was awarded the Homi Bhabha Prize for research in applied sciences by the University Grants Commission, India, in 1986, and a National Award for Excellence in Computer Engineering by the Indian Society for Technical Education, in 1988. He was a member of the Electronics Commission from 1979–1982 and is currently a Fellow of the Indian National Science Academy, Indian Academy of Sciences and Indian National Academy of Engineers, and an Honorary Professor of the Jawaharlal Nehru Centre for Advanced Scientific Research.

Computer Operating Systems in Electrical Engineering Curriculum

Chyan Yang, *Senior Member, IEEE*

Abstract—**Diversified in their background knowledge, electrical engineering students cover a wide spectrum of strengths and weaknesses when they first attempt to study the subject of operating systems. Commonly, students are initially more concerned with the hardware design of digital systems and, therefore, less appreciative of the principles of software systems. In light of this, this paper presents and examines a practical syllabus for a course on operating systems. It also discusses possible derivations from this syllabus so as to provide versatility in implementation.**

I. Introduction

WHEN the earliest operational electronic computer provided the first services to mankind, with it came a set of operating procedures dedicated to the efficient utilization of the computer. First generation computers required complex procedures to maintain their operations. These procedures, though usually requiring some human interactions, were perhaps optimistically labeled as "operating systems." With advances in technologies and experiences in computer organizations, various computer architectures have evolved through the last three decades. The accompanying operating systems also have become more sophisticated. Many manual operating procedures are now automated and the need for human operations diminishes mainly to policy monitoring and file backups. Despite the changes, however, the issues concerning operating systems remain concrete. Although the relative importance of various issues fluctuates over time, the fundamental core of operating systems stems from the coordination of system resources and maximization of performance. Comparatively, electrical engineers design computers much in the same way that mechanical engineers design cars. We put functional components together to satisfy a hardware specification. The engineering efficiency of a car, however, is mainly determined by the entire construction or architecture of the car and not by the engine alone. Similarly, the performance of a computer relies not only on the hardware but also on the operating system that interfaces the hardware and the users. Many operating-system features may be the direct consequence of the underlying hardware support and many hardware specifications come directly from the operating system requirements. For instance, with indivisible instruction support, an operating system can provide concurrency control whereas an optimized compiler may call for a hardware design specification of register windows. The relationship between an operating system and its underlying hardware may sometimes appear as a chicken-and-egg problem: whether the hardware specification came from the operating system requirements or the operating system requirements came from the hardware reality? One could stake a convincing argument on either or both cases. In order to appreciate the wonder they set out to design, electrical engineering students must understand the importance and ramifications of operating systems.

Manuscript received July 1992.

The author is with the College of Management, National Chiao Tung University, Taiwan, Republic of China.

IEEE Log Number 9205768.

II. What to Learn and How Much?

It is not difficult to explain a CPU (central processing unit) to students of electrical engineering if they have had digital systems as their major. With a major in digital systems most students should have taken at least a course that uses the assembly language of some microprocessors. Instruction set architectures, macro definitions, and conventional branch instructions are terms covered in a standard syllabus of microprocessor programming. In addition, even students majoring in circuits and electronics, electromagnetics, control or communications commonly have knowledge of at least one high-level programming language. Be it as it may, not every student attending a class on operating systems is equipped with a proper background in basic computer organization. From experience, the understanding of the CPU functionality is a crucial part that cannot be skipped. Just as with an engine alone one cannot drive, with a lone CPU one cannot compute. An overall computer organization has to be reviewed. Hopefully, the functional components of a computer system can be covered within one lecture depending on the background of students. Only a few novices may have problems in this area.

TABLE I
COURSE OUTLINE

Subject	Reading
Why do we need an operating system?	(1.1)
History of OS	(1.2)
Minimal skills for reading C programs and using UNIX	handouts
Fundamentals	(1.3)
System Calls	(1.4)
OS Structure	(1.5)
Processes and Communication	(2.1)
Mutual Exclusion	(2.2)
Semaphores	(2.2)
Event Counters and Monitors	(2.2)
Message Passing	(2.2)
Classical IPC Problems	(2.3)
Process Scheduling	(2.4)
I/O Hardware	(3.1)
I/O Software	(3.2)
Deadlocks	(3.3)
Disks	(3.6)
Terminals	(3.8)
Memory management	(4.1 and 4.2)
Virtual Memory	(4.3)
Page Replacement Algorithms	(4.4)
Design Issues for Paging Systems	(4.5)
Segmentation Systems	
File Systems Design	(5.1 and 5.2)
File Servers and Network File Systems	(5.3)
Security	(5.4)
Protection Mechanisms	(5.5)

Comparable to exception handling, these students can be dealt with on an individual basis.

Table I lists a typical operating systems course outline for a quarter. Subject items and the associated reading sections are taken from Tanenbaum's book [9]. When adopting other textbooks, the reading sections in Table I should change to reflect the chosen texts although most of the subject items can still remain intact. The author has tried the same syllabus on different textbooks [1]–[5], [11], [12] while periodically supplementing them with handouts as required.

How much should an average engineering student know about the computer operating systems? This depends heavily on the preparation of students. The same subject may be covered at different depths in accordance to the variety of background knowledge. Absolute coverage of all materials can never work well. Students in each institution show different learning styles and tolerate different expectations from instructors. The actual level of details, regardless of the textbook chosen, depends on the preparation of students. It is to the benefit of both the instructor and students to adjust their level of detail dynamically and consistently. Dynamic adjustments come from the feedback of students: exercises, examinations, and questions. Consistency must be maintained in communicating the main issues of each subject. Additionally, the time span of instruction also plays an important role; one quarter's appetite is inherently more intensive than a semester's dose since most instructors tend to cram all essential subjects in rapid sessions. The temptation of lecturing faster than necessary can be avoided when one regulates the progression by observing a well-tested schedule.

III. METHODOLOGY OF LEARNING

To cook one meal for many and please everyone is possible only when carefully designed: know the preparation of students. There is no universal 'average' electrical engineering student when designing a course. Each department of each school has its own 'average' student. This student may not exist in reality yet the curriculum must be catered to a standard coverage. At the school where the author teaches, the rule of thumb is to teach students at the 80th percentile. Yes, some students may be bored and some may be lost. Confucious insisted on the same rule in his educational philosophy: there is not much one can do for those outside the normal range, either too bright or too dim.

UNIX[1] operating systems are commonly available in today's universities. Due to its wide availability and the design philosophy of reusability, UNIX is a good example operating system to explain and to learn in schools. Advanced workstations or personal computers can now run UNIX operating systems. Assignments may be designed around the UNIX systems. Therefore, it is assumed that UNIX operating system is available to the instructors and students in the following discussions. The method of having two phases of performing assignments are tested and have proven successful. The first phase is to familiarize oneself with UNIX as a user. The second phase is to do some projects with UNIX as a designer.

A minimum set of UNIX commands is introduced to students right after the elements of computer organization are covered. Meanwhile, the instructor has to explain the function of a command interpreter. A command interpreter provides part of the view that an operating system is an extended machine to users. Having explained how a command interpreter works, an instructor may start the assignment of shell scripts. Programming UNIX shell scripts is basically programming the command language in other commercial machines.

Shell programming is the best exercise with which to cultivate a good user since it forces students to study relevant commands by browsing through manuals. The author's favorite shell programming assignments always require the use of 'awk' commands. New users of UNIX operating systems generally find their first few hours' experiences miserable and dissatisfying. Like many software tools, this uncomfortable learning hurdle will soon fade. With the experiences of using 'awk' students come to appreciate the terseness of manual pages and the reusability of programs. If a strange command name 'awk' can be accepted, any other command names in UNIX are reasonable. Besides, awk is the first command in the UNIX manual. It has been noted that many students later write their own shell scripts for their research work since there are so many tasks that can be automated and many UNIX programs are reusable.

During shell programming, the concept of 'pipe' is also introduced. The usage of pipes will help in explaining concurrent processes. Since time is at premium, students are not expected to read the entire UNIX manual. Becoming a good user is a continual effort and shell programming is just a spark to jump start students' enthusiasm in learning

[1] UNIX is a trademark of Bell Laboratories.

the operating systems. Moreover, shell programming helps in explaining UNIX system calls. Covering system calls early shows students how an operating system can be used. Having covered the fundamentals, an instructor may start launching the enumeration techniques of process management to motivate students' interest and humble their distaste of the 'soft' science. Soon students will find process management is a difficult subject. When most students have a tough time in learning process management, an instructor is under pressure to explain it well.

With experience as a user, students should at least have learned part of the operating system's functions. Namely, several processes can be running concurrently, several useful programs are already available. To be a good designer one must be capable of performing logical reasoning, an indispensable ingredient of a good programmer. Although not everyone will be future designer of operating systems, the training of design principles undisputedly provides the students with a system perspective of computers.

Today's textbooks in operating systems are abundant and the fundamental issues are well understood. The key to design principles exhibits no exception to other engineering disciplines: coordination and management of various components and resources in a computer system. Students have to be reminded repeatedly on why the subjects are important and where they fit in the broader perspective.

Homework problems should be assigned and selective ones discussed in class. By doing homework problems students are given the opportunity to reread the text and to think. Programming assignments are the best tools to experiencing how important and how realistic certain design principles are. Again, the temptation of giving sizable programming assignments may jeopardize students' appetite. If the programming assignments are not handled with care, many students later may only remember sleepless nights of doing the programming assignments and can not recall specifically what the basic issues of operating systems are. A single, large, comprehensive programming assignment may be good in some computer science departments. In electrical engineering departments, however, several small programming assignments might be a better choice. Recall that in learning to be a good user, an assignment of shell programming was given which can generally be completed within a week. Similarly, each programming assignment should be done within one or two weeks.

IV. Variants and Enhancements

Instructors may emphasize different aspects of the modern operating system according to the needs of succeeding courses. In electrical engineering education, it is most likely that there would be a necessary follow-on course on computer networks. However, it is entirely possible for one to have the course of operating systems as the terminal course in computer engineering. For this possible exception, one needs to cover some tertiary subjects. Three main subjects whose relative importance will emerge in the coming decade are parallel processing, distributed systems, and computer security. A brief discussion on these three subjects is given in this section.

A. Parallel Processing

One example of parallel processing is the use of iPSC/2[2] (Intel Personal Super Computer). Readers interested in details should consult technical manuals [6]. In an iPSC/2 system, a number of processors or nodes work concurrently on parts of a problem. An iPSC/2 system consists of compute nodes and a front-end processor called the host. A typical iPSC/2 application has a host program that runs on the host and a node program that runs on a group of allocated nodes called a cube. The host program executes in the UNIX environment as a process. It initializes the application, provides any necessary human interface, and loads the node program onto the nodes. Generally, a node program performs calculations, exchanges messages with other nodes, and sends the results back to the host. There are several useful language constructs for parallel programming in the form of as system calls in iPSC/2. These system calls allow each node to communicate with other nodes within a parallel computer.

B. Distributed Systems

A stand alone computer generally runs a single operating system. With a multitude of computers within an organization, possibly dispersed geographically, there may be many different operating systems. What is the best strategy for resource sharing and load balancing? How should a file system be organized? Several subjects covering this avenue are system performance, process migration, file systems, file recovery, synchronization and concurrency control.

The collection of networks and gateways which use the TCP/IP[3] protocol suite and function as a single, cooperative virtual network is normally called internet. The internet allows researchers around the world to share information by a program called ftp (file transfer protocol). All formal specifications of internet can be found as 'Request For Comments' (RFC documents) and are stored on-line at many sites. If students can access the internet, it is very instructive to have short assignments on using internet commands. For example, using ftp to get RFC1094 [9] and study the network file systems. Alternatively, one may ftp RFC1057 [8] and study the remote procedure calls. There are several sites open to public ftp. One site with complete archive is ftp.nisc.sri.com.

C. Computer Security

Why computer security? Although some computer systems are still operated in the same mode of one or two decades ago, today's computers are no longer a set of huge boxes locked in an airconditioned room. With the advent of internetworking of computers, most computers are reachable from outside the machine room; generally a computer is reachable from remote locations. One may illustrate the following possibility to students: with a built-in modem, a personal computer can access numerous computers around the world by dialing a

[2] iPSC/2 is a trademark of Intel Corporation.

[3] Defense Advanced Research Project Agency (DARPA) has funded internetworking research. The DARPA technology includes a set of network standards and conventions of how computers communicate and networks interconnect. This technology is commonly named TCP/IP internet. TCP (Transmission Control Protocol) and IP (Internet Protocol) are the two main protocols among many protocols in the internet.

local number. For engineering students, the hardware aspects that make the above scenario possible have to be sorted out. Having motivated the importance of computer security, an instructor may cover the subject in different depth and breadth. In a regular course on operating systems computer security may take from one to two hours of lectures. A list of possible subjects related to computer security are: password security, data encryption, digital signatures, privacy-enhanced electronic mail, and network file systems [10]. The first three items in the list are basics.

V. Evaluation

Evaluation through tests in a class is a two-way measure and should be treated differently from the qualifying examinations. Except for the final examination, results of midterm tests can be used for modifying the lecture style. Different levels of difficulty and complete coverage are two important factors for measuring the capability and knowledge of students. The grades should conform to the standard set in the course syllabus. 'Difficulty' is a relative term.

Easy problems are those asking for facts or simple illustrations. Answers can be found either directly from the textbook or course notes with minimum derivation. Explaining technical terms and computing basic measures are thought of as easy problems. Average problems are those asking for some logical reasoning as opposed to simple memorization. Answers can be found indirectly from textbook or course notes with some computation. Problems that may fail about ten percent of students, on the average, are thought of as average problems. Difficult problems are those requiring sequences of logical reasoning or a series of computations. Answers usually cannot be found from the textbook or course notes. 'Difficult' sometimes means that the success rate is relatively lower and does not merely imply that the idea is difficult to grasp. As an engineer, one is expected to perform a series of correct computations. Problems that only about ten percent of students can correctly answer, on the average, are thought of as difficult problems.

VI. Conclusions

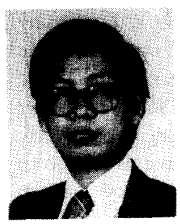
In this paper, a practical course syllabus for operating systems that can be used for electrical engineering students is examined. Methodology that can achieve the instructional goals and possible derivatives from the syllabus are also discussed. Most guidelines discussed in this paper can be used equally well in other curricula.

Acknowledgement

The author would like to thank Prof. M. Cotton for sharing a concurrent class as well as many fruitful discussions. The author is grateful to all former students in the classes; especially P. Nguyen for his assistance in the preparation of this paper.

References

[1] L. Bic and A. Shaw, *The Logic Design of Operating Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
[2] P. Brinch Hansen, *Operating System Principles,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1973.
[3] D. E. Comer, *Operating System Design–The XINU Approach.* Englewood Cliffs, NJ: Prentice-Hall, 1984.
[4] H. M. Deitel, *An Introduction to Operating Systems,* 2nd ed. Reading, MA: Addison-Wesley, 1990.
[5] A. N. Habermann, *Introduction to Operating System Design.* Chicago: Science Research Associates, 1976.
[6] Intel Corporation, iPSC/2 User's guide, Order Number 311532-004, Beaverton, OR, Oct. 1989.
[7] A. Kelly and I. Pohl, *A Book on C,* 2nd ed. Reading, MA: Addison-Wesley, 1991.
[8] Network Working Group, RFC1057, RPC: Remote Procedure Call Protocol Specification, vol. 2, June 1988.
[9] Network Working Group, RFC 1094, NFS: Network File Systems Protocol Specification, Mar. 1989.
[10] C. P. Pfleeger, *Security in Computing.* Englewood Cliffs, NJ: Prentice-Hall, 1989.
[11] A. Silberschatz, J. Peterson, and P. Galvin, *Operating System Concepts.* 3rd ed. Reading, MA: Addison-Wesley, 1991.
[12] A. S. Tanenbaum, *Operating Systems Design and Implementation.* Englewood Cliffs, NJ: Prentice-Hall, 1987.



Chyan Yang (S'86–M'87–SM'90) received the Master's degree in information and computer science from Georgia Institute of Technology, Atlanta, GA, and the Ph.D. degree in computer science from the University of Seattle, WA.

He was an Assistant Professor of Electrical and Computer Engineering at the US Naval Postgraduate School from 1987 thru 1992. He is currently an Associate Professor of College Management, at National Chiao Tung University where he carries out research in computer networks and distributed systems. He is a member of ACM.