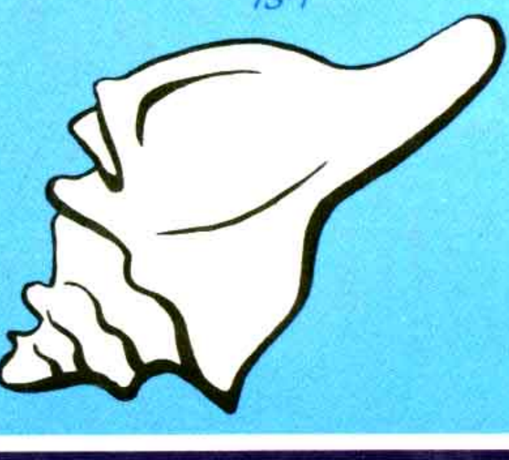
# NEWNES UNIX™ POCKET BOOK

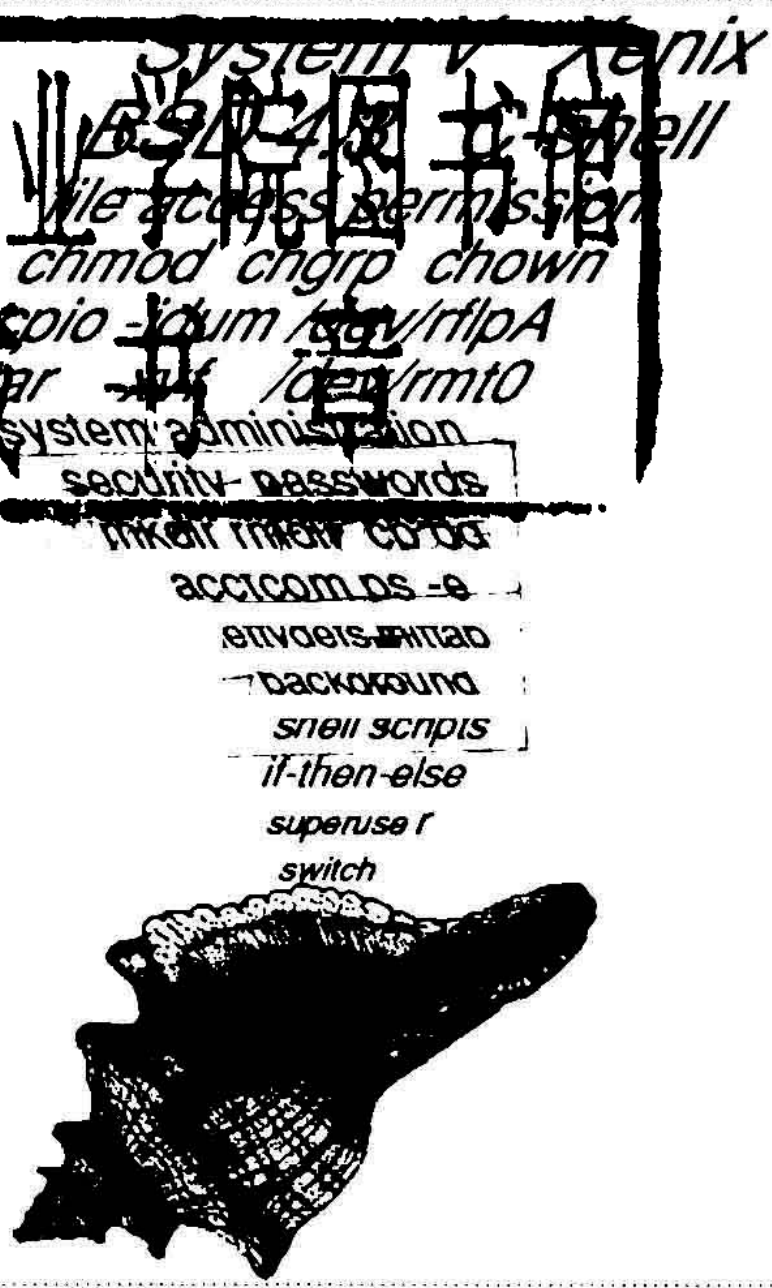
*System V Xenix*
*BSD 4.3 C-shell*
*file access permission*
*chmod chgrp chown*
*cpio-idium/dev/rflpA*
*tar-xvf /dev/ rmt0*
system administration
security passwords
mkdir rmdir cp dd
acctom ps- e
gettydefs inittab
background
*shell scripts*
*if-then-else*
*superuse r*
*switch*
*ls-l*



*Steve Heath*

A Butterworth-Heinemann Book

# NEWNES UNIX POCKET BOOK



Steve Heath

NEWNES

Newnes
An imprint of Butterworth-Heinemann Ltd
Linacre House, Jordan Hill, Oxford OX2 8DP

PART OF REED INTERNATIONAL BOOKS

OXFORD LONDON BOSTON
MUNICH NEW DELHI SINGAPORE SYDNEY
TOKYO TORONTO WELLINGTON

First published 1992

**British Library Cataloguing in Publication Data**
Heath, Steve
Newnes UNIX pocket book.
I. Title
004.165

ISBN 0 7506 03917

Printed and bound in Great Britain

*Dedicated to those unfortunates who have switched off their UNIX system without running* shutdown.

# Preface

The UNIX operating system combines everything about software that people dislike together with everything that they want. In other words, it is extremely complex but very powerful and flexible. The documentation that comes with the systems can only be described as immense, especially when compared with the user manuals supplied with MSDOS systems. This presents problems in making the documentation available, let alone guiding a user to the essential information that is needed to perform both simple and complex tasks. The aim of this book is to address this problem and provide an easy to use reference covering the majority of user commands and system functions for the majority of UNIX systems. Also included are plenty of examples and additional information which users need to know but frequently cannot find. This book is based on UNIX System V and the Bourne Shell but also includes the C shell and BSD environments. These form the basis of almost all the UNIX implementations available today. The commands and data are organised in terms of their functions, rather than as a straight alphabetical listing.

This book can be used in several different ways. It could be read from cover to cover but, for many, the best way is to either refer or browse as necessary. Chapter 1 provides a brief history of UNIX and explains how it works internally. Concepts like virtual memory, disk caching, multiuser and multi-tasking are explained here. For explanations and examples of the various commands, consult Chapter 2. The commands included in this chapter are generally common to most systems, although there is an appendix which cross references the command differences between System V and BSD. Advanced users can learn more about the Bourne and C shells in Chapters 3 and 4. These concentrate on the more complex features and shell script programming and include a large number of examples, including several scripts that act as DOS command equivalents.

UNIX editors are covered in Chapter 5 and Chapter 6 provides information for system administrators. This is essential reading for all users who have the responsibility of looking after a UNIX system — from a large multi-user system to a desktop workstation. It explains how to add users and control their access to the system, using printers, how to check the file system and several techniques for improving performance. The final chapter is concerned with the use of the C compiler — with the emphasis on the tools rather than the language itself. While there are many good books on C, they rarely include information on how to use the compilers, assembler, linker and libraries to their best advantage, yet using these tools is as important as understanding the C language itself. Finally, there are several appendices covering what to do when a command does not work or appears to be missing, a BSD command reference, an MSDOS to UNIX command cross reference and how to transfer data using serial communications.

I have used several conventions within this book:

- All the text in `courier` font is taken directly from the screen of a UNIX system - either a Motorola MC68010, MC68020, MC68030 CISC or MC88100 RISC based system running UNIX System V release 3 with BSD extensions/environment.
- Control characters are identified by preceding the character with a ^ e.g. ^C is the result of pressing the control key and the C key at the same time, ^BACKSPACE is the combination of control and backspace keys, and so on.
- As most of the commands are in lower case and UNIX is case sensitive, I have maintained lower case for these words, even when grammar dictates capitalisation of the first letter. To fully differentiate them, they are in *italics*.

- In command descriptions, square brackets indicate that the arguments or options within the brackets are optional. In other words, the command does not expect the square brackets when it is entered from a terminal. However, there are cases when square brackets form part of a command, such as file name generation. If in doubt, look at an example to see how the command is written and used in practice.

My thanks must go to Sue Carter once again for lots more coffee, editing, constructive criticism and support. Without her help, this book would not have been as much fun to write as it was.

*Steve Heath*

# Acknowledgements

# Contents

# Chapter 1

# Inside UNIX

UNIX has probably established itself as the most well known multi-tasking operating system within the microprocessor and mini computer environment because of a group of dedicated individuals, an interest in computer games and despite an apparent lack of interest from major corporations.

## Origins and beginnings

UNIX was first described in an article published by Ken Thompson and Dennis Ritchie of Bell Research Labs in 1974, but its origins owe much to work carried out by a consortium formed in the late 1960s, by Bell Telephones, General Electric and the Massachusetts Institute of Technology, to develop MULTICS - a MULTIplexed Information and Computing Service. Their goal was to move away from the then traditional method of users submitting work in as punched cards to be run in batches - and receiving their results several hours (or days!) later. Each piece of work (or job) would be run sequentially - and this combination of lack of response and the punched card medium led to many frustrations - as anyone who has used such machines can confirm. A single mistake during the laborious task of producing punched cards could stop the job from running and the only help available to identify the problem was often a 'syntax error' message. Imagine how long it could take to debug a simple program if it took the computer several hours to generate each such message!

The idea behind MULTICS was to generate software which would allow a large number of users simultaneous access to the computer. These users would also be able to work interactively and online in a way similar to that experienced by a personal computer user today. This was a fairly revolutionary concept. Computers were very expensive and fragile machines that required specially trained staff to keep users away from and protect *their* machine. However, the project was not as successful as had been hoped and Bell dropped out in 1969. The experienced gained in the project was turned to other uses when Thompson and Ritchie designed a computer filing system on the only machine available - a Digital Equipment PDP-7 mini computer.

While this was happening, work continued on the GE645 computer used in the MULTICS project. To improve performance and save costs (processing time was very expensive), they wrote a very simple operating system for the PDP-7 to enable it to run a space travel game. This operating system, which was essentially the first version of UNIX, included a new filing system and a few utilities.

The PDP-7 processor was better than nothing - but the new software really cried out for a better, faster machine. The problem faced by Thompson and Ritchie was one still faced by many today. It centred on how to persuade management to part with the cash to buy a new computer, such as the newer Digital Equipment Company's PDP-11. Their technique was to interest the Bell legal department in the UNIX system for text processing and use this to justify the expenditure. The ploy was successful and UNIX development moved along.

The next development was that of the C programming language, which started out as attempt to develop a FORTRAN language compiler. Initially, a programming language called B which was developed, which was then modified into C. The development of C was crucial to the rapid movement of UNIX from a niche within a research environment to the outside world.

UNIX was rewritten in C in 1972 - a major departure for an operating system. To maximise the performance of the computers then available, operating systems were usually written in a low level assembly language that directly controlled the processor. This had several effects. It meant that each computer had its own operating system, which was unique, and this made application programs hardware dependent. Although the applications may have been written in a high level language (such as FORTRAN or BASIC) which could run on many different machines, differences in the hardware and operating systems would frequently prevent these applications from being moved between systems. As a result, many man hours were spent porting software from one computer to another and work around this computer equivalent of the Tower of Babel.

By rewriting UNIX in C, the painstaking work of porting system software to other computers was greatly reduced and it became feasible to contemplate a common operating system running on many different computers. The benefit of this to users was a common interface and way of working, and to software developers, an easy way to move applications from one machine to another. In retrospect, this decision was extremely far sighted.

The success of the legal text processing system, coupled with a concern within Bell about being tied to a number of computer vendors with incompatible software and hardware, resulted in the idea of using the in-house UNIX system as a standard environment. The biggest advantage of this was that only one set of applications needed to be written for use on many different computers. As UNIX was now written in a high level language, it was a lot more feasible to port it to different hardware platforms. Instead of rewriting every application for each computer, only the UNIX operating system would need to be written for each machine - a lot less work. This combination of factors was too good an opportunity to miss. In

September 1973, a UNIX Development Support group was formed for the first UNIX applications, which updated telephone directory information and intercepted calls to changed numbers.

The next piece of serendipity in UNIX development was the result of a piece of legislation passed in 1956. This prevented AT&T, who had taken over Bell Telephone, from selling computer products. However, the papers that Thompson and Ritchie had published on UNIX had created a quite a demand for it in academic circles. UNIX was distributed to universities and research institutions at virtually no cost on an 'as is' basis - with no support. This was not a problem and, if anything, provided a motivating challenge. By 1977, over 500 sites were running UNIX .

By making UNIX available to the academic world in this way, AT&T had inadvertently discovered a superb way of marketing the product. As low cost computers became available through the advent of the mini computer (and, later, the microprocessor), academics quickly ported UNIX and moved the rapidly expanding applications from one machine to another. Often, an engineer's first experience of computing was on UNIX systems with applications only available on UNIX. This experience then transferred into industry when the engineer completed training. AT&T had thus developed a very large sales force promoting its products - without having to pay them! A situation that many marketing and sales groups in other companies would have given their right arms for. Fortunately for AT&T, it had started to licence and protect its intellectual property rights without restricting the flow into the academic world. Again, this was either far sighted or simply common sense, because they had to wait until 1984 and more legislation changes before entering the computer market and starting to reap the financial rewards from UNIX.

The disadvantage of this low key promotion was the appearance of a large number of enhanced variants of UNIX which had improved appeal - at the expense of some compatibility. The issue of compatibility at this point was less of an issue than today. UNIX was provided with no support and its devotees had to be able to support it and its applications from day one. This self sufficiency meant that it was relatively easy to overcome the slight variations between UNIX implementations. After all, most of the application software was written and maintained by the users who thus had total control over its destiny. This is not the case for commercial software, where hard economic factors make the decision for or against porting an application between systems.

With the advent of microprocessors like the Motorola MC68000 family, the Intel 8086 and the Zilog Z8000, and the ability to produce mini computer performance and facilities with low cost silicon, UNIX found itself a low cost hardware platform. During the late 1970s and early 1980s, many UNIX systems appeared using one of three UNIX variants.

XENIX was a UNIX clone produced by Microsoft in 1979 and ported to all three of the above processors. It faded into the background with the advent of MSDOS, albeit temporarily. Several of the AT&T variants were combined into System III, which, with the addition of several features, was later to become System V. The third variant came from work carried at out at Berkeley (University of California), which produced the BSD versions destined to became a standard for the Digital Equipment Company's VAX computers and throughout the academic world.

Of the three versions, AT&T were the first to announce that they would maintain upward compatibility and start the lengthy process of defining standards for the development of future versions. This development has culminated in AT&T System V release 4, which has effectively brought System V, XENIX and BSD UNIX environments together.

What distinguishes UNIX from other operating systems is its wealth of application software and its determination to keep the user away from the physical system resources. There are many compilers, editors, text processors, compiler construction aids and communication packages supplied with the basic release. In addition, packages from complete CAD and system modelling to integrated business office suites are available.

## Inside UNIX

The key to understanding UNIX as an operating system is to understand how much UNIX protects the user from the computer system it is running on - from having to know exactly where the memory is in the system, what a disk drive is called and other such information. Many facets of the UNIX environment are logical in nature, in that they can be seen and used by the user - but their actual location, structure and functionality is hidden. If a user wants to run a 20 Mbyte program on a system, UNIX will use its virtual memory capability to make the machine behave logically like one with enough memory - even though the system may only have 4 Mbytes of RAM installed. The user can access data files without knowing if they are stored on a floppy or a hard disk - or even on another machine many miles away and connected via a network. UNIX uses its facilities to present a logical picture to the user while hiding the more physical aspects from view.

While it is perfectly possible to simply accept the logical world that UNIX presents, it is extremely beneficial to understand its inner workings and how UNIX translates the logical world to the physical world.

## The UNIX file system

UNIX has a hierarchical filing system which contains all the data files, programs, commands