

[home](#)[library](#)[techtopia](#)[virtuatopia](#)[ebook store](#)[FOLLOW US ON twitter](#)**On-line Guides**

- [▶ All Guides](#)
- [▶ eBook Store](#)
- [▶ iOS / Android](#)
- [▶ Linux for Beginners](#)
- [▶ Office Productivity](#)
- [▶ Linux Installation](#)
- [▶ Linux Security](#)
- [▶ Linux Utilities](#)
- [▶ Linux Virtualization](#)
- [▶ Linux Kernel](#)
- [▶ System/Network Admin](#)
- [▶ Programming](#)
- [▶ Scripting Languages](#)
- [▶ Development Tools](#)
- [▶ Web Development](#)
- [▶ GUI](#)
- [▶ Toolkits/Desktop](#)
- [▶ Databases](#)
- [▶ Mail Systems](#)
- [▶ openSolaris](#)
- [▶ Eclipse Documentation](#)
- [▶ Techtopia.com](#)
- [▶ Virtuatopia.com](#)

**How To Guides**

- [▶ Virtualization](#)
- [▶ General System Admin](#)
- [▶ Linux Security](#)
- [▶ Linux Filesystems](#)
- [▶ Web Servers](#)
- [▶ Graphics & Desktop](#)
- [▶ PC Hardware](#)
- [▶ Windows](#)
- [▶ Problem Solutions](#)
- [▶ Privacy Policy](#)

# Advanced Bash-Scripting Guide

An in-depth exploration of the art of shell scripting

Mendel Cooper

<thegrendel@theriver.com>

3.7

23 October 2005

**This tutorial assumes no previous knowledge of scripting or programming, but progresses rapidly toward an intermediate/advanced level of instruction . . . *all the while sneaking in little snippets of UNIX® wisdom and lore.* It serves as a textbook, a manual for self-study, and a reference and source of knowledge on shell scripting techniques. The exercises and heavily-commented examples invite active reader participation, under the premise that the only way to really learn scripting is to write scripts.**

**This book is suitable for classroom use as a general introduction to programming concepts.**

## Dedication

For Anita, the source of all the magic

### Table of Contents

#### Part 1. Introduction

1. Why Shell Programming?
2. Starting Off With a Sha-Bang

#### Part 2. Basics

3. Special Characters
4. Introduction to Variables and Parameters
5. Quoting
6. Exit and Exit Status
7. Tests
8. Operations and Related Topics

#### Part 3. Beyond the Basics

9. Variables Revisited
10. Loops and Branches
11. Internal Commands and Builtins
12. External Filters, Programs and Commands
13. System and Administrative Commands
14. Command Substitution
15. Arithmetic Expansion
16. I/O Redirection
17. Here Documents
18. Recess Time

#### Part 4. Advanced Topics

19. Regular Expressions
20. Subshells

- 21. Restricted Shells**
- 22. Process Substitution**
- 23. Functions**
- 24. Aliases**
- 25. List Constructs**
- 26. Arrays**
- 27. /dev and /proc**
- 28. Of Zeros and Nulls**
- 29. Debugging**
- 30. Options**
- 31. Gotchas**
- 32. Scripting With Style**
- 33. Miscellany**
- 34. Bash, versions 2 and 3**

## **35. Endnotes**

- 35.1. Author's Note**
- 35.2. About the Author**
- 35.3. Where to Go For Help**
- 35.4. Tools Used to Produce This Book**
- 35.5. Credits**

## **Bibliography**

- A. Contributed Scripts**
- B. Reference Cards**
- C. A Sed and Awk Micro-Primer**

- C.1. Sed**
- C.2. Awk**

- D. Exit Codes With Special Meanings**
- E. A Detailed Introduction to I/O and I/O Redirection**
- F. Standard Command-Line Options**
- G. Important Files**
- H. Important System Directories**
- I. Localization**
- J. History Commands**
- K. A Sample .bashrc File**
- L. Converting DOS Batch Files to Shell Scripts**
- M. Exercises**

- M.1. Analyzing Scripts**
- M.2. Writing Scripts**

- N. Revision History**
- O. Mirror Sites**
- P. To Do List**
- Q. Copyright**

## **List of Tables**

- 11-1. Job identifiers**
- 30-1. Bash options**
- 33-1. Numbers representing colors in Escape Sequences**
- B-1. Special Shell Variables**
- B-2. TEST Operators: Binary Comparison**
- B-3. TEST Operators: Files**
- B-4. Parameter Substitution and Expansion**
- B-5. String Operations**
- B-6. Miscellaneous Constructs**
- C-1. Basic sed operators**
- C-2. Examples of sed operators**
- D-1. "Reserved" Exit Codes**
- L-1. Batch file keywords / variables / operators, and their shell equivalents**
- L-2. DOS commands and their UNIX equivalents**
- N-1. Revision History**

## **List of Examples**

- 2-1. cleanup: A script to clean up the log files in /var/log**
- 2-2. cleanup: An improved clean-up script**
- 2-3. cleanup: An enhanced and generalized version of above scripts.**
- 3-1. Code blocks and I/O redirection**

- 3-2. Saving the results of a code block to a file
- 3-3. Running a loop in the background
- 3-4. Backup of all files changed in last day
- 4-1. Variable assignment and substitution
- 4-2. Plain Variable Assignment
- 4-3. Variable Assignment, plain and fancy
- 4-4. Integer or string?
- 4-5. Positional Parameters
- 4-6. `wh`, `whois` domain name lookup
- 4-7. Using shift
- 5-1. Echoing Weird Variables
- 5-2. Escaped Characters
- 6-1. `exit` / exit status
- 6-2. Negating a condition using `!`
- 7-1. What is truth?
- 7-2. Equivalence of `test`, `/usr/bin/test`, `[ ]`, and `/usr/bin/`
- 7-3. Arithmetic Tests using `(( ))`
- 7-4. Testing for broken links
- 7-5. Arithmetic and string comparisons
- 7-6. Testing whether a string is *null*
- 7-7. `zmore`
- 8-1. Greatest common divisor
- 8-2. Using Arithmetic Operations
- 8-3. Compound Condition Tests Using `&&` and `||`
- 8-4. Representation of numerical constants
- 9-1. `$IFS` and whitespace
- 9-2. Timed Input
- 9-3. Once more, timed input
- 9-4. Timed read
- 9-5. Am I root?
- 9-6. `arglist`: Listing arguments with `$*` and `$@`
- 9-7. Inconsistent `$*` and `$@` behavior
- 9-8. `$*` and `$@` when `$IFS` is empty
- 9-9. Underscore variable
- 9-10. Inserting a blank line between paragraphs in a text file
- 9-11. Converting graphic file formats, with filename change
- 9-12. Emulating *getopt*
- 9-13. Alternate ways of extracting substrings
- 9-14. Using parameter substitution and error messages
- 9-15. Parameter substitution and "usage" messages
- 9-16. Length of a variable
- 9-17. Pattern matching in parameter substitution
- 9-18. Renaming file extensions:
- 9-19. Using pattern matching to parse arbitrary strings
- 9-20. Matching patterns at prefix or suffix of string
- 9-21. Using `declare` to type variables
- 9-22. Indirect References
- 9-23. Passing an indirect reference to *awk*
- 9-24. Generating random numbers
- 9-25. Picking a random card from a deck
- 9-26. Random between values
- 9-27. Rolling a single die with `RANDOM`
- 9-28. Reseeding `RANDOM`
- 9-29. Pseudorandom numbers, using *awk*
- 9-30. C-type manipulation of variables
- 10-1. Simple for loops
- 10-2. for loop with two parameters in each `[list]` element
- 10-3. *Fileinfo*: operating on a file list contained in a variable
- 10-4. Operating on files with a for loop
- 10-5. Missing in `[list]` in a for loop
- 10-6. Generating the `[list]` in a for loop with command substitution
- 10-7. A `grep` replacement for binary files
- 10-8. Listing all users on the system
- 10-9. Checking all the binaries in a directory for authorship
- 10-10. Listing the symbolic links in a directory
- 10-11. Symbolic links in a directory, saved to a file
- 10-12. A C-like for loop
- 10-13. Using *efax* in batch mode
- 10-14. Simple while loop
- 10-15. Another while loop
- 10-16. while loop with multiple conditions

- 10-17. C-like syntax in a while loop
- 10-18. until loop
- 10-19. Nested Loop
- 10-20. Effects of break and continue in a loop
- 10-21. Breaking out of multiple loop levels
- 10-22. Continuing at a higher loop level
- 10-23. Using "continue N" in an actual task
- 10-24. Using case
- 10-25. Creating menus using case
- 10-26. Using command substitution to generate the case variable
- 10-27. Simple string matching
- 10-28. Checking for alphabetic input
- 10-29. Creating menus using select
- 10-30. Creating menus using select in a function
- 11-1. A script that forks off multiple instances of itself
- 11-2. printf in action
- 11-3. Variable assignment, using read
- 11-4. What happens when read has no variable
- 11-5. Multi-line input to read
- 11-6. Detecting the arrow keys
- 11-7. Using read with file redirection
- 11-8. Problems reading from a pipe
- 11-9. Changing the current working directory
- 11-10. Letting "let" do arithmetic.
- 11-11. Showing the effect of eval
- 11-12. Forcing a log-off
- 11-13. A version of "rot13"
- 11-14. Using eval to force variable substitution in a Perl script
- 11-15. Using set with positional parameters
- 11-16. Reassigning the positional parameters
- 11-17. "Unsetting" a variable
- 11-18. Using export to pass a variable to an embedded awk **script**
- 11-19. Using getopts to read the options/arguments passed to a script
- 11-20. "Including" a data file
- 11-21. A (useless) script that sources itself
- 11-22. Effects of exec
- 11-23. A script that exec's itself
- 11-24. Waiting for a process to finish before proceeding
- 11-25. A script that kills itself
- 12-1. Using ls to create a table of contents for burning a CDR disk
- 12-2. Hello or Good-bye
- 12-3. Badname, eliminate file names in current directory containing bad characters and whitespace.
- 12-4. Deleting a file by its *inode* number
- 12-5. Logfile: Using xargs to monitor system log
- 12-6. Copying files in current directory to another
- 12-7. Killing processes by name
- 12-8. Word frequency analysis using xargs
- 12-9. Using expr
- 12-10. Using date
- 12-11. Word Frequency Analysis
- 12-12. Which files are scripts?
- 12-13. Generating 10-digit random numbers
- 12-14. Using tail to monitor the system log
- 12-15. Emulating "grep" in a script
- 12-16. Looking up definitions in Webster's 1913 Dictionary
- 12-17. Checking words in a list for validity
- 12-18. toupper: Transforms a file to all uppercase.
- 12-19. lowercase: Changes all filenames in working directory to lowercase.
- 12-20. Du: DOS to UNIX text file conversion.
- 12-21. rot13: rot13, ultra-weak encryption.
- 12-22. Generating "Crypto-Quote" Puzzles
- 12-23. Formatted file listing.
- 12-24. Using column to format a directory listing
- 12-25. nl: A self-numbering script.
- 12-26. manview: Viewing formatted manpages
- 12-27. Using cpio to move a directory tree
- 12-28. Unpacking an *rpm* archive
- 12-29. Stripping comments from C program files
- 12-30. Exploring /usr/X11R6/bin
- 12-31. An "improved" *strings* command

- 12-32. Using `cmp` to compare two files within a script.**
- 12-33. `basename` and `dirname`**
- 12-34. Checking file integrity**
- 12-35. Uudecoding encoded files**
- 12-36. Finding out where to report a spammer**
- 12-37. Analyzing a spam domain**
- 12-38. Getting a stock quote**
- 12-39. Updating FC4**
- 12-40. Using `ssh`**
- 12-41. A script that mails itself**
- 12-42. Monthly Payment on a Mortgage**
- 12-43. Base Conversion**
- 12-44. Invoking `bc` using a "here document"**
- 12-45. Calculating PI**
- 12-46. Converting a decimal number to hexadecimal**
- 12-47. Factoring**
- 12-48. Calculating the hypotenuse of a triangle**
- 12-49. Using `seq` to generate loop arguments**
- 12-50. Letter Count"**
- 12-51. Using `getopt` to parse command-line options**
- 12-52. A script that copies itself**
- 12-53. Exercising `dd`**
- 12-54. Capturing Keystrokes**
- 12-55. Securely deleting a file**
- 12-56. Filename generator**
- 12-57. Converting meters to miles**
- 12-58. Using `m4`**
- 13-1. Setting a new password**
- 13-2. Setting an erase character**
- 13-3. secret password: Turning off terminal echoing**
- 13-4. Keypress detection**
- 13-5. Checking a remote server for *identd***
- 13-6. `pidof` helps kill a process**
- 13-7. Checking a CD image**
- 13-8. Creating a filesystem in a file**
- 13-9. Adding a new hard drive**
- 13-10. Using `umask` to hide an output file from prying eyes**
- 13-11. `killall`, from `/etc/rc.d/init.d`**
- 14-1. Stupid script tricks**
- 14-2. Generating a variable from a loop**
- 14-3. Finding anagrams**
- 16-1. Redirecting `stdin` using `exec`**
- 16-2. Redirecting `stdout` using `exec`**
- 16-3. Redirecting both `stdin` and `stdout` in the same script with `exec`**
- 16-4. Avoiding a subshell**
- 16-5. Redirected *while* loop**
- 16-6. Alternate form of redirected *while* loop**
- 16-7. Redirected *until* loop**
- 16-8. Redirected *for* loop**
- 16-9. Redirected *for* loop (both `stdin` and `stdout` redirected)**
- 16-10. Redirected *if/then* test**
- 16-11. Data file "names.data" for above examples**
- 16-12. Logging events**
- 17-1. broadcast: Sends message to everyone logged in**
- 17-2. dummyfile: Creates a 2-line dummy file**
- 17-3. Multi-line message using `cat`**
- 17-4. Multi-line message, with tabs suppressed**
- 17-5. Here document with parameter substitution**
- 17-6. Upload a file pair to "Sunsite" incoming directory**
- 17-7. Parameter substitution turned off**
- 17-8. A script that generates another script**
- 17-9. Here documents and functions**
- 17-10. "Anonymous" Here Document**
- 17-11. Commenting out a block of code**
- 17-12. A self-documenting script**
- 17-13. Prepending a line to a file**
- 20-1. Variable scope in a subshell**
- 20-2. List User Profiles**
- 20-3. Running parallel processes in subshells**
- 21-1. Running a script in restricted mode**
- 23-1. Simple functions**

- 23-2. Function Taking Parameters**
- 23-3. Functions and command-line args passed to the script**
- 23-4. Passing an indirect reference to a function**
- 23-5. Dereferencing a parameter passed to a function**
- 23-6. Again, dereferencing a parameter passed to a function**
- 23-7. Maximum of two numbers**
- 23-8. Converting numbers to Roman numerals**
- 23-9. Testing large return values in a function**
- 23-10. Comparing two large integers**
- 23-11. Real name from username**
- 23-12. Local variable visibility**
- 23-13. Recursion, using a local variable**
- 23-14. The Towers of Hanoi**
- 24-1. Aliases within a script**
- 24-2. unalias: Setting and unsetting an alias**
- 25-1. Using an "and list" to test for command-line arguments**
- 25-2. Another command-line arg test using an "and list"**
- 25-3. Using "or lists" in combination with an "and list"**
- 26-1. Simple array usage**
- 26-2. Formatting a poem**
- 26-3. Various array operations**
- 26-4. String operations on arrays**
- 26-5. Loading the contents of a script into an array**
- 26-6. Some special properties of arrays**
- 26-7. Of empty arrays and empty elements**
- 26-8. Initializing arrays**
- 26-9. Copying and concatenating arrays**
- 26-10. More on concatenating arrays**
- 26-11. An old friend: *The Bubble Sort***
- 26-12. Embedded arrays and indirect references**
- 26-13. Complex array application: *Sieve of Eratosthenes***
- 26-14. Emulating a push-down stack**
- 26-15. Complex array application: *Exploring a weird mathematical series***
- 26-16. Simulating a two-dimensional array, then tilting it**
- 27-1. Using `/dev/tcp` for troubleshooting**
- 27-2. Finding the process associated with a PID**
- 27-3. On-line connect status**
- 28-1. Hiding the cookie jar**
- 28-2. Setting up a swapfile using `/dev/zero`**
- 28-3. Creating a ramdisk**
- 29-1. A buggy script**
- 29-2. Missing keyword**
- 29-3. `test24`, another buggy script**
- 29-4. Testing a condition with an "assert"**
- 29-5. Trapping at exit**
- 29-6. Cleaning up after Control-C**
- 29-7. Tracing a variable**
- 29-8. Running multiple processes (on an SMP box)**
- 31-1. Numerical and string comparison are not equivalent**
- 31-2. Subshell Pitfalls**
- 31-3. Piping the output of echo to a read**
- 33-1. shell wrapper**
- 33-2. A slightly more complex shell wrapper**
- 33-3. A generic shell wrapper that writes to a logfile**
- 33-4. A shell wrapper around an awk script**
- 33-5. A shell wrapper around another awk script**
- 33-6. Perl embedded in a Bash script**
- 33-7. Bash and Perl scripts combined**
- 33-8. A (useless) script that recursively calls itself**
- 33-9. A (useful) script that recursively calls itself**
- 33-10. Another (useful) script that recursively calls itself**
- 33-11. A "colorized" address database**
- 33-12. Drawing a box**
- 33-13. Echoing colored text**
- 33-14. A "horserace" game**
- 33-15. Return value trickery**
- 33-16. Even more return value trickery**
- 33-17. Passing and returning arrays**
- 33-18. Fun with anagrams**
- 33-19. Widgets invoked from a shell script**
- 34-1. String expansion**

- 34-2. Indirect variable references - the new way**
- 34-3. Simple database application, using indirect variable referencing**
- 34-4. Using arrays and other miscellaneous trickery to deal four random hands from a deck of cards**
- A-1. mailformat: Formatting an e-mail message**
- A-2. rn: A simple-minded file rename utility**
- A-3. blank-rename: renames filenames containing blanks**
- A-4. encryptedpw: Uploading to an ftp site, using a locally encrypted password**
- A-5. copy-cd: Copying a data CD**
- A-6. Collatz series**
- A-7. days-between: Calculate number of days between two dates**
- A-8. Make a "dictionary"**
- A-9. Soundex conversion**
- A-10. "Game of Life"**
- A-11. Data file for "Game of Life"**
- A-12. behead: Removing mail and news message headers**
- A-13. ftpget: Downloading files via ftp**
- A-14. password: Generating random 8-character passwords**
- A-15. fifo: Making daily backups, using named pipes**
- A-16. Generating prime numbers using the modulo operator**
- A-17. tree: Displaying a directory tree**
- A-18. string functions: C-like string functions**
- A-19. Directory information**
- A-20. Object-oriented database**
- A-21. Library of hash functions**
- A-22. Colorizing text using hash functions**
- A-23. Mounting USB keychain storage devices**
- A-24. Preserving weblogs**
- A-25. Protecting literal strings**
- A-26. Unprotecting literal strings**
- A-27. Spammer Identification**
- A-28. Spammer Hunt**
- A-29. Making wget easier to use**
- A-30. A "podcasting" script**
- A-31. Basics Reviewed**
- A-32. An expanded cd command**
- C-1. Counting Letter Occurrences**
- K-1. Sample .bashrc file**
- L-1. VIEWDATA.BAT: DOS Batch File**
- L-2. viewdata.sh: Shell Script Conversion of VIEWDATA.BAT**
- P-1. Print the server environment**

---

[Next](#)  
[Introduction](#)