Novel Approaches to Teach and Learn Courses on Computer Operating Systems

Pinaki Chakraborty, P. C. Saxena

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi – 110067, India.
E-mail: pinaki_chakraborty_163@yahoo.com

Abstract

The courses on computer operating systems are indispensable in any curriculum in Computer Science. However, it has been observed that it is difficult to competently teach these courses using the conventional ways. So, some customized approaches are required to be designed for these courses. The current paper contends the use of pedagogical operating systems, models, simulations and performance measurement techniques to teach and learn the courses on computer operating systems.

Keywords: *Computer operating system, pedagogical operating system, model, simulation, performance measurement*

1. Introduction

A computer operating system is a piece of software that manages all the resources of a computer system, both hardware and software, and provides an environment in which the users can execute their programs in a convenient and efficient manner [1, 2]. A sound knowledge of the computer operating systems is a desirable qualification for all information technology professionals and an essential for all computer engineers. However, traditional ways of classroom teaching fails to generate much interest about the courses on operating systems among the students. This phenomenon has been observed around the globe and over the years. Students often complain of having difficulty in apprehending the courses on operating systems. And when these students graduate as professionals, they find it difficult to obtain a comprehensive and logical view of the operating systems and fail to optimally utilize the available computer systems.

Perhaps the best way to solve this longstanding problem is to take support from the laboratory. However, it is not easy to arrange a suitable programming laboratory for the courses on operating systems. The key problem with the operating systems is that even the small and simple ones are too large and too complex [3]. So, the students cannot be asked to develop an operating system as an exercise. Although it is feasible to ask students to design and implement a small piece of an operating system, like a CPU scheduler, it has been observed over the years that nobody learns much about operating systems in this approach. Alternatively, the concepts of API programming or shell programming can be introduced to the students in the laboratory but that also does not help very much in understanding the basic structure and the behavior of the operating systems. Nevertheless, some unconventional approaches have evolved over the years to study operating system. Four such approaches have been discussed in Sections 2 to 5.

2. Pedagogical Operating Systems

It has been observed that the students learn by experimenting and not merely by listening [3]. If the source code of a simple and well documented operating system is available to the students then they can study the structure and the internal working of the operating system. Some of the students can even go ahead to modify some parts of the given operating system. This approach improves the understanding of the subject among the students [4]. As a result, quite a few pedagogical operating systems, also known as instructional operating systems, have been

developed. The characteristic features that make these operating systems pedagogical are listed below.

1. The source code of the operating system, in full, should be available to the users [5]. Since the source code of an operating system typically spans several thousand lines, it should be properly modularized. Moreover, the source code should be readable and for this purpose appropriate comments should be used in the source code wherever necessary.
2. The structure and the functioning of the operating systems must be thoroughly documented and published.
3. As one of the requisites of being pedagogical, the operating system should have an option for a verbose mode of operation [6]. When used in this mode, the operating system should explain the internal working of the entire computer system to the users in a stepwise manner.
4. It is often desirable to have tools to compile, link and debug the source code available along with the operating system. Such tools allow the learners to experiment with the source code and rebuild the operating system.

3. Models of Operating Systems

The use of models to teach and learn intricate concepts is common in various subjects. The use of formal models has been quite successful in several disciplines of Computer Science too [7]. Similarly, formal and semiformal models can be also developed for operating systems. Such a model should broadly describe the various components in an operating system, their internal functioning, and the interactions between them. It should explicate how an operating system forms the control mechanism of the entire computer system. Such a model must illustrate how an operating system acts as a resource allocator for the system, a control program for the users and an extended machine for the system developers. Such models should be able to integrate the different principles and concepts used in operating systems, negotiate various design considerations, and also lay down some basic guidelines for implementation. There are some features, as follows, that are essential in a good model of computer operating systems.

1. A model of operating systems must illustrate the controlling and the management of the entire computer system by the operating system.
2. A model of operating systems should be independent of the hardware architecture of the computer system and preferably cover all peripheral devices.
3. A model of operating systems should encompass all operating systems without any prejudice for their designs and sizes.
4. It is desirable to have a model that can be extended to cover multiprocessor computer systems.

4. Performance Simulation of Operating Systems

A computer system comprises of several types of resources including one or more processors and peripheral devices. An operating system uses some scheduling algorithm or the other to appropriately distribute these resources among various competing tasks in the computer system. The performances of these scheduling algorithms can be modeled using the probability theory and their functioning can be precisely simulated. Such simulations can be used to study different arrival and service patterns of the tasks, and can prove immensely beneficial in apprehending the relative performances of the scheduling algorithms under various circumstances [8].

5. Performance Evaluation of Operating Systems

At advanced levels, a qualitative study of operating systems is not enough and quantitative analyses are stressed upon. A quantitative study of an operating system requires experimental measurement of the performance of the operating system at the runtime [9]. These measurements are performed by special programs known as profilers. Profilers can be employed for evaluating performance measures related to both timescale and reliability. A typical profiler determines the

time taken to perform each elementary task in the computer system and the reliability associated with it. From these measurements, the relative suitability of the various algorithms and methods used in operating systems can be judged. This approach becomes most evocative when coupled with a precise model of operating system. In such a case, it illustrates a comprehensive statistical model of the entire operating system.

6. Concluding Remarks

Four unconventional and novel approaches to teach and learn courses on computer operating systems have been briefly overviewed in the current paper. The pedagogical operating systems and the models of operating systems can be used to study the courses on operating systems at both elementary and expert levels of learning. Alternatively, simulation and performance measurement based approaches are suitable only at an expert level of study. However, these four approaches should be employed in tandem (Figure 1) for a comprehensive study of computer operating systems at the research levels.

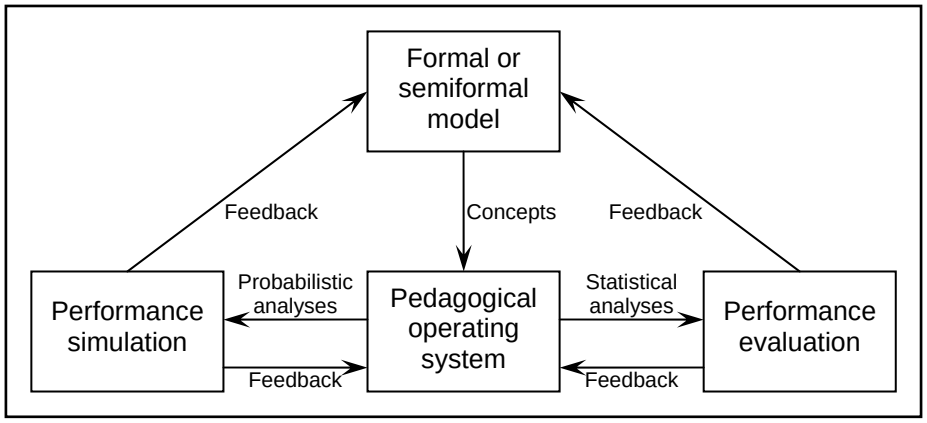

Figure 1: A scheme to facilitate comprehensive study of operating systems.

References

[1] Silberschatz, A., Galvin, P.B. and Gagne, G., *Operating System Principles*. 7th ed., John Wiley & Sons, 2006.

[2] Chakraborty, P. and Gupta, R.G., A structural classification and related design issues of operating systems. *Proceedings of Second National Conference on Methods and Models in Computing*, 2007, pp. 265-273.

[3] Tanenbaum, A.S., A Unix clone with source code for operating systems courses. *ACM SIGOPS Operating Systems Review*, 1987, **21**(1): 20-29.

[4] Chakraborty, P. and Gupta, R.G., The design of a pedagogical operating system. *Proceedings of Second National Conference on Computing for Nation Development*, 2008, pp. 517-527.

[5] Tanenbaum, A.S. and Woodhull, A.S., *Operating Systems Design and Implementation*. 3rd ed., Prentice-Hall, 2006.

[6] Chakraborty, P. 2008. Verbose mode of operation of a pedagogical virtual machine operating system, *Proceedings of Third National Conference on Methods and Models in Computing*, pp. 40-48.

[7] Chakraborty, P., A language for easy and efficient modeling of Turing machines. *Progress in Natural Science*, 2007, **17**(7): 867-871.

[8] Hansen, P.B., *Operating System Principles*. Prentice-Hall, 1973.

[9] Meurs, R., *Building Performance Measurement Tools for the Minix 3 Operating System*. M.Sc. Thesis, Vrije University, Amsterdam, 2006.

Article received: 2009-01-17