

THE EXPERT'S VOICE® IN OPEN SOURCE

From Bash to Z Shell

Conquering the Command Line

Oliver Kiddle, Jerry Peek,
and Peter Stephenson

Apress®

Praise for *From Bash to Z Shell: Conquering the Command Line*:

"Some areas are covered in other books, but this one goes into some little seen side streets and alleyways to show you the shortcuts to more efficient use of the shell."

—Raymond Lodaro, Slashdot contributor (www.slashdot.org)

"The material here is invaluable: you're not going to get it from the manual pages! If you work on UNIX systems, or if you'd like to make your Windows environment vastly more powerful, you need this book. I strongly recommend it."

—Ernest J. Friedman, JavaRanch (www.javaranch.com)

"This is a totally neat idea for a book . . . the command line gets addictive quickly."

—Bill Ryan, Bill's House Of Insomnia (<http://insomaps.com/~williamryan/>)

"This! is one of the best books I've read on the Bash and Z Shell, and is highly recommended to anyone learning to work with either of these two of the most popular of all Linux shells."

—Harold McFarland, Midwest Book Review (www.midwestbookreview.com)

From Bash to Z Shell: Conquering the Command Line

OLIVER KIDDLE, JERRY PEEK, AND PETER STEPHENSON

Apress®

From Bash to Z Shell: Conquering the Command Line

Copyright © 2005 by Oliver Kiddle, Jerry Peek, and Peter Stephenson

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN (pbk): 1-59059-376-6

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

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Jason Gilmore

Technical Reviewers: Bart Schaefer and Ed Schaefer

Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, John Franklin, Jason Gilmore, Chris Mills, Dominic Shakeshaft, Jim Sumser

Project Manager: Beth Christmas

Copy Edit Manager: Nicole LeClerc

Copy Editor: Liz Welch

Production Manager: Kari Brooks-Copony

Production Editor: Laura Cheu

Compositor: Susan Glinert

Proofreader: Linda Seifert

Indexer: Kevin Broccoli

Artist: Kinetic Publishing Services, LLC

Cover Designer: Kurt Krames

Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013, and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders@springer-ny.com, or visit <http://www.springer-ny.com>. Outside the United States: fax +49 6221 345229, e-mail orders@springer.de, or visit <http://www.springer.de>.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

Contents at a Glance

About the Authorsxi
About the Technical Reviewersxiii
Acknowledgments	xv
Preface	xvii

PART 1 ■■■ Introducing the Shell

CHAPTER 1	Introduction to Shells	3
CHAPTER 2	Using Shell Features Together	27
CHAPTER 3	More Shell Features	53

PART 2 ■■■ Using bash and zsh

CHAPTER 4	Entering and Editing the Command Line	71
CHAPTER 5	Starting the Shell	107
CHAPTER 6	More About Shell History	123
CHAPTER 7	Prompts	141
CHAPTER 8	Files and Directories	167
CHAPTER 9	Pattern Matching	197
CHAPTER 10	Completion	231
CHAPTER 11	Jobs and Processes	261

PART 3 ■■■ Extending the Shell

CHAPTER 12	Variables	279
CHAPTER 13	Scripting and Functions	307
CHAPTER 14	Writing Editor Commands	347
CHAPTER 15	Writing Completion Functions	371

APPENDIX A	Unix Programs	409
APPENDIX B	External Resources	415
APPENDIX C	Glossary	417
INDEX	425

Contents

About the Authorsxi
About the Technical Reviewersxiii
Acknowledgments	xv
Preface	xvii

PART 1 ■■■ Introducing the Shell

■ CHAPTER 1	Introduction to Shells	3
	What's a Shell?	3
	Getting Started	6
	Simple Commands	7
	Command Lines	8
	Shell Types and Versions	14
	The Filesystem: Directories, Pathnames	15
	Where the Commands Are Located	17
	Relative Pathnames and Your Current Directory	18
	Building Pathnames with Wildcards	20
	Expansion and Substitution	21
	Building Pathnames by Completion	22
	Command-Line Editing	24
	Command History	24
	More Unix Programs	26
	Summary	26
■ CHAPTER 2	Using Shell Features Together	27
	Writing Output to Files: Redirection	28
	Using Programs Together: Pipes	32
	Joining Forces	34
	Editing Data, Continuing Command Lines	36
	Command Substitution	38
	for and foreach Loops	39

More About for Loops	41
Building Our Script.....	43
Making a Script File.....	46
Running the Script.....	47
Programs and the Path	49
Using the Results (via Aliases and Functions)	51
Summary	52

CHAPTER 3 More Shell Features

Passing Info to Processes with Environment Variables.....	53
Starting and Stopping Processes: Signals, Job Control.....	56
Control Shell Interpretation with Quoting	62
Quick Directory Changes with <code>cdpath</code>	65
Summary	66

PART 2 ■■■ Using bash and zsh

CHAPTER 4 Entering and Editing the Command Line

Terminals and Shells.....	71
The Command Line	76
Line Editing Basics.....	76
Beyond Keystrokes: Commands and Bindings.....	87
Options for Editing	98
Multiline Editing and the <code>zsh</code> Editor Stack.....	99
Keyboard Macros.....	101
Other Tips on Terminals	102
Summary	106

CHAPTER 5 Starting the Shell

Starting Shells	107
Startup Files, Login and Interactive Shells	108
Shell Options	113
Getting Started with <code>Cygwin</code>	118
Summary	121

CHAPTER 6	More About Shell History	123
	Setting Up Variables	123
	“Bang” History: The Use of Exclamation Marks	124
	More Options for Manipulating History	133
	A Few More History Tricks	137
	Summary	139
CHAPTER 7	Prompts	141
	Basic Prompting	141
	Prompts in bash	143
	Prompts in zsh	150
	Checking for Mail and Idle Terminals	162
	Summary	165
CHAPTER 8	Files and Directories	167
	Types of Files	167
	Finding Commands and Files	173
	Managing Directories with the Shell	178
	More Argument Handling: Braces	185
	Redirection	187
	Here-Documents and Here-Strings	190
	Summary	195
CHAPTER 9	Pattern Matching	197
	Basic Globbing	197
	Internationalization and Locales	206
	Globbing in Bash	208
	Globbing in Zsh	211
	Glob Qualifiers in Zsh	220
	Globbing Flags in Zsh	228
	Summary	229

CHAPTER 10 Completion	231
Getting Started with Completion	232
Listing and Formatting Possible Matches	235
Types of Completion	242
Controlling Matching	248
Reducing the Number of Matches	251
Automatically Added Suffixes	257
Exact Ambiguous Matches	259
Summary	260
CHAPTER 11 Jobs and Processes	261
Mastering Job Control	261
High-Power Command Substitutions	266
Resource Limits	271
Lying About the Program Name	274
Summary	275

PART 3 ■■■ Extending the Shell

CHAPTER 12 Variables	279
Arrays	280
Variable Attributes	289
Numeric Variables and Arithmetic	290
Complex Variable Expansions	293
Associative Arrays	300
Variable Indirection	303
Summary	306
CHAPTER 13 Scripting and Functions	307
Programming with the Shell	308
Input and Output	322
Propagating Functions	333
Traps and Special Functions	335
Defining New Globbing Qualifiers	338
Debugging Scripts	343
Summary	345

CHAPTER 14	Writing Editor Commands	347
	Widgets and Functions	347
	Simple Recipe for a Widget	348
	Making Widgets Behave Naturally	352
	Case Study 1: Multiple Tasks in Widgets	355
	Case Study 2: Overriding Widgets	357
	Case Study 3: Locating Command-Line Arguments	359
	Case Study 4: Chaining Widgets Together	360
	Approximate Matching	361
	An Example Widget: Correcting Spelling	362
	Input and Output Within Widgets	364
	Summary	368
CHAPTER 15	Writing Completion Functions	371
	Completions	371
	Helper Functions	382
	Handling Styles	392
	Making Full Use of Tags	395
	Tags, Labels, and Descriptions	400
	Prefixes and Suffixes	403
	Stand-alone Completion Widgets	405
	Summary	408
	And Finally	408
APPENDIX A	Unix Programs	409
APPENDIX B	External Resources	415
	bash	415
	zsh	415
	General	416
APPENDIX C	Glossary	417
INDEX		425

About the Authors



■ **OLIVER KIDDLE** was first introduced to Unix systems while studying at the University of York. Since graduating in 1998, Oliver has worked as a software developer and system administrator. Over the past five years, Oliver has been actively involved with the development of the Z shell.



■ **JERRY PEEK** is a freelance writer and instructor. He has used shells extensively and has taught users about them for over 20 years. Peek is the “Power Tools” columnist for *Linux Magazine* and coauthored the book *Unix Power Tools* (O’Reilly Media).



■ **PETER STEPHENSON** grew up in northeast England and studied physics at Oxford. After nine years as a researcher in computational physics, he became a software engineer with Cambridge Silicon Radio, where he now works on short-range digital radio. He has been involved with zsh since the early 1990s and currently coordinates its development.

About the Technical Reviewers



BART SCHAEFER has served as a key architect and senior developer of e-mail systems for more than 15 years, creating flexible and scalable solutions with an emphasis on open standards. Before cofounding iPost he was a founder of Z-Code Software, whose groundbreaking multiplatform e-mail application, Z-Mail, won numerous awards. Dr. Schaefer contributes regularly to open software projects, including SpamAssassin and zsh. He holds a PhD in computer science from the Oregon Graduate Institute, focusing on automated process distribution

and scheduling for massively parallel computer systems, and a BSS in computer science from Cornell College.



ED SCHAEFER is an ex-paratrooper, an ex-military intelligence officer, and an ex-oil field service engineer. He's not a total has-been. Presently, he's a software developer and DBA for a Fortune 50 company—a Unix island in a sea of Windows. He's also a contributing editor to *Sys Admin*, *the Journal for UNIX and Linux Systems Administrators*, and edits *Unix Review*'s monthly "Shell Corner" column at <http://www.unixreview.com>.

Acknowledgments

The authors would like to thank Martin Streicher for the initial concept behind this book and for bringing us together at the start of the project. Thanks also to the Apress staff and our technical reviewers who've done so much work "behind the scenes" to bring this book to you. We authors did only a small part of the job.

Jerry Peek's portrait is by Meredith Hayes.

Preface

A shell is a sophisticated way to control your computer—Unix, Linux, Microsoft Windows, Mac OS X, and others. If you do more than the most basic operations, you can do many of them more powerfully and quickly with a shell and your keyboard than by using a mouse.

The history of shells goes back some 30 years. In the early days of the Unix operating system, choosing and customizing your interface to a computer was a new idea. (It still *is* new to many people today, users of “one-size-fits-all” window systems that can be changed only superficially.) Before windows and a mouse were common, programmers began developing an interface that used the keyboard: typing one or a few words to run programs, then reading results from the same screen. As time went on, more shells were developed, giving users more choices.

New features have been added continually over the years, making the modern shell an incredibly rich environment that saves power users hours of time and frustration. Tasks that take lots of repetitive work with a mouse can be automated. For example, shell features such as *completion* let you accomplish a lot with little typing.

A shell can work in two ways. You can use it interactively to do things by hand. You can also automate a task by packaging those same operations into a *script* or *function*. Learning shell features lets you do both of those because a shell is a user interface and a programming language in one.

The shells we discuss run on many operating systems. What you learn about shells will let you use all of these operating systems in the same way. If you use more than one operating system, a shell gives you a powerful and familiar interface to all of them.

There are several major shells. Because each has its differences, covering all of the shells could make a book that’s both confusing and unwieldy. We’ve concentrated on bash and zsh, two of the most modern and powerful shells. Both are freely available; in fact, they’re installed on many of the systems we’ve listed and can be downloaded from the Internet for the rest.

- bash is the de facto standard shell on Linux. bash runs most scripts written for other Bourne-type shells, including the original Unix shell sh, and it has a growing list of features.
- zsh, also called Z shell, is an extremely powerful shell that’s not as well known as bash. zsh combines most of the best features of several shells, including C-type shells such as tcsh. However, its basic usage is similar to bash.

This book provides the first comprehensive Z shell coverage that we know of. If you consider yourself a power user (or if, after reading what shells can do, you want to *become* a power user!), you owe it to yourself to get familiar with all that zsh can do to make your work easier.

Covering both bash and zsh shows you what features the two shells have in common as well as their different approaches to the same tasks.

How This Book Is Structured

This book is divided into three parts consisting of 15 chapters. Part 1, *Introducing the Shell*, contains Chapters 1 through 3. Part 2, *Using bash and zsh*, is made up of Chapters 4 through 11. Part 3, *Extending the Shell*, includes Chapters 12 through 15. The book also has three appendices: a list of Unix-like commands, a list of resources, and a glossary. In this section we offer a brief introduction of each chapter.

Chapter 1: Introduction to Shells

This chapter covers the highlights of shells. Topics include: what a shell is, how to start one, the parts of a command line, running simple Unix commands and getting help with them, an introduction to the filesystem and how to use files by typing only part of their names, how the shell finds the programs you need, how the shell processes command lines, and recalling and editing command lines.

Chapter 2: Using Shell Features Together

Here we introduce several important features, including rerouting a program's input and output, using utility programs to edit text automatically, running loops (repeating a series of commands), handy command-line techniques, and more. The emphasis, though, is on one of the capabilities that make shells so useful: that you can combine programs, together with features of the shell, to do completely new things.

Chapter 3: More Shell Features

This chapter moves more slowly through several other major shell topics: passing information between programs, managing processes, using quoting to control how the shell interprets a command line, and a time-saving technique for moving through the filesystem.

Chapter 4: Entering and Editing the Command Line

When you use a shell interactively, you tell it what to do by typing commands on its command line. (When programming the shell, as we'll discuss in Chapter 13, you put those same commands into a shell function or a file.) Chapter 4 covers the command line in depth—including how to fix errors and how to save time by reusing previous command lines. This chapter also covers the interaction of a shell with its window (a terminal).

Chapter 5: Starting the Shell

A shell can be customized to work the way you want it to by setting its options and variables, by installing your own commands, and more. You can do this interactively (from the command line) after the shell has started. You can also customize the shell automatically each time it starts. Chapter 5 shows how.

Chapter 6: More About Shell History

Chapter 4 introduced shell history—a remembered list of previous command lines. Chapter 6 describes how to save, recall, and share history between shells. It also covers a way to let you reuse parts of previous command lines in later ones.

Chapter 7: Prompts

When the shell needs to ask you something (for instance, “Do you want to change the spelling of this word?”) or tell you something (such as “You can enter a command line now”), it prompts you. Like almost everything about the shells, the prompt can be customized to show information you want, as you want it. This chapter shows you how.

Chapter 8: Files and Directories

One of a shell’s greatest strengths is the power and control it gives you for working with files. This chapter highlights some useful information about files on a Unix-type system such as Linux and Mac OS X (also under Microsoft Windows, with Cygwin). Next it discusses how to refer to and use files from the shell.

Chapter 9: Pattern Matching

This chapter carries on from Chapter 8, showing one of the most useful and work-saving techniques in a shell: finding one or many files by their names and other characteristics. After seeing these powerful techniques, you might wish that the file-handling menus on your graphical applications (word processors, for instance) had a shell built in.

Chapter 10: Completion

One handy shell feature that *is* available on some graphical file-handling menus is filename completion: the ability to type the first few characters of a filename and have the remaining characters completed for you. As this chapter shows, modern shells have extended this basic idea in many ways. This chapter covers the many aspects of bash’s and zsh’s completion systems and the ways in which their behavior can be configured.

Chapter 11: Jobs and Processes

Shells let you control multiple programs (multiple processes) from a single terminal: starting them, suspending them, ending them early, and more. The shell lets you control some of the resources that a process uses—the amount of memory, for example. And, shells being the flexible tools that they are, there’s more.

Chapter 12: Variables

Variables are places to store information within the shell and to pass information between the shell and other programs (processes). Variables are used to customize the shell and to keep something you want to reuse later (like a filename), and they are especially useful if you’re programming.

Chapter 12 shows how data stored in variables can be manipulated. In addition, Chapter 12 shows how to use the shell's built-in math facilities.

Chapter 13: Scripting and Functions

The shell implements a full programming language. Chapter 13 covers features like loops and condition tests, which allow power-of-things to be done just from the command line. Often, however, it can be useful to save a set of commands for later reuse—including some commands you just ran from the command line.

The `alias` and `alias` capabilities of the shell allow it to let you use the same language for controlling your system interactively as for writing programs (to do things automatically). This, as we've said, one of the great things about shells. The focus of Chapter 13 is on how you can write programs with the shell and how you can use these programs to extend the basic functionality of the shell.

Chapter 14: Writing Editor Commands

The Z shell has a completely configurable editor built in. This chapter explains how you can add new commands to your `zsh` editor.

Chapter 15: Writing Completion Functions

Chapter 10 shows how completion works "out of the box." If that's not enough for you, both `ksh` and `zsh` also let you write your own custom completion definitions. Chapter 15 explains how.

Who Should Read This Book

Although shells are sophisticated, they aren't just for experts. For those of you without considerable shell experience, we've carefully chosen topics and a manner of instruction that will enable you to immediately begin using shells at a new level of proficiency. In particular, Part 1 of this book will help prepare you for some of the more advanced topics that follow throughout the remainder of the book.

Expert users interested in maximizing their already efficient use of the command line will find the hundreds of tips, tricks, and hidden gems that we present throughout the book quite useful. Based on our years of experience immersed in command-line interaction, we're well aware of the features that can even further improve your shell proficiency, and condense that knowledge into this book.

Prerequisites

This book covers `ksh` version 3.0 and `zsh` version 4.2. Most of this material applies to either versions of the two shells—especially `tcsh` 6.04 and `zsh` 4.0—and the concepts apply to other shells as well.

You can download the latest versions of both of these shells—as well as the `Cygwin` package you'll need for Microsoft Windows—free on the Internet. If you haven't read `Free as in Beer`, available at www.fsf.org, don't be concerned: this software is the highest quality, maintained by groups of professional programmers who want the very best software for their own use.

There's more information about the software and where to get it in Appendix B.

Tips for Reading Technical Material

If you've found that reading technical topics can be a challenge, here are tips that may help:

- **Reading technical material is different than reading a novel:** In technical writing, the authors aren't trying to disguise secrets or surprise you. They're giving you information as clearly as they can. Instead of trying to obscure the clues to the "mystery," they're laying them out in front of you. But even words spelled out clearly don't always mean that a concept will be obvious at first.
- **Put on your detective's cap:** If the puzzle doesn't seem to be coming together, go back and see what the missing parts are. This can take some time and effort; learning something new isn't always easy! But, like reading a good mystery novel, finding the answer—putting the puzzle together—is well worth the time you spend.
- **If something's missing, find it:** Read each paragraph, or each group of a few paragraphs, then be sure the new concepts make sense before you go on. Sometimes they just won't make sense; there might be a missing piece to the puzzle. If that happens, try going back and reviewing what you've read before.
- **Check for understanding as you go:** Most sentences aren't "fillers"; we're trying to make a point, either to introduce a new idea or to tie some ideas together. For instance, after you've read a while and learned some new concepts, you might see a sentence like this:
 The list of directories comes from the standard output of `tr`.
 It's best not to just say "Oh, umm-humm" and keep reading. Instead, you should ask yourself a question like "On the basis of what I've read before, does that sentence make sense?" or "Do I agree with what the sentence says?" You could also ask yourself about each part of the sentence, like "What list?" or "What's a directory?" or "What's the standard output again?" or "What does `tr` do and why does it write to its standard output?" If you aren't sure, don't read too much more before you go back to hunt for the missing clues.
- **Talk it over:** If you have some questions, and you have some computer-literate friends, talk to one or two of them. Discuss the problem—and, if you'd like to learn more than just the answer, discuss the topic in general. Explaining a problem to someone else, and being sure that each of you understands the other, is a great way to increase understanding. (This is true even if your friends don't know about shell scripts; that they won't only be impressed at what you're doing, soon your friends will be studying shells, too—and possibly asking you for advice!)
- **Please experiment:** Experimenting in your session is a great way to learn and to check your understanding. If that is, you're careful about programs that could do damage: like `rm`, a program that removes files. For instance, as you read through a section, take some of the previous examples and change them slightly to see how that affects the results... then be sure you understand why.

We don't assign formal exercises in this book, but you could come up with a practical problem and see how to handle it. As an example, after you've learned how to remove many files at once by using a wildcard, create a lot of dummy files and try to remove them.

Contacting the Authors

You can send e-mail to the authors at shellbook@jpeek.com. While we can't promise to answer every message, we will do our best, and you can be sure that we *will* read every one. Thanks in advance for your messages.

We've listed some good places to get more information in Appendix B.

- [**download online The Saturated Self: Dilemmas Of Identity In Contemporary Life**](#)
- [click Self Leadership and the One Minute Manager](#)
- [Web Designer \[UK\], Issue 236 pdf, azw \(kindle\)](#)
- [*China's Super Consumers: What 1 Billion Customers Want and How to Sell it to Them pdf, azw \(kindle\), epub*](#)

- <http://flog.co.id/library/The-Total-Dumbbell-Workout--Trade-Secrets-of-a-Personal-Trainer.pdf>
- <http://conexdx.com/library/The-Color-of-Death.pdf>
- <http://fortune-touko.com/library/Neural-Nets-WIRN10--Proceedings-of-the-20th-Italian-Workshop-on-Neural-Nets--Volume-226-.pdf>
- <http://thermco.pl/library/Fairy-Tales--Barnes---Noble-Classics-Series-.pdf>