Reviews

A Scientist's and Engineer's Guide to Workstations and Supercomputers: Coping with Unix, RISC, Vectors, and Programming, by Rubin H. Landau and Paul J. Fink Jr. ISBN 0-471-53271-1 (paperback), 1993, $49.95, xxi+349 pp., IBM PC-compatible 5-1/4" 1.2 Mb diskette included. Available from John Wiley & Sons, Inc., New York, NY.

Not another introduction to Unix and workstations! Aren't there already enough of these? What's the point?

One of the goals of the book is to introduce scientists and engineers to Unix. While this goal is at least partly met, one can find better introductory Unix texts. However, Landau and Fink contribute more to scientific programmers than just another introduction to Unix; they have written what might be called a guide to how a non-computational scientist can progress to the fuller use of Unix-based computers to do science; if, in the journey, the scientist needs to find supplementary guidebooks, at least the authors have provided an overview. Experienced scientific programmers should also find value in this book, as I hope to demonstrate later.

In their introduction, Landau and Fink also ask the question of why two physicists are writing a book about computers. Their answer is that the computer is a tool to "advance science and engineering", which they hope will "help increase the creativity, productivity, and effectiveness" of scientists and engineers. Implicit in this claim is the assumption that most scientists and engineers are not now using computers effectively, productively, or creatively; such a claim may or may not be true, of course.

Specifically, the authors advocate Unix as the operating system of choice for serious computational science, by which they mean the use of computers to solve complex scientific problems, whether through conventional "number crunching", or through techniques such as visualization and graphics to assist in the understanding of various phenomena.

Their choice of Unix seemed at first surprising, since one of the authors (Fink) is employed by IBM. However, as the authors point out, Unix has the considerable advantage of letting the user write more-easily portable code than it is possible to do on other operating systems. In addition, the availability of many applications for Unix platforms, much of it in the public domain, allows the user to collect many tools quickly and relatively inexpensively. An equivalent collection for other operating systems would be either very hard to find, or very expensive, or both.

Let me confess from the outset that I am not a 'Unix person', although I have used, and do use, Unix systems. I found the first part of the text to be a reasonably complete and reasonably understandable introduction to Unix, one which taught me some things I had not known. However (as is true throughout this text), there is in one sense too much material, and in another sense too little. As an example of too little information, Landau and Fink mention the vi editor, but limit their discussion quite severely, although they present a one-page summary of vi commands. Anyone who has tried to use vi know that the editor demands more; similarly, their discussion of Emacs is crammed into two pages. Part of this brevity may result from their stated bias in favor of X windows systems; part may stem from the need to keep the text at a reasonable length.

On the other hand, in Chapter 2, Landau and Fink discuss at length the use of subdirectories, along with suggestions for naming conventions, and then unnecessarily repeat much of the information in Chapter 3. In Chapter 2, which introduces Unix, and discusses file systems, Unix commands, home directories, and shells, the authors are careful to specify commands for the System V and for the BSD versions where necessary. Likewise, if a command is different in the C shell, the Korn shell, and the Bourne shell, the difference in the commands is specified. One item, however, should be mentioned in connection with

ISSN 1058-9244/97/$8
Scientific Programming, Vol. 6, pp. 391–393 (1997)

the names of shells: the Bourne shell is almost invariably called the 'Borne' shell, but once it is the 'Born' shell, and once it is the 'Bourne' shell. This is an inconsistency – and a set of errors – which I found quite distracting.

On the positive side, in their discussion of hardware and how it works, Landau and Fink present valuable information (supplemented by programs on the included floppy disk, which the reader can freely copy) about the effect of different word sizes, single-precision and double-precision representations of numbers, and fixed- or floating-point specifications on different computers; all of these affect the results of calculation, of course, and must be taken into consideration in design and creation of programs – and all work differently on different hardware, even if the computers use the same implementation of the same operating system. The book explains the consequences of the various choices, and give examples of pseudocode (a term which goes undefined, by the way) to help determine how a particular computer's design is related to the results of the programs.

Landau and Fink provide an overview of workstation hardware, definitions of various components, and an explanation of how to begin using a workstation. They suggest some guidelines for managing a Unix system; and they very sensibly note that learning to use the workstation should come before any attempts to configure it. They show how to set variables; how to manage files, devices, and the system itself; and how to enable or disable the various shell programs.

Their information on networking, which will be useful if a workstation is already networked, does not help the novice Unix user set up a node on a network. (Perhaps the authors assume the workstation will be networked by whatever support personnel are available.) Their discussion of networks and network commands presupposes connection to the Internet, or at least to a TCP/IP network, in one form or another. For example, they show how to use nslookup and telnet to find and connect to the computer physics.orst.edu. They then discuss use of tape drives, the tar command with tapes and floppy diskettes, and the mt command for tape drives.

Chapter 4, computer–computer interactions, contains a lot of information, not all of it new. The authors return to networks, reprise information about the telnet command and about the Internet (which they don't usually capitalize), and discuss mail systems and remote login programs. The chapter also contains information on file transfer, both between computers on a network, and between a host and a client computer (e.g., a PC and a mainframe) using a terminal-emulation program.

One area of difficulty, which Landau and Fink point out in the text, but do little to help the beginning Unix user solve, concerns the problem of using the files on the DOS-formatted floppy disk included with the book. There are approximately 300 kb of information on the disk, including graphics files, sample programs of various kinds, and sample shell scripts and formatting control files (for example, the LaTeX control files which they used in formatting the text). The authors could have put, say, a version of Kermit, and perhaps several other public-domain or copyable programs, on the disk rather than leave the other 900 kb empty. Fortunately for the beginning user, they do include on the floppy a script for copying all files from the floppy disk to a workstation – after the user manually copies the script file to the workstation. (In fairness, let me say that Landau and Fink recognize the 'catch-22' nature of file transfer, pointing out that one must have a file-transfer program in order to get a file-transfer program. Also, they do discuss the Unix dosread and doswrite commands.)

Of all their chapters, I found the one on X Windows the most difficult to read and the least useful. It was difficult because it presumed more knowledge of the X system than I have, even though it is intended as an introduction to X Windows. It is the least useful mainly because it is too ambitious for a novice, particularly in connection with configuring X Windows system. While the more experienced user may be confident – dare I say 'courageous'? – enough to integrate different pieces of X systems from different vendors and to attempt more than minor personalization of X Windows, it appears to me that Landau and Fink have provided just enough information to get the novice into trouble.

This appears to be a classic case of "a little knowledge is a dangerous thing". On the other hand, the floppy disk does provide a configuration file (.mwmrc) for Mosaic Windows Manager, which the reader can use for experimentation – assuming the presence of Mosaic, of course, on the user's workstation. Other included dot files help the user set X defaults and initialization parameters.

From the point of view of the scientist or engineer for whom the book is written, the chapters on graphics packages and on programming in Fortran or C on Unix systems are quite useful: Landau and Fink offer specific examples of how or where a particular application can be used. Most of their examples are based on problems which would be familiar to the intended audience, for example, the use of Gnuplot to draw a surface plot of an antikaon–nucleon T matrix (p. 169). Data files for the various examples are included on the floppy disk. In many cases, the authors also describe where to find the packages by ftp on the Internet, thus helping the beginning scientific programmer assemble a working library of tools.

I found the discussion of Fortran and C compilers and compiler options both interesting and valuable. Although Landau and Fink go into more detail than the beginning scientist or engineer needs, their examples and

explanations would reward more-advanced programmers, and thus are of interest even to those who have experience in scientific programming. For example, their discussion of matrices and of the difference in row-major order (in C and in Pascal) and column-major order (in Fortran) in connection with the programming of nested loops clearly shows the importance of matching program flow to the characteristics of the language being used and to the implementation of that language by a specific combination of compiler and operating system. Unfortunately, the illustration accompanying the discussion on if–then–else constructions suffers from poor proofreading: Figure 11.5, p. 261, shows both results of an if-test leading to the same action.

RISC and supercomputer architectures are covered at length, as is the difference between vector and scalar processing. In discussing architectures, authors show how optimization of code is related to both the extent of vectorization and the architecture; and they provide a principled way to decide which parts of a program one should attempt to vectorize, and to decide at which point the attempt becomes self-defeating. Again, this discussion is more likely to be understood by someone who has extensive programming experience than by their intended audience. But for that experienced person, the explanation is quite worthwhile, as are their examples and discussions of good and bad code in relation to cache and register architectures, and vector and scalar loops and of when to unroll loops. Similarly, their discussion of supercomputer architectures can serve as an introduction to the topic for the more-experienced programmer.

For the novice, much of this text is too sophisticated; for the experienced scientific programmer, especially one who knows something about the various architectures and the difference between CISC and RISC computers, much of the information is ‘old hat’. However, both groups can benefit from this text. As the beginner becomes more proficient, the more advanced parts of the text becomes accessible.

Particularly intriguing, to me at least, was the discussion of the future of computing. Landau and Fink believe that “the technical work on which [their] book focuses, and the present desktop computing paradigm in which we now exist, will continue within the next paradigm, which is believed to be ‘networked computing”’. They predict the further development of open systems; of Unix as a major operating system; of Fortran and C as the programming languages of choice (although with an emphasis on massively-parallel versions of these languages), through the further development of Fortran 90 and its successors (and analogous C and C++ programming languages); of transparently-networked computers which share tasks as required; and of “a continuing emphasis on structuring, documentation and publication”. If they are correct, readers of this journal might best begin now to learn much more about the parallel processing and vectorizing techniques which Landau and Fink discuss, in order to prepare themselves to work on the next generation of computers and of programming languages.

Louis “Lou” Hillman
Department of English
SUNY College at Brockport
Brockport, NY 14420, USA
and
Applied Computer Technology Department
Rochester Institute of Technology
Rochester, NY 14618, USA