
)	(r)

- For any other special character, such as #, substitute (xxx), where xxx is the three-digit decimal ASCII code for the character. For #, you would use (035).

For example, to convert the Internet address

`oldvax!Mutt#Jeff@cartoon.com`

into something that can be sent from an X.400 service such as MCI Mail, you need to turn it into this:

`oldvax(b)Mutt(035)Jeff(a)cartoon.com`

Gateway Translation Specifics

Using the following instructions should be fairly easy. To send mail to CompuServe from an Internet mail account, see the translation instructions in the "CompuServe" section later in this chapter.

Parts of the address that you have to replace with appropriate information are given in italics. For instance, with

userid@aol.com

you need to replace *userid* with the recipient's account name or number. *domain* is the part of the Internet address after the @.

Note:

The ! is replaced with (b) because computer users like short names, and refer to an exclamation point as a *bang*.

If you are sending mail from one service to another through the Internet, for example from WWIVNet to CompuServe, you have to do two translations. First, check the "CompuServe" section and see how to translate the ID "From Internet." Then check the "WWIVNet" section and see how to translate that address "To Internet." If you do this from one strange network to another, the name may be a crawling horror, but at least it should be possible.

America Online

America Online (AOL) is a major commercial information system that recently joined the Internet (although it has had Internet email for a while). Its Internet email is seamless from an Internet point of view.

From Internet: America Online looks just like any other normal Internet site.

userid@aol.com

Example: *jjones@aol.com*

To Internet: There's no need to do anything special; just use the regular Internet format.

userid@domain

Example: *bsmith@wubba.edu*

To Others: America Online lets you use special abbreviated domains for mail to AppleLink, CompuServe, or GENie. Send your mail to *userid@applelink*, *userid@cis*, or *userid@genie*, respectively.

Example: *11111.2222@cis*

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#) [Table of Contents](#) [Next](#)

AT&T Mail

From Internet: Use standard Internet addressing:

`userid@attmail.com`

To Internet: Use the following. Note the backward order here—this is the old bang-path type addressing. Oh well.

Example: `internet!wubba.edu!bsmith.`

CompuServe

CompuServe is a large commercial system. Although it finally bowed to the pressures of Internet standardization, CompuServe still allows the format shown in the next section.

From Internet: Use standard Internet addressing with one difference: CompuServe IDs are in the form `77777,7777`. Because the Internet dislikes commas in addresses, you need to change the comma to a period:

`77777.7777@compuserve.com`

Example: `12345.677@compuserve.com`

To Internet: You need to add a prefix to the standard Internet addressing:

>INTERNET;userid@domain

Example: >INTERNET;bsmith@wubba.edu

EasyLink

This is a set of commercial Internet services from AT&T.

For more information on AT&T's EasyLink, you can contact them at <http://www.att.com/easycommerce/easylink/mail.html>.

FidoNet

FidoNet is a large international BBS network—sort of the Internet for the BBSing crowd. It's not as fast as the Internet, but access is usually very cheap, and chances are there's a FidoNet BBS in your area.

Because it's run over phone lines, the BBS operators will rack up long-distance charges for any mail transferred, so please don't send large messages to FidoNet sites. Many sites even chop your messages to 8,000 or 16,000 bytes, so much of your message won't get through.

From Internet: First, you need to know the network address of the BBS your recipient is on. It will be in a form such as Z;N/F.P. Then send the mail to the following address:

userid@pP.fF.nN.zZ.fidonet.org

If the network address of the BBS doesn't have a P component, leave the pP. part out of the address. For the *userid* replace any nonalphanumeric characters (such as spaces) with periods (.).

Example: Jim_Jones@p4.f3.n2.z1.fidonet.org

To Internet: Use standard Internet addressing with a suffix:

userid@userid ON gateway

The *gateway* is a special FidoNet site that acts as a gateway to the Internet. You can use 1:1/31 unless you find a better one.

Example: bsmith@wubba.edu ON 1:1/31

Prodigy

Prodigy is a large commercial service, Prodigy Information Services (jointly developed by Sears and IBM).

From Internet: Use standard Internet addressing:

-
`domain@prodigy.com`

Example: `jone45a@prodigy.com`

To Internet: When online, Jump to ABOUT MAIL MANAGER and proceed from there.

SprintMail

Hmm...AT&T and MCI have commercial mail services. Sprint has to have one, if only for the principle of the matter. Actually, to be fair, Sprint has always been one of the more network-oriented phone companies. You may have used their Telenet network.

From Internet: Use this addressing:

`/G=first/S=last/O=organization/ADMD=TELEMAIL/C=US/@sprint.com`

first and *last* are the recipient's first and last names, of course, and *organization* is the recipient's SprintMail organization name.

Example: `/G=Chris/S=Smith/O=FooInc/ADMD=TELEMAIL/C=US/@sprint.com`

To Internet: Use this addressing:

`C;USA,A;TELEMAIL,P;INTERNET,"RFC-822":<userid(a)domain>) DEL`

Again, refer to the section "X.400 Addressing" earlier in the chapter, to see how to handle nonstandard characters in addresses.

Example: `C;USA,A;TELEMAIL,P;INTERNET,"RFC-822":<bsmith(a)wubba.edu>) DEL`.

Other Gateways

There are other gateways, and more are sure to appear. Most services offering this type of gateway should have at least some clue of how the address translation needs to be done—ask the service if you need to know.

Finding Addresses

Many places on the World Wide Web keep track of people's names and their corresponding email addresses. Many sites also provide a nice "front end" to these databases, allowing you to search for someone's email address.

One such site is <http://www.four11.com>. It allows you to narrow your search based on geographic regions, such as country and state. It also has a handy "smart name" feature, that expands a search for certain variations in a name (for instance, Robert = Bob). It currently contains more than 6.5 million listings. Other worthwhile sites include <http://www.iaf.net> and www.bigfoot.com. Finally, be sure to check out <http://www.starthere>.

com/index.html. It has links and descriptions of many sites, which, in turn, do the actual email address searches on the Web.

To find someone in the communications field, try RPI's address server. Send mail to Internet address `comserve@vm.its.rpi.edu` with `help` as the body of the message.

UNINETT of Norway maintains an X.500 address registry service. Send mail to Internet address `directory@uninett.no` with `help` as the body of the message.

PSI runs an X.500 service at Internet address `whitepages@wp.psi.com` with `help` as the message body.

Usenet Address Server

MIT keeps track of every person who has ever posted an article to Usenet since the late 1980s (many Usenet readers would be shocked to know this). This includes those from other networks who use a news gateway. If the person you are looking for has posted an article to Usenet since then, he or she might be in this database.

Send mail to the Internet address `mail-server@rtfm.mit.edu`. In the body of the message, put this:

```
send usenet-addresses/key1 key2 key...
```

The keys should include all the words you think might appear in the address, usually parts of the person's name. In many cases, you will use only `key1`. The keys are case insensitive.

You can try the following:

```
send usenet-addresses/dippold
```

to return several entries. The server returns only 40 matches, so if your keys are overly general (Smith) you will need to give more keys, such as a first name, to narrow the search.

You can do several searches at once by placing several `send usenet-addresses/keys` lines in the message.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#) [Table of Contents](#) [Next](#)

Mail Programs

Three main components are involved in sending mail. First there's the *link level transport layer*. Directly above the transport layer is the mail transport agent (MTA). This layer is responsible for the movement and delivery of mail messages. An MTA has several components, including routing mechanisms, a local delivery agent, and a remote delivery agent. The MTA for most UNIX systems is the `sendmail` program. An MTA that takes liberties in modifying the contents of a message is known as a hostile MTA. Finally, the mail user agent (MUA) provides the user interface. It allows you to read, send, and otherwise manipulate mail messages. This is what people usually mean when they talk about a *mail program*. This is covered in more detail in Chapter 30, "Mail Administration."

There are many mail programs from which to choose. The next section covers the elements common to them all.

Using Mail Programs

As mentioned earlier, there are many mail programs, each with its own quirks. But they try to accomplish the same task and tend to present the messages in a similar format.

To learn more about the many common UNIX mail programs available and their various features, see Chapter 30, “Mail Administration.”

SMTP—Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (SMTP), or some variation of it (such as Extended SMTP) is used by computers on the Internet that handle mail to transfer messages from one machine to another. It’s a one-way protocol—the SMTP client contacts the SMTP server and gives it a mail message.

■ Most mail client programs support SMTP for sending outgoing mail, simply because it’s easy to implement. Few mail clients support SMTP for incoming mail because normally your mail computer can’t contact your personal computer at will to give it mail. It’s possible if your personal computer happens to be permanently networked to the mail computer via Ethernet, for instance, or if your mail computer knows how to use a modem to call your personal computer, but in most cases this isn’t done.

POP3 (Post Office Protocol 3)

The standard protocol used by most mail clients to retrieve mail from a remote system is the post office protocol POP3. This protocol enables your mail client to grab new messages, delete messages, and do other things necessary for reading your incoming mail. POP only requires a rather “stupid” mail server in the sense that your mail client needs to have most of the intelligence needed for managing mail. It’s a simple protocol and is offered by most mail clients.

POP3 is somewhat insecure in that your mail client needs to send your account name and password every time it calls. The more you do this, the greater the chance that someone with a network snooper might get both. (We’re not trying to scare you, but it’s possible.) An extension known as APOP uses a secure algorithm known as MD5 to encrypt your password for each session.

Finally, note that standard POP3 has no way to send mail back to the mail server. There is an optional extension to POP3 known as XTND XMIT that allows this, but both the client and the server have to support it. Generally, a mail client uses SMTP to send messages and POP3 to retrieve them.

Mailing Lists

With email, you can carry on a conversation with another person. But why not with three others? Easy enough—just use the Cc header or specify multiple recipients on the To header. What about hundreds? Well, that might be tough. But what if there were enough interest in something (such as the band REM) that someone agreed to serve as a central dispatch point? All mail to that account would be sent to all other people in the discussion. This is known as a mailing list, and they are quite popular. The REM list mentioned has more than 800 subscribers.

The first thing you have to realize is that when you join (subscribe to) a mailing list, all of a sudden you’re going to have a lot of messages in your mailbox. Can you handle the extra time it’s going to take to read these new messages? Are you paying

for mail? Many people don't comprehend exactly what they're getting into when they sign up for a mailing list. Remember to save the instructions on how to unsubscribe from the group, so you don't send your unsubscribe request to all the members of the group and feel like a fool.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)

ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Finding Lists

First you need to find some lists. Every month several informative postings are made to the Usenet group `news.answers`, describing hundreds of mailing lists and how to subscribe to them. For example, Stephanie da Silva posts "Publicly Accessible Mailing Lists." If you have Usenet access, `news.answers` is your best bet. Perhaps some of the people you correspond with know of some lists.

If neither approach works, you can use the `uga.cc.uga.edu` mail server described in the following paragraph.

LISTSERVs are nifty automatic programs that handle much of the drudgery involved in maintaining a mailing list. There are several such `LISTSERVs`, but you need only one to get started. We suggest you use `listserv@uga.cc.uga.edu`. Others include `listserv@mizzoul.missouri.edu`, `listserv@jhvm.bitnet`, `listserv@vm1.nodak.edu`, `listserv@ucsd.edu`, `listserv@unl.edu`, `LISTSERV@PSUVM.PSU.EDU`, and `LISTSERV@SJSUVM1.SJSU.EDU`.

Commands to these sites are simple. You can give a new instruction on each line of the body if you want, although generally most of your requests consist of a single line.

To start with, try sending mail to `listserv@uga.cc.uga.edu` with only the text `help` in the body of the message (the subject doesn't matter). You should get back a

list of valid commands. Probably the most interesting for you will be `listserv` `refcard`, which returns a reference card, and `lists global`, which returns a big list of all known mailing lists on many LISTSERVERs—it's more than 300,000 bytes. You're in mailing list heaven! If that's too big, try just `lists`.

Joining and Dropping

If your mailing list is managed by a LISTSERVER, joining a list is easy. Send mail to `listserv@domain`, with the following message line:

```
SUB LISTNAME Firstname Lastname
```

LISTNAME is the name of the list, such as HUMOR. *Firstname* and *Lastname* are your first and last names.

To sign off the list, use this:

```
SIGNOFF LISTNAME
```

Do not send your unsubscribe request to the mailing list itself. You'll just irritate people, and they'll laugh at you.

If you would rather get one mailing a day—consisting of all the posts to the mailing list in one big chunk—rather than receiving dozens of little messages during the day, use this:

```
SET LISTNAME DIGEST
```

To get each piece as it is sent, use this:

```
SET LISTNAME MAIL
```

There are other commands—the `help` command should get them for you.

If the mailing list isn't being handled by a LISTSERVER, you're at the mercy of the mailing list maintainer as to how subscriptions are handled.

— Generally, the address to send messages to for a mailing list is this:

```
listname@domain
```

The address to send messages to for subscribing and unsubscribing is this:

```
listname-request@domain
```

However, you can't always count on these. In this case, you have to rely on the instructions for the specific list, which you need to get from the maintainer or a friend.

Automatic Mail Sorting

We're not going to go into too much detail about mail sorting because it's a rather complex subject, but sometimes you get to the point where you can't treat your incoming mail file as a single entity.

We get literally hundreds of messages a day, and we would go insane if we didn't use a program known as a mail filter. These look at your incoming mail, and based on criteria you set regarding the contents of header items or message text, they sort the mail into several mailboxes before you even see them.

For instance, Ron subscribes to several mailing lists. He routes messages from each of these into a separate mailbox for reading at his leisure. He has Usenet voting ballots arriving all the time—these go into a special voting file for processing by the voting software. Everything that's left goes into a general mailbox for normal reading.

Actually, mail filters can often do more than this. You can use them to selectively forward mail to other users, or to send automatic responses to certain messages. You can even have them send only a single informational message to anyone who mails you while you're on vacation, no matter how many messages they send you during that time.

The drawback to a filter program is that it can be tough to set up, unless you're using a mail client with the capability built in (for example, Eudora). Carefully check your configuration files to make sure that you aren't accidentally dropping messages on the floor!

procmail

`procmail` is probably the most popular of the mail filters. You have quite a bit of control over your messages and can even pass them through other programs, such as a formatter, before they are saved. It can execute other programs on demand and can be used to run simple mailing lists or mail servers. It's been extensively tested, it is stable, and it is fast. Be careful, though, that you don't accidentally tell it to send some of your mail into a black hole.

You can get the latest version by anonymous ftp to <ftp://ftp.informatik.rwth-aachen.de> as `/pub/packages/procmail`.

deliver

Although `procmail` is the king of the hill for mail filter programs, we personally like `deliver`. You write shell scripts to handle all incoming messages. This requires more work on your part, usually, than would `procmail`, but it's very clean, almost infinitely flexible, and limits what you can do with your email only to how well you can program scripts. The speed shouldn't be too much of a concern on that fast machine of yours.

We found `deliver` by anonymous ftp at `sunsite.unc.edu` as <ftp://sunsite.unc.edu/pub/Linux/distributions/slackware/contrib/deliver.tgz>.

mailagent

mailagent is another well-known email filter. This one is written in the Perl language, which again means that you can do anything with your email by extending mailagent yourself (if you know Perl). It comes with quite a few built-in features. We suggest this if you know Perl. Anonymous ftp to <ftp://ftp.foretime.co.jp> and get /pub/network/mail/mailagent.

elm

elm comes with a support program named filter, which does mail filtering.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Register for EarthWeb's
Million Dollar
Sweepstakes!



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Usenet

As described in the introduction, Usenet is the world's largest electronic discussion forum. One of the most popular features of the Internet, Usenet allows people all over the world to discuss topics and exchange ideas on a wide variety of subjects.

One way to describe Usenet is in terms of email. Think of your mailbox, with all its new and old messages. Imagine what it might be like if everyone on the Internet could read that mailbox, enter new messages, and leave replies. Now imagine having 20,000 mailboxes. This is analogous to how Usenet works.

Usenet is a huge public messaging system. It is divided into thousands of discussions of different subjects—each separate piece is known as a newsgroup, or group. When someone enters a message while “in” a group, that message goes to all other Usenet sites in the world, and people reading that same group can read the message and reply to it if they want. Generally, dozens of different conversations (also known as *threads*) are going on in any particular group—each is distinguished by a subject name, much like the Subject in a mail message. Thousands of new messages are posted each day.

Usenet is commonly thought of as being the same thing as the Internet, but they're not the same thing. The Internet is an international network of computers tied together via dedicated lines. Usenet is just one service that uses the Internet. If you're familiar with

bulletin board systems (BBSes), you might think of the Internet as the BBS hardware, and Usenet as the message bases.

Not all computers on the Internet have Usenet (it can take a lot of space!). Not all computers carrying Usenet groups are on the Internet—like email, some systems call Internet systems to exchange Usenet messages.

USENET Is Usenet Is NetNews

Frankly, capitalization standards on Internet are quite relaxed. You can call it USENET, you can call it Usenet, you can call it UseNet. People will know what you mean. If you call it UsEnEt, people will start edging nervously for the exits. You can even refer to it by the ancient moniker Netnews (or NetNews). People will understand what you mean.

Usenet Messages

Usenet messages are much like the Internet mail messages described earlier in this chapter. They consist of a header, which has information about the message, and the body, which has the actual message. They even use the same format as mail messages, and most of the same headers are valid. There are a few new ones, which are covered in the following sections.

The Usenet Distribution Model

Every computer that gets Usenet keeps a database of Usenet messages. When a new message is entered, it is sent to neighboring Usenet sites using NNTP (Network News Transfer Protocol). These distribute the post to other sites, until it is on every machine on Usenet. Various mechanisms prevent a message from showing up on the same machine more than once, which we don't need to get into here. Only occasionally does a broken machine (usually a FidoNet gateway) regurgitate old articles back onto the Net.

- We said that all machines get all posts. Well, sort of—because Usenet is so huge, many sites only carry a subset of all the available groups. A site won't get posts for groups it doesn't care about, or if it does, it won't keep them. In addition, there's something called a Distribution header that you can put in your message to try to restrict its distribution to a geographical area, such as San Diego. This is useful for messages that affect only San Diego.

Newsgroup Names

Newsgroups are named like this:

```
comp.sys.ibm.pc.games.action
```

This is a hierarchy reading down from left to right. Reading the group name, you have a computer group for computer systems from IBM, the PCs to be exact. You're talking about games for those systems, more specifically action games.

Here's another one:

```
talk.politics.guns
```

You have a group for talk about politics, more specifically gun control. We'll talk more about these hierarchies later.

The newsgroup with which your post is associated is given in the header of the message, in the Newsgroups item. It looks like this:

```
Newsgroups: news.announce.newgroups
```

Unlike traditional bulletin board systems, each post can go in multiple groups. If we do this:

```
Newsgroups: alt.usenet.future,news.groups
```

my post appears in both groups. This is known as *crossposting*. Although you should know it is possible, you shouldn't actually do this until you've looked around a while, because frivolous crossposting is frowned on.

In fact, another header can be used to send any replies back to a specific group. For instance, you might make a wide informational post to several groups but specify that the discussion (if any) should be only in a single group. This is the Followup-To header. Together, the headers look like this:

```
Newsgroups: rec.arts.comics.misc,rec.arts.comics.strips,  
rec.arts.comics.animation  
Followup-To: rec.arts.comics.animation
```

Remember from the email header discussion that one header can spread over several lines, as long as succeeding lines are indented. That's what you did to split Newsgroups over two lines. All replies to the post go to rec.arts.comics.animation, unless the person replying overrides that.

Crossposting can be abused, but more on that later.

Threads

An original post and all the replies to it are considered to be a single "thread" of conversation. This can actually look more like a Christmas tree than a straight line, as there are replies to replies, and replies to those replies, which branch off until each sub-branch dies of finality or boredom.

Each Usenet message has a Subject associated with it that is supposed to summarize the contents of the message (although this is often not the case). One way to track a thread is to note the message subjects, which those who reply to the post are supposed to preserve until the discussion wanders too far from the original subject. The only way to fully keep track of threads is to use a threaded newsreader, which is discussed

in the next section.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Newsreaders

The first item of business is which program you will use to read Usenet. Your choice of these programs (known as newsreaders) can hugely impact how you read the Net, how much information you get out of it, and how much garbage you have to sludge through.

rn (readnews)

rn is free, so there's a good chance the system you use to read mail has it, and a good chance that it will be offered to you as your default newsreader. Avoid using it if you can.

Back when rn was first written, one person could read every single message posted to Usenet and still have time for a life. It reflects those simpler times—its default is to dive in and show you all the messages in the group, one at a time.

This sounds reasonable, but it's a fact that the majority of the posts on most newsgroups you read are of no interest to you. There will come a time when you no longer want to slog through every post on the group and become choosy about which posts you read. rn does not let you do this easily. Because popular groups can get more than 100 messages a day, rn's preference for showing you every single message really wastes your time.

Message Overview and Threading

Just how much of your time `rn` wastes is evident the first time you run another news program that first gives you an overview of the group. It provides you with a summary line for each post, just as a mail program does—it gives you the poster's name, the subject, and possibly the message size. Scroll through the pages of summaries and choose which posts look interesting. When you're done choosing, read the posts you've selected.

Now we'll add another concept to that—the newsreader should keep track of which posts are related to each other and group them, so that you can select or ignore whole groups of posts at once. It can do this by noticing the threads and subject names mentioned before.

These two changes account for an almost unbelievable difference in speed between a good threaded newsreader and something like `rn`. When you've gotten good at determining which threads look promising and which don't, you can read Usenet literally 100 times faster than you could before. We'll recommend some right after this....

Kill Files

In a group where more than half the discussion is about something you don't care about (for instance, a particular author on a fantasy group), having the newsreader kill all articles relating to that author can save you time and make you less likely to lose valuable articles in the crush.

There's also the opposite of a kill file. If you know you want to read every posting on a particular subject or from a particular person, a selection file lets you have the - newsreader automatically mark them for reading. This isn't quite as common as the kill file.

NN (No News)

NN is fast, flexible, and configurable; has nice kill and selection options; sorts messages in several ways, and offers several ways to manage the old messages. It even has its own group, `news.software.nn`. This is definitely worth a look.

Other UNIX Readers

Other UNIX readers that are worth looking at (if your site offers them) are TRN, STRN, and TIN. TIN happens to have the largest number of UNIX readers at this time. They meet or exceed the criteria given. You can also read the Usenet group `news.software.readers` for the latest information.

Netscape

The Netscape Web browser provides facilities for tracking, replying to, and initiating user group postings. To access a particular newsgroup, invoke the File and Open

Location menu items, and enter the URL for the newsgroup. The URL for a newsgroup consists of the word `news`, followed by a colon (`:`) and the name of the group. For example, to access the Oracle database newsgroup, enter `news:comp.databases.oracle`. You can even use an asterisk (`*`) to display all items at a particular level in the hierarchy. For example, the URL `news:comp.databases.*` would list all database discussion groups.

When you have opened a particular group, a set of command buttons that perform some common Usenet functions appears. For example, buttons are available to subscribe/unsubscribe to groups, as well as initiate and receive postings.

When you subscribe to a newsgroup, the entry is maintained for future use by the Netscape software. The list of all your newsgroups can be accessed by selecting the Directory and Go To Newsgroups menu options.

Other Readers

For other systems, you should be reading the Usenet groups `comp.os.msdos.mail-news` and `news.software.readers`. There are, most likely, programs out there for your system. For instance, there's Trumpet for DOS and WinTrumpet for Windows. If you have a complete TCP/IP package, you might want to see whether it includes a mail reader (other than `rn`).

Offline Readers

Just as you can use a mail client to do your mail processing offline, you can use an offline reader to do your Usenet processing offline. This is useful if you're paying by the minute for your connect time. See the group `alt.usenet.offline-reader` for help with these.

Finding Your Groups

You can participate in literally thousands of newsgroups. This section helps you find the groups in which you are interested.

The Hierarchies

As mentioned earlier, group names are arranged in hierarchies from left to right. The left item is known as the top-level of the hierarchy. In the case of a group such as this:

`alt.tv.animaniacs`

it is said that the group is “in the `alt` hierarchy” (or “`alt.` hierarchy”). The Net is organized into eight major hierarchies, one anarchic hierarchy, and a bunch of smaller, less important hierarchies.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



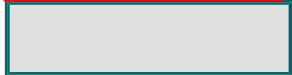
Brief

Full

Advanced

Search

Search Tips



To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

The Big Eight Hierarchies

The big eight hierarchies are the following:

- | | |
|-------|--|
| comp. | Computer topics—This ranges from programming to hardware to peripherals to folklore. Most popular computer systems and operating systems have their own set of groups here. |
| misc. | Miscellaneous—When nobody can figure out where to put a new group, it often ends up under misc.. For example, the misc.jobs groups don't clearly belong in any of the other six hierarchies, so they go under misc.. |
| news. | The business of USENET—This is where people talk about Usenet administration, propose new groups, and argue about when Usenet is going to die of its own excesses. |
| rec. | Recreational topics—This is where most of the hobbyist stuff, such as rec.crafts.jewelry, goes. It also contains artistic and music discussions, crafts, and more in that vein. |

<code>sci.</code>	Science—This is where the math and physics types hang out. Medical, too, such as <code>sci.med.radiology</code> .
<code>soc.</code>	Social topics—This is a grab bag of many cultural groups for different regions, such as <code>soc.culture.chile</code> , social research groups, religious discussion groups, and alternative lifestyle groups. It's something of a milder version of the talk hierarchy.
<code>talk.</code>	Heated debate—Incredibly vicious personal attacks by people (most of whom seemingly haven't even heard of the concept of "critical thinking") that go on interminably about all the things you would expect—politics and religion. See <code>talk.politics.mideast</code> , for example. No debate here is ever really ended.
<code>humanities.</code>	Literature and fine arts—This hierarchy contains a wealth of discussion regarding music, philosophy, and fine art. For example, see <code>humanities.lit.author.shakespeare</code> .

These hierarchies are sometimes known as Usenet proper and are considered by many news administrators to be the only "real" hierarchies. For a new group to be created in any of these eight hierarchies, it has to go through a group interest polling procedure that discourages overly frivolous group creation. More on this later.

Where Do I Go?

Back to your original question—how do you know where to go for a particular subject? There are several ways.

First, your newsreader might be smart enough to find part of a group name. If I tell NN to go to group `beer`, for instance, it asks me whether I mean `alt.beer` or `rec.food.drink.beer`. In this way, I just found two groups, and if I look for `brewing`, I'll find more.

Dave Lawrence posts "List of Active Newsgroups" and "Alternative Newsgroup Hierarchies" to `news.groups` and `news.answers`. This is the mother lode—all "official" groups (although with `alt`. "official" doesn't mean much), each with a short description. Get it if you can.

Your newsreader probably has a way to show you a list of all groups. This might take some digging to find. (It's `:show groups all` in NN.)

Next, you can look through a file your newsreader leaves in your home directory, named `.newsrc` or something similar. This is just a list of group names, but they might give you some hints.

You can always ask for help on the group `news.groups.questions`, which is

just for this sort of question.

Signature Files

Most newsreaders enable you to attach a signature to every post you make. It takes the contents of the file `.signature` in your home directory and attaches it to the end of the post. This is intended to be used for identification purposes—perhaps your name and place of work if it's not obvious from the header. Or sometimes it's used for disclaimers.

By far, the most common use is as a small personality statement—this usually involves your name, Internet address, a favorite quote, and maybe a small picture drawn with text characters.

Read the FAQ!

One day, the people of Usenet noted that new users all tended to ask the same few questions. They decided to create a Frequently Asked Questions List (FAQ—the L just didn't sound good), which would present the answers to these questions and prevent them from being asked over and over.

That worked pretty well, and now many groups have FAQs. This means that if you pop up on a group and ask a question that is in the FAQ, you're going to get some negative responses ("Read the FAQing FAQ!") If you enter a new group for the purpose of asking a question, make sure that you look for a post with "FAQ" in the title. If you find any, read them first. Your answers (and answers to questions you hadn't even thought of yet) may be in there.

If you're looking for information in general, most FAQs are posted to `news.answers`. You can go there and browse all the beautiful FAQs.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Brief](#) [Full](#)

[Advanced](#)

[Search](#)

[Search Tips](#)

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Talk

Talk is a program that allows two users to communicate in real-time using a split screen interface. A user “talks” to another user by typing text in one area of the split screen and “listens” as the other user’s text appears in another area of the screen. It can be used for users on the same system, or over a TCP/IP network.

Before initiating a talk session, you need the other person’s address. If the user is connected to the same local machine as you, the login name will suffice.

Next, you need to make sure that the other user is logged in. You can find out with the `finger` command. For example:

```
$ finger userid
leibniz 24: finger trimblef
Login name: trimblef                In real life: Frederick Trimble
Directory: /users/leibniz/INFO780-543/trimblef  Shell: /bin/csh
On since Apr 28 00:21:37 on pty/ttys0 from ts2.noc.drexel.e
```

No Plan.

\$

In the preceding example, the `finger` command indicates that user `trimblef` is logged on to the system on pseudo-terminal `pty/ttys0`. The `finger` command can also determine whether a remote user is logged in by specifying a remote address. For example:

```
finger userid@domain
```

After you verify that the user with whom you want to speak is logged on, he must agree to talk with you. To initiate a talk session, first issue the `talk` command:

```
talk userid@domain
```

The talk initiator's screen clears, and the talk header appears at the top of the screen:

```
[Waiting for connection...]
```

On the other screen, the following text appears:

```
talk: connection requested by username@host
talk: respond with: ntalk username@host
```

After the user responds with the appropriate message, the connection is established. Everything typed at this point appears on the other terminal, until the connection is terminated. The talk session is terminated when one of the users presses Control+C.

In certain situations, receiving a talk connect request can be disrupting. You can use the following command to disable any such request from a remote user:

```
mesg n
```

To enable such requests, use the `mesg` command with the `y` option:

```
- mesg y
```

To see the current status of your talk request mode, use the `mesg` command with no options.

The `talk` command is based on a set of protocols that allow communication to take place. There are two protocols for the `talk` command: One is based on version 4.2 BSD UNIX, and the other on version 4.3 BSD UNIX. Unfortunately, these versions are not compatible. Therefore, you cannot establish a talk session between UNIX systems whose `talk` command is based on different versions of the protocol.

Another variation of the `talk` command is the `ytalk` command. The most interesting feature of `ytalk` is that it allows more than two users to partake in a conversation. In addition, it supports both versions of talk protocols. Therefore, the `ytalk` command can establish a connection with either version of the `talk` command.

To establish a `ytalk` session with multiple users, type the address of each user on the command line. For example:

```
ytalk mary@gwyned.edu fred@drexel.edu katherine@nova.edu
```

The `ytalk` command then splits the screen into several panes. Each screen is labeled with the corresponding user, so you always know who is typing.

If you need assistance with any `ytalk` options, simply press the ESC key. A small menu of `ytalk` commands appears as follows:

```
#####
# a) add a new user to session                               #
# b) delete a user from session                             #
# c) output a user to a file                                #
# Your choice:                                             #
#####
```

Tip:

Because typing is slow compared to real conversation, it can be annoying watching the other party backspacing over misspelled words. If you feel the other party should be able to figure out the intention of the misspelled word, it is considered acceptable to continue typing after a spelling

mistake.

Also, it is not uncommon for more experienced users to abbreviate commonly used phrases. Here is a list of abbreviations that you may encounter:

BCNU	Be seeing you
BRB	Be right back
BTW	By the way
BYE	Good-bye
CU	See you
CUL	See you later
FYI	for your information
FWIW	For what it's worth
GA	Go ahead and type
IMHO	In my humble opinion
IMO	In my opinion
JAM	Just a minute
O	Over
OO	Over & out
OBTW	Oh, by the way
ROTFL	Rolling on the floor laughing
R U THERE	Are you there
SEC...	Wait a second
WRT	With respect to

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Internet Relay Chat (IRC)

Each day, thousands of people worldwide hold “keyboard conversations” using Internet Relay Chat (IRC). Like the `ytalk` facility, IRC allows multiple people to converse at the same time. When it is your turn to type, the characters appear on all other workstations that are logged in to the same channel.

Basic IRC Structure

Note:

During the attempted Communist coup in Russia in 1993, an IRC channel was set up to relay eyewitness accounts of the event. IRC channels have also been set up during other natural disasters, such as earthquakes and hurricanes.

IRC uses a client-server model. The IRC “universe” consists of hundreds of channels with names such as `#initgame`. Users join (using their client software) in a channel that interests them and are then in conversation with everyone else who is on that same channel. You can talk with everyone or direct your comments to certain individuals. This is a flexible format that allows something as free-form as a general babble to many pairs of private conversations to a game of IRC Jeopardy, which plays much like the TV show. Some channels are private.

In addition, IRC users have their own nicknames and become quite attached to them

(because your reputation goes with your nickname, this is understandable).

Getting IRC Clients

Before you can do anything, you need an IRC client. You need to grab the source code appropriate for your machine and compile it.

You can get the UNIX IRC client by pointing your Web browser to <ftp://cs-ftp.bu.edu>. The software is located in the `irc/clients` directory. Look to see which file the symbolic link `CURRENT` points to—it will be linked to the latest UNIX source code for `ircII`.

A PC client running under MS-DOS, OS/2, or Windows can anonymous `ftp` to <ftp://cs-ftp.bu.edu> and look under `/irc/clients/pc`. You have your choice of several for each operating system. MIRC is now the most popular client for the Windows environment.

A Mac client can also anonymous `ftp` to <ftp://cs-ftp.bu.edu> and look under `/irc/clients/macintosh`. Grab the latest version of Homer you find there.

Connecting to a Server

After you have your client, you need to figure out which IRC server you will be talking to. Anonymous `ftp` to `cs.bu.edu` and look under `/irc/support`. There should be a file named `servers.950301` (the last number is the date, so that part will change). Grab this and look for a server that's close to you.

Then tell your client to connect to this server. With luck, it'll talk back to you, and you'll be in the world of IRC.

Choosing Channels

After you get on an IRC server, all commands start with a `/`.

`/help` gives you a list of commands. To get the new user help, do `/help intro` and then `/help newuser`.

`/list` shows all the current IRC channels. It looks something like this, except that there will be many more channels:

```
*** Channel      Users  Topic
*** #wubba       3      Wherefore the wubba?
*** #hoffa       5      i know where the body is
*** #litldog     2      where oh where has he gone
```

`/names` might be more interesting. It shows who's logged on each channel and whether it's a private or public channel:

```
Pub: #wubba      @wubba jblow jdoe
```

```
Prv: *      marla donald ivana bill hillary
Pub: #litldog  @yakko dot
```

Then use `/join channel` to participate on *channel*. Here you might do a `/join #wubba`.

`/nick nickname` enables you to change to a new nickname in case your old one is too stodgy.

`/msg nickname message` enables you to send a private *message* to *nickname*. Use the `/query nickname` to enter a private conversation with *nickname*. Use `/query` to exit it.

If you get ambitious and create a channel (using `/join` on a nonexistent channel creates it), be sure to look at the `/mode` command, which lets you determine the behavior of the channel.

Need Help?

`/join #Twilight_zone` is where IRC operators often hang out, and some are willing to help. Just ask your question—don't announce that you need to ask a question first.

Bad Moves

Don't use someone else's nickname if you can help it—people are very protective about them.

Never type anything that someone asks you to type if you aren't sure what it does. You might find that you've just given someone else control of your client!

Don't abuse the telnet server. If you're going to IRC a lot, get your own client.

Further Info

More information on IRC can be found via anonymous ftp on <ftp://cs-ftp.bu.edu> in the `/irc/support` directory. IRC also has several `alt.` groups dedicated to it: `alt.irc.corruption`, `alt.irc.ircii`, `alt.irc.lamers`, `alt.irc.operators`, `alt.irc.questions`, `alt.irc.recovery`, and `alt.irc.undernet`.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Multimedia

Multimedia is defined as the presentation of information in which more than one medium is used at a time. Using animation and sound in addition to ordinary text is such an example. By using more than one medium, multimedia enhances our ability to communicate and understand one another. The advent of powerful desktop computers equipped with high-resolution color monitors and stereo sound has increased the demand.

Internet Infrastructure

Delivering multimedia to the desktop over the Internet presents several obstacles. First, the Internet and its supporting protocols were designed to transmit 7-bit ASCII text to support email and file transfer. Second, the original NSFNET was made up of 56KB data communication lines. (The Internet backbone has been upgraded in recent years with higher network speeds.) Although this was sufficient for its original purpose of supporting email and file transfer, it is not adequate for supporting the growing demand for multimedia.

Files containing multimedia data require large amounts of disk space. When such files are transferred across a network, they require large amounts of network bandwidth. When a router handles a packet of data, it has no knowledge of data flow. It only sees individual packets and handles them separately. When transferred across a network using a connectionless oriented protocol like IP, individual packets of data may arrive

out of order. The TCP protocol is responsible for reassembling the packets before they are made available to an application. There is also no priority information specified in the IP packet, so that real-time data could take precedence over other types of data with a lower priority. This type of protocol was fine for supporting applications such as email and text-based chat sessions. It is not acceptable, however, for packets of data that are sensitive to time delay, such as real-time audio and video. Thus, to support large-scale multimedia, fundamental changes in the Internet infrastructure are necessary, including the data communication lines, routers, and protocols.

Delivering Multimedia Over the Internet

As mentioned previously, the current Internet infrastructure is not adequate for supporting multimedia. This section examines two attempts at updating the infrastructure to deliver large-scale multimedia.

RTP/RSVP

- The connectionless nature of the IP protocol does not lend itself to the time-sensitive nature of data packets carrying real-time audio and video. The Real-Time Transport Protocol (RTP) and the ReSerVation Setup Protocol (RSVP) protocols are currently being developed by the IETF (Internet Engineering Task Force) to make such multimedia support a reality. One of the major challenges of this effort is to minimize the amount of change in the existing Internet infrastructure to support them.

The initial specification of RSVP defines four levels of Quality of Service (QoS) without requiring wholesale changes to the Internet:

- Guaranteed delay
- Controlled delay
- Predictive service
- Controlled load

Although these QoS specifications vary in the priority in which they are handled, each adds a higher degree of determinism to the time in which packets are routed. RSVP/RTP advocates claim that this is sufficient for meeting the needs of multimedia applications.

The fundamental idea behind RSVP is to create a reservation with each router through which the data packets pass. This reservation consists of a flow identifier, to identify the data stream, and a flow specification, which identifies the QoS that must be provided to the data flow. In essence, the reservation defines a contract for service between the network and the requesting application.

Multimedia applications access these protocols by using a WINSOCK version 2 compliant application programming interface (API). The interface calls even allow specification of the QoS. Changing the API to support new features, while minimizing the amount of changes that need to be made to existing software will not be easy.

Another issue that must be resolved is payment for additional services. How will users be billed for specifying a higher level of service?

Multicast Backbone

As previously mentioned, the backbone of the Internet consists of high-speed data communication lines. Many experiments are being conducted to find ways of upgrading the physical hardware to support the transmission of real-time audio and video. One such experiment is known as the Multicast Backbone (MBONE). MBONE is not separate from the Internet. Rather, it is a set of Internet sites with powerful hosts and high-speed connections between them.

Unfortunately, MBONE can handle the display of only three to five frames per second. Full-motion video, on the other hand, requires the display of 30 frames per second. Although its potential does not approach broadcast quality, it is sufficient for a number of useful applications, such as teleconferencing. For more information, you can visit the <http://www.mbone.com> website.

Note:

On July 20th, 1996, the National Science Foundation and NASA sponsored a live broadcast over the MBONE. The presentation, given to commemorate the 20th anniversary of the Mars Viking Landings, discussed the Mars Pathfinder and Mars Global Surveyor missions.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Audio Over the Internet

At this time, there are two methods for handling audio data over the Internet. The first technique requires that an audio data file be transferred to a workstation, which is then handled by the appropriate audio player. The second technique does not require the complete file to be transferred before the file can begin to be played.

Audio File Transfer

Many sound files are on the Internet in a variety of formats. Each format has a unique file extension associated with it, such as .wav or .au. The following list shows file extensions that you are likely to see, along with their associated platform:

File Extension	Platform
AU	UNIX
SND	Macintosh
WAV	PC
AIF, AIFF	Macintosh
RA (RealAudio)	Macintosh, PC, UNIX

Each file type requires a special “player” utility. In most cases, these utilities can be configured to work with your favorite Web browser, so that they can be played

automatically when referenced within the browser. These are known as “helper” applications in Web terminology.

The major disadvantage of this technique, however, is the amount of time it takes to transfer the files. Even though the files are usually compressed, they are still large.

Streaming Audio

A technique known as *streaming audio* was developed to improve the performance of the plain file transfer method. This method allows the file to be played at the same time that the file is being transferred. To utilize this technique, you must use audio files and a player application capable of supporting streaming audio. The most popular audio streaming technology today is RealAudio.

- While the audio file is being played, the audio server and audio player exchange information about the connection. If it is a low-speed connection, a smaller version of the file is sent. If it is a high-speed connection, a larger, higher quality version of the file is used.

To reduce the amount of time necessary to transfer the data over the Internet, the file is compressed. The User Datagram Protocol (UDP) is used in conjunction with the IP protocol to transfer the data packets. Unlike TCP, UDP does not resend packets if problems in the transmission occur. If this were the case, the sound player would not be able to play the file due to frequent interruptions.

Phone Calls Over the Internet

Another form of audio over the Internet is making phone calls. Technically speaking, you can call anyone who has an email address. All that is needed are a speaker and microphone for your desktop computer, along with software to interpret the digitized packets of data. A number of competing companies make phone products for the Internet, including WebTalk by Quarterdeck, NetPhone by Electric Magic, and Internet Phone from VocalTec.

The main benefit of making phone calls over the Internet is the price. The only charge incurred is the cost of an Internet connection. The main disadvantages are voice quality and compatibility between Internet phone products. Other users must have the same exact software as you to have a phone conversation.

Video Over the Internet

Just like audio, there are two primary methods for handling video data over the Internet. The first technique requires that a video data file be transferred to a workstation, which is then handled by the appropriate video player. The second technique does not require the complete file to be transferred before the file is processed.

Video File Transfer

Many video files are on the Internet in a variety of formats. Each format has a unique file extension associated with it. The following list shows file extensions for video files that you are likely to see, along with their associated platform:

File Extension	Platform
QT (QuickTime)	Macintosh, PC
AVI	PC
MPG, MPEG	Macintosh, PC, UNIX
MOV	Macintosh, PC

Just like audio files, there are corresponding “player” applications for each file type. Even when compressed, they suffer from the same problem as audio files: They are simply too large.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Streaming Video

Conceptually, streaming video works in the same way as streaming audio. That is, compressed files are transferred over the Internet using the UDP/IP protocol. The user actually sees the file being played before the file transfer is complete. This can deliver reasonable performance when sent over a high-speed network, such as the MBONE.

The first attempt at implementing streaming video over the Internet is a product called VDOLive. Just like RealAudio, it tries to adjust the quality of the video based on speed of the connection. VDOLive can deliver 10 to 15 frames per second on a two-inch section of the screen over a 28.8 Kbps line. Over an ISDN line, 30 frames per second are possible. Before video data is transmitted, it must be compressed. Therefore, it does not lend itself to live broadcasts. Despite these limitations, VDOLive has a lot of potential.

Two newer products are RealPlayer and Streamworks. They combine both audio and video.

Videoconferencing

An application that uses both audio and video over the Internet is videoconferencing. In addition to a specially equipped workstation, including a microphone and a video camera, special software is needed.

The most popular videoconferencing software in use to date is a product called CU-SeeMe. This technology works much like streaming audio and video. When someone wants to participate in a videoconference, he first must log in to a special system on the Internet known as a *reflector*. A reflector hosts many videoconferences that you can join. After you log in, voice and video data are digitized and compressed before transport over the Internet. For efficiency reasons, the UDP protocol is used rather than TCP. Any missing packets are ignored by the application.

Note:

NASA has quite a few reflector sites from which live videos can be seen using CU-SeeMe. They also have an excellent collection of audio and video clips that are available for downloading. See <http://www.nasa.gov> for details.

CU-SeeMe also tries to reduce the amount of network bandwidth needed by only sending relevant portions of the images. For example, if someone is speaking, but rarely makes any motion in the field of the camera, only the changes from previous video frames need to be sent.

It is also possible to have a videoconference without a reflector site. If you know the other person's IP address, you can contact them directly and have a two-way conference.

Summary

The Internet, also known as the Information Superhighway, is still evolving. It was built to support applications such as email and file transfer. For it to support multimedia, such as audio and video, the infrastructure needs to be upgraded. Researchers are busy at work trying to figure out how to upgrade the infrastructure without requiring a major overhaul. Efforts such as RSVP/RTP are promising but are still on the horizon.

A more compelling problem is the data communications structure in place that connects to our homes and schools. To support multimedia, more bandwidth is needed at this juncture. This is known as the "last mile" problem.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

[Click Here!](#)



ITKnowledge

[home](#)

[account
info](#)

[subscribe](#)

[login](#)

[search](#)

[My
ITKnowledge](#)

[FAQ/
help](#)

[site
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

Chapter 7

Text Editing with vi and Emacs

by David B. Horvath, CCP, and Jeffrey A. Simon

In This Chapter

- What Is `vi`?
- Getting Started with `vi`: The Big Picture
- Advanced Editing with `vi`: Tips and Techniques
- What Is Emacs?
- Basic Editing with Emacs: Getting Started
- Advanced Editing with Emacs: Tips and Techniques
- Command Summary `vi` and Emacs

A text editor is one of the most common and useful tools found in any computer system. It is a program used to create and modify text or other data-type objects, usually interactively by the user at a terminal. It is distinguished from a word processor or desktop publishing program in that a text editor is generally expected to produce plain ASCII text files that do not have embedded formatting information. Those other programs are intended to produce more complex documents that contain much more formatting information. For example, a typical word processor has a graphical user interface and is capable of producing “what-you-see-is-what-you-get” (abbreviated as WYSIWYG) printed output.

Common uses of a text editor are to produce simple ASCII text files, program source code, email, and so on. Therefore, text editors are often extended to provide features that assist with specific aspects of such tasks, such as the formatting of a specific programming language. For example, such extended modes exist for C++, Lisp, and HTML, to name only a few. Detailed examples of some of these features are described later in this chapter.

This chapter examines two of the most popular and widely used editors in the UNIX world, `vi` and `Emacs`. In addition to being useful tools, each of these editors has its own group of devoted users, ready to “sing praises” to the virtues of using their favorites. In any case, you can get a lot of work done with either of these tools.

Full-Screen Editors Versus Line Editors

A full-screen editor displays on the user’s terminal a view of all or a portion of the document he or she is working on. For example, on a 25-line display, the user sees a 24-line section of the document. When using an editor, you are not actually making edits to the file that is stored on the hard disk. When the editor is commanded to begin working with a particular file, a working copy of that file is made. This working copy is often called the *buffer*. Adding, changing, and deleting of text (“editing”) is done only within the buffer until the file is saved. You often hear the advice to “save your work.” This advice is applicable to using a full-screen editor as well as any other computer work that uses a buffer in the same way.

You can think of the screen as a movable viewport into the buffer. This viewport is also often called a *window*. Editing actions take place at or relative to a specific point on the screen referred to as the *cursor*. The cursor is usually indicated on the screen by some sort of highlighting, such as an underscore or a solid block, which may or may not be blinking. Edits to the buffer are shown on the screen as soon as they are entered. This type of user-interface makes simple editing functions convenient for the user.

In contrast, a line editor does not attempt to show the appearance of a continuous section of the document being edited. It concentrates on editing one line at a time. Thus, its user interface is more primitive. The type of editing that you would naturally do in a full-screen editor becomes more cumbersome under such an arrangement.

However, do not be misled into thinking that the primitive user interface of the line editor means that a line editor lacks power or that all line editors are obsolete. (A great many line editors are obsolete; the trick is recognizing those that are not!)

Certain powerful editing functions are most easily executed by using a line editor. For example, if you had to reverse the order of the lines in a file, you could do that with eight keystrokes in `vi`! So it might be a good thing if there was an editor that could take advantage of the power of both the full-screen and line-oriented modes.

What Is `vi`?

`vi` (usually pronounced *vee-eye*) is a full-screen text editor that comes with nearly every UNIX system. Many versions of `vi` or similar programs have been made for other operating systems. Such versions exist for Amiga, Atari, Macintosh, MS-DOS,

OS/2, Windows 3.1/95/NT, and probably many other operating systems.

The Relationship of vi and ex

vi is “closely” related to the line editor ex. (In fact, they are one and the same.) vi is the visual (or open) mode of ex. This means that you could start editing a file with the ex editor. At any time, you can invoke the visual mode of ex. Voila`—you are in vi! From vi, you can at any time “drop down into ex” because all ex commands are available from within vi. Thus you can easily go back and forth between the visual and line-oriented modes, executing the particular editing operation you need from the mode in which it is most effectively accomplished. Later in this chapter, you see examples of such operations.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Why Would I Be Interested in Using vi?

Many computer users are familiar with the powerful word processing programs widely available on personal computers. If you are used to such a tool, you might be disappointed to find out that `vi` is not a WYSIWYG word processor. However, it is rare that such a word processing program is available on the typical UNIX system. `vi` on the other hand is nearly always available. One of the strongest reasons for knowing at least the rudiments of `vi` is the fact that it is nearly always available on any UNIX system. It becomes particularly invaluable to those who have to periodically go into a UNIX environment away from their everyday system.

Although the lack of a graphical user interface might be a hindrance to the novice, many “power users” believe that the fastest and most productive interaction with online tools is through command-based interfaces. This is also true of `vi`. After the keystrokes of the commands become second nature, the productivity of a `vi` user is not easily surpassed. In fact, the apprehension of the uninitiated toward command-based interfaces is probably due to the following common misconception: People think they have to memorize an obtuse, counter-intuitive set of command keys when, in fact, it is more a matter of finger training than memorization.

For example, suppose that you want to move the cursor (the point where actions on the text take place) down one line. After learning to use `vi` and becoming comfortable with it, there is no mental process (such as “Move down—let’s see that’s a “j”). Rather it is a physical motion (such as “Move down—finger motion to press the “j” key). Think of it as like learning to drive a car. After having mastered the process, if you see a ball bouncing into the road ahead, you do not have a mental process (such as “Ball—child—STOP! Let’s see, which pedal is it? Where’s the instruction manual?!). Rather, your body reacts instantly to press the brake pedal. It is the same way with a command-based interface. After you learn it, your fingers effectively execute the command.

Starting Up and Exiting vi

The first thing to learn with any new program is how to start it and how to get out of it. The simplest way to start `vi` is to type its name along with the name of the file you want to edit. If no name is specified, `vi` responds with an empty screen, except for a column of tildes along the left side. Your screen looks similar to the following:

[illegible]

If the `TERM` environment variable has not been set or `vi` does not recognize the terminal, it starts up in `ex` mode. You will see a message about not recognizing the terminal and one line from the file. You will also see just a colon (`:`) as a prompt.

```
export TERM=vt100
```

And start `vi` again.

If wise, a commander is able to recognize changing circumstances and to act expediently. If sincere, his men will have no doubt of the certainty of rewards and punishments. If humane, he loves mankind, sympathizes with others, and appreciates their industry and toil. If

Shen Pao-hsu ... said: 'If a general is not courageous he will be unable to conquer doubts or to create great plans.'

```
"art1"  8 lines, 576 characters
```

Note:

Entering the command ZZ saves your file and exits. The other ways of exiting involve ex mode commands. To enter ex mode, enter the colon character :. The screen display changes so that a colon is displayed on the bottom line of the screen, and the cursor is positioned immediately to the right of this colon, waiting for your command.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

□

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(Publisher: Macmillan Computer Publishing)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

The `:q` key “quits” the file, if no changes have been made since the last save of the file. If a change has been made, you are prevented from exiting, and the following warning is displayed: `No write since last change (use ! to override)`. The command `wq` can be used to handle this situation, by writing the file before exiting. Or you can go ahead and use the `:q!` as the message indicates, to go ahead and quit anyway, abandoning all your edits since the last save of the file. (It’s good to keep in mind the `:q!` command for those cases in which you have truly messed up and want to get rid of your mess.)

Table 7.1 summarizes the exiting commands presented so far.

Table 7.1.Exiting commands.

Keystrokes	Result
<code>ZZ</code>	Save file and immediately exit
<code>:wq</code>	Save file and immediately exit (same as <code>ZZ</code>)
<code>:q</code>	Exit; prevented if file not saved
<code>:q!</code>	Exit; forced exit whether saved or not

Getting Started with vi: The Big Picture

Let’s look at some pieces of the big picture that give `vi` its character.

vi Has Modes

`vi` was created back when the keyboard and screen method of interaction with computers was new. In those primitive days, keyboards did not have all the useful function keys that are now familiar. Therefore, `vi` was designed to allow you to enter and modify text using only the typewriter keys plus the Escape key. With newer versions, other key sequences are sometimes recognized, such as the cursor control keys.

Although it might seem like a limitation not to take advantage of the many additional keys available on the

modern keyboard, the “silver lining” of this limitation is that all functions can be executed without taking your hands away from the touch-typing position. The result makes for efficient and rapid typing.

To enable the many editing functions necessary for interactive, full-screen editing, `vi` is operated in three modes. The insert mode is used for entering text. While in insert mode, every typewriter key pressed is displayed on the screen and entered into your text. The command mode is used for most editing functions. While in command mode, nearly every typewriter key pressed causes some action other than the direct entry of text, such as moving around to different points in your text, deleting blocks of text, copying blocks of text, and so on. A third mode, called `ex` mode is used to execute additional functions, such as searching, global replacement, manipulation of multiple files, and many more. The `ex` mode is based on the underlying `ex` editor and is described in greater detail later in the section “Using the Power of `ex` from `vi`.”

Tip:

vi has the capability of showing the mode it currently is in. On the screen, you can type the following:

```
:set showmode
```

This enables the display of the insert status for this editing session only. Not all versions of `vi` support this capability. See the “Changing `vi` Settings” section of this chapter for more information on the settings and how to save them permanently.

Starting vi

When `vi` is started up, the default mode is command mode. Test this out: start `vi` by typing in the program name only:

§ vi

`vi` can also be started with the `view` command, which starts `vi` in a read-only mode (preventing you from saving any changes). Some versions of UNIX have a `vedit` command, which is a novice version of `vi`. One nice feature of `vi` is that it tracks changes to a file until you save it. If the system goes down or you are disconnected, you can recover your changes using `vi -R yourfilename`.

You see something similar to the following:

~~~~~

~~~~~

Most UNIX systems send you email reminding you to use `vi -R` to recover your file.

Shen Pao-hsu ... said: 'If a general is not courageous he will be unable to conquer doubts or to create great plans.'

~~~~~

Unfortunately, there is no readily visible indication of which mode you are in unless you enable `showmode`. However, it is pretty easy to see what mode you are in. If the keystrokes go into the text, you are in insert mode; if your screen jumps around wildly, beeps, and all kinds of weird things are happening, you are probably in command mode. If you are unsure of what mode you are in, just press `Esc` twice to get the beep confirming that you are in command mode.

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc.  
All rights reserved. Reproduction whole or in part in any form or medium without express written  
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Click Here!

ITKnowledge

home

account  
info

subscribe

login

search

My  
ITKnowledge

FAQ/  
help

site  
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

- Brief Full
- Advanced
- Search
- Search Tips

Bookmark It

Search this book:

Previous

Table of Contents

Next

Moving Around and Simple Editing

It’s time to look at the most basic movement commands, the ones that you must train your fingers to execute automatically.

The Most Important Movement Keys

Editing commands in vi are composed of objects and commands. Objects are used by themselves to move around, or “navigate,” in the buffer. A single object keystroke either causes the cursor position to move on the screen, or to reposition the “viewport” in the buffer. Let’s see how the various movement commands affect the cursor position in our sample text.

h—Cursor Left

First, move the cursor back five positions by pressing the h key five times (if you see five h’s go into the text, you forgot to press the Esc key). The cursor should now be under the “p” of “plans” (see the following example):

If wise, a commander is able to recognize changing circumstances and to act expediently. If sincere, his men will have no doubt of the certainty of rewards and punishments. If humane, he loves mankind, sympathizes with others, and appreciates their industry and toil. If courageous, he gains victory by seizing opportunity without hesitation. If strict, his troops are disciplined because they are in awe of him and are afraid of punishment.

Shen Pao-hsu ... said: 'If a general is not courageous he will be unable to conquer doubts or to create great plans.'

~  
~  
~  
~  
~  
~  
~

## k—Cursor Up

If wise, a commander is able to recognize changing circumstances and to act expediently. If sincere, his men will have no doubt of the certainty of rewards and punishments. If humane, he loves mankind, sympathizes with others, and appreciates their industry and toil. If courageous, he gains victory by seizing opportunity without hesitation. If strict, his troops are disciplined because they are in awe of him and are afraid of punishment.

~~~~~

Other commands work like the h and k keys. Table 7.2 describes their functions. The best way to get used to how they work is to practice using them. Table 15.2 shows the most frequently used movement keys.

Table 7.2.Frequently used movement keys.

Keystroke(s)	Moves
h	one character left
j	one line down
k	one line up
l	one character right

w, W	one word forward (W ignores punctuation)
b, B	one word backward (B ignores punctuation)
\$	to end of line
^	to first non-space character of line
0	to beginning of line
G	to top of buffer
nG	where <i>n</i> is a whole number, to line <i>n</i>

The upper- and lowercase versions of the word movement commands have a subtle difference. The lowercase version counts most punctuation marks as “words.” The uppercase version skips over them as if they were not present.

Practice moving around in your sample text, using the previously described commands. Although they might seem awkward at first, you will soon get used to them as your fingers are trained.

The Most Important Editing Procedures

Let’s look at some of the simplest and most often used editing procedures:

Changing Text

Nobody is perfect. Sooner or later you will want to change some text that you have created. In fact, more text editing time is probably spent modifying existing text than in entering brand-new text, so you need some easy ways of changing text. This section shows how.

x—Delete Character

The simplest way to delete text is with the `x` command. This command causes the character that the cursor is over to be deleted, and the remaining characters on the line to be shifted one character to the left. You can think of “x-ing” out the text you want to get rid of. If the character deleted is the last one on the line, the cursor moves one character to the left, so as not to be over nonexistent text. If there is no more text on the line, the beep sounds.

d—Delete Object

The delete command requires a text object on which to operate. A text object, or object for short, is the block of text that would be traversed by the use of a movement command. For example, `w` advances to the next word. So `dw` deletes to the beginning of the next word. `5w` advances to the beginning of the fifth word (remember, punctuation symbols count as “words” to the `w` command). So `5dw` (or alternatively `d5w`) deletes to the beginning of the fifth word. Both forms work because `5dw` mean “do five delete-words”; `d5w` means “do delete five words.”

dd—Line Delete

One of the most often used forms of the `d` command is the special version, `dd`, which deletes an entire line. As before, `5dd` deletes five lines.

D—Big Delete

The uppercase form `D` is used to delete from the cursor position to the end of the line. It has the same action as `d$`.

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



ITKnowledge

home

account
info

subscribe

login

search

My
ITKnowledge

FAQ/
help

site
map

contact us

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

u—Undo

After learning how to do deletes, the first thing I want to know is whether there is an undo function. There is. It is invoked naturally by the `u` command. The `u` command undoes the most recent change to the file (not only deletes, but any edits). The cursor does not need to be at the location of that most recent change.

Caution:

Unfortunately, standard `vi` has only one level of undo. After a second change is made, you cannot undo the first. If you press `u` a second time, it will “undo the undo,” which is sometimes known as “redo.” Repeated presses of the `u` key toggles the text back and forth between the two versions.

This also applies for `U`—the big undo.

U—Big Undo

The “big brother” of the `u` command, the `U` command undoes all changes made to the line that the cursor is on, since the cursor was last moved on to that line. After the cursor is moved off a line, the `U` command no longer works on the edits that have already been made to that line.

.—Repeat

Repeats the last editing command.

How Commands Are Formed

By now you have probably noticed that there is a pattern to the structure of the `vi` commands. First, the commands are fairly mnemonic, which means that the letter of the command should remind you of the function being executed. Second, many commands have an uppercase version, which is usually a modified form of the basic, lowercase command. Third, all commands can be multiplied by a repeat count, entered as a prefix.

Table 7.3 shows the easiest ways to see how the commands are formed. You can see that there are several ways of combining command elements to get the result you want. To repeat a command, just enter the repeat count prior to the command itself, as in the previous examples of cursor motion and deletion.

Table 7.3.How `vi` commands are formed.

<i>General form of vi commands</i>	
<code>{count} {command}</code> <code>{object}</code>	All parts are optional (see the following). The <code>{count}</code> , if present, causes the command to be executed <i>count</i> number of times. The <code>{command}</code> , if present, causes some action to take place. If absent, the cursor is moved according to the object. The <code>{object}</code> is used together with the command to indicate what portion of the text is to be affected.
<i>Specific Forms of vi Commands</i>	
<code>{count}</code>	Position the cursor <i>count</i> lines down, if terminated with <code>return</code> or <code>+</code> ; position the cursor <i>count</i> line up if terminated with <code>-</code> .
<code>{command}</code>	Execute the command.
<code>{object}</code>	Move the cursor over the indicated block.
<code>{count} {command}</code>	Execute the command <i>count</i> times.
<code>{count} {object}</code>	Move the cursor over <i>count</i> indicated blocks.
<code>{command} {object}</code>	Execute the <i>command</i> over the indicated block.

Table 7.4 shows examples of combining some of the commands that you already know.

Table 7.4.Examples of combining commands.

Command	Result
h	Move cursor left one character
3h	Move cursor left three characters
dd	Delete one line
3dd	Delete three lines
w	Move cursor forward one word
dw	Delete one word
3dw	Delete three words

You now have the basic editing commands that will enable you to get started. You might want to start practicing these commands right away. With these commands, you could do any text editing project. But you wouldn't want to. By adding some additional commands, you can make your work much faster and easier. The whole point of computers is to make work easier, so why not use the power of `vi` to have the computer do what it is good at!

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

$$\sim \sim \sim \sim \sim \sim \sim \sim \sim \sim \sim \sim \sim$$

You can insert a repeated string sequence by using one of the insert or append commands with a repeat count. For example, to insert 78 asterisk characters, you could type **78i*Esc**.

~~~~~

```
ais able to <Esc>
```

## - A—Big Append

### c—Change Object

To change text, you can use the `c` command. The `c` command takes an object to indicate the block of text that will be changed. The `c` command works like the `d` command followed by the `i` command. That is, it first performs the deletion that would be performed by the `d` command with the same object and then allows the insert of any amount of text (including line feeds) until the Esc key is pressed. This behavior makes it especially useful in such situations where you want to change the text from

the position of the cursor to the beginning of the line (using `c0`) or to the end of the line (using `c$`).

## cc—Change Line

In a similar vein, the `cc` command works like the `dd` command followed by an `i` command. It deletes the line the cursor is on and then inserts all keystrokes typed until the Esc key is pressed.

## C—Big Change

The `C` command works like the `D` command followed by the `i` command; it deletes the text from the cursor position to the end of the line; then enters insert mode. The `C` command has the same action as `c$`.

## r—Replace Character

The `r` command replaces the single character where the cursor is placed. After the `r` key is pressed, no change is seen on the screen. The next key typed replaces the character at the cursor position and then `vi` returns to command mode. It is a simple way to change just one character.

When used with a numeric count, the same replacement occurs over count characters. For example, suppose that the screen looks like the following, with the cursor under the “c” of “commander”:

If wise, a commander recognize changing

~~~~~


insert mode.

A Copy Is a “Yank”

Many text editors have features known as “cut and paste” or “copy and paste.” `vi` calls the copy part of “copy and paste” a *yank*. You can use the yank command to save any block of text in the undo buffer. The undo buffer is a special place that `vi` keeps internally. You can’t directly see the contents of this buffer. The contents of this buffer can be put into the text with the `p` command.

Each use of the `y` command overwrites the contents of the undo buffer, as does any delete command. There is a more advanced version of the yank command explained in the section “How To Use Buffers,” which can be used to save text in multiple named buffers.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition
(*Publisher: Macmillan Computer Publishing*)
Author(s): Robin Burk
ISBN: 0672314118
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

y—Yank

The yank command works with an object in the same way as the c and d commands. You can yank a word with yw, yank to the end of the line with y\$, yank three lines with 3yy.

Y—Big Yank

There is an exception to the pattern, however. For some reason, the Y command does not take its action to the end of the line as C and D do. Instead, it yanks the whole line and is therefore identical to yy.

Copying Text

The commands discussed in the following sections are used to copy text.

p—Put

The p command takes whatever is in the undo buffer and inserts it into the text after the cursor position.

P—Big Put

The P command takes whatever is in the undo buffer and inserts it into the text before the cursor position.

Moving Text

In addition to text that you specifically yank being placed in the undo buffer, each portion of text that is deleted goes into the same undo buffer, replacing the previous contents each time. So to perform a cut and paste, you would use any delete function, move the cursor to the desired insertion point, and then use one of the put commands to insert the text. You have to pay attention to the location of the cursor and whether to use the p or P commands to get exactly what you want.

For example, suppose that your screen appears as shown in the following example:

```
      If wise, a commander is able to recognize changing circumstances and
```

Shen Pao-hsu ... said: 'If a general is not courageous he will be unable to conquer doubts or to create great plans.'

Now suppose that you want to change the order of the paragraphs so that the paragraph beginning with “Shen Pao-hsu” comes first. First, move the cursor on to any character in the first line of the “Shen Pao-hsu” paragraph, as shown here:

Shen Pao-hsu ... said: 'If a general is not courageous he will be unable to conquer doubts or to create great plans.'

Then press `2dd` to delete the second paragraph. Your screen appears as in the following example:

If wise, a commander is able to recognize changing circumstances and to act expediently. If sincere, his men will have no doubt of the certainty of rewards and punishments. If humane, he loves mankind, sympathizes with others, and appreciates their industry and toil. If

courageous, he gains victory by seizing opportunity without hesitation. If strict, his troops are disciplined because they are in awe of him and are afraid of punishment. ~

~~~~~

Now move the cursor to any character on the top line (1G takes you to the first line), as shown here:

If wise, a commander is able to recognize changing circumstances and to act expediently. If sincere, his men will have no doubt of the certainty of rewards and punishments. If humane, he loves mankind, sympathizes with others, and appreciates their industry and toil. If courageous, he gains victory by seizing opportunity without hesitation. If strict, his troops are disciplined because they are in awe of him and are afraid of punishment.

~~~~~

Now use the `p` command to put the text. Oops! Your screen looks like this:

If wise, a commander is able to recognize changing circumstances and to

Shen Pao-hsu ... said: 'If a general is not courageous he will be unable to act expediently. If sincere, his men will have no doubt of the certainty of rewards and punishments. If humane, he loves mankind, sympathizes with others, and appreciates their industry and toil. If courageous, he gains victory by seizing opportunity without hesitation. If strict, his troops are disciplined because they are in awe of him and are afraid of punishment. ~

2

~~~~~

You probably noticed during this exercise that you were able to use the `put` command repeatedly to put the same text. You can do this because the `put` command does not change the contents of the undo buffer. This feature sometimes comes in handy.

[Previous](#) [Table of Contents](#) [Next](#)

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#). Copyright © 1996-2000 EarthWeb Inc. All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

2



ITKnowledge

home

account  
info

subscribe

login

search

My  
ITKnowledge

FAQ/  
help

site  
map

contact us

To access the contents, click the chapter and section titles.

## Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

## Searching for Patterns

### /, //, ?, ??, n, and N—Search

One of the more useful ways of moving around in your text is to search for a pattern. You might be editing a long source code file and want to go back to a routine that you remember contains a specific instruction. You can do this by using / , the forward search command, or ? , the backward search command. As soon as you type the slash or question mark while in command mode, the cursor moves to the command line. You then type the pattern that you want to find. This pattern can be a literal text string, which is the exact character sequence you want to find. Or it can be a regular expression, described in detail in the “Regular Expressions” section.

After pressing the Return key, the text is repositioned so that the line containing the first occurrence of the pattern is displayed approximately in the center of the screen (assuming there is enough surrounding text to permit this), and the cursor is positioned to the first character of the matched text. If the pattern is not found, the message “Pattern not found: pattern” is displayed on the command line.

You use the / command to search forward in the text (that is, from the cursor’s position to the end of the buffer). You use the ? command to search backward in the text (that is, from the cursor’s position to the top of the buffer). You can repeat either forward or backward searches without reentering the pattern by using the two search again commands (/ and ? without any search text—// and ?? also work). You must

also press the Return key after these search again commands. After a pattern has been entered, you can intermix the forward and backward search commands.

Another variation on repeating search commands is to use the `n` command, which repeats the previous search in the same direction, whether forward or backward. The `N` command repeats the previous search in the opposite direction.

`vi` searches “wrap around” the top and bottom of the buffer. When searching for the pattern, if `vi` hits one end of the buffer, a message displays on the command line notifying you of this fact. For example, the message “search hit BOTTOM, continuing at TOP” may appear. At this point, you may press Return to continue the search. This behavior can be changed; refer to the “For the Power User: Customizing `vi`” section to see how you can use special settings to change `vi`’s default behavior.

---

**Tip:**

Because there are different versions of `vi`, the behavior you see might be different. Look at the man page and check your personal `vi` settings.

---

## \How To Use Buffers

The undo buffer contains only the most recent yanked or deleted text. This means that if you were intending to yank some text and copy it in somewhere, but before you put the text you performed any other deletion, you would be disappointed with the result. To keep various text snippets available for putting, you have to use named buffers.

### *Named Buffers*

Named buffers allow you to keep up to 26 separate places where text can be deleted or yanked. Using the named buffers allows you to overcome the problem of intermediate deletes replacing the text that you have yanked or deleted. The contents of a named buffer remain unchanged until the end of your `vi` session, unless you use commands to deliberately change it.

Buffers are named by using a ``` followed by a lowercase letter. So the buffers are named from ``a` to ``z`. To yank or delete into a named buffer, prefix the yank or delete command with the name. For example, to yank two lines from the cursor position into buffer `z`, you would use the following keystrokes: ``z2yy`. To put from the named buffers, the key sequence is the buffer name followed by the `p` command. Table 7.5 shows some examples of using named buffers (some of these object commands have not been introduced yet; they will be explained further):

**Table 7.5.**Examples of using named buffers.

---

| Keystrokes         | Result                                                 |
|--------------------|--------------------------------------------------------|
| <code>`a2dw</code> | Delete next two words into named buffer <code>a</code> |

---

|      |                                                                  |
|------|------------------------------------------------------------------|
| "jD  | Delete from cursor to end of line into named buffer j            |
| "jp  | Put the contents of named buffer j after the cursor              |
| "by) | Yank from cursor position to end of sentence into named buffer b |

---

As with a normal yank into the regular (unnamed) undo buffer, the action of the yank or delete into the named buffer replaces the previous contents of that buffer. If, instead, you want to collect text in a named buffer by appending it to what is already there, you may do this by uppercasing the letter of the buffer's name. Yanks and deletes can be intermixed when using the appending method, as shown in the sequence in Table 7.6.

**Table 7.6.** Intermixed sequence of yanks and a delete.

| Keystrokes | Result                                                                   |
|------------|--------------------------------------------------------------------------|
| "a2yy      | Yank two lines into named buffer a, discarding the previous contents     |
| "Ad4w      | Delete the next four words and append them into buffer a                 |
| "Ay)       | Yank from the cursor to the end of the sentence and append into buffer a |

---



---

**Caution:**

In executing such a sequence, do not forget to use the capital letter. If you forget, the previous contents are obliterated, and your careful work is lost. For this reason, I seldom use this technique, and when I do use it, I do it carefully!

---

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

[Click Here!](#)



ITKnowledge

[home](#)

[account  
info](#)

[subscribe](#)

[login](#)

[search](#)

[My  
ITKnowledge](#)

[FAQ/  
help](#)

[site  
map](#)

[contact us](#)

To access the contents, click the chapter and section titles.

## Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

[Bookmark It](#)

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

## Delete Buffers

In addition to the named buffers, `vi` provides numbered delete buffers. A normal undo can only undo the last delete and only if it was the last edit. However, `vi` saves the most recent nine deletes in buffers numbered from 1 to 9. The most recent delete is kept in buffer 1, the next most recent in buffer 2, and so on. To recover the contents of one of these buffers, use the number as the buffer name with the `p` command, as in `"2p` to put from the second delete buffer.

---

### Tip:

How do you know what is in each of the delete buffers? A special feature of the `.` (repeat) command (described later in the “Recovering Deleted Text: Cutting and Pasting” section) allows you to easily choose from among the numbered buffers. When the repeat command is used after a command referencing a numbered buffer, the buffer number is incremented.

Thus you can put from numbered buffer 1, see the text that is inserted, use the `u` command to undo the put, and then just press `.` (the *repeat* command) to see the next buffer’s contents. Continuing this process with a series of `u` commands and `.` (*repeat*) commands quickly scans through the nine most recent deletions. (When you get to the ninth one, continued key presses “stick” on the ninth buffer’s contents.) This is another technique that is much easier to do than to describe.

---



## The Complete Guide to Movement and Editing: Command Reference Tables

By this point, you have seen the most commonly used `vi` commands. However, many useful and powerful movement and editing commands that, although less frequently used, might become invaluable to you as you learn to use them. Some of them are likely to be included in your repertoire of often-used commands, sooner or later. This section is intended to provide complete coverage of the movement, editing, and other commands, for easy reference.

---

### Note:

So called Control-key commands are entered in a similar fashion, with the `Ctrl` key held down while the other key is pressed. Ctrl-key commands are indicated in the `vi` sections by prefixing the command with a caret symbol (^), or alternatively, by the sequence `Ctrl+key`. So, for example, when you see in the text the symbol `^A`, you are to hold down the `Ctrl` key while you press the letter `a`.

Later, in the `Emacs` sections, the standard `Emacs` way of indicating control keys will be used—do not let this confuse you.

---

Note that some commands are shifted commands. Commands represented by uppercase letters are entered by holding down the `Shift` key while the alphabetic letter is pressed.

Be aware that all the commands shown in Table 7.7 can be combined in the ways described in Table 7.3. That is, you can amplify the effect of commands by using a count or an object or both, as appropriate.

Some commands included here have not been introduced yet. They will be explained in detail in the advanced editing section.

### ***Movement***

In Table 7.7 the following words in the command column have the specified meaning: *char* means any character; *number* (or *nbr*) is a whole number; *pattern* is a string of characters or a regular expression.

In the description column, a small word is a word that can be either a string of alphanumeric characters (plus the underscore) or a string of punctuation characters, delimited by whitespace (spaces, tabs, and line feeds); a big word is a sequence of nonblank characters. These precise but technical definitions are saying, in effect, that small words consider the punctuation to be separate “words”; big words include the punctuation as part of the word. The easiest way to see the difference is first to try a repeated sequence of moves using the lowercase version of a command; then try the uppercase version of the command.

**Table 7.7.**Movement commands.*Single character cursor motion*

| Command     | Result                    |
|-------------|---------------------------|
| h           | Moves one character left  |
| ^H          |                           |
| left-arrow  |                           |
| j           | Moves one line down       |
| ^J          |                           |
| down-arrow  |                           |
| k           | Moves one line up         |
| ^P          |                           |
| up-arrow    |                           |
| l           | Moves one character right |
| right-arrow |                           |

*Movement within a line*

| Command | Result                                                                |
|---------|-----------------------------------------------------------------------|
| ^       | Moves to first non-space character on the line                        |
| 0       | Moves to beginning of the line                                        |
| \$      | Moves to end of line                                                  |
| fchar   | Moves to next occurrence of character <i>char</i>                     |
| Fchar   | Moves to previous occurrence of character <i>char</i>                 |
| - tchar | Moves to character before next occurrence of character <i>char</i>    |
| Tchar   | Moves to character after previous occurrence of character <i>char</i> |
| ;       | Repeats previous f, F, t, or T command; same direction                |
| ,       | Repeats previous f, F, t, or T command; opposite direction            |

*Motion to a specified line*

| Command | Result                                                 |
|---------|--------------------------------------------------------|
| Enter   | Moves to next line                                     |
| +       | Moves to next line (usually used with preceding count) |

|                 |                                                            |
|-----------------|------------------------------------------------------------|
| -               | Moves to previous line (usually used with preceding count) |
| <i>number</i> G | Moves to line <i>number</i>                                |
| <i>number</i>   | Moves to column <i>number</i>                              |

*Screen positioning*

| Command          | Result                                  |
|------------------|-----------------------------------------|
| H                | Moves to top line displayed onscreen    |
| L                | Moves to bottom line displayed onscreen |
| M                | Moves to middle line displayed onscreen |
| ^D               | Scrolls down one-half screen            |
| <i>number</i> ^D | Scrolls down <i>number</i> lines        |
| ^U               | Scrolls up one-half screen              |
| <i>number</i> ^U | Scrolls up <i>number</i> lines          |
| ^F               | Scrolls forward one screen              |
| ^B               | Scrolls backward one screen             |
| ^E               | Scrolls down one line                   |
| ^Y               | Scrolls up one line                     |

*Lexical object positioning*

| Command | Result                                   |
|---------|------------------------------------------|
| w       | Moves forward one small word             |
| W       | Moves forward one big word               |
| b       | Moves backward one small word            |
| B       | Moves backward one big word              |
| e       | Moves to end of next small word          |
| E       | Moves to end of next big word            |
| (       | Moves to beginning of previous sentence  |
| )       | Moves to beginning of next sentence      |
| {       | Moves to beginning of previous paragraph |
| }       | Moves to beginning of next paragraph     |
| [[      | Moves to beginning of next section       |
| ]]      | Moves to beginning of previous section   |

*Screen redrawing*

| Command | Result                                                |
|---------|-------------------------------------------------------|
| z       | Redraws screen with current line at top of the screen |

|                 |                                                          |
|-----------------|----------------------------------------------------------|
| <code>z-</code> | Redraws screen with current line at bottom of the screen |
| <code>z.</code> | Redraws screen with current line at center of the screen |

*Positioning by pattern searching*

---

| Command                    | Result                                                                                                                                  |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <code>/pattern</code>      | Moves to next line containing pattern                                                                                                   |
| <code>?pattern</code>      | Moves to previous line containing pattern                                                                                               |
| <code>/</code>             | Repeats last search forward                                                                                                             |
| <code>?</code>             | Repeats last search backward                                                                                                            |
| <code>n</code>             | Repeats last search in same direction                                                                                                   |
| <code>N</code>             | Repeats last search in opposite direction                                                                                               |
| <code>/pattern/+nbr</code> | Moves to <i>nbr</i> lines after next line containing <i>pattern</i>                                                                     |
| <code>?pattern?-nbr</code> | Moves to <i>nbr</i> lines before previous line containing <i>pattern</i>                                                                |
| <code>/pattern/z-</code>   | Redraws screen with next line containing pattern at bottom of the screen<br>(Other <i>z</i> options give the corresponding positioning) |
| <code>%</code>             | Moves to parenthesis or brace matching the one at the current cursor position                                                           |

*Positioning to marked text locations*

---

| Command            | Result                                                                                            |
|--------------------|---------------------------------------------------------------------------------------------------|
| <code>mchar</code> | Marks the current cursor position with the letter <i>char</i>                                     |
| <code>`char</code> | Moves to mark specified by <i>char</i>                                                            |
| <code>^char</code> | Moves to beginning of line containing mark specified by <i>char</i>                               |
| <code>``</code>    | Moves to previous location of the current line (after a cursor movement)                          |
| <code>“</code>     | Moves to beginning of line containing previous location of current line (after a cursor movement) |

---

**Note:**

Some versions of `vi` do not allow you to use the arrow keys on the keyboard for cursor movement. This is typically an operating system or terminal (terminal emulator) issue, not really a `vi` problem.

The other cursor movement keys will work.

---

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)  
All rights reserved. Reproduction whole or in part in any form or medium without express written  
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

ITKnowledge

home

account info

subscribe

login

search

My ITKnowledge

FAQ/help

site map

contact us

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Editing

In Table 7.8, the words in the command column have the specified meaning: *object* means an object command from Table 7.7; *letter* means one of the 26 alphabetic characters from a to z. All editing commands can take nearly any movement command as an object. The text insertion commands cause entry into *insert* mode; *vi* then stays in that mode until you press the Esc key.

Table 7.8.Editing commands.

Inserting text

| Command | Result                                                                 |
|---------|------------------------------------------------------------------------|
| i       | Inserts text before the cursor                                         |
| I       | Inserts text before first nonblank character of line                   |
| a       | Inserts text after the cursor                                          |
| A       | Inserts text at the end of the line                                    |
| o       | Adds an empty line below the current line and enters insert mode there |
| O       | Adds an empty line above the current line and enters insert mode there |

## Changing text while in insert mode

---

| Command | Result |
|---------|--------|
|---------|--------|

---

(Note: These commands are only available while in insert mode.)

---

| Command          | Result                                                                       |
|------------------|------------------------------------------------------------------------------|
| <code>^H</code>  | Backspaces and erases the previous character (only since insert began)       |
| <code>^W</code>  | Backspaces over and erases the previous small word (only since insert began) |
| <code>\</code>   | Quotes the erase and kill characters                                         |
| <code>Esc</code> | Ends insert mode and goes back to command mode                               |
| <code>^D</code>  | Goes back to previous auto-indent stop                                       |
| <code>^^D</code> | (Caret followed by Ctrl+D) no auto-indent on current line only               |
| <code>0^D</code> | Moves cursor back to left margin                                             |
| <code>^V</code>  | Enters any character into text (do not interpret control characters)         |

### Changing text

---

| Command              | Result                                                                                                                    |
|----------------------|---------------------------------------------------------------------------------------------------------------------------|
| <code>cobject</code> | Changes the text object to the text inserted until the Esc key is pressed                                                 |
| <code>C</code>       | Changes the rest of the line to the text insert until the Esc key is pressed (same as <code>c\$</code> )                  |
| <code>cc</code>      | Changes the whole line to the text inserted until the Esc key is pressed                                                  |
| <code>- rchar</code> | Replaces the character the cursor is on with <i>char</i> ; then returns to command mode                                   |
| <code>R</code>       | Overwrites text until the Esc key is pressed; if you go past the end of the line, appends new text to the end of the line |
| <code>s</code>       | Substitutes characters (same as <code>c </code> )                                                                         |
| <code>S</code>       | Substitutes lines (same as <code>cc</code> )                                                                              |

### Deleting text

---

| Command        | Result                                  |
|----------------|-----------------------------------------|
| <code>x</code> | Deletes the character under the cursor  |
| <code>X</code> | Deletes the character before the cursor |

|         |                                            |
|---------|--------------------------------------------|
| dobject | Deletes the text object                    |
| D       | Deletes the rest of the line (same as d\$) |
| dd      | Deletes the line                           |

*Using buffers*

---

| Command        | Result                                                                                                                      |
|----------------|-----------------------------------------------------------------------------------------------------------------------------|
| u              | Undoes the last change                                                                                                      |
| U              | Restores the current line to the state it was in when the cursor was last positioned to it                                  |
| yobject        | Places the text of the object into the undo buffer                                                                          |
| YY             | Places the line the cursor is on into the undo buffer                                                                       |
| Y              | Places the line the cursor is on into the undo buffer (same as yy, which is a departure from the pattern set up by C and D) |
| p              | Inserts the text in the undo buffer after the cursor                                                                        |
| P              | Inserts the text in the undo buffer before the cursor                                                                       |
| "letterdobject | Deletes the object into the letter buffer                                                                                   |
| "letteryobject | Yanks (copies) the object into the letter buffer                                                                            |
| "letterp       | Inserts the text in the letter buffer after the cursor                                                                      |
| "numberp       | Inserts the <i>number</i> -th last delete of a complete line or block of lines                                              |

*Other editing commands*

---

| Command | Result                                                                                                                                                            |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .       | Repeats the last editing command (and increments n in a "np command)                                                                                              |
| ~       | Changes the case of the letter under the cursor and moves cursor to the left one character (does not support a count in standard vi)                              |
| J       | Joins two lines                                                                                                                                                   |
| >>      | Shifts line <i>shiftwidth</i> characters to the right (use :set sw to change the <i>shiftwidth</i> )                                                              |
| >L      | Shifts all lines from the line the cursor is on to the end of the screen <i>shiftwidth</i> characters to the right (use :set sw to change the <i>shiftwidth</i> ) |



<< Shifts line *shiftwidth* characters to the left (use :set sw to change the *shiftwidth*)

<L Shifts all lines from the line the cursor is on to the end of the screen *shiftwidth* characters to the left (use :set sw to change the *shiftwidth*)

---

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)  
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(Publisher: Macmillan Computer Publishing)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

Other vi Commands

In Table 7.9, commands that start with : (colon) are *ex* commands. If these are being executed from within the *ex* editor, you do not need the colon. When the ! modifier is included with a command, some form of override is performed. Not all combinations that include the ! are shown.

In the command column, the following words have the specified meaning: *file* means the name of a disk file; *number* or *nbr* means a positive whole number; *command* or *cmd* means a UNIX shell command; *tag* means a function identifier created using the *ctags* program; *addr* means an *ex* line address (defined in the “Using the Power of *ex* from *vi*” section following).

Table 7.9. Other *vi* commands.

*Saving the buffer to a file*

| Command | Result                                                          |
|---------|-----------------------------------------------------------------|
| :w      | Writes (saves) the buffer to disk, using the original file name |
| :w file | Writes the buffer to disk, to file                              |
| :w!     | Writes the buffer to disk, overwriting file                     |

| Command | Result                                                       |
|---------|--------------------------------------------------------------|
| ZZ      | Writes the buffer to disk and exits the program              |
| Q       | Enters the ex editor (same as typing :)                      |
| :q      | Quits vi, unless you have an unsaved buffer                  |
| :q!     | Always quits vi, overriding warning about an unsaved buffer  |
| :wq     | Writes the buffer to disk and exits the program (same as ZZ) |

*Editing other files*

| Command                  | Result                                                                                                                                                                                                                                |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| :e <i>file</i>           | Edits <i>file</i> , unless you have an unsaved buffer                                                                                                                                                                                 |
| :e!                      | Discards any changes and starts over with the last saved version of the file from disk                                                                                                                                                |
| :e + <i>file</i>         | Edits <i>file</i> , unless you have an unsaved buffer; places cursor on bottom line                                                                                                                                                   |
| :e + <i>nbr file</i>     | Edits file, unless you have an unsaved buffer; places cursor on line <i>nbr</i>                                                                                                                                                       |
| :e #                     | Edits alternate file                                                                                                                                                                                                                  |
| :n                       | Edits the next file (applies when a list of files was entered on the command line)                                                                                                                                                    |
| :n <i>file file file</i> | Sets up a new list of files to edit                                                                                                                                                                                                   |
| :r <i>file</i>           | Reads (inserts) contents of file into the buffer on the line below the cursor                                                                                                                                                         |
| :r ! <i>command</i>      | Runs the shell command and inserts the output of the command on the line below the cursor                                                                                                                                             |
| ^G                       | Displays information about the current file (filename, current line number, number of lines in file, percentage through the file)                                                                                                     |
| :ta <i>tag</i>           | Jumps to the file and the location in the file specified by <i>tag</i> (before you can use this function, you must use the <i>ctags</i> program to create the <i>tags</i> file. Refer to the section on the :ta command for details.) |

*Redrawing the screen*

| Command | Result |
|---------|--------|
|---------|--------|

|                      |                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------|
| <code>^L</code>      | Redraws the screen (implementation depends on terminal type)                                       |
| <code>^R</code>      | Redraws the screen; eliminates blank lines marked with @ (implementation depends on terminal type) |
| <code>znumber</code> | Sets screen window to <i>number</i> lines                                                          |

#### *UNIX shell commands*

| Command                  | Result                                                                                                                                                                                                                                         |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>:sh</code>         | Executes a shell; remain in shell until shell exit command given (^D)                                                                                                                                                                          |
| <code>:!command</code>   | Executes the shell command and returns to vi (After the ! command, certain special characters are expanded. # is expanded to the alternate file name; % is expanded to the current file name; ! is expanded to the previous shell command.)    |
| <code>:!!</code>         | Repeat the previous shell command                                                                                                                                                                                                              |
| <code>!object cmd</code> | Executes the shell <i>cmd</i> ; replaces the text <i>object</i> with the shell <i>cmd</i> output. If the shell <i>cmd</i> takes standard input, the designated text object is used.                                                            |
| <code>nbr!!cmd</code>    | Executed the shell <i>cmd</i> ; replaced <i>nbr</i> lines beginning at the current line with the shell <i>cmd</i> output. If <i>nbr</i> is missing, 1 is assumed. If the shell <i>cmd</i> takes standard input, the designated lines are used. |

#### *ex editing commands*

| Command                  | Result                                                                                      |
|--------------------------|---------------------------------------------------------------------------------------------|
| <code>:vi</code>         | Enters visual mode from the ex command line                                                 |
| <code>:addr d</code>     | Deletes the lines specified by <i>addr</i>                                                  |
| <code>:addr m nbr</code> | Moves the lines specified by <i>addr</i> after line <i>nbr</i>                              |
| <code>:addr c nbr</code> | Copies the lines specified by <i>addr</i> after line <i>nbr</i>                             |
| <code>:addr t nbr</code> | Copies the lines specified by <i>addr</i> after line <i>nbr</i> (same as <i>co</i> command) |

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)  
All rights reserved. Reproduction whole or in part in any form or medium without express written  
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Register for EarthWeb's  
Million Dollar  
Sweepstakes!



ITKnowledge

home

account  
info

subscribe

login

search

My  
ITKnowledge

FAQ/  
help

site  
map

contact us

To access the contents, click the chapter and section titles.

## Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

[Previous](#)

[Table of Contents](#)

[Next](#)

## Advanced Editing with vi: Tips and Techniques

You may be ready for any of the topics included in this section at any time while you are learning vi. Do not let the title of the section deter you from browsing for features that interest you. Although some vi commands are less often used, the real power of the vi editor will not be fully yours until you are comfortable with at least some of these features.

### Using the Power of ex from vi

As mentioned earlier, vi is actually the visual mode of the ex editor. As such, all the power and features of the ex editor are available at any time while editing in vi, without leaving your place in the file. vi commands that are actually ex commands are shown in the command reference tables and elsewhere in this chapter prefixed with a : (colon). You can think of this prefix in either of two ways: (1) as a prefix to a special vi command; or, (2) the command that takes the editor into ex mode, at which time the screen display changes so that a colon is displayed on the bottom line, and the cursor is placed immediately to the right of the colon. Thereafter, the editor acts exactly the same as if you were in the ex editor, except that you still have the vi screen displayed on all lines but the bottom line. Certain commands return you to the vi mode; others leave you in the ex mode. To return explicitly to the vi mode, just enter the :vi command.

The real power of using `ex` commands from within `vi` is that certain specific editing functions are provided in this way that are usually not available in most text editors. (The power of such operations is approached by macro languages included with PC-based word processing programs; however, the simplicity and elegance of the `ex` commands are not.)

The types of operations that are available only from the `ex` command line are using basic `ex` commands to manipulate blocks of text, search and replace operations, global search and replace with regular expressions, and edit multiple files.

## - Using Basic `ex` Commands to Manipulate Blocks of Text

`ex` has its own versions of delete, copy, and move commands. Sometimes these commands are preferable to the `vi` versions, particularly when you want to manipulate the file as a whole. The main `ex` commands for these operations are covered here.

First, let's look at the general form of an `ex` command. An `ex` command is composed of an object and an operation to perform on the lines in the file that are selected by the object. The general form is

```
:object command Return
```

where `Return` means to press the `Return` or `Enter` key. All `ex` commands require the `Return` key (which is labeled `Enter` on a PC keyboard) to be pressed. The spaces shown in the preceding example can be used if desired for readability, but they are not necessary.

Rather than using the `vi` concepts of a full-screen display with the cursor position to indicate where actions take place, `ex` has the concept of a current line. This concept means that `ex` takes its action on or relative to that line.

Both the *object* and *command* are optional. If the *object* is missing, the default is to apply the *command* to the current line. If the *command* is missing, the default is the `ex` print command, which displays the selected lines on the screen. Spaces between the parts of the command are also optional. In the examples, spaces are included for clarity.

When using the `ex` editor from the command line (that is, you are not running from within `vi`), `ex` responds to each command by displaying the lines affected. In the examples that follow, the behavior of the `ex` editor is shown with the assumption that you are using it from the command line, rather than from within `vi`. You might want to get a feel for the pure `ex` mode of interaction by trying it from the command line. When operating from within `vi`, the effect on the text you are editing is shown, just as if you executed the equivalent `vi` command. The screen display is repositioned if necessary.

The `ex` editor can be entered from within `vi` to edit the file you are currently editing by typing `:`. It can also be started from the command line with the name of the file you want to edit. If you start from the command line in this way, you might see the

following on your screen (assuming that you are editing the same sample file that we have been using all along):

```
$ex art1  
"art" 8 lines, 576 characters  
:
```

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)  
All rights reserved. Reproduction whole or in part in any form or medium without express written  
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪



Brief

Full

Advanced Search

Search Tips

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(*Publisher: Macmillan Computer Publishing*)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

### Selecting Lines to Edit

Sets of lines may be selected in several ways. Because you already know that ex is a line editor, it is not surprising that line addresses refer to lines in the file without regard to content. The simplest way to address a line is with its number. For example, to print line three of your file, you could enter the following command:

```
:3p
rewards and punishments. If humane, he loves mankind, sympathizes
with others,
```

Another way to give an address is with a pattern search. A pattern search is indicated by surrounding the exact character string you are looking for with forward slashes. For example, to display on the screen the first line of your file containing the word “general,” you could enter the following command:

```
:/general/p
Shen Pao-hsu ... said: 'If a general is not courageous he will be unable
```

In both of these preceding two examples, you could leave off the p command because as mentioned already, the p command is the default.

You may also specify a range of lines by entering two addresses separated by commas, as in the following example:

```
:3,6p
rewards and punishments. If humane, he loves mankind, sympathizes with
others, and appreciates their industry and toil. If courageous, he gains
victory by seizing opportunity without hesitation. If strict, his troops
are disciplined
```

Patterns also work in range selection, as in the following example:

```
:/humane/,/hesitation/p
rewards and punishments. If humane, he loves mankind, sympathizes with
```

others, and appreciates their industry and toil. If courageous, he gains victory by seizing opportunity without hesitation. If strict, his troops are disciplined

You can mix patterns and line numbers too:

`:4,/awe/p`  
and appreciates their industry and toil. If courageous, he gains victory by seizing opportunity without hesitation. If strict, his troops are disciplined because they are in awe of him and are afraid of punishment.

Special `ex` line addressing symbols can be used as addresses. Line addressing takes on greater flexibility when you add these capabilities to the ones you already know. The special symbols are shown in Table 7.10:

**Table 7.10.**Special `ex` line addressing symbols.

| Command         | Result                    |
|-----------------|---------------------------|
| <code>.</code>  | The current line          |
| <code>\$</code> | The last line of the file |
| <code>%</code>  | Every line in the file    |

Another feature to allow greater flexibility in line addressing is line number arithmetic. This feature allows you to use the `+` and `-` symbols along with numbers to refer to offsets from the position specified. For example, to refer to 20 lines from the current line number, you would use `.+20`.

When using two line addresses, the second address cannot be less than the first address. Sometimes when you try to use search patterns to select a line or line number arithmetic, you may get the error message `ex:` The first address cannot exceed the second address. This is because both line addresses are determined relative to the current line. In this case, you get an error message. What you really wanted was to have the second address be determined relative to the first address. `ex` has a feature that causes the second line address to be relative to the first. You use this feature by using a semicolon between the two addresses instead of a comma.

Table 7.11 shows several examples of the various methods of line addressing presented so far.

**Table 7.11.**Various methods of line addressing.

| Command                  | Result                                                                                                   |
|--------------------------|----------------------------------------------------------------------------------------------------------|
| <code>1,5</code>         | Lines 1 through 5                                                                                        |
| <code>.,20</code>        | From the current line to line 20                                                                         |
| <code>.,.+20</code>      | 20 lines beginning at the current line                                                                   |
| <code>.,+20</code>       | Another way of specifying 20 lines beginning at the current line (the second <code>.</code> is optional) |
| <code>.,\$</code>        | From the current line to the end of the file                                                             |
| <code>1,\$</code>        | All lines in the file (same as <code>%</code> )                                                          |
| <code>8,/pattern/</code> | From line 8 to the next line containing <i>pattern</i>                                                   |
| <code>5;.+20</code>      | From line 5 to 20 lines beyond line 5                                                                    |

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)  
All rights reserved. Reproduction whole or in part in any form or medium without express written  
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

Unix Unleashed, Third Edition

(Publisher: Macmillan Computer Publishing)

Author(s): Robin Burk

ISBN: 0672314118

Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Basic ex Commands

ex has the property that every command has a name. You can enter the full name of the command, or any length abbreviation of the command that sufficiently distinguishes it from all other commands. As new ex commands are introduced in this chapter, the full command name will be used first. However, the examples will use the shortest possible abbreviation of the command because that is the way you will want to use them. Table 7.12 shows a few of the basic ex commands:

Table 7.12. Basic ex commands.

| Command | Result                |
|---------|-----------------------|
| d       | Delete                |
| m       | Move                  |
| co      | Copy                  |
| t       | Copy (synonym for co) |

Table 7.13 presents examples of various ex editing commands. It summarizes the information given in this section on manipulating blocks of text.

Table 7.13. Examples of ex editing commands.

| Command | Result                    |
|---------|---------------------------|
| 1, 5d   | Deletes lines 1 through 5 |

|                            |                                                                                                   |
|----------------------------|---------------------------------------------------------------------------------------------------|
| <code>.,20m\$</code>       | Moves the current line through line 20 to the end of the file                                     |
| <code>.,.+20co0</code>     | Copies 20 lines beginning at the current line to the top of the file                              |
| <code>8,/pattern/t.</code> | Copies from line 8 to the next line containing <i>pattern</i> to the point after the current line |

---

In this section, some basic `ex` commands that are useful to extend the power of `vi` were introduced. Several additional `ex` commands provide an alternative way of doing various editing tasks. However, these tasks are more easily done by using the features of `vi`. Therefore, such commands are not covered here.

## Search and Replace

One of the main uses for `ex` commands from within `vi` (in addition to working with files and exiting the program) is to execute search and replace operations. In this section, basic search and replace operations are introduced. The next section introduces the topic of regular expressions.

- Regular expressions are extraordinarily powerful tools to search for text. If you are familiar with so-called “wildcard” searches offered by certain text manipulation tools, you can think of regular expressions as “wildcards on steroids”! The use of regular expressions for searching is covered in the second section following this one.

Simple search and replace operations are done in `ex` (and, therefore, in `vi`) by using the substitute command. Unless line addressing is used (see the following), the substitute command operates on the current line, so it is necessary to move the cursor to the line you want to edit first. The following example assumes that you want to substitute the word “opportunities” for the word “plan” in line 8 of the sample text. Note that the final slash is required:

```
:8                Position to line 8 of the buffer
:s/plans/opportunities/ Replace "plans" with "opportunities" on the
                        current line
```

You can also perform search and replace operations on the entire file, or a selected range of lines. Table 7.14 shows examples of using the substitute command to operate on all lines in the buffer, or a selected range of lines.

**Table 7.14.**Using the substitute command.

| Command                                | Result                                                                                                                       |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <code>:%s/warrior/general/g</code>     | Replaces every occurrence of <code>warrior</code> in the buffer with <code>general</code> .                                  |
| <code>:.,.+20s/warrior/general/</code> | Replaces the first occurrence of <code>warrior</code> with <code>general</code> on 20 lines beginning with the current line. |

---

Another way to search is to use the global command, as shown here:

```
:g/plans/command
```

When used in this way, the command is performed on all lines that match the pattern. To negate the action of the search (that is, to act on all lines that do not match the pattern), use `:g!`.

If the global command is used without the final slash and the command, the cursor is positioned to the last line in the file that contains the pattern (or does not contain the pattern, if ! is used). If no match is found, the screen does not change, and the cursor stays where it is. (There is little point to using the global command to search in this way—use the `vi`, `/`, or `?` instead.)

You can use the global command with line addressing to limit the scope of its action. Table 7.15 shows examples of using the global command with other commands.

**Table 7.15.**Using the global command with other commands.

| Command                     | Result                                                      |
|-----------------------------|-------------------------------------------------------------|
| <code>:g/22/d</code>        | Deletes all lines containing 22                             |
| <code>:g/plans/p</code>     | Displays all lines containing plans                         |
| <code>:g!/22/d</code>       | Deletes all lines not containing 22.                        |
| <code>:8,12g/plans/p</code> | Displays all lines between lines 8 and 12 containing plans. |

The global command can also be used to perform replacements. However, the real power of this command for replacements does not emerge until you begin to use it with regular expressions, which are explained in the next section.

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(Publisher: Macmillan Computer Publishing)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Regular Expressions

Regular expressions are patterns used in search and replace operations that vastly extend the power and flexibility of the editing you can do. Regular expressions include in addition to literal characters, combinations of so-called *metacharacters*, which have special properties. Table 7.16 shows all the metacharacters available for use within `vi`. (Although a number of UNIX tools can operate on regular expressions such as `grep`, `sed`, and `awk`, certain metacharacters shown in the table are not implemented for these other tools. Such `vi`-and-Emacs-only metacharacters are indicated by the comment “editors only.”)

Table 7.16. Metacharacters available with `vi`.

| Metacharacter | Matches                                                                      |
|---------------|------------------------------------------------------------------------------|
| .             | Any single character, except a newline.                                      |
| *             | Zero or more occurrences of the previous character.                          |
| ^             | When the first character of the regular expression, the beginning of a line. |
| \$            | When the last character of the regular expression, the end of a line.        |
| \<            | The first character of a word (editors only).                                |

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\&gt;</code>    | The last character of a word (editors only).                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>\</code>        | The escape character; alters (“escapes from”) the standard interpretation of the following character. For example, to search for the literal presence of a metacharacter, you must escape the character by preceding it with a backslash.                                                                                                                                                                                             |
| <code>[ ]</code>      | Any single character within the brackets; ranges may be used with a hyphen. For example, <code>[a-z]</code> matches all lowercase letters, <code>[a-zA-Z]</code> matches both lower- and uppercase characters. When metacharacters (other than <code>^</code> ) appear within square brackets, they do not need to be escaped. The literal hyphen can be included by placing it as the first character after the left square bracket. |
| <code>[ ^ ]</code>    | Any single character not within the brackets.                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>\( \)</code>    | In a search pattern, saves the text matched within the escaped parenthesis in a numbered buffer for later “replaying.” (The number is the position in the line as it is scanned from left to right; the first occurrence of this metacharacter pair is numbered 1, the second occurrence 2, and so on).                                                                                                                               |
| <code>\n</code>       | In a replacement pattern, where <i>n</i> is a digit from 1 to 9, “replays” the text saved by the escaped parenthesis.                                                                                                                                                                                                                                                                                                                 |
| <code>&amp;</code>    | Uses the entire search pattern which produced the match; used to save typing.                                                                                                                                                                                                                                                                                                                                                         |
| <code>\u or \l</code> | In a replacement pattern, causes the next character to be either upper- or lowercased.                                                                                                                                                                                                                                                                                                                                                |
| <code>\U or \L</code> | In a replacement pattern, causes the rest of the replacement pattern (or until a <code>\e</code> or <code>\E</code> is scanned) to be upper- or lowercased.                                                                                                                                                                                                                                                                           |
| <code>\e or \E</code> | In a replacement pattern, terminates the action of <code>\U</code> or <code>\L</code> .                                                                                                                                                                                                                                                                                                                                               |
| <code>~</code>        | Matches the search pattern of the last regular expression search.                                                                                                                                                                                                                                                                                                                                                                     |

---

### Caution:

The use of metacharacters might vary in different contexts. The shells use metacharacters for filename expansion; however, the interpretation of the metacharacters by the shells is slightly different from the interpretation by the utilities and text editors.

A shell is the name given to the UNIX command processor. There are several common versions of shell programs. The most common are `sh`, `csh`, `ksh`, and `bash`. Refer to Part II, “UNIX Shells” for an extensive discussion on this topic.

This can be a source of confusion, especially to the newcomer. To make matters worse, the implementation of metacharacters differs between different UNIX versions.

---



---

**Tip:**

All regular expression matches are limited to a single line. That is, a match that “wraps around” from the end of one line to the beginning of the next is not allowed.

---

---

**Tip:**

All regular expression searches are case sensitive. You have to explicitly use the features of the metacharacters to perform case-insensitive searches. For example, if you want to perform a search for the word “general” that is case-insensitive on the initial “g,” you should use `/[Gg]eneral/` for the search string.

---

---

**Tip:**

Regular expressions are usually delimited by forward slash (/) characters. However, any nonalphanumeric character other than “\”, “|”, or “#” can be used. This is especially helpful when the slash is one of the characters in the search string and you don’t want to escape the slash.

---

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#) All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(Publisher: Macmillan Computer Publishing)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Global Search and Replace with Regular Expressions

You have already seen how to perform searches from vi using the / and ? commands. As mentioned earlier, you can use these commands with regular expressions as well as with literal text strings. There are two ways to use regular expressions to perform search and replace operations. Both methods work from the ex command line, and both are extensions of commands you have been exposed to in the “Search and Replace” section. One way is to use the substitute command; the other way is through the use of the global command.

Using the Substitute Command with Regular Expressions

When you want to make a global replacement, you can use the substitute command with line addressing and regular expressions. A commonly used form is to use the % addressing symbol to refer to all lines in the file. The general form of such a command is as follows:

`:address s/searchexpression/replaceexpression/options`

(The space after address in the preceding example is for clarity and is optional.) The options refer to one of the options shown in Table 7.17.

Table 7.17.Substitute options.

| Command  | Result                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>g</b> | Makes the substitution global. (Without this option, the substitution only occurs on the first occurrence in the line; with it, all occurrences on the line are substituted. Do not confuse this option to the substitute command, which is placed at the far right of the command, with the global command itself, which occurs on the left of the command line near the colon.)                                                                |
| <b>c</b> | Confirm. <b>vi</b> displays each line found and indicates the text to be substituted with ^ symbols as follows:<br><pre>this is some text ^^^^</pre> <p>You must enter a “y” to make the substitution; any other response causes the substitution not to be made. (Note: Some versions of <b>vi</b> handle this a little differently, using text highlighting to indicate the pattern matched and allowing additional choices at each step.)</p> |

You may combine both of these options into one substitute command.

## Using the Global Command with Regular Expressions

- The global command becomes very powerful when combined with the substitute command. One interesting way to use this combination is to use the global command to select the lines and then use the substitute command to cause a change on the lines that does not directly relate to the text that caused the line to be selected.

In effect, the global command provides a two-step editing function. First, a set of lines is selected using several of various techniques (line addressing and pattern matching). Then another command is used on the lines selected. When the global command was first introduced in this chapter, it was used in a simple way with other commands to display or delete text. In fact, the global command can be used with most any other command. (Some creative techniques, indeed, have been invented that use the global command. Some are shown in the examples of Table 7.18.)

The best way to show how the substitute command works and how the global and substitute commands can be used together is to present some examples. Table 7.18 shows a few of the types of search and replace operations that can be done. (Note: For clarity, the bolded lowercase letter **b** is used to indicate a single blank space.)

**Table 7.18.**Examples of search and replace with the **s** and **g** commands.

| Command                  | Result                                                                   |
|--------------------------|--------------------------------------------------------------------------|
| <code>:%s/ex/vi/g</code> | Substitutes every occurrence of <b>ex</b> in the buffer with <b>vi</b> . |

|                                                |                                                                                                                                                                                                                              |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>:.,\$s/ex/vi/c</code>                    | Substitutes the first occurrence of <code>ex</code> with <code>vi</code> on every line from the current line to the end of the buffer, confirming each substitution.                                                         |
| <code>:%s/\&lt;author\&gt;/contractor/g</code> | Substitutes the word <code>contractor</code> for each occurrence of the full word <code>author</code> in the buffer; note that text objects containing “author” as a substring, such as “authority” will not be substituted. |
| <code>:g/editor/s/line/full-screen/g</code>    | Substitutes every occurrence of <code>line</code> with <code>full-screen</code> on all lines containing the pattern <code>editor</code> .                                                                                    |
| <code>:g/editor/s//word-processor/g</code>     | Substitutes every occurrence of <code>editor</code> with <code>word-processor</code> ; note that when the second search string is missing, the first search string is used; in this case, the string <code>editor</code> .   |
| <code>:%s/bb*/b/g</code>                       | Substitutes a single space for every occurrence of one or more spaces (note: the <code>b</code> stands for a single space).                                                                                                  |
| <code>:%s/[:.]bb*\([a-z]\)/\.\bb\u\1/g</code>  | Searches for all occurrences of a colon or a period followed by one or more spaces and a lowercase letter; substitutes a period, two spaces, and the uppercase form of the letter.                                           |
| <code>:g/^\$/d</code>                          | Deletes all blank lines (lines that have only a beginning followed immediately by the end).                                                                                                                                  |
| <code>:g/^/m0</code>                           | Reverses all the lines in the buffer.                                                                                                                                                                                        |
| <code>:g!/Complete/s/\$/ To be done/</code>    | Appends <code>To be done</code> to all lines not containing the string <code>Complete</code> .                                                                                                                               |

---

### Caution:

You need to have write permission to the file you are changing. `vi` does not warn you when you start, only when you try to save.

---

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc. All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(*Publisher: Macmillan Computer Publishing*)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

Working with Files

The basic commands for saving files were introduced in the “Starting Up and Exiting vi” section. This section more fully explains these commands and shows how to use them to work with more than one file at a time.

Saving Changes to a File

:w—Write

The `w` command is used to write the buffer to the current disk file. The current disk file is the one most recently loaded for editing, either from the command line when `vi` was started, or using the `:e` command. If there is no current disk file (perhaps because `vi` was loaded without specifying a file to edit), `vi` displays an error message and no action occurs. In this case, you can give a name to the current disk file with the version of the `:w` presented next.

Until the buffer is written to disk, all edits to the file are only stored temporarily. It is, therefore, a good habit to save your work frequently during your session to minimize the inconvenience of a system failure or major editing error that may occur.

:w filename—Write to filename

This version of the write command saves the buffer to the named *filename*. If there was previously no current file defined, the *filename* becomes the current file. Otherwise, the current file remains the same as it was before the `:w filename` command was issued.

If the *filename* file already exists, as usual `vi` warns you of this fact with an error message and gives advice on how to override the warning, as follows for a file name `art1`:

```
"art1" ex: The file already exists. Use w! art1 to force the write.
```

If you do in fact want to overwrite the existing version, you can use the syntax `:w! filename`.

### **`:address w filename`—Write addressed lines to *filename***

This version of the write command further refines the action of the `:w filename` version. The difference is that only the lines selected by the address are written. The same caveat regarding existing files applies.

### **`:address w >> filename`—Append addressed lines to *filename***

This version of the write command uses the UNIX redirect and append operator to add the addressed lines on to the end of an existing *filename* file. As you would expect, if you omit the address portion, the full buffer is appended.

## **Editing a Different File**

### **`:e filename`—Edit file**

Begins to edit the *filename* file. If the file does not exist, then a new file is started, and *filename* becomes the current file. (That is, a subsequent `:w` command writes the *filename* file.)

If the current file has been changed since the last time it was saved, the `:e filename` command is not allowed. Instead, `vi` gives the usual warning about the unsaved file, which may look as follows:

```
ex: No write since the file was last changed. The edit! command will
    force the action.
```

Once again, you could use `:e! filename` “to force the action.”

### **`:e!`—Revert**

This special use of the edit command forces the last saved version of the current file to be loaded for editing. This has the effect of reverting back to the last saved version, discarding all edits you made since then. (If you omit the exclamation point, in effect you are telling `vi` to begin editing the current file. If there were no changes yet, `vi` would oblige; however, this is a useless function. If there were changes, `vi` would do its normal routine of warning you that there has been “no write, etc.” So the only useful version of this command is with the exclamation point.)

### **`:e + filename`—Edit *filename* at end**

This command edits the *filename* file and places the cursor at the end of the file.

### **`:e + number filename`—Edit *filename* at line number**

This command edits the *filename* file and places the cursor on line *number*.

### **`:address r filename`—Import (read) *filename***

The read command allows you to import the full contents of another file into the buffer. If the optional address is present, the *filename* file’s lines are imported immediately after the line selected by the address. If the address is omitted, the lines are imported immediately after the line the cursor is presently on. The address can be a line number (with 0 indicating the top of the file), `$` to indicate the last line of the file, or a search pattern.

## **Editing More Than One File**

### **`:n`—Next file**

When `vi` is started, more than one file can be listed on the command line. For example, to edit the three files `art1`, `art2`, and `art3`, you would enter the following shell command:

```
$vi art1 art2 art3
```

`vi` loads the file `art1` as usual, and you may begin editing it. After saving the file, you can go on to the next file with the `:n` command. If you try to use the next command before you have saved the current file, you get the usual warning, which can be overridden in the usual way (yes, you guessed it—with `n!`). There is no “previous file” command.

## Alternating Between Two Files

`vi` keeps two filenames available via special symbols for use with `ex` commands. These are `#` for the alternate file, and `%` for the current file. After you have switched files with the `:e` command, the previous (“alternate”) file can be referenced via the `#` symbol. You can toggle back and forth between the two files by just typing `:e#` each time you want to switch.

The `%` symbol is mainly used to save typing within shell commands.

## Moving Text Between Files

You can use named buffers to copy or move text from one file to another. The method is just like copying or moving text from one part of a single file to another part of that file using the buffers as already described in the “How To Use Buffers” section. The difference is that after (say) a yank is done, an `:e` or `:n` command is used to change to another file. Then the put is done into the new file.

This technique works because when you change files with the `:e` or `:n` commands, the contents of all the buffers are retained. This is still a single `vi` session, which is why text can be moved between files using the named buffers.

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)  
All rights reserved. Reproduction whole or in part in any form or medium without express written  
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

.

Brief

Full

Advanced

Search

Search Tips

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(*Publisher: Macmillan Computer Publishing*)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

## Using the Power of UNIX from Within vi

You may have already seen in Chapter 4, “General Commands,” the general UNIX strategy for combining the functions of single-purpose or specialized tools to achieve results. `vi` shares in the capability to interact with the shell, filters, and utilities, and, in general, with any program that reads from the standard input and writes to the standard output. This section explains the various `vi` commands that make this possible.

### :sh—Shell

This `:sh` command invokes the shell. The shell that is run depends on your UNIX environment but can be controlled with the `sh` setting (see the “Changing `vi` Settings” section). You use this command when you want to temporarily leave `vi` to enter the shell and then return to the `vi` session at the point where you left off. You would usually enter the shell to run one or more shell commands, staying within the shell between each command. Any output to the terminal from the command you type is displayed normally, and the display of the `vi` session scrolls upward.

To return to your `vi` session, press `^D` (or use the proper convention for the shell that you are using). You are then returned to `vi`, with the screen still showing the output of your shell session. In addition, at the bottom of the screen, the text “Press return to continue” is displayed. After pressing Return, the screen is cleared and redrawn. You are then at the exact point in your editing session where you left off.

### :!—Shell

In some versions of `vi`, this command is the same as the `:sh` command. Other versions give an error message prompting you to use the `:sh` form.

### !:shellcommand—Shell Command

This command executes the `shellcommand` and returns to `vi`. Note that in the shell command, the `ex` special symbols for files (introduced in the section “Editing Multiple Files”) are expanded. These are `%` for the current file, and `#` for the alternate file. If you want to prevent these from being expanded, you must escape them with a `\` (backslash). The role of the escape character is explained in detail next, under “Repeat Shell Command.”



## **:!!—Repeat Shell Command**

This command repeats the last `:!shellcommand`. Note that any unescaped `!` that follows the `:!` command is expanded to the last shell command issued via `:!`. For example, if you enter the following sequence, you get a result similar to what follows (the text in *italics* are my comments):

```
:!ls                                to display your directory
    art1 art2 !test1 lsefair        contents of the directory (perhaps)

[Press return to continue]
:!ls -l !*                          attempt display of long listing for
                                     all files starting with !
-rw-rw-rw-  1 jas      system      2886 Nov  3 10:07 lsefair
[Press return to continue]         not what you wanted!
```

When you entered `:!ls -l !*`, the final exclamation point was expanded to the last command issued.

- Because the last command issued was `ls`, the actual command passed to the shell was `ls -l ls*`, where the `!` was replaced with `ls`. To prevent expansion of the `!`, you must escape it by preceding it with a `\` (backslash). The correct way to enter the command is `:!ls -l \!*`.

## **:address!*shellcommand*—Filtering text through a shell command (ex)**

This command works with `ex` line addressing to select some text to send to a shell command. The output of the command then replaces the selected lines. You could use this command, for example, to sort a section of your text. The following example shows how to sort lines 100 to 108:

```
:100,108!sort
```

Note that if the shell command does not take any standard input, then the selected lines are just replaced.

## **!object *shellcommand*—Filtering text through a shell command (vi)**

You can also select `vi` objects to filter, using certain of the keystrokes that are used to select objects. This technique does not work on anything less than a whole line, so you must either use keystrokes that cover more than a line, or use a count prefix to extend the object selected. The following example shows how to uppercase the next paragraph using the `tr` filter:

```
!}tr '[a-z]' '[A-Z]'
```

---

### **Tip:**

You can test a script that you've just written and saved (assuming that the file already has execute permissions) by using the following command:

```
:!%
```

`vi` substitutes the current filename for the percent sign (%).

---

When you use this technique, note that `vi` responds in a special way. First, when you type the initial exclamation point, nothing is displayed on the screen. After the keystrokes are typed to select an object, an exclamation point appears on the command line, but the keystrokes for the selected object do not. At this point, you can type the shell command, as shown previously. In addition, certain special features are available. If you type another exclamation point, it is expanded to the text of any previous shell command. So for example, if you wanted to continue to move around in your file and uppercase various text objects after typing the string in the previous example, you could just type `!object!` each time after positioning to the chosen spot.

## **count!!*shellcommand*—Filtering text through a shell command !count!*shellcommand***

These two commands are equivalent. They are variations on the preceding `vi` text filtering command. They apply to *count* number of lines relative to the line the cursor is on. The *count* is optional; if absent, the current line is processed.

## Marking Your Position

It is often necessary when editing larger files to move back and forth between specific points. Many editors implement the concept of a *bookmark*, which is a mechanism for marking a place in your buffer and then allowing an easy way to return to that exact place. In `vi`, this concept is implemented in several ways.

### **mx—Mark**

This command marks the position that the cursor is presently at with the letter *x*, where *x* is any letter. The mark is not visible on the screen.

### **`x (back quote)—Move to mark x**

The ``x` (back quote) command moves the cursor to the exact position marked by the letter *x*.

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), Copyright © 1996-2000 EarthWeb Inc.  
All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪

Brief

Full

Advanced Search

Search Tips

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(*Publisher: Macmillan Computer Publishing*)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

Previous

Table of Contents

Next

**‘x (apostrophe)—Move to beginning of line of mark x**

The ‘x (apostrophe) command moves the cursor to the beginning of the line marked by the letter x.

**`` (two back quotes)—Move to previous location**

This command moves to the exact position before the previous repositioning of the cursor via pattern search, G command, or move to a mark.

**‘‘ (two apostrophes)—Move to beginning of line of previous location**

Moves to the beginning of the line of the cursor position before the previous repositioning of the cursor via pattern search, G command, or move to a mark. You can also use marks with addresses such as:

: 'a ,+20w foo

**For the Power User: Customizing vi**

If you have stuck with this chapter this far, you are probably looking for the “Power User” features. Here they are!

**Automation of Common Chores**

As you become more familiar with an editor, you may find yourself seeking to automate certain keystroke sequences that seem to be occurring over and over. There are two ways to do this in vi: abbreviations and key mappings. Both of these methods are described in the following sections.

**Tip:**  
Although the abbreviation feature is used primarily to save retyping common strings, another useful technique can be used with the feature: correcting common mistakes. Just set up as an abbreviation the mistyped word.

For example, because I am in the habit of typing “wrok” when I mean “work,” and “flies” when I

mean “files.” I could set both of these as abbreviations and see `vi` automatically change my mistake into the correct word.

However, do not overdo this—as you can see, if I used “flies” as an abbreviation for “files,” then I lose the ability to use the word “flies.”

---

## Abbreviations

`vi` has the built-in capability of using abbreviations that you set up. For example, suppose that you are often found typing the phrase “SAMS Publishing, an imprint of Macmillan Computer Publishing USA.” You could abbreviate this phrase for example as `SAMS`, with the following command:

```
:ab SAMS SAMS Publishing, an imprint of Macmillan Computer Publishing USA
```

`vi` expands the abbreviation while you are in insert mode into the full text after you have typed the abbreviation plus either a space or a punctuation character. Note that the abbreviation is case-sensitive, so you can retain the capability to omit the expansion by a judicious choice of upper- and lowercase characters in the abbreviation.

To see all the abbreviations currently in effect, just enter `:ab` with no arguments (that is, by itself). To eliminate an abbreviation (perhaps so that you can include the literal characters of the abbreviation itself), use the `:unab` command.

## Creating Macros with the `:map` Command

The `:map` command is used in a similar way as an abbreviation. The `:map` command allows you to assign command mode keystroke sequences to a single (or multiple) keystroke sequence. Such commands are often called keyboard macros. Often `:map` commands are used to assign keystrokes available on your keyboard (such as the cursor control arrow keys) to the desired `vi` commands (`h`, `j`, `k`, and `l`).

A `:map` command is entered just like an abbreviation. For example, to map the command sequence `dwelp` (which reverses the order of two adjacent words) to the letter “v,” enter this command:

```
:map v dwelp
```

To use a control character in the mapping, you must precede the entry of that control character with a `^V`. Otherwise, `vi` immediately tries to interpret the control character you are typing. The `^V` tells `vi`, in effect, “do not interpret the next character, just enter it into text.” When you type the `^V`, only the `^` (caret) shows. For example, when you want to insert an Esc character into the mapping, you would type `^V` followed by the Esc key. Suppose that you wanted to map the “v” key to the Esc key. What you would see on the screen after typing the `^V` would be the following:

```
:map v ^
```

---

### Note:

The `=` key is not available if the `lisp` setting is in effect.

---

Then when you pressed the Esc key, your screen would look like this:

```
:map v ^[
```

After pressing Return, your mapping would be in effect.

To remap certain keys that might be on your keyboard to the equivalent function in `vi`, you may not even

need to know what character sequence is assigned to your keyboard. Just use the `^V` key; then type the key that you want to assign. This might not work, depending on the terminal mapping in effect at your terminal. See your system administrator for help if you are having trouble setting up your keyboard mappings.

It is useful to see what keys are not already used by `vi` and are, therefore, available for mappings without losing any functions (there are, in fact, a few keys left unused). They are as follows: `g`, `K`, `q`, `V`, `v`, `^A`, `^K`, `^O`, `^T`, `^W`, `^X`, `_`, `*`, `\`, `=`.

To remove the mapping, use the `:unmap` command. To see all mappings currently in effect, use the `:map` command with no arguments.

Other creative uses for the `map` command are shown in Table 7.19, which shows several examples of both the `:ab` and `:map` commands.

**Tip:**  
In addition to using the `0` and `$` commands to see where lines begin and end, you can also use line numbering with `:set nu` to see how `vi` is assigning line numbers. Another technique that might help is using `:set list`, which shows tabs and line ends with `^I` and `$`, respectively.

**Table 7.19.**Some examples of the `ab` and `map` commands.

| Command                        | Result                                                 |
|--------------------------------|--------------------------------------------------------|
| <code>:ab L1 Level One</code>  | Sets <code>L1</code> to expand to “Level One”          |
| <code>:ab</code>               | Displays all abbreviations                             |
| <code>:unab L1</code>          | Removes the expansion of <code>L1</code>               |
| <code>:map v dwelp</code>      | Reverses two adjacent words (not at the end of a line) |
| <code>:map ^A i"^[ea"^[</code> | Surrounds a word with quote characters                 |

To access the contents, click the chapter and section titles.

**Unix Unleashed, Third Edition**  
(*Publisher: Macmillan Computer Publishing*)  
Author(s): Robin Burk  
ISBN: 0672314118  
Publication Date: 12/30/98

Bookmark It

Search this book:

### Changing vi Settings

Until this section, I have been describing the default behavior of `vi` under various circumstances. You can modify these default behaviors by changing the `vi` settings. The easiest way to understand these settings is with an example. One such default behavior is what `vi` does while you are inserting text, and you get to the end of the line. The default behavior is to continue adding text to the line; however, the display gives the appearance of wrapping to the next line. You can confirm this by using the `0` and `$` commands from command mode to see where `vi` considers lines to begin and end. You can alter this default behavior to suit your requirements.

If you are typing the text of a memo for example, you might want `vi` to automatically wrap lines. By using the setting `wrapmargin`, (abbreviated as `wm`), you can cause `vi` to insert a line feed automatically when you get close to the end of the line (the behavior of the `wm` setting is described later). If, on the other hand, you are editing a source code file, most likely you would want there to be no such wrapping, so the default behavior is what you want.

The `vi` settings can be modified in four ways, as shown in Table 7.20.

**Table 7.20.** Ways to modify default `vi` characteristics.

| Command | Result |
|---------|--------|
|---------|--------|

---

|              |                                                                                                                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| set commands | Executed during a vi session to temporarily change the characteristic, until they are explicitly changed again or the vi session ends.                                                                                       |
| .exrc file   | Changes included in the .exrc file go into effect whenever the editor is started, until overridden during the session by a set command.                                                                                      |
| EXINIT       | An environment variable that can be used just like the .exrc file. (If the same setting is changed by both an entry in an .exrc file and the EXINIT environment variable, the setting from the .exrc file takes precedence.) |

---

The methods are listed in precedence order, from highest to lowest. This means that any of the commands overrides the effect of the same command of a type shown lower in the table. For example, a set command overrides the same command from the .exrc file, which in turn overrides the effect of the same command from the EXINIT environment variable).

There are two types of vi settings. The first is a toggle setting, which may be either on or off. To put the setting into effect, you would use the command `:set option`, where *option* is the desired setting to put into effect. To remove the setting from effect, you use the command `:set nooption`, where *nooption* is the same name as you would use to put it into effect with the characters “no” preceding the name.

The other type of setting is a numeric setting, which has a numeric value. For example, the `wrapmargin` setting may have a value of 10. To use a numeric setting, you would use the command `:set option=x`, where *option* is the name of the setting, and *x* is the numeric value. For example, use `:set wrapmargin=10` to set the `wrapmargin` setting to a value of 10.

---

**Note:**

In System V UNIX, the feature to look for an .exrc file in the current directory is only enabled if the `exrc` setting is enabled (it is off by default). This setting is a security feature. An attack on the security of a system could be made by planting an .exrc file in a directory that causes some unintended and undesirable action.

---

Many of the settings have abbreviations. For example, you have already been introduced to the `wrapmargin` abbreviation of `wm`.

The .exrc file is a plain text file located in the user’s home directory. It consists of `set`, `ab`, and `map` commands, entered one per line. (The `ab` and `map` commands were described in the “Automation Of Common Chores” section.) If such a file exists, vi reads it immediately after starting and the commands it contains will be placed into effect just as if you had typed them from the `ex` command line before starting to edit. The following shows an example of an .exrc file:

```
set wrapmargin=10 nowrapscan  
ab SAM SAMS Publishing
```

With the preceding example as your `.exrc` file, the `wrapmargin` is set to 10, `wrapscan` is turned off, and an abbreviation is created every time you start `vi`. In addition to reading the `.exrc` file in your home directory, `vi` tries to read an `.exrc` file in the current directory you are in when you start `vi`. If such a file is found, the entries there are added to the ones put into effect from the `.exrc` file in your home directory. This feature allows you to have your preferred settings for all `vi` sessions, plus special tailoring for separate projects.

You can see what settings you have placed in effect by entering `:set` without any options. By entering `:set all`, the value of all settings is displayed.

|                          |                                   |                      |
|--------------------------|-----------------------------------|----------------------|
| <a href="#">Previous</a> | <a href="#">Table of Contents</a> | <a href="#">Next</a> |
|--------------------------|-----------------------------------|----------------------|

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

---

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)  
All rights reserved. Reproduction whole or in part in any form or medium without express written  
[permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

▪