Introduction

Shell scripts are a fundamental part of the OS X programming environment. As a ubiquitous feature of UNIX and UNIX-like operating systems, they represent a way of writing certain types of command-line tools in a way that works on a fairly broad spectrum of computing platforms.

Because shell scripts are written in an interpreted language whose power comes from executing external programs to perform processing tasks, their performance can be somewhat limited. However, because they can execute without any additional effort on nearly any modern operating system, they represent a powerful tool for bootstrapping other technologies. For example, the `autoconf` tool, used for configuring software prior to compilation, is a series of shell scripts.

You should read this document if you are interested in learning the basics of shell scripting. This document assumes that you already have some basic understanding of at least one procedural programming language such as C. It does not assume that you have very much knowledge of commands executed from the terminal, though, and thus should be readable even if you have never run the Terminal application before.

The techniques in this document are not specific to OS X, although this document does note various quirks of certain command-line utilities in various operating systems. In particular, it includes information about some cases where the OS X versions of command-line utilities behave differently than other commonly available versions such as the GNU equivalents commonly used in Linux and some BSD systems.

This document is not intended to be a complete reference for shell scripting, as such a subject could fill entire libraries. However, it is intended to provide enough information to get you started writing and comprehending shell scripts. Along the way, it provides links to documentation for various additional tools that you may find useful when writing shell scripts.

For your convenience, many of the scripts in this document are also included in the “Companion File” Zip archive. You can find this archive in the heading area when viewing this document in HTML form on the developer.apple.com website.

Organization of This Document

This document is organized as a series of topics. These topics can be read linearly as a tutorial, but are also organized with the intent to be a quick reference on key subjects.

- Before You Begin—explains how to get a command prompt in OS X and other operating systems, provides pointers to documentation about using the command line interactively, and provides useful command-line tips (such as how to enter control characters).
- Shell Script Basics—introduces basic concepts of shell scripting, including variables, control statements, file I/O, pipes, redirection, and argument handling.
- Subroutines, Scoping, and Sourcing—describes how to obtain result codes from outside executables, how to write and call subroutines, subroutine variable scoping rules, how to include one shell script inside another (sourcing), and how to use job control to run tasks in the background.
- Paint by Numbers—explains how to use integer math in shell scripts. This section also explains how to use the `bc` command-line utility or Perl to handle more complex math, such as floating-point calculations.
- Regular Expressions Unfettered—describes basic and extended regular expressions and how to use them. This section also describes the differences between these regular expression dialects and the dialect supported by Perl, and shows how to use Perl regular expressions through inline scripting.
- How AWK-ward—explains the AWK command, which provides a data-driven programming language based on regular expressions and tabular data.
- Designing Scripts for Cross-Platform Deployment—describes key differences in the shell scripting environments provided by various operating systems and provides tips for writing portable scripts.
- Advanced Techniques—shows you how to simulate data structures and pointers, perform nonblocking I/O, write timing loops, trap signals, use special built-in shell variables, draw styled text using ANSI color and formatting commands, find the absolute path of a script, use `osascript` to manipulate graphical applications, and use file descriptors and named pipes to treat command-line tools as filters.
- Performance Tuning—describes techniques for improving the performance of complex scripts.
- Other Tools and Information—provides a basic summary of various commands that may be useful to shell script developers, including links to OS X documentation for each of them.
- Starting Points—provides several sample shell scripts and snippets that automate real-world tasks. This appendix also provides links to other complete examples elsewhere in the book.
- An Extreme Example: The Monte Carlo (Bourne) Method for Pi—provides a complex example to showcase the power of shell scripts to perform complex tasks (slowly). The code example shows a shell script implementation of the Monte Carlo method for approximating the value of Pi. The code example takes advantage of a number of numerical and string handling techniques described in the previous chapters. By showing some of the same calculations written in multiple ways, it also illustrates why it is often beneficial, performance-wise, to embed scripts written in other languages such as Perl or AWK when attempting tasks that suit those languages better.

Happy scripting!