2008 14th IEEE International Conference on Parallel and Distributed Systems

A Practical Implementation Course of Operating Systems: Curriculum Design and Teaching Experiences

Shiao-Li Tsao
Department of Computer Science,
National Chiao Tung University, Hsinchu Taiwan
sltsao@cs.nctu.edu.tw

Abstract

The embedded software engineers are highly demanded in recent several years in order to support fast development of SoCs and embedded systems. These engineers need both strong hardware/software knowledge and hands-on experiences of system-level software. Unfortunately, the practical training of the system software development such as the OS design and implementation is often insufficient for computer science students in Taiwan recently. To minimize the gap between theory and practical implementation, a practical implementation course of OSs is thus developed in National Chiao Tung University. The course provides students trainings on the design details of modern OSs and comprehensive hands-on practices on OS implementation. In this paper, the curriculum and hands-on lab design, the teaching experiences and student feedbacks for the trial run are presented.

1. Introduction

The development of SoCs and embedded systems grows rapidly in recent several years. Besides the embedded hardware designers, the demands for embedded software designers, especially system-level software designers, are largely increasing. Different from application software designers, the embedded software designers require strong hardware and system software knowledge. Moreover, the embedded software engineers, especially system-software engineers, need professional skills in implementation. Unfortunately, the practical training of system software development such as the OS design and implementation is often insufficient for computer science students in Taiwan recently. To design a course for introducing practical OS implementation and training students with system-programming skills is quite essential.

Operating system is one of the fundamental courses for computer science students. For example, there are 12 courses related to operating systems in the department of computer science of National Chiao Tung University [1]. Figure 1 illustrates curricula for OS or embedded software related courses. The basic, intermediate, and advanced courses are given to the 2nd, 3rd, and 4th years undergraduates. The detail description of each course can refer [2]. In order to minimize the gap between theory and practical implementation, the design and implementation of OSs course is thus developed in our university. The course aims to give students the design and implementation details of modern OSs such as Linux and Microsoft Windows. The course especially emphasizes the hands-on practices. Based on the course design, students are asked to implement their own operating systems step by step and from scratch based on an X86 virtual machine. The students are expected to understand the practical aspect of an OS, and have comprehensive exercises on OS implementation. This paper presents the course and hands-on lab designs, and the teaching experiences and student feedbacks for the trial run.

The rest of the paper is organized as follows. Section 2 presents the curriculum design. Section 3 describes the hands-on lab design which helps students to learn OS development step by step. The teaching experiences and student feedbacks are discussed in Section 4 and finally we conclude this paper in Section 5.

2. Curriculum Design

The course is for graduated students and senior undergraduates. The objectives of the course are (a) to introduce the practical design and implementation of modern OSs, (b) to introduce research topics such as benchmarking, optimization, real-time characteristics, etc. for modern OSs, and (c) to provide comprehensive hands-on training to students to improve their system-level programming skills. The prerequisite of the course is the Introduction to OS. The course includes

1521-9097/08 $25.00 © 2008 IEEE
DOI 10.1109/ICPADS.2008.99

three hours lecture per week, five take-home hands-on labs, and one term project.

The courseware has to be newly designed since the course shall cover the theoretical part of the OS design, and also has to give students examples or sources code traces which implement a specific module of an OS. Moreover, we would like to give students side by side comparisons between different modern OSs such as Linux and Microsoft Windows Research Kernel [3]. We use process management of an OS as an example. We first review the how and why we need the process management and how to design it. Theoretical overview based on the OS textbook such as [4] is summarized and presented first. Then, we present how Linux process management is designed [5]. Besides presenting the lecture slides, we frequently switch to the source codes of the Linux to demonstrate its implementation. Microsoft Windows is also used as the case study [6]. Source codes of the Windows Research Kernel are also illustrated for cross references of the Windows design. Figure 2, Figure 3 and Figure 4 give examples of some slides presenting the general process management design, Linux process management design and Windows process management design.

Students are also required to study research papers from OSDI etc. [7] and we also encourage students to think and improve the design of an OS. The course takes 17 weeks, and the syllabus and labs plan are shown in Figure 5.

During the lecture, we spend less than 10% time in reviewing the theoretical design of the OS. About 50% time is used to present Linux case study and its source codes. About 30% time is used to discuss Windows case study and its source codes. Also, we have about 10% time for open discussions, mainly on why and how to improve the components of an OS.

3. Hands-on Lab Design

The hands-on labs are step-by-step and closely related to the lectures. Each lab has a number of incremental requirements. Students have to complete mandatory requirements and could continue working on bonus requirements which are optional. There are five hands-on labs and one term project during the entire 17 weeks. The homework is given to students after the subjects are presented. Students are encouraged to discuss to each other, refer open sources, Linux kernel sources, and Windows Research Kernel sources, but they are asked to implement the OS code by themselves. They can refer all kind of open sources but the cut and paste from the open sources or from other students are prohibited.

The development environment for the hands-on labs is PC/IA-32. Several virtual machine platforms are recommended. They are Bochs [8], Virtual PC [9], VMWare [10], and ProjectOZ [11]. The programming languages are Assembly and C. Each student has to upload his/her report, kernel sources and image to the FTP server before lab demo deadline. TAs will run their kernel before the demo. Students have to demo their kernel to TAs and answer the TAs' questions.

The five labs are

- Lab #1: Bootloader
  The mandatory requirement of this lab is that students have to write a simple bootloader for X86. The bootloader has to boot the PC and run another program displaying "hello world" message on the screen using BIOS services. The bonus requirement is to support multi-boot which could boot DOS OS, Linux, and Windows XP kernel images.
- Lab #2: Task, multi-tasking and task scheduler
  Based on Lab 1, the mandatory requirement of this lab is to allow a PC/X86 running several "hello world" applications at the same time. The cooperative multi-tasking and process management services are implemented and applications can call the services to switch the tasks. The bonus requirement is to implement time-slice based multi-tasking.
- Lab #3: Memory management
  Based on lab 2, students have to implement memory-related system services such as **malloc** and **free**. The program has to manage free/allocated memory spaces. The bonus requirement is that students can switch the PC/X86 to the protected mode so that the paged memory management can be used.
- Lab #4: Interrupt and ISR
  Students are asked to modify interrupt vectors and write their own interrupt services routines. The bonus requirement is to handle exceptions.
- Lab #5: Device driver
  Based on the lab 4, students are asked to implement a simple keyboard driver and related OS services. Applications can use the system services to get keyboard inputs. The BIOS service is no longer used at this stage. The bonus requirement is implement a simple monitor driver.

For the term project, students have to submit a two-page proposal first. After the proposal is approved, they could work on their term project. The term project

must be based on their own OSs which they develop during this course. The term project could be improving the stability of their OSs, adding a device driver, adding OS modules etc. In the trial run, we received 83 proposals, and we categorize and summarize them in Table 1.

Table 1. Summary of the term project proposals

Category	Number of proposals
Add a file system such as FAT	33
Device drivers such as NIC, mouse, VGA	22
Kernel enhancement such as IPC, preemptive, system call, semaphore	15
Middleware such as GUI APIs	8
Protocol stacks such as TCP/IP	5

4. Teaching Experiences and Students Feedbacks

There were more than 200 students who pre-registered the course for the trial run. After we overviewed the course and labs, 120 students finally decided to enroll the course. About 40 students dropped the course before the deadline of the first lab, i.e. the bootloader. There were finally 83 students including 8 undergraduates, 3 PhD students and 72 master students enrolling the course. The student evaluation demonstrates the students satisfy the course design and training. The course got an overall score of 4.33 (1-5(highly satisfied)). We received a number of positive feedbacks from students. First, they enjoy the courses and feel interested to know practical aspect of the OS and know details of OS design and implementation. Most of the students feel excited to know Linux and Windows Internal and compare the implementation of Windows and Linux. Many students said that their system programming skills improve significantly. We also had some negative comments. For example, some students said the hands-on labs are too difficult and too heavy. Moreover, labs are strongly related and some students felt frustrated for the new assignment if they could not be able to complete previous one.

We also had an internal review with TAs and other professors on the course after the trial run. We have a couple of observations. First, we found that the course heavily rely on good supporting tools and teaching materials. The courseware should be continually improved and enhanced. We setup a student discussion forum for this course and found it is very useful for student discussion. The posts could be used for FAQs or accumulating experiences for references. We had more than 400 posts for the course during the trial [12]. TA loading is quite heavy and they have to be very experienced in system programming. Capable TA team is one of the most important factors for the success of the course. Dividing the requirements of a lab into incremental phases is a good approach. Also, it is suggested to announce good examples after each lab so that students who did not write the homework well can learn from others and their examples.

5. Conclusions

In this paper, we presented the curriculum and hands-on lab design for a practical OS implementation course. Also, the teaching experiences and student feedbacks were presented. The experiences show students are interested for such a practical course and learn a lot although the hands-on labs are difficult and heavy. With the comprehensive training on both theory and practical implementation, we could have capable embedded software engineers to support the development of SoCs and embedded systems.

References

[1]Department of Computer Science, National Chiao Tung University, http://www.cs.nctu.edu.tw/en/about_cs/index.php

[2]Shiao-Li Tsao, T. Y. Huang, and C. T. King, "The Development and Deployment of Embedded Software Curricula in Taiwan," ACM SIGBED Review, Special Interest Group on Embedded Systems, Vol.4, No.1, 2007 Jan.

[3]Microsoft Windows Research Kernel, http://www.microsoft.com/resources/sharedsource/windowsacademic/researchkernelkit.mspx

[4]Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne, "*Operating System Concepts*," Wiley, 2004.

[5]Robert Love, "Linux Kernel Development (2nd Edition)," Novell Press, 2005.

[6] Mark E. Russinovich and David A. Solomon, "*Microsoft Windows Internals*," Microsoft Press, 2005.

[7] Operating Systems Design and Implementation (OSDI), http://www.informatik.uni-trier.de/~ley/db/conf/osdi/

[8] BOCHS project, http://bochs.sourceforge.net/

[9] Virtual PC, http://www.microsoft.com/windows/downloads/virtualpc/default.mspx

[10] VMWare, http://www.vmware.com/

[11] ProjectOZ, http://www.microsoft.com/resources/sharedsource/windowsacademic/projectoz.mspx

[12] http://brass.cs.nctu.edu.tw/forum/viewforum.php?f=31

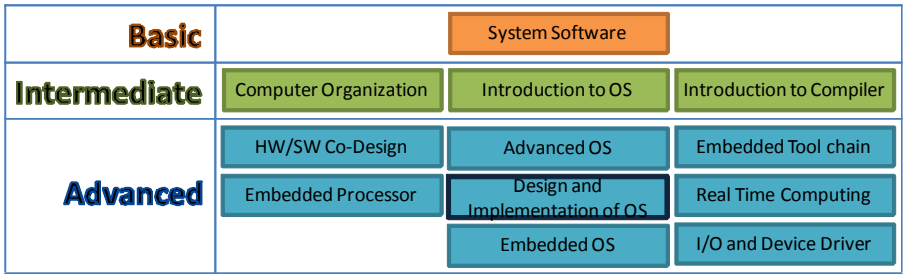

Figure 1. OS related courses in CS of NCTU

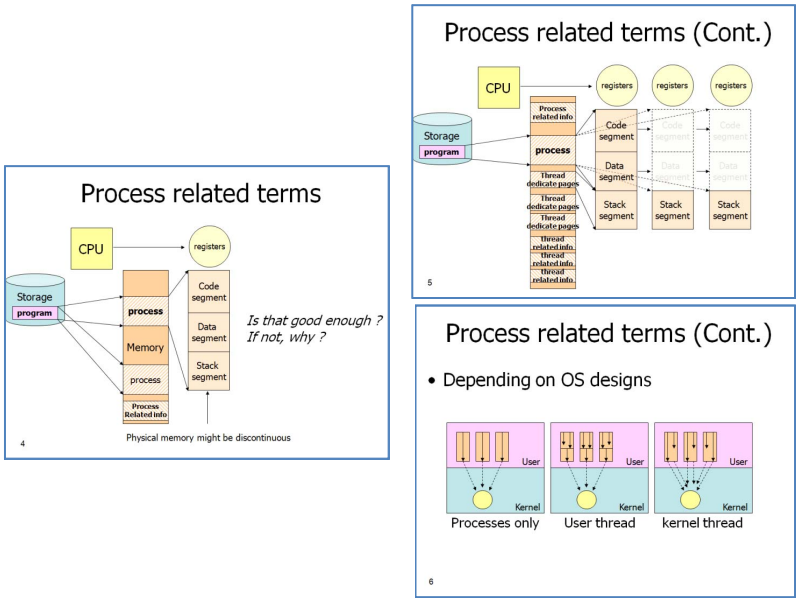

Figure 2. Slide examples presenting general idea of the process management design

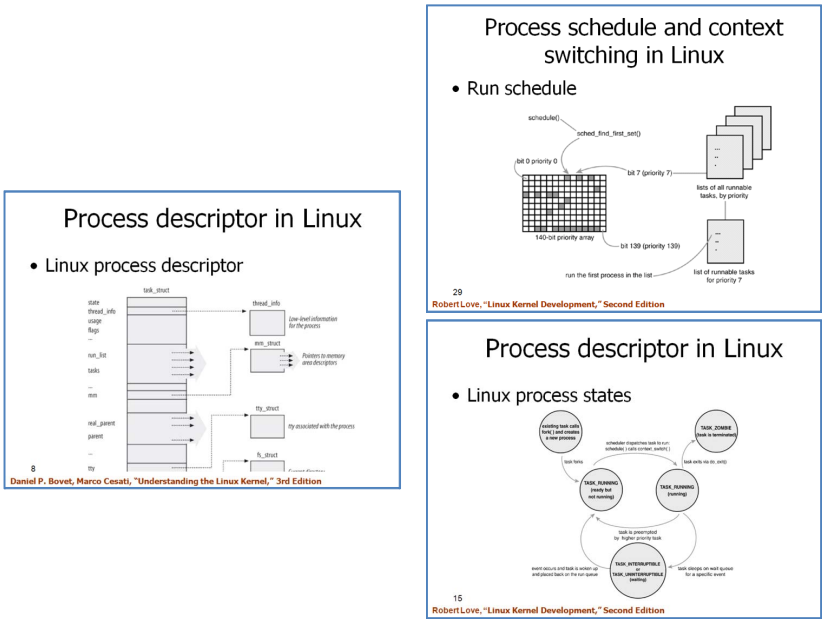

Figure 3. Slide examples presenting Linux process management design [5]

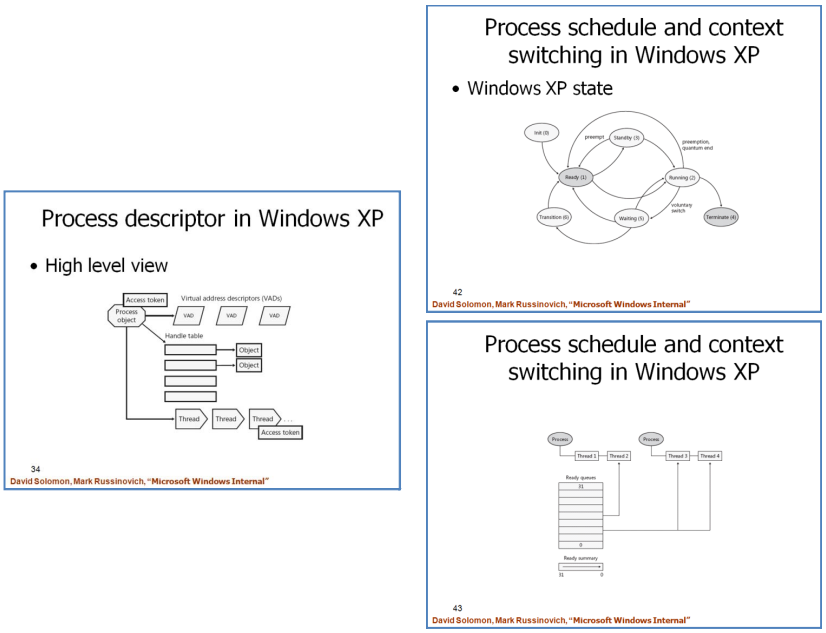

Figure 4. Slide examples presenting Windows process management design [6]

Week	Agenda	Hands-on homework	Homework demo
Week 1	Opening and Course Briefing		
Week 2	80x86 Architecture	Lab1: Bootloader	
Week 3	OS Briefing		
Week 4	Bootloader		
Week 5	Process/Schedule	Lab2: Process/Schedule	Lab1: Demo
Week 6	Process/Schedule		
Week 7	Memory management		
Week 8	Memory management	Lab3: Memory	Lab2: Demo
Week 9	Synchronization and IPC		
Week 10	Timers/Exceptions/Interrupts		
Week 11	Timers/Exceptions/Interrupts	Lab4: Interrupt	Lab3: Demo
Week 12	Device Driver		
Week 13	System Calls/Call Gates		
Week 14	I/O subsystem and File Systems	Lab5: Device driver	Lab4: Demo
Week 15	Network Protocol Stacks		
Week 16	Paper Presentation		
Week 17	Paper Presentation		Lab 5: Demo

Figure 5. Syllabus of the design and implementation of OS course