2751

UNIX® V AND XENIX® SYSTEM V PROGRAMMER'S TOOL KIT

MYRIL CLEMENT SHAW AND
SUSAN SOLTIS SHAW

FOR
C PROGRAMMERS
USING
VERSION 3.0!

UNIX® V AND XENIX® SYSTEM V PROGRAMMER'S TOOL KIT

MYRIL CLEMENT SHAW AND
SUSAN SOLTIS SHAW

TAB TAB BOOKS Inc.
Blue Ridge Summit, PA 17214

FIRST EDITION
FIRST PRINTING

Library of Congress Cataloging in Publication Data

Shaw, Myril Clement.
UNIX V and XENIX system V programmer's tool kit.

Includes index.
1. UNIX (Computer operating system) 2. XENIX (Computer operating system) 3. C (Computer program language) I. Shaw, Susan Soltis. II. Title.
QA76.76.063S552 1986 005.43 86-14434
ISBN 0-8306-0251-8
ISBN 0-8306-2751-0 (pbk.)

UNIX® V AND XENIX® SYSTEM V PROGRAMMER'S TOOL KIT

Preface: The UNIX V Programming Environment

THE PUBLICATION OF THE *SYSTEM V INTERFACE DEFINITION* MARKED the entry of UNIX into a new era. UNIX was suddenly thrust from the role of technical wonder into the role of an exciting and complete application development environment. The *Interface Definition* provided specifications and standards by which application programs could achieve nearly total hardware independence. Applications can now be written that are truly portable from mainframe to PC—but only if the application developer writes for UNIX V using the *Interface Definition*.

UNIX V is the *de facto* industry standard for all UNIX systems, as well as UNIX clones and look-alikes. All UNIX-like operating systems are migrating toward the UNIX V standard. Microsoft and SCO (The Santa Cruz Operation) have released the XENIX V operating system, which provides a superset of UNIX V running on virtually all 8086-, 8088-, 80286-, and 80386-based microcomputers. The AT&T 3B line of computers, ranging from supermicro to supermini, all run UNIX V and the Amdahl UTS mainframe system runs a UNIX V operating system.

It is into this environment of increasing consistency and standardization that the application developer is placed. Fortunately, UNIX V and its cohorts provide a rich and exciting application development environment.

Unfortunately, even gaining a glimmer of the content of this wonderful operating environment can be difficult. This book was written in the hope of remedying that situation. It is the book the authors would have liked to have had available when they were first introduced to the UNIX V application development environment.

Acknowledgment

The authors wish to acknowledge the gracious assistance of Bruce Steinberg and Brigid Fuller and the Santa Cruz Operation for providing XENIX System V.

Introduction

THE UNIX V AND XENIX SYSTEM V PROGRAMMER'S TOOL KIT IS A DIFFERENT type of computer book. Strictly speaking, it is not a how-to book, nor is it a book for teaching the minutiae of UNIX commands. Both of these events occur during the course of the text, and are related to the main purpose for the book; they do not encompass the purpose itself.

This book is designed to be a comprehensive resource, defining the tools available to the application programmer in the UNIX V and XENIX System V environment. The reader of this book should come away with knowledge of what tools are and are not available for the application development process under these systems.

The title of the book, particularly the word "toolkit," is meant to provoke an image in the reader's mind. When a programmer begins work in the UNIX V environment, a large toolkit is opened. Not only is the vast array of UNIX shell commands suddenly available, but also revealed is a very specific set of utilities, commands, functions, and system calls explicitly devoted to improving the application development environment. Thus UNIX is an application programmer's toolkit: it contains all the necessary tools of the programmer's trade. [illegible]

There are two important steps in honing the skills required to become a master craftsman in the UNIX environment. First, one must become aware of what tools are available, and what purpose those tools serve. Second, the proper use of the tools must be learned through a combination of instruction and practice. This book is designed to facilitate the first of these steps.

The reader of this book will become aware of what UNIX V and XENIX System V have to offer. UNIX tools are defined and described, and during the course of this exploration, some detailed explanation of usage will be found.

The book is written for an (already) technically oriented individual who will be able to learn how to handle the tools offered. The reader should be able to accomplish the second step of the honing process, the use and practice, alone.

This is not a general-purpose introduction to UNIX. Generalities regarding UNIX usage and the basic shell commands are not explained. Instead, the book focuses on the features of UNIX which are of special interest to the application programmer. [illegible]

The book will probably not be of interest to the casual reader, and most certainly will not make *The New York Times* bestseller list. It is aimed at those who are technically oriented managers, and at programmers who already have a general working knowledge of UNIX and programming. These readers will find the book of value both as a means of increasing their knowledge of UNIX and for determining UNIX suitability for solving specific problems. A technical manager may read the book cover to cover to get a firm grasp on the application development capabilities available through UNIX. The programmer may refer only to specific chapters as an aid to understanding a particular tool—perhaps simply to find that slightly different slant in an explanation that can bring clarity to a confusing point of information. [illegible]

The terms UNIX and UNIX V are used throughout the book interchangeably, and are also used to encompass XENIX. Where UNIX and XENIX diverge, the divergence is specifically noted.

The Santa Cruz Operation (SCO), in conjunction with Microsoft, has released XENIX System V. XENIX System V is a true and excellent port of UNIX V as released by AT&T. All of the commands, functions, utilities, and system calls work as described in both UNIX V and XENIX System V with few exceptions, which are noted as they occur.

This book is divided into three distinct but integrated parts. Part I provides an overview of the UNIX toolkit. The intrepid programmer or technician should be able to begin practice and experimentation with the basic UNIX tools after reading only Part I. Part II provides detailed explanations of the most common UNIX tools, and is a good reference. Part III provides an overview of the more esoteric UNIX tools—those useful, sophisticated tools which are less commonly used.

Finally, this book does not stand alone. The *UNIX Reference Manual* will still be necessary for points of precise syntax, much as a dictionary is always necessary for an author. Reading about UNIX is like reading about playing golf: without actual practice, very little is learned. Understanding of UNIX ultimately comes with use. This book should provide a springboard from which the tools contained in the vast UNIX toolkit can become familiar. It is up to the reader to practice using them.

Contents

Part I

Chapter 1

An Overview of The UNIX V Programming Environment

UNIX V OFFERS THE PROGRAMMER ALL THE TOOLS NECESSARY FOR A successful programming experience. Useful features range from highly structured programming languages to sophisticated debugging aids, and include editors, calculators, compilers, linkers, source code control mechanisms, compilation aids, and a host of other accessible tools. The aggregate makes UNIX V in many respects an ideal programming environment.

A HISTORY OF UNIX

The somewhat eccentric evolution of UNIX is entering the realm of folklore. Because of its origins, the existence and present form of UNIX could hardly have been anticipated by its developers. The growing acceptance and increasing use of UNIX in production environments was never expected. The developers of this operating system initially wanted nothing more than a mechanism for handling files on a Digital Equipment Corporation PDP-7.

In the late 1960s, Ken Thompson (now UNIX legend, then just another mortal at Bell Laboratories in Murray Hill, New Jersey) worked under an operating system called MULTICS. By corporate fiat, MULTICS was eliminated as an allowable operating environment. Ken Thompson was left without an operating system under which to do his research.

The *modus operandi* at Bell Labs in those days was to encourage considerable independent research by providing certain employees with approximately 50% unstructured time during each work day. Ken Thompson's particular interest was an orrery (planetary motion) simulation application. After MULTICS was phased out, he acquired a DEC PDP-7 with no operating system sufficient for the complexities of the simulation.

To solve the problem, Thompson developed UNIX, deriving its name from a semisatirical pun on the previous system, MULTICS. UNIX was to function as a file handler and minimal operating system for the PDP-7. Thus, in 1969, UNIX was born, the result of a part-time project to allow files to be stored and manipulated on a PDP-7. It was written in B, an interpreter for CPL (Combined Programming Language), and in Assembler, which handled the hardware-dependent aspects of the project. The initial UNIX was a single-user, multiprogramming operating system.

The environment at Bell Labs at that time encouraged others to join in the development of this new operating system—but just for fun. Thus, after its initial release, Dennis Ritchie joined Ken Thompson in efforts to enhance UNIX. Others began to make contributions as they saw fit.

UNIX was not designed to be a coherent operating system for large applications in production environments. It was a kind of corporate hobby rather than a project with a clear objective. Thompson, Ritchie, and the others enjoyed the technical opportunities involved in UNIX development, but they absolutely were not trying to create a product. They were simply programmers building a programmer's environment.

After one intermediate step, Ritchie and Thompson rewrote UNIX in C—but only after Ritchie had written the C language under UNIX. This version, available in 1973, was the beginning of UNIX as it is known today. It consisted of approximately 10,000 lines of C code and roughly 2000 lines of Assembler for the machine-dependent functions. The on-line user's manual was a part of this version, cleverly allowing others from the lab to make contributions to the UNIX code, write documentation, and include it in the manual as they worked.

As UNIX continued to be developed, it was either freely given or sold at minimal cost to colleges and universities. Students began to learn and experiment with UNIX. Enhancements ensued, and UNIX-knowledgeable people began to appear in the data processing marketplace. The UNIX bandwagon had begun to roll.

Notable among the schools using UNIX was University of California at Berkeley. The enhancements emerging from the Berkeley campus were widely acclaimed and began to be absorbed into the body of UNIX proper. Among others, one of Berkeley's most significant contributions to UNIX was vi, UNIX full-screen editor.

Today, UNIX has evolved into UNIX 5.*x*. UNIX System 5 is intended to be a practical production environment. From its small beginnings, UNIX today is approximately 50,000 lines of C code and 5000 lines of Assembler. It has not lost its appeal to programmers. For many of the same reasons that UNIX and its ideal development environment are popular with programmers, it is much less suitable for the naive end user.

Anyone at all familiar with UNIX is aware of its unusual and arcane commands and messages. It is the job of the programmer to use those commands to write "bulletproof" applications that shield the user from the complexities of UNIX—while admitting the user to the full benefits of the UNIX timesharing application environment.

THE UNIX PROGRAMMING PHILOSOPHY

Despite what might be construed as scattershot development, there is a definite and consistent philosophy behind UNIX, a philosophy which has been with it since its inception by Kernighan and Ritchie. This philosophy can be briefly described in two sentences.

1. Build small, reliable, and reusable software tools.
2. Use those tools to build larger and more complex modules.

A *software tool* is a set of code that performs one particular function. That set of code, or *tool*, should be general enough to be useful from one application to another, and simple enough to use that a user will employ it rather than "reinvent the wheel" each time a new application is developed. UNIX is full of software tools.

The UNIX commands themselves illustrate the concept of software tools. For example, the command **whodo** (which tells who is doing what) is not an original concept. The command is a combination and merging of **who** (which tells who is logged on) and **ps** (which tells what processes are active and for whom). The **whodo** command uses the software tools **who** and **ps** to build on. Both **who** and **ps** solve general problems in ways that are simple to use. By itself, **whodo** solves a different general problem and becomes a software tool that stands alone.

UNIX was developed by programmers using the software tools concept. One programmer would look at the work of another, use it, and build on it to produce new results. C language encourages the use of small, reusable code modules to produce large and complex programs.

As an example, the lines of C code shown in Fig. 1-1 represent the entire structure of a complex C application. Because small, reusable pieces of code called *functions* are used, even the main controlling module is simple to understand and use.

THE UNIX OPERATING ENVIRONMENT

There are three parts to the UNIX operating system—the *kernel* with its file system, the *shell*, and the *command set*. These comprise the entirety of the UNIX operating system. The kernel is the heart, providing process control and performing the actual hardware/software interface. The file system controls files throughout the UNIX system, allowing files to be created, deleted, modified, located, and accessed. The shell, the removable and modifiable part of the UNIX operating system, provides an interpreter for a programmable command set. The command set is accessible by the programmer for easy use

```
/*help.c -- the main program for HELP software*/

#include "help.h"

main(argc, argv)

int argc;
char *argv;
                                /*if argc=1 then give help desc*/
                                /*if argc=2 then help describes
                                  a command*/
                                /*if argc>2 then help must parse
                                  a command line*/

{
clrscrn (RW1,CL1,LSTRW,LSTCL);          /*clear entire screen*/
helpmsk();                              /*put help mask up*/

if(argc==1)
     helpexp();                         /*give help desc*/
else
     opntxt();                          /*open help text file*/

if(argc==2)
     {
     cmdfind();                         /*locate command in file*/
     if(offset>0)
          dishelp();                    /*display command help*/
     }
else
     parseln();                         /*parse command line*/
}
```

Fig. 1-1. Through the use of reusable functions and program modules, even complex applications can be developed using relatively few lines of code.

and maintenance of programs and operating system functions.

The kernel is the portion of the operating system that shields the user from hardware and system software intricacies. The shell shields the user from the intricacies of the kernel. The file system is the file-handling mechanism for the kernel and the shell. With a few exceptions, the user can be shielded from all of the UNIX oddities by application programs written to interface with the kernel, shell, and file system. These applications should provide the user with a friendly, simple, and easily understood operating environment.

The programmer can also be shielded from the kernel by using the shell and its utilities—although the power of UNIX V comes from the programmer's