Preface

A language is not worth knowing unless it teaches you to think differently.
—Larry Wall (the creator of Perl) and Randal Schwartz

I decided to write a book on UNIX without having previously taught the subject. I didn't like any of the books then available in the market, and there were hardly any experts whom I could turn to for help. Having learned the subject the hard way, I decided to present my experience in the form of a book, but without knowing its intended audience. When I taught the subject many years later, I found the answers to the questions posed by students to be already in the book. I discovered I was on the right track and that I had actually written a textbook.

UNIX has survived the test of time and is likely to do so for some time to come. Yet UNIX is still described by many as "unfriendly" and "unforgiving." Beginners continue to feel overwhelmed by its apparent complexity. Even experienced computer professionals have been unable to embrace the paradigm of doing work using the command line and its myriad options and complex syntaxes. All UNIX and Linux systems today offer the graphical user interface (GUI), but the command line remains the backbone of the system.

The excitement that UNIX generates lies in the fact that many of its powers are hidden. UNIX doesn't offer everything on a platter; it encourages you to create and innovate. Figuring out a command combination or designing a script that does a complex job is a real challenge to the UNIX enthusiast. This is what UNIX is, and it had better remain that way. If you appreciate that *there is a method to this madness*, then you are on the right track and this book is meant for you.

How This Book Is Different

Facing a UNIX box was my first encounter with computers, and prolonged periods of struggle with the system have led me to believe that the stumbling blocks to understanding UNIX are often different from what they are perceived to be. It's not that UNIX is difficult, but that its authors have not been imaginative enough to make it appear otherwise. Books often present, say, 20 options to a command even though only a couple of them could actually be useful. It's frustration that led me to develop my version of the "true" UNIX book—a book that knows what users actually need and one they would like to have with them all the time.

Real-Life Examples UNIX concepts are simple, but they are also abstract, and it's often not obvious why a certain feature is handled in a particular way. The mastery of this operating system requires a clear understanding of these concepts. I have made sure that the key features are explained clearly to reveal both their design considerations and their relevance in the real world. You'll find that many examples of this text refer to real-life situations.

Both a User's and Programmer's Guide There are mainly two categories of UNIX users: those who *use* its native tools, and others who *develop* tools using shell scripts and the UNIX system call library. This book—probably the only one of its kind—adequately addresses both of these segments but has a size that is convenient to read and carry.

The "user" category is served by the first 11 chapters, which is adequate for an introductory UNIX course. The "developer" is a shell or systems programmer who also needs to know how things work, say, how a directory is affected when a file is created or linked. For their benefit, the initial chapters contain special boxes that probe key concepts. This arrangement shouldn't affect the beginner, who may quietly ignore these portions. UNIX shines through Chapters 16, 17, and 18, so these chapters are compulsory reading for systems programmers.

Strong Learning Aids The pedagogical aids are a strong feature of this book. They have various names, for example, Note, Caution, and Tip. I consider Linux to be an important member of the UNIX family, so I have separately highlighted Linux features using the penguin as identifier.

I don't agree with the approach adopted by many authors of treating each shell in a separate chapter. Instead, I have discussed key concepts using mainly the Bash shell. Deviations are addressed by separate asides for the C shell, Korn, and Bourne shells.

Numerous Questions and Exercises This book features an enormous number of questions that test the reader's knowledge. More than a third of them are Self-Test questions, and their answers are provided in Appendix I. These questions are all targeted toward beginners who will do well to answer them before moving on to the next chapter.

More rigorous and extensive questioning is reserved for the Exercises sections. Some of them pose real challenges, and it may take you some time to solve them. These exercises reinforce and often add to your knowledge of UNIX, so don't ignore them. The answers to these questions are available to adopters of the book at the book's Web site, *http://www.mhhe.com/das*.

What's New In This Edition

Some of the reviewers seemed to be quite comfortable with the second edition as it is, and had warned that the improvement may not be "very productive." Keeping this view in mind, I have made the following changes in this edition:

- While the chapter on **vi/vim** has been retained, the one on **emacs** has been condensed and relegated to an appendix. To make the transition to **vi** easier for beginners, Chapter 3 features the **pico** editor.

- The **bc** calculator utility makes a return in this edition, but only as an efficient computing tool for shell programmers.
- A separate chapter on the essentials of C programming has been added. The treatment, though brief, is just adequate to understand the two chapters on systems programming that follow.
- Chapter 15 now includes the popular Concurrent Version System (CVS), which is found on all Linux systems. SCCS and RCS continue to find place in this edition, but at least one of them might be removed in the next edition.
- The GNU debugger (**gdb**) has been included in this edition since it is superior to **dbx**, even though the latter has been retained.

These changes have resulted in a nominal increase in the size of the book. In spite of increased coverage on Linux and GNU tools, the generic character of the book has been retained; it doesn't focus on any particular flavor of UNIX, but variations found in Solaris and Linux have been highlighted.

Understanding the Organization

This edition is *logically* divided into user and programming sections. Essential UNIX is confined to the first 11 chapters, which culminate with a discussion on networking tools. Programming material comprising **awk**, shell programming, systems programming, and **perl** are covered in the the next seven chapters. The final chapter presents the essentials of system administration.

Introducing UNIX Chapter 1 reveals the key UNIX concepts through a simple hands-on session. This is followed by a brief history and presentation of the features of UNIX. You are introduced to the *kernel* and *shell*, who between them, handle the system's workload. You will also understand the role played by standards bodies like POSIX and The Open Group in building the framework for developing portable applications.

Chapter 2 presents the structure of the UNIX command line. It also discusses the techniques of using the **man** command to look up the online documentation. You learn to use an email program, change your password and see what's going on in the system. Things can and will go wrong, so you also need to know how to use the keyboard for corrective action.

Files The *file* is one of the two pillars that support UNIX, and the next three chapters discuss files. Chapter 3 discusses the various types of files you'll find on your system and the commands that handle them. You'll learn to create directories, navigate a directory structure, and copy and delete files in a directory. You'll also learn to edit a file with the **pico** editor. UNIX also offers a host of compression utilities that you need to use to conserve disk space.

Files have attributes (properties), and Chapter 4 presents the major attributes, especially the ones displayed by the **ls -l** command. Be aware that your files and directories are open to attack, so learn to protect them by manipulating their permissions. Use *links* to access a file by multiple names. You'll also forget where you have kept your files, so you need to be familiar with the **find** command.

How productive you eventually are also depends on how well you exploit the features of your editor. Chapter 5 presents **vi**, one of the most powerful text editors found in any operating environment. A programmer probably uses the editor more than anyone else, so most examples in this chapter use snippets of program code. Appendix D presents a summary of the features of **vi**.

The Shell and Process You now need to understand a very important program that is constantly interacting with you—the shell. Chapter 6 presents the interpretive features of the shell, including many of its *metacharacters*. Learn to use *wild cards* to match a group of similar filenames with a single pattern. Manipulate the input and output of commands using *redirection* and *pipes*. The shell is also a programming language, so you have to wait until Chapter 13 to understand it completely.

Chapter 7 introduces the *process* as the other pillar of the UNIX system. Processes are similar to files, and processes also have attributes. Understand how the *fork-exec* mechanism is used to create a process. Learn to control processes, move them between foreground and background, and also kill them by sending *signals*.

The UNIX shell provides excellent opportunities to customize your environment (Chapter 8). Understand and manipulate shell variables, create command *aliases* and use the *history* mechanism to recall, edit and re-execute previous commands. Choose a suitable shell that offers all of these features and learn to use the initialization scripts to save the changes you've made to the environment.

Filters The next two chapters deal with *filters*—those special commands in the UNIX tool kit that handle all text manipulation tasks. Chapter 9 presents the simple ones and shows how they are most effective when they are connected to one another. A special examples section features three real-life applications that are handled by these filters working in pipelines.

Chapter 10 discusses two powerful filters—**grep** and **sed**—that, between them, handle all pattern search, edit, and replace operations. At this stage, you'll be introduced to *regular expressions*, an elaborate pattern-matching mechanism that often makes searching and replacement a lot easier. Filters are followed by a chapter on networking tools (Chapter 11), which concludes the first part of this book.

Programming The next seven chapters handle most of the programming features of UNIX. The **awk** command makes its appearance as a filter and a programming language in Chapter 12. Knowing **awk** and its standard programming constructs (like the **if**, **for**, and **while** constructs) should prepare you well for shell programming, **perl**, and C programming.

Eventually, you'll place all of your commands and pipelines in *shell scripts*. Use the programming features of the shell discussed in Chapter 13 to develop both interactive and noninteractive scripts. Learn to design a script whose behavior depends on the *name* by which it is invoked. The three sample scripts featured in the chapter are compulsory reading for a shell programmer.

The next four chapters are directly or indirectly related to C programming. Chapter 14 presents a primer on C programming, the only new chapter in this edition.

This is followed by the program development tools (Chapter 15). Use the **make** utility and a powerful debugger (**gdb**) for managing and debugging programs. Also, learn to maintain multiple versions of a program using SCCS, RCS, and CVS.

Chapter 16 is the first of two chapters that feature the use of *system calls* in the C programming environment. This chapter discusses the system calls related to files and I/O. Write programs that perform directory-oriented functions like listing files. Also learn to fetch and manipulate file attributes stored in the inode.

Chapter 17 discusses the system calls related to processes. Learn to create processes using the **fork** and *exec* family of system calls. Once you've understood how the kernel maintains the metadata of an open file in memory, you'll be able to implement both redirection and pipelines and to handle signals in your programs.

We encounter **perl** in Chapter 18 as the most powerful filter and scripting language in the UNIX world. Most UNIX concepts are embedded in the design of **perl**, the reason why many UNIX users can't do without it. Even though we can't do justice to **perl** in a single chapter, Chapter 18 represents a useful beginning.

Finally, every user must know the routine tasks related to system administration, and Chapter 19 addresses the basic issues in this domain. Understand the important security features provided by the system. Be familiar with the activities associated with system startup and shutdown, and how file systems are *mounted* and checked for consistency. Also learn to do some elementary backups.

Acknowledgments

Every edition of this book has provided me with an opportunity to work with a new team. The prime mover for this project is Raghu Srinivasan, the global publisher, and it was a delight to work with him and Melissa Leick, the project manager. I am impressed with the way they have guided the team at McGraw-Hill. I must also thank Brenda Rolwes, the design coordinator, and Curt Reynolds for the marketing arrangements that he was responsible for. Special mention must be made of Melinda Bilecki, the developmental editor, who remains as agile as ever. She is the only person associated with the book since its conception, and this continuity has benefitted me immensely.

Credit must also go to the reviewers who have played a significant role in shaping the form and content of this edition:

Ivan Bajic, Sas Diego State University
Bruce Elenbogen, University of Michigan–Dearborn
Ziad Kobti, University of Windsor
Galina Piatnitskaia, University of Missouri–St. Louis
Paul Piwowarski, University of Kentucky
Darren Provine, Rowan University
Quazi M. Rahman, The University of Western Ontario

I am thankful to Deepti Narwat, the project manager at Cenveo Publisher Services, who handled the entire production process with confidence and the utmost sincerity. There have been many others who couldn't be mentioned by name, but have contributed just the same.

Final Words of "Wisdom"

Most of the examples have been tested on Solaris and Linux, but I can't guarantee that they will run error-free on every system. UNIX fragmentation makes sweeping generalizations virtually impossible. If some commands don't work in the way specified in this text, don't conclude that the system has bugs. Nevertheless, bugs in these examples are still possible, and I welcome ones that you may hit upon.

Before I take leave, a note of caution would be in order. Many people missed the UNIX bus through confused and misguided thinking and are now regretting it. Let this not happen to you. Once you have decided to exploit UNIX, you'll learn to build on what's already provided without reinventing the wheel. Sooner rather than later, you'll find a world of opportunity and excitement opening up. Approach the subject with zeal and confidence; I am with you.

Sumitabha Das