Yukun Liu
Yong Yue
Liwei Guo

UNIX Operating System

The Development Tutorial via UNIX Kernel Services

UNIX 操作系统

依据 UNIX 内核服务的开发指南（英文版）

高等教育出版社
HIGHER EDUCATION PRESS

Yukun Liu
Yong Yue
Liwei Guo

UNIX Operating System

The Development Tutorial via UNIX Kernel Services

UNIX操作系统是一个源代码操作系统，广泛应用于企业级行业应用领域以及嵌入式设备中。本书全面地、系统地介绍了IUNIX操作系统的开发和管理原则、内核服务、shell、计算机联网和应用。内容包括五个部分：背景以及如何开始、文本编辑器、UNIX的内核服务、UNIX的命令解释以及编程和UINX的网络连接。

本书可作为高等院校计算机专业研究生和高年级本科生的教学参考书，也可供程序设计员参考。

Yukun Liu（刘玉坤） 河北科技大学计算机系副教授，英国贝德福特大学应用计算研究所博士研究生。

Yong Yue（岳勇） 英国贝德福特大学教授，应用计算研究所主任，计算机科学与技术系执行主任。

Liwei Guo（郭立炜） 河北科技大学教授，信息科学与工程学院院长。

关键词：UNIX 操作系统 内核服务 Shell 计算机联网

高等教育出版社与Springer公司合作出版
海外发行ISBN：978-3-642-20431-9

学科分类：计算机

ISBN 978-7-04-031907-1

9 787040 319071 >

定价 69.00 元

http://academic.hep.com.cn

Liu • Yue • Guo

UNIX Operating System
UNIX 操作系統

Yukun Liu
Yong Yue
Liwei Guo

UNIX Operating System

The Development Tutorial via UNIX Kernel Services

UNIX 操作系统

依据 UNIX 内核服务的开发指南 (英文版)

UNIX Caozuo Xitong
Yiju UNIX Neihe Fuwu de Kaifa Zhinan

With 132 figures

高等教育出版社·北京
HIGHER EDUCATION PRESS BEIJING

Authors

Associate Professor Yukun Liu
College of Information Science and Technology
Hebei University of Science and Technology
Hebei 050018, China
Institute for Research in Applicable
Computing, University of Bedfordshire
E-mail: lyklucky@ hebust. edu. cn

Professor Yong Yue
Faculty of Creative Arts, Technologies
and Science
University of Bedfordshire
Park Square Luton Bedfordshire
LU1 3JU, United Kingdom
E-mail: yong. yue@ beds. ac. uk

Professor Liwei Guo
College of Information Science and Technology
Hebei University of Science and Technology
Hebei 050018, China
E-mail: guoliwei@ hebust. edu. cn

图书在版编目（CIP）数据

UNIX 操作系统：依据 UNIX 内核服务的开发指南 = UNIX Operating System: The Development Tutorial via UNIX Kernel Services：英文 / 刘玉坤，岳勇，郭立炜编著. —北京：高等教育出版社，2011.4
ISBN 978-7-04-031907-1

Ⅰ. ①U… Ⅱ. ①刘… ②岳… ③郭… Ⅲ. ①UNIX 操作系统-英文 Ⅳ. ①TP316.81

中国版本图书馆 CIP 数据核字(2011)第 042816 号

策划编辑 陈红英　**责任编辑** 陈红英　**封面设计** 张　楠　**责任印制** 刘思涵

出版发行	高等教育出版社	**购书热线**	010-58581118
社　　址	北京市西城区德外大街 4 号	**咨询电话**	400-810-0598
邮政编码	100120	**网　　址**	http://www. hep. edu. cn
			http://www. hep. com. cn
经　　销	蓝色畅想图书发行有限公司	**网上订购**	http://www. landraco. com
印　　刷	北京中科印刷有限公司		http://www. landraco. com. cn
		畅想教育	http://www. widedu. com
开　　本	787×1092　1/16	**版　　次**	2011 年 4 月第 1 版
印　　张	24.25	**印　　次**	2011 年 4 月第 1 次印刷
字　　数	660 000	**定　　价**	69.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

物料号　31907-00

Sales only inside the mainland of China　（仅限中国大陆地区销售）

Yukun Liu
Yong Yue
Liwei Guo

UNIX Operating System

The Development Tutorial via UNIX Kernel Services

Preface

This book presents UNIX as a practical case of operating systems for the reader to understand and master deeply and tangibly the theory and algorithms in operating systems. It gives discussions and studies on the hierarchical structure, principles, applications, shells, development, and management of the UNIX operation system multi-dimensionally, systematically and from the elementary to the profound. It brings readers to go into the inside of the UNIX operating system and lets them understand clearly what and how UNIX operating system functions.

Subject Matter

This book consists of 11 chapters. The first two chapters discuss the background of UNIX operating system (OS), and give a whole picture of what UNIX OS looks like and the evolution of UNIX.

Chapter 3 focuses on the editors that will be used frequently by UNIX users, no matter who are regular users or seasoned programmers.

Chapters 4, 5, 6, and 7 concentrate on the services of the UNIX kernel. Chapter 4 zooms in the process management, which is usually hidden from the final users. Chapter 5 is to discuss the UNIX memory management, which cooperates with the process management to accomplish the processes' concurrently running. Chapter 6 introduces the UNIX file management, which is involved almost in every service that the UNIX kernel provides to the users. Chapter 6, however, for UNIX users, is fundamentally useful to understand how UNIX works. Chapter 7 explores UNIX I/O, I/O redirection and piping. As UNIX treats its hardware devices as special files, this mechanism brings a whole different concept on UNIX input and output devices. I/O redirection and piping are two useful tools that can be used to deduce different commands to control the terminals through UNIX system calls.

UNIX has almost as many shells as versions of the UNIX operating system. Chapter 8 introduces some types of shells, shell evolution, and some common concepts in UNIX shells. As there are so many kinds of shells, it

is difficult to put all of them in one book. Hence, our strategy is to try to make one of them clear and integral in this book. Our choice is the primary shell for all other shells—Bourne shell. From this point, the readers can learn other shells by themselves from references. Therefore Chapters 9 and 10 focus on the discussion of Bourne shell as a programming language: Chapter 9 introduces basic facilities and Chapter 10 is for the advanced level.

Different from the studies in the previous chapters, which are concentrated on the local services and applications of UNIX in individual computers, Chapter 11 discusses the remote and network functions and services of UNIX in servers and workstations. Since the late 1960s, UNIX has had many original contributions to the development history of the computer networking and Internet.

Even though this book includes 11 chapters, it does not mean they are totally divided and irrelevant. Contrarily, they are coherent and relevant each other. Just like UNIX itself, its knowledge should be a link-up and cooperative "system". And we try very hard to unfold the information gradually and step by step in this book. When you, dear friends and readers, finish this book, you will have a relatively whole and systematical idea about UNIX. From that point, you can develop your applications on UNIX or other operating systems, or even build up a new operating system for a certain computer hardware system. This is just what the authors of this book really expect.

Historic and Active UNIX and Meaningfully UNIX Learning

As an open-source operating system, UNIX made its history during two decades of 1969 –1989. Maybe some say it has gone. However, UNIX's openness, which brought different groups of developers together to communicate their developing ideas and to respond feedback each other, really cultivated an excellent generation of operating system developers. We should remember these names: Dennis M. Ritchie, Ken Thompson, Robert S. Fabry, William N. Joy, Chuck Haley, Samuel J. Lefflerand, and more. The first two made their contribution to the premier UNIX System series and were honored by the ACM Turing Award in 1983 because of their work in UNIX, and the latter four did their great work on the primary UNIX BSD versions. Just as they made the earlier UNIX code open to the academic world and strived to move UNIX from one machine to another, UNIX grew and evolved. And even more, it left a lot of valuable academic papers about operating systems for the successors.

For its development process as an intact software system, UNIX, which presented solutions in detail, is unchallenged for generations of programmers Compared to UNIX, its commercial counterparts usually provide a perfect environment that hides almost all the development details of lower levels of operating systems, which may leave a limited space for application program-

mers and also confine their imagination and creativity. This tendency can also affect the ability of system development newcomers to develop an intact software system that can handle software as well as hardware by restricting the field of vision to some detached modules or applications so as to result in software maintenance costly and complicated.

Further, just understanding the theory of operating systems, readers cannot image and understand well how the operating system works. With UNIX as a real case, readers can map the abstract algorithms, mechanisms and strategies of operating system theory into the real modules, functions and programs of UNIX implementation one-to-one. The abstract theory can be exemplified. In this way, as the promising programmers, readers can understand well and master these algorithms and mechanisms, practice them in their own development, and stimulate novel algorithms and mechanisms that may be more effective and efficient to their own context.

It seems as if a repetition of the old tale when considering the discussion on UNIX, which, all in all, reached its heyday around 1980s. In the latest two decades, however, due to commercial purposes and activities, there are no other operating systems like UNIX, which is so thoroughly open for the academic community to learn and do research.

In addition, there are plenty of references about UNIX that have been published, but most of them were originally published around 1980s. For the recent promising programmers, the published classics may be somewhat obscure because of the sparse context that might not be necessary for readers in those days but can be unfamiliar to nowadays readers. As the well-known rapid development of computer hardware in the latest decades, computer architecture and structure have made a big change. This change has also wielded a deep influence on the theories and concepts of computer, which makes the difficulty for recent readers to understand well descriptions and expressions in the published UNIX classics, and to map them properly into practical cases. It is possible to build an obstacle for readers to learn from them. Otherwise, for the operating system construction, which belongs to software developments but resides the one of the most exciting and integrated of software development, it would be a pity and defect if losing an operational means. Fortunately, this means can be gained by doing research on UNIX.

It is taken that UNIX has its own philosophy and several items in the philosophy are written in different references. If having the right, we can say that the most important one should be the UNIX programmers' dedication and passion to their work. UNIX is also deemed to a programmer's OS. UNIX programmers have done a wonderful work just as for tackling a necessary affair, from which others else really benefit. It is critical for the academic community.

UNIX benefited also from those days. If AT&T, at that time, could market computer products without a 1956 Consent Decree signed with the Federal Government, and if Bell Laboratories did not withdraw Ken Thompson and others from the MULTICS project, and if Professor Robert S. Fabry of the

University of California at Berkeley did not contact Ken Thompson at the Symposium on Operating Systems Principles at Purdue University in November 1973, we would have a totally different story about UNIX. It needs the open and free soil to breed an academic activity. The more relieved the outside environment is, the more natural the academic activity develops within the environment. UNIX was destined for being flourishing in its day.

Even though being just observers on this period of history, the authors of this book are impressed by the passion and concentration that UNIX developers had in the day. During five years of teaching UNIX in their campuses, the authors realized that if this fantastic piece of history was not introduced to more readers, it would be a pity for authors as well as readers. In this high-technology and high-material-civilization age, UNIX development process can give readers some new inspiration—a glowing motivation from inside to accomplish a meaningful work.

A General Survey of UNIX Development

Observing different versions of UNIX emerging, the authors and readers can discover that it is a process of constant development, amendment and enhancement. In this process, UNIX developers' thoughts were adjusted and enriched with the development of computer hardware and peripherals, and the proposal of new application demands. It resulted in UNIX' s being moved to execute on different hardware platforms and fitting in different projects, which also naturally made UNIX' s portability and scalability practice and reinforce repeatedly and concretized the concepts of portability and scalability in operating system theory.

UNIX drew on a lot of ideas of the earlier operating systems, and its openness made the idea-drawing expand into different UNIX versions and different groups of developers. For a new programmer, it is also necessary to derive the precursors' development thoughts and experiences. Only if learning from the precursors, newcomers can enrich their knowledge gradually and effectively, and the novel thinking can just grow from thick knowledge reserves.

For promising developers, the UNIX development process was also a training program. Linux is a successful example of UNIX derivatives. Through this training program with deducing mentally and programming physically, developers can get familiar with the computer system as a whole, including both hardware and software.

With the advent of commercial operating systems, most of the readers do their jobs on encapsulated and transparent operating systems. On the other hand, many students and graduate students of computer disciplines mostly start their studies from the theory of operating systems. A transparent, well-designed and inextricable operating system seems like saving the users a lot of time and effort, but it also cuts the exploring road towards the inside

of operating systems and the underlying hardware parts. For real developers and programmers, it may take a big risk to sit on a seemingly-transparent but unfamiliar system to do their developments—finally they may encounter some bugs that they cannot tackle. They have to experience something that can let them understand what really make the construction of an operating system, what the kernel of an operating system does for users, and how algorithms and mechanisms in the theory of operating systems are implemented. Even though the disassembled UNIX cannot tell all the story of a well-designed modern or future operating system, it can give the mapping or clues to different functions and services, which can be treated as an anatomy lecture of a promising surgeon.

In other words, a well-designed operating system may be daunting for a promising developer, which is complicated and confused. The simplicity and clarity of UNIX can help readers walk out of the swamp and sort out the confusion, and lead them to face and tackle more sophisticated problems.

Targets and Strategy of this Book

Knowledge needs to renew and information needs to update. The updating includes the expression of a convincing, successful and classical process in a proper, timely and new way. Maybe the UNIX story is old, but it can give different inspirations to people in different ages, which is still developing. The authors hope the developing can be reflected in this book.

One of the targets of this book is to let the UNIX philosophy propagate and carry on. Let more readers comprehend this philosophy's results—the fast development, maintainability and scalability of an operating system.

The authors also want to present readers (especially, programmers) two aspects of a whole operating system and any one of its subsystems, in other words, to give not only the inside implementation process, which is viewed by the system or application programmers, but also the outside application performance, which is usually felt by the end users. In this way, readers cannot only keep the view of the system constructors but also be considerate of the end users when developing their systems. During development, a system can benefit from that its developers can consider more for its end users.

For readers, it is easy to enter the learning from user interfaces of UNIX operating systems since they have usually had the experience of using one of operating systems in their daily works or lives. Thus, in this book, we take this strategy: when starting one topic, we present it from its user interface, and then go into the kernel with the system calls and lower-level algorithms if possible. In this way, readers can be brought from a familiar environment into elusive and deep techniques.

To describe algorithms, we try to use common English language rather than some computer language, such as C or assembly language. The primary

reason is: we try to make algorithms more readable and help readers save their time and effort. For a programmer, it is often time-consuming to read some code that is written by others.

Intended Audience

This book is written for the wide groups of readers who want to master the professional knowledge of operating systems through a real and open-source case. Its main readers include graduates, senior undergraduates and teachers of computer and software majors, and potential researchers on applicable computing and engineering modeling. The readers can also be ones who maybe have some or have not much knowledge related to Computer Science and Technology and Software Engineering, but have a strong interest in these fields and want to get into them quickly, acquire some useful and important knowledge and reach an advanced level in the relevant fields after learning. This book can help readers construct, not only as the users of operating systems but also in the view of the operating system designers, the knowledge on the UNIX operating system, and even on other kinds of operating systems. From this point, readers can build up their projects on an operating system. Or on this basis, readers can go deep into how UNIX and other operating systems to be designed and programmed because many versions of UNIX are open-source code, like Linux, and adjust and modify the operating systems on their own computer systems.

For readers whose mother tongues are not English, it may be more difficult to read and learn an English edition of the academic book than books written with their mother tongue. However, it is necessary for readers to have the ability to read the English editions of academic books, especially for computer and software professionals, because most papers on the advanced and update science and technology are written in English, especially in the field of computer hardware and software. Why not to try to gain this ability just from your learning process? Maybe it is difficult for you now. But nothing is easy when starting it. Maybe when you finish this book, you say, "It is not that hard, is it?" So try it now.

Yukun Liu
Yong Yue
Liwei Guo
January 2011

Acknowledgements

This book is funded by Academic Work Publication Fund of Hebei University of Science and Technology.

The authors of this book would like to give their thanks to Ms. Hongying Chen, Editor of High Education Press in China, who gave generously of her time and expertise to edit this book.

Contents

1 Background of UNIX Operating System 1
1.1 Introduction of Operating System 1
1.2 Types of UNIX 3
1.3 History of UNIX 4
1.4 Summary 6
Problems 7
References 7

2 How to Start 9
2.1 UNIX Software Architecture 9
2.1.1 UNIX Kernel 10
2.1.2 System Call Interface 12
2.1.3 Standard Libraries and Language Libraries 14
2.1.4 UNIX Shell 14
2.1.5 Applications 14
2.2 UNIX Environment 15
2.3 Character User Interface Versus Graphical User Interface 16
2.4 UNIX Command Lines 17
2.4.1 UNIX Command Syntax 18
2.4.2 Directory Operation Commands 19
2.4.3 File Operation Commands 24
2.4.4 Displaying Online Help 30
2.4.5 General Utility Commands 32
2.4.6 Summary for Useful Common Commands 34
2.5 UNIX Window Systems 35
2.5.1 Starting X 35
2.5.2 Working with a Mouse and Windows 36
2.5.3 Terminal Window 37

2.5.4 Using a Mouse in Terminal Windows 37
2.6 Shell Setup Files 38
2.7 Summary 40
Problems 41
References 43

3 Text Editors 45
3.1 Difference Between Text Editors and Word Processors 45
3.2 Introduction of Pico Editor 46
3.2.1 Start pico, Save File, Exit pico 47
3.2.2 Create a New File with Pico 48
3.2.3 Cursor-moving Commands in Pico 49
3.2.4 General Keystroke Commands in Pico 50
3.3 The vi Editor and Modes 52
3.3.1 Three Modes of the vi and Switch Between Them 52
3.3.2 Start vi, Create a File, Exit vi 53
3.3.3 Syntax of the vi Commands 55
3.4 Practicing in Insert Mode of the vi Editor 56
3.5 Practicing in Command Mode and Last Line Mode of the vi Editor 62
3.6 Using Buffers of the vi Editor 65
3.7 The vi Environment Setting 67
3.8 Introduction of the emacs Editor 69
3.8.1 Start emacs, Create File, Exit emacs 70
3.8.2 Buffers, Mark and Region in emacs 71
3.8.3 Cursor Movement Commands 72
3.8.4 Keyboard Macros 73
3.8.5 Search and Replace 73
3.8.6 Operation Example 74
3.8.7 Programming in emacs 76
3.9 Summary 77
Problems 77
References 79

4 UNIX Process Management 81
4.1 Multiple Processes' Running Concurrently 81
4.1.1 Fundamental Concept for Scheduler and Scheduling Algorithm 83
4.1.2 UNIX Scheduling Algorithm and Context Switch 84
4.2 Process States 86