FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 18, No. 2, April 2005, 309-317

Advanced System Software Curricula

S. Đorđević -Kajan, Dragan Stojanović, Aleksandar Stanimirović

Abstract: An advanced System Software curricula at the Faculty of Electronic Engineering in Niš is presented in this paper. The system software track consists of two important themes of Computer Science and Computing in General organized now as two separated courses: Operating Systems course and System Software Development and System Programming course. Both courses offer extensive teaching of foundational concepts and principles of Operating Systems and System Programming along with design and implementation of presented topics in real operating systems and system software, such as Unix, Linux and Windows 2000/XP. Laboratory environments and exercises for both courses offer both examination of main algorithms and structures within operating systems and system software through simulation, and what is more important, hands-on experience with operating system internals and code.

Keywords: System software, Operating Systems, System programming, Laboratory exercises

1 Introduction

Operating systems and system software in general are an essential part of any computer system. Similarly, a course on operating systems and system software is an essential part of any computer science and computing education. This field is undergoing change at a breathtakingly rapid rate, as computers are now prevalent in virtually every application, from games for children through the most sophisticated planning tolls for governments and multinational firms. All these applications rest on services and foundational concepts of appropriate operating systems and system software, because the design of an operating system and associated system software exerts a major influence on the overall function and performance of the entire computer, as well as its applications.

Manuscript received February 28, 2004
The author are with Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia and Montenegro, E-mail: sdjordjevic@elfak.ni.ac.yu

The concept of operating system as primarily providing its users with a convenient interface is a top-down view and belongs to ordinary users. An alternative, bottom-up view, holds that operating system is there to manage all the pieces of a complex system including processors, memories, timers, disks, mice, network interfaces, printers and a wide variety of other devices and belongs to operating system designers and system programmers. For them operating system provide an orderly and controlled allocation of the processors, memories, I/O devices and other logical and physical resources among the various programs competing for them. The system software beyond an operating system provides services for program development and execution, distributed and network computing, virtual machines for programming abstractions, database management, an so on. Thus, system software development and system programming are the important activity that enables proper design, implementation and performance of all other application software developed on top of system software in general. The System software course, covering both operating systems and system programming, is intended for such audience providing an up-to-date overview and practical information on understanding and designing modern operating systems, blending a theoretical foundation of operating systems design with real, contemporary implementations. It explains key mechanisms of modern operating systems, types of design tradeoffs and decisions involved in system design, and the context within which the operating system functions, as well as principles and design of system software residing on top of operating system.

2 System Software Track at the Faculty of Electronic Engineering

The System Software theme is currently offered as unique two-semester course at the Faculty of Electronic Engineering, University of Niš, at 7^{th} and 8^{th} semester of the undergraduate studies. It consists of two important Computer Science topics, Operating Systems and System Programming, which are taught in each semester. The Faculty of Electronic Engineering started new Computing Curricula in 2004 according to Bologna Declaration and Process and to the ACM/IEEE work on Computing Curricula specification and recommendations [1]. In the new Curricula the System Software course is logically split in two courses making a System Software track. The first one is traditional Operating Systems course, taught in 4^{th} semester mandatory for all Computing undergraduate programme: Computer Science, Computer Engineering, Software Engineering, Information Systems and Information Technologies. The second part of current System Software course is extended and formed as System Software Development and System Programming Course scheduled for 7^{th} and 8^{th} semester of the new Computing Curricula. The

course is one of the core courses for Computer Engineering and Software Engineering programmes and elective for other programmes.

3 Teaching System Software Principles and Concepts

The first part of the System Software track, the Operating Systems course, follows traditional sequence of topics regarding modern operating systems principles and concepts [2], [3], [4]. Introductory lecture topic represents so-called bird's-eye view of the operating system. It reviews the history of operating system starting from the early batch systems to modern multiprogramming systems and personal computer systems. Since operating systems interact closely with the computer hardware, the topic reviews shortly computer hardware components and their interconnections. The basic components on which all operating systems are built: processes, memory management, I/O management, the file system and security are introduced leaving to subsequent chapters to describe them thoroughly. The concept of system calls as the programmer interface to an operating system, as well as design principles of operating system as monolithic, layered, virtual machine, exokernels and client-server system are presented. The next lecture topics present the most central concept in any operating system: the process as an abstraction of the running program. It introduces the concept of multiple threads of control within a single process, which are scheduled independently, each one having its own stack but sharing a common address space within the process they belong to. The topic discusses process and thread properties and how they communicate with one another using interprocess primitives, such as semaphores, monitors or messages. It also gives a number of detailed examples of how interprocess communication works and how to avoid some of the pitfalls. The topic is concluded by the review of scheduling algorithms. Next follows the topic, which introduces deadlocks as potential problems in any operating system. It briefly shows what deadlocks are and discusses the ways to prevent or avoid them. The obligatory topic in this sequence is the memory management, which is presented in detail. The important topic of virtual memory is examined along with closely related concepts such as paging and segmentation. The next topic provides introduction to the principles and techniques used within Input/Output subsystem. The tree ways in which I/O can be accomplished: programmed I/O, interrupt driven I/O and DMA are discussed. The four level structure of I/O: interrupt service procedures, the device drivers, the device-independent I/O software, and the I/O libraries are described. Several important devices, including disks, clocks, keyboards, displays and network terminals are used as examples. Within the next topic, detailed overview of both the file system interface and the file system implementation. The lecture offers a basic view

to a file system through operations on collection of files and directories, along with operations of them, but more thoroughly from inside, concerning how storage is allocated, how system keeps track of files and containing blocks, how directory structure is maintained, how free disk space is managed, and so on.

Mentioned themes present completed study of the basic principles of single-CPU operating systems, which any course on operating systems should cover. While we recently introduced an elective course in computer security, we could not guarantee that our computer science graduates would be exposed to a formal treatment of even the most basic computer security topics. Since computer security is a hugely important subject nowadays, we added some fundamental computer security concepts and activities to our Operating Systems course. Also, The Computing Curricula 2001 Computer Science model curriculum [1] lists security as an elective component of the Operating System course. Among the topics discussed in this lecture are threats (e.g. Trojan Horses, viruses, worms, Denial of Service, etc.), protection mechanisms and security models.

The most valuable component of the Operating Systems course are the case studies of real operating systems, and design and implementation of presented operating systems components and functions. The Operating Systems chosen are UNIX/Linux [5] and MS Windows, because of their widespread usage and popularity. The case studies present how operating system principles and concepts covered by previous chapters are applied in the real world. Due to high diversity of UNIX clones and versions, the fundamental principles, system calls, general implementation strategies, algorithms and data structures are primarily drawn upon examples of 4.4BSD (FreeBSD), System VR4 and Linux implementations [6]. The design principles of Windows 2000 and Windows XP are also given.

The System Software Development and System Programming course of the System software track represents an introduction to the design and implementation of various types of system software, beyond the operating system [7]. A central theme of the course is the relationship between machine architecture and systems software. System software consists of software programs that support the operation of a computer. A variety of system software programs (assemblers, macro processors, loaders and linkers, interactive debugging systems, virtual machines, distributed object and file systems, middleware, database management systems, embedded software systems, etc.) are covered in the course and some software engineering concepts and issues related to system software development are also introduced. The course presents the fundamental concepts and basic design of each type of system software in a machine-independent way. Both machine-dependent and independent extensions to the basic concepts are discussed, and examples of the actual system software are given.

4 Laboratory Environment and Exercises for System Software Courses

The purpose of the lab component of the Operating System course is to reinforce topics covered in lecture and to give students hands-on experience with design of operating systems. The lecture and lab are complimentary. Lecture material typically covers the theory behind some concept, while the lab addresses the same material from a pragmatic standpoint. Our basic idea was to give students opportunity to study and implement real operating system code, which is fundamental for this kind of course.

In order to provide students a better understanding of operating system internals we have considered several solutions. First we have considered typical approach where student projects simulate operating systems internals using system programming. In this way they get experience with system programming while learning about operating system concepts, but they don't get opportunity to get "hand-on" experience with operating system internals. Also, we have considered, using an operating system simulation environment such as Nachos [8]. In this way students get opportunity to implement operating system internals. But this kind of environment adds an extra layer of complexity and often developers have to develop their often design tools. This approach also have to deal with fact that in a real life students have to deal with real operating systems not with simulations.

We believe that "hands-on" experience is an essential component of computer science education and that most current curricula rely far too heavily on simulation when teaching systems issues. In order to provide students with access to real operating system code we have choose solution based on open source operating systems. Operating systems with open source code, such as Linux, are not only available but becoming serious competitors to commercial products in the market. For this reason we have choose Linux as a operating system for our lab exercises. We chose to use Linux for several reasons [9]:

- wide range of hardware supported,
- the sophistication of its kernel,
- availability of source code [10] and
- rich documentation.

In order to achieve our goals appropriate laboratory is needed for Operating System lab exercises. Faculty of Electrical Engineering, with help from TEMPUS Project CD-JEP 16160/2001, has built a computer laboratory. This laboratory has 15 workplaces. Each workplace is equiped with 1.7MHz Pentium Celeron IV PC machines running the Windows XP operating system. Each machine has 256MB of

RAM, 40GB hard disk, 3.5" floppy, and a DVD ROM drive disk. This laboratory is available to all CS students. It is not dedicated Operating Systems lab.

Operating System course must provide students with laboratory allowing them unrestricted access to the machines, but in a safe manner both for students in the class and for students outside the class. In order for students to modify the operating system code, they need super-user permission. In this way, each machine becomes insecure because any file, on their file system, can be compromised, both in terms of privacy and integrity. This means that open source project machines cannot be easily time-shared because they can be unstable and not private. There must be a quick way to selectively repair or re-install the software on a machine in the event that a system is compromised. Moreover, a super-user can infiltrate the network, using his time in the lab to compromise other systems. This requires additional protection in order to provide productive but protected environment.

In order to provide this kind on environment (without dedicated Operating Systems lab) we have experimented with virtual machines (VirtualPC [11], Vmware [12], BOCHS [13]). We have installed VMware Workstation in the school computers, running a commercial open source operating system in the virtual machines. The virtual machines run fully isolated from the host operating system (the operating system running on the physical computer), meaning that any errors or crashes in the virtual machines do not affect the physical computer. Students now perform their coursework in the virtual machines, gaining the hands-on experience that allows them to fully test and develop their skills, without posing a risk to the school machines. In this way students have opportunity to explore all aspects of developing an operating system, including modifying, designing, implementing, and testing kernel code. The virtual machines also give students the freedom to make mistakes and learn from them. Using undoable disk capabilities offered by VMware, students can "undo" their errors and restart the session as many times as they need to master the material.

The main goal of lab exercises for System Software Development and System Programming course is to provide students with knowledge of how to use topics, covered in lectures, in realistic situations. In this way students can learn to construct effective software that interact directly with operating system, rather than with higher-level abstract machines (such as database managers, windows systems and high-level file systems). Same time, using system calls directly, students can gain important insight into way how operating system is designed and implemented.

System Software Development and System Programming lectures provide students with concepts and issues. In this way course is concentrated on ideas and technology that transcend any particular operating system. The 1991 (and draft 2001) IEEE/ACM undergraduate course recommendation describes a course that

consists of substantial amount of time spent on issues, but also includes a significant laboratory component. Even though the trend is toward courses based on conceptual materials, hands-on expirience is invaluable in learnin about operating systems and system programming [14]. Lab exercises have to give students opportunity to try concepts, they have learned in lectures, in real life. For a long time Linux was the best real operating system on which to build lab exercises for system programming. It provides a contemporary implementation of Unix, it has a rich documentation and samples available over Internet and source code is also publicly available [10], [15]. On the other hand Windows operating systems (Windows 2000, Windows XP) dominate commercial market of operating systems. Literally every PC machine has a copy of some Windows installed on it. For that reason students are more familiar with Windows environment [16], [17]. For this reasons we have decide to support both, Linux and Windows, in our laboratory exercises.

In order to help students during preparation of their lab exercises we have published a lab manual. Lab manual "System software – Operating systems and system programming" [18] is fully supported by Tempus project CD JEP 16160/2001 that is financed by European Commission. This lab manual contains all necessary material that students need in order to prepare their lab exercises. This material is written in order to complement with System Software Development and System Programming course lectures. It contains definitions of most important system calls and data structures provided by operating system kernel (both Unix/Linux and Windows) that are necessary for system programming and for managing components of operating system. Source code, that illustrates this system calls and data structures, is also provided as a part of this lab manual.

For lab exercises we are using the same laboratory used for lab exercises in Operating System course. Although we treat Unix/Linux and Windows as equal operating system during lab exercises we slightly favor Linux. There are several reasons for this. First, most of our CS students have not used any operating system other than Windows before. Using Linux in our class gives students experience with another major operating system. Second, although development GUIs exists for Linux, we chose to use the command-line interface. Using the command-line reinforces the ideas behind topics, rather than teaching students how to use a particular GUI. A non-GUI approach is not feasible in an easy way in Windows. Third, Linux is easily customized to provide a secure environment in which students can work.

Our System Software Development and System Programming course have 7 projects. In the projects the students were required to study the current operating system and his system calls, design and implement a solution, evaluate the performance of their solution and answer some short questions regarding extensions to their project. The first project was designed to get students familiar with the Linux

system. It included a series of “cook-book” type instructions that walked students through the addition of user accounts, the use of some common Unix tools such as find and grep. Second project is designed in order to get students familiar with internal parameters of current operating systems and to use some tools to access those parameters. Other projects prepare students to deal with projects regarding processes, synchronization, deadlock, security, virtual memory and file management.

5 Conclusions

Providing the great importance of Operating Systems and System Programming topics for undergraduate Computer Science and Computing in general education, we organize those topics in advanced System Software track, comprising two courses, at the Faculty of Electronic Engineering in Niš, for Computing and Informatics new curricula programme. Those courses blend up-to-date theory with broad coverage of fundamentals, offering a comprehensive treatment of operating systems and system programming, with an emphasis on internals and design issues of both operating systems and system software. The Operating Systems course and System Software Development and System Programming courses provide a thorough discussion of the fundamentals of operating systems and system software design and relates these principles to contemporary design issues and to current trends in the development of operating systems and system software. It helps students develop a solid understanding of the key structures and mechanisms of operating systems and system software beyond them, the types of trade-offs and decisions involved in operating system and system software design, and the context within which the operating system functions (hardware, other system programs, application programs, interactive users, etc).

Acknowledgement

The work on this paper is partially supported by the Tempus project CD JEP 16160/2001 financed by European Commission.

References

[1] “Computing curricula 2001: Computer science (final report),” in *The Joint Task Force on Computing Curricula*. IEEE Computer Society, Association for Computing Machinery.

[2] G. Nutt, *Operating Systems: A Modern Perspective*, 2nd ed. Addison Wesley, 2000.

[3] J. L. Peterson and A. Silberschatz, *Operating System Concepts*, 6th ed. John Wiley & Sons Inc, 2002.

[4] A. S. Tanenabum, *Modern Operating Systems*, 2nd ed. Prentice Hall, 2001.

[5] S. Đorđević-Kajan and E. Kajan and I. Popović, *UNIX sistemski prilaz*. Niš: Elektronski fakultet, 1996.

[6] (2004) The single unix specification version 3. Open Group Publications. [Online]. Available: http://www.unix.org/single_unix_specification/

[7] L. Beck, *System Software: An Introduction to Systems Programming*, 3rd ed. Addison-Wesley, 1997.

[8] (2005) Nachos. [Online]. Available: http://www.cs.washington.edu /homes /tom /nachos/

[9] R. Chapman and W. Carlisle. (2005, April) A linux-based lab for operating systems and network courses. [Online]. Available: http://www.linuxjournal.com/

[10] (2005) Linux source code browser. [Online]. Available: http://lxr.linux.no/source

[11] (2005) Microsoft virtual pc. [Online]. Available: http://www.microsoft.com/windows/virtualpc/default.mspx

[12] (2005) Vmware. [Online]. Available: http://www.vmware.com/

[13] (2005) Bochs ia 32 emulator project. [Online]. Available: http://bochs.sourceforge.net/

[14] G. Nutt, *Kernel Projects for Linux*. Addison-Wesley, 2000.

[15] W. R. Stevens, *Advanced Programming in the UNIX Environment*. Addison-Wesley Publishing Company, 1992.

[16] J. M. Hart, *Win32 System Programming: A Windows(R) 2000 Application Developer's Guide*, 2nd ed. Addison-Wesley Professional, 2000.

[17] G. Nutt, *Operating System Projects Using Windows NT*. Addison Wesley, 1999.

[18] S. Đorđević-Kajan and D. Stojanović and A. Stanimirović and B. Predić, *Laboratory Manual for System Software Course: Operating Systems and System Programming*. Niš, Serbia and Montenegro: Faculty of Electornic Engineering, University of Niš, 2004.