Springer Books on PROFESSIONAL COMPUTING

Peter P. Silvester

The Unix™ System Guidebook

An Introductory Guide for Serious Users

Springer-Verlag
New York Berlin Heidelberg Tokyo

Peter P. Silvester

The Unix™ System Guidebook

An Introductory Guide for Serious Users

Springer-Verlag
New York Berlin Heidelberg Tokyo

Peter P. Silvester
Department of Electrical Engineering
McGill University
Montreal, Quebec
Canada H3A 2A7

Series Editor
Henry Ledgard
Human Factors Ltd.
Leverett, Massachusetts 01054
U.S.A.

With 6 Figures.
Unix is a trademark of Bell Laboratories.

Library of Congress Cataloging in Publication Data
Silvester, Peter P.
The Unix system guidebook.

(Springer books on professional computing)
Bibliography: p.
1. UNIX (Computer system) I. Title. II. Series.
QA76.6.S564 1983 001.64'25 83-16846

Media conversion by Science Typographers, Inc., Medford, New York.
Printed and bound by R. R. Donnelley & Sons, Harrisonburg, Virginia.
Printed in the United States of America.

9 8 7 6 5 4 3 2 1

ISBN 0-387-90906-0 Springer-Verlag New York Berlin Heidelberg Tokyo
ISBN 3-540-90906-0 Springer-Verlag Berlin Heidelberg New York Tokyo

Preface

Well suited to medium-scale general purpose computing, the Unix time-sharing operating system is deservedly popular with academic institutions, research laboratories, and commercial establishments alike. Its user community, which until recently was made up mostly of experienced computer professionals, is now attracting many people concerned with computer applications rather than systems. Such people are mainly interested in putting Unix software to work effectively, hence need a good knowledge of its external characteristics but not of its internal structure. The present book is intended for this new audience, people who have never encountered the Unix system before but who do have some acquaintance with computing.

While helping the beginning user get started is a primary aim of this book, it is also intended to serve as a handy reference subsequently. However, it is not intended to replace the definitive Unix system documentation. The Unix operating system as it now exists at most installations (popularly, though somewhat inaccurately, called Version 7 Unix) is substantially as described by the Seventh Edition of the system manuals. This book emphasizes Version 7 and systems closely related to it, but it does also describe some other facilities in wide use.

Many people have been instrumental in shaping this book and the author wishes to express his gratitude to them all. Particular thanks are due to David Lowther, for our many helpful discussions; and to the many students whose suggestions enlivened the task.

PETER P. SILVESTER

Contents

Chapter 1

Introduction

The Unix time-sharing system is rapidly becoming one of the most popular computer operating systems ever designed. Its unique popularity may be the result of portability; Unix systems are available for various different computers, while practically all other operating systems are tied to specific machines. Whether for this reason or any other, the Unix system is becoming universal, much as Fortran became the universal language in its day. And just as Fortran influenced the style of other programming languages, so Unix software characteristics are becoming visible—both by emulation and deliberate avoidance—in other operating systems. For computer users, some acquaintance with the Unix system is therefore taking on increasing importance.

A Multimachine Operating System

Although it was originally intended for the PDP-11 family of computers, Unix software has been recreated for use on many other machines, both smaller and larger. There now exist versions of the Unix system, or other operating systems which very closely resemble it, for many widely used small computers based on 16-bit microprocessor chips. Upmarket from the PDP-11, Unix systems (in some cases several) exist for the Interdata 8/32, the PERQ, the VAX-11 family, and other large minicomputers. Other versions run on large mainframe computers like the Amdahl. At the opposite end of the computer spectrum, Unix-like operating systems are available for several eight-bit microprocessors.

System Characteristics

Three main reasons are usually cited for the current popularity of Unix and Unix-like operating systems with users. First, they provide a simple and logically almost consistent command language through which the user can interact with the system; a language easy to learn, fairly easy to understand, and not very easy to forget. Second, Unix systems provide a very wide variety of software tools and services, so that program development can progress rapidly. Third, and perhaps most important, both system services and user programs are insured against too rapid obsolescence, by being nearly machine-independent. Programs can be moved to new computers along with the operating system, while new system services become available on practically all versions of the Unix system at once.

Traditionally, many computer manufacturers have regarded operating system software as an unpleasant hurdle to be overcome before a new machine could be marketed. The relative portability of the Unix system has endeared it to hardware makers, for computers can be designed to run under this operating system by investing only a modest amount of software effort. New hardware can be made ready for the market not only quickly, but with all the sureness of an already accepted product. To the user, a knowledge of Unix software structure and command language is of long-term value, for it is very likely that his next computer will employ a variant of the same system or one of its close cousins. Relative machine-independence also enriches the range of general utility programs available; because programs can migrate to new computers along with the operating system, development of new general-purpose programs becomes attractive.

Not surprisingly, the Unix operating system is not quite perfect. Its major shortcomings include, first of all, that it assumes a friendly user community. The file security provided, for example, is not nearly good enough to make it attractive in such sensitive applications as banking or finance. Secondly, many of its command structures and conventions bear the marks of having been developed by a circle of friends, without much regard for subsequent distribution to others. For instance, many commands are abbreviated to very short forms and appear easy to confuse with others. Finally, protection against operator error is imperfect; certain users can even accidentally destroy all files on the system, including the operating system itself. This latter disadvantage can be very serious, especially in commercial or financial applications. But fortunately it only matters to very experienced users, who have gradually acquired a knowledge of pretty well everything the system can do. Novices are very unlikely to have access to quite so much destructive power.

Portability

Because Unix programs are almost entirely written in a high-level programming language called C, this system is practically guaranteed to become available on many future computers as well as quite a few more already existing ones. To install a Unix system on yet another computer, two main things are necessary: a C compiler and a modest amount of machine-dependent coding. A compiler for the C language is always required, so that the Unix operating system itself can be translated to the native language of the new machine. Construction of such a compiler generally takes a few man-months or perhaps a man-year of programming effort. In addition to the compiler, transporting Unix to another machine requires a few machine-dependent input-output hardware service routines. These must necessarily be written in the native language of the new machine, so that they are strictly locked to that computer. Fortunately, they are usually quite short, so that much programming effort is not needed. Usually, a matter of man-weeks or, at worst, man-months, is involved. These amounts of time are tiny when compared to the investment required to design and write a new operating system. The initial effort that produced the Unix kernel amounted to two or three man-years, but the addition of the many utility programs that make Unix systems useful has taken much, much more.

The great majority of Unix system services available—editors, compilers, file sorting and merging programs, and others—are written in high-level languages, with C the most widely used language by far. New utility programs constructed by the now quite wide Unix user community are also written in high-level languages, C being again the most frequent choice. As a result, the new programs can be incorporated in almost any Unix installation without alteration.

Past and Future

Although it has gained wide popularity only recently, the Unix system is mature software which has undergone years of testing and rewriting. To assess its probable future, its history may deserve at least brief mention.

Ancient History

The first Unix system was written by D. M. Ritchie and K. Thompson at Bell Laboratories, around 1969 or 1970, to run on the now all-but-forgotten PDP-7 and PDP-9 computers. Their primary objective was to produce a system convenient for inexperienced users, and they succeeded at least well enough to be encouraged to construct a second version to run on a PDP-11/20. Because the PDP-11 family of computers became enormously popular in the decade of the seventies, a third version, again fully rewritten, appeared in due course; it supported the PDP-11/34, /40, /45, /60, and /70. By 1973, the system authors had abandoned assembler language coding, for it was becoming evident that transportability from machine to machine would be greatest if a very large part (ideally but impossibly, all) of the system were written in a high-level language. A language called C was developed for the purpose. C is well suited to writing operating systems while retaining most other characteristics of good high-level languages, and it remains the principal language of the Unix operating system. C resembles Pascal in many respects, but it does allow programming a little closer to the machine register level—as if Pascal were to recognize the existence of registers and bits! The structure and capabilities of C thus allowed building the Unix system in a fashion which made it largely independent of the machine hardware structure: at least transportable, if not actually portable.

A paper on the Unix operating system was published by Ritchie and Thompson in 1974 in the *Communications* of the Association for Computing Machinery. This paper quickly became a defining landmark for the system. It outlined the basic system structure and methods of work; although these have been refined considerably since that time, the basic notions have remained almost unchanged. What has changed, to be sure, is the range of system services and utilities available. Unix probably contains a better selection of software tools than any other operating system. Not only is their range wide, but they have for the most part been written to go together well.

The Unix system is currently available on all computers in the PDP-11 family, including the PDP-11/23, /24, /44, and other recent additions. It represents a fourth broad rewriting of the system, although it is popularly (though not quite correctly) called Version 7. The number 7 in fact refers to the seventh edition of the *Unix Programmer's Manual*, the document which describes the operation of the current system version. Very few people, incidentally, possess a complete copy of that document—not because it is secret, but because it is about the size and shape of the Manhattan telephone directory!

A significant influence on the course of Unix software development came in the late 1970s, when a major development project at the University of California, Berkeley, began to bear fruit. The Berkeley Unix system adhered closely to the spirit and objectives of its Bell Laboratories cousin, but it introduced substantial extensions and improvements. Many of the improvements were internal, invisible to the casual user. Others, like the Berkeley `vi` text editor, are immediately visible.

The Modern Age

Unix systems for computers other than the PDP-11 started appearing in the late seventies. A version for the Interdata 8/32 computer became available around 1978, and one for the VAX-11/780 came soon thereafter. Several versions for other processors, notably the model 68000 and Z8000 16-bit microprocessor chips and thus for the many computers built around them, followed. Between 1978 and 1982, other operating systems also appeared, developed independently but with a remarkable similarity to Version 7 (and its predecessor Version 6) Unix systems. Some resemble Version 7 only in what the system looks like to the user at the terminal. In others, the similarity extends to such internal details as file formats, so that not only programs but even disk or tape files can be moved between systems. Most of the look-alikes are intended for computers that employ 8-bit or 16-bit microprocessor chips, but at least one is available on the commercial market for the PDP-11 series of machines.

Until 1980–1981, the creation and marketing of look-alike Unix systems was lent strong encouragement by the fact that the Bell Laboratories Unix system itself was, for all practical purposes, available for academic research and teaching use only. Look-alike systems therefore appeared to fill the commercial gap. For example, the Idris and Unix systems are independent but look similar to the user, and they are compatible with each other. Since about 1981, Unix has been made available commercially through retail vendors, under various names such as Xenix, Unisis, or Unity. These are not look-alikes, but Unix itself dressed in a commercial suit. They are not only entirely compatible with Unix systems; they *are* Unix systems. Most such derivative systems are enhanced, modified, or adapted to perform well in particular environments.

Both the independently developed systems and the licenced variants of Version 7 appear on the market under names other than Unix. Different names are used for both commercial and legal reasons, among which trademark protection probably ranks high. With Unix systems coming into widespread use and commercially available almost everywhere short of drugstore counters, Bell Laboratories are presumably concerned lest their trademark pass into the public domain through excessively great success—along with aspirin, bakelite, and many others. At present, there is no generic name to cover Unix, Xenix, Coherent, Zeus, Cromix, Flex, Qunix, ... and much of the computer press refers to them all as "Unix-like operating systems."

Through a Glass Darkly

Will the Unix operating system prove sufficiently long-lived and sufficiently universal to merit study and practice? No one can tell for sure, but the answer is more likely yes than no. This system is no longer new; it has been seasoned by more than a decade of development, through several stages, and has settled down. The prospect for its commercial success is excellent, and the future of Unix systems in the 1980s may easily resemble that of the Fortran language in the sixties. Both were initially developed with particular computer systems in mind, but they quickly outgrew their original hosts. Both suffer from structural and logical deficiencies which seemed minor or unimportant at the outset but became irksome after the first decade. Both seem better suited to their tasks (warts and all) than any currently available competitor. Both may therefore live on long after better languages and more portable systems became available. Like the English language, with its constrictive syntax, difficult grammar, incomprehensible spelling, ... and two or three billion people who continue to use it simply because they all understand it.

Things to Read

Until recently, very little has been available by way of a simple introduction to the Unix operating system. Most beginners have been expected to cut their teeth on photocopies of the admittedly excellent brief articles that survey the system characteristics—and by perusal of the *Unix Programmer's Manual*, which in its seventh (current) edition is a large volume some ten centimeters thick. Indeed it is the definitive work, but hardly easy for the beginner.

With the growing popularity of Unix systems, textbooks and articles of a tutorial nature have begun to appear. These tend to be a good deal more readable than the full manual and should constitute the major reference point for the beginner. A brief bibliography listing most of these, annotated to give some idea of their contents, appears at the end of this book.

The *Unix Programmer's Manual* is the defining document of the system and is furnished both as paper copy and in machine-readable form. Provision is made in Unix systems for keeping much of the user documentation available as disk files, so that users can read particular portions of the manual without having access to a printed copy, or without even needing one. Keeping the system manuals in computer-readable form and therefore easy to modify is vital for most Unix installations, for there are probably no two installations exactly alike. While every system manager strives to keep documentation current for his system, in many places no up-to-date paper manual exists; the disk-file version is the only true version.

Typographic and Lexical Curios

The words used in Unix system commands, the way words are abbreviated, and occasionally the ways they are spelled are a bit idiosyncratic. Presumably this is the result of having been developed initially within a circle of friends prepared to put up with each other's foibles. Some users take to the strange habits of Unix like a duck takes to water; others show enthusiasm more appropriate to a cat. There is no choice; Unix commands come the way they come.

Lowercase letters are used almost exclusively in Unix system commands, programming languages, and naming conventions. Although they are rare, exceptions do exist; it is not simply a matter of using lower case instead of upper. Both are used, but for some reason capital letters occur much less frequently than they do in English. A particularly irksome idiosyncrasy to some is the failure to accept initial capitals even where the conventions of English demand it. The author, for example, will probably never grow quite accustomed to identifying himself as *silvester*, without a capital S.

The almost, but not quite, total use of lower case causes certain problems when documentation is written in natural languages. For example, the Unix text editor program is called **ed**, and the phototypesetter program has the name **troff**. (Not unreasonably, some users would have preferred mnemonically more useful names, e.g., *Editor* and *Phototype*.) They are never called **Ed** and **Troff**, because the system regards the letters **E** and **e** as simply two different, unrelated, characters. But what should one do when a sentence begins "**troff** is a program for ..."? In this book, the Unix program naming convention is followed strictly: program names exist in lower case only, they do not acquire capitals even if they occur at the beginnings of sentences. But much Unix system literature, occasionally even including the definitive manual, is somewhat inconsistent about usage in this matter.

Word usage in the Unix system shows up a lot of curious details rooted in history, but it is often difficult for the newcomer to master. For example, the verb "print" is used almost everywhere in the manuals to imply that output is to be sent to the user terminal. That verb may have been accurate at some past time, but today few users employ printing terminals; display screens are much more common. (The verb "print" to most computer users implies use of a line printer, not the user terminal.) As another example, the verbs "move" and "remove", when referring to files, are employed to mean "rename" and "delete". The reference here is to the software methodology employed: deletion is achieved by removing a linking pointer.

Using This Book

The best way of learning to use an operating system is to use it. To allow the beginner to learn in this natural way, the first (short) part of this book contains an introduction to the system and its use, in quite brief and very simple form. It is sufficiently concise to be usable while sitting at the terminal, trying out the commands. The main part of this book is longer. It is intended for reading away from the terminal, and for reference; it therefore consists of a few explanatory chapters, followed by a summary description of the more important system commands.

Chapter 2

Getting Started

The Unix operating system is generally considered easy to learn and easy to use. But even the easiest operating system takes a little getting used to, especially at the very start when nothing looks even remotely familiar and every response from the system appears vaguely ominous—if indeed the system responds at all! Most computer users dislike the first hour or so spent with a new operating system, when the initial difficulties of a new command language, new name conventions, and new protocol rules all appear together. This chapter is intended to provide a launching pad for the novice user and to help him overcome the problems of that first hour. It is brief enough to be read at the terminal, trying out the various commands on the spot or it can be read at another time and place, in preparation for that first hour.

Communicating with the System

Several users can be logged in to the same computer at the same time under the Unix timesharing multi-user operating system. Learning to use the Unix system therefore begins with becoming an authorized system user, then acquiring familiarity with the procedures for using a terminal.

User Names and Numbers

To keep track of users and their needs, every Unix system has a human *system manager*. In large computing centers, the system manager may well consist of a whole office establishment with receptionists and secretaries to