A SIMULATOR SUPPORTING LECTURES ON OPERATING SYSTEMS

Luiz Paulo Maia[1] and Ageu C. Pacheco Jr.[2]

Abstract ¾ A common problem faced by teachers and Computer Science students is the difficulty in attaining a proper understanding of the real dynamic nature of the computing events involved. No matter how solid the knowledge and the communication ability of the instructor might be as well as the concentration and close attention paid by the students, the proper understanding of the concepts presented is impaired by the implicit static nature of the class or lecture presentation. In the specific case of operating systems, after many years lecturing on the subject, we started looking for a way to improve the approach in which the concepts and techniques were presented. This paper is the outcome of this research. It implements a simulator (SOsim) with visual facilities to serve as an effective tool for the better teaching & learning of concepts and techniques in modern operating systems, serving as a way to render the whole process more efficient.

Index Terms ¾ computer science education, operating systems, simulation.

INTRODUCTION

There are some disciplines in the Computer Science (CS) curriculum that are more abstract and difficult to teach than others, e.g., teaching programming language techniques which can be implemented in Pascal or C. This is so that what is taught in the classroom can be tested and analyzed using any kind of programming language. In this case, if some concept was not perfectly understood, the student's program will probably not work or will give wrong results. In the case of operating systems (OS), lectures are generally limited to only presenting concepts and mechanisms. On the whole, no implementation is done in the classroom, making classes boring and lectures challenging.

After many years lecturing on OS, we began looking for a way to improve the approach in the teaching of concepts and techniques. Our idea was to implement an OS simulator (SOsim) with visual interface to serve as a tool for better teaching, and also, learning of the concepts and techniques implemented in modern OS, serving as a way to render the whole process more efficient [1].

Educational software is part of a broader effort to improve OS teaching. First, the book "Introduction to Operating Systems Architecture" [2]-[3] was published to give teachers and students a formal introduction to the subject. Now, in its third edition, "Operating Systems Architecture" [4] has been used for more than ten years by many educational institutions around the country. A web site [5] was created to give support in the form of slides, exercises and email, improving the communication amongst the OS community. The simulator is a step further in this effort. It is available on the Internet [6], so that free access to its educational contents is ensured to students and instructors.

The SOsim has been used at the CS department of Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil. There it was able to clearly demonstrate how a simple OS lecture can be improved by showing those aspects demanded in the complete understanding of the subject. In this way, the teaching and learning process becomes far more efficient.

This paper will describe the software and how it can help teachers and students to have a better teaching and learning experience. First, we look at what had been done in this area. Next, the paper examines the pedagogic issues. Then, we present the simulator model and implementation. Next, we show SOsim benefits, an assignment and its result. Finally, concluding remarks and future works are outlined.

PRIOR WORK

There are several papers describing "closed labs" [7] which propose different ways to improve OS classes. They can be divided into three main categories: short projects, source code modification and simulators.

Basically, short projects propose shell interaction and programming using system calls. Programs can include developing inter-process communication (IPC) and synchronization programs, system utilities or some other classic OS algorithm [8]-[11]. Short projects are easy to implement and there is no doubt that they really help students to better understand OS concepts.

Source code modification can be done using instructional OS, e.g. Minix [12] and Xinu [13], or real OS, e.g. FreeBSD and Linux. To take advantage of this kind of lab, students need a very good knowledge of computer architecture, Unix and C/C++ programming. Because of this, code modification is very hard to be implemented. Only a few courses have the time, resources and the skill to try such an endeavor.

On the other hand, simulators like BASI [14], NACHOS [15], OSP [16] and RCOS.java [17] are easier to be handled than source code modification. Nevertheless, the process to learn, install and run such tools frequently takes time and requires some knowledge of Unix and programming.

[1] Luiz Paulo Maia, Federal University of Rio de Janeiro, NCE/UFRJ, PoBox 2.324, Rio de Janeiro, RJ, Brazil, 20001-970 lpmaia@ism.com.br
[2] Ageu C. Pacheco Jr., Federal University of Rio de Janeiro, NCE/UFRJ, PoBox 2.324, Rio de Janeiro, RJ, Brazil, 20001-970 ageu@nce.ufrj.br

LEARNING AND TEACHING CONSIDERATIONS

Operating Systems is an essential discipline in CS education (CSE). It involves high levels of theory, abstraction and design [18]. SOsim tries to integrate all these requisites to help students build a viable mental model [19].

As an educational tool, SOsim should be used as a way to improve pedagogic goals. Operating systems' classes are, in general, based on the theory of Behaviorism, more specifically on Instructionism, where the students passively receive information and knowledge from their instructors [8]. Teachers indicate a book, follow the chapters according to a pre-planned syllabus, and finally students are quizzed. This means that there is little space for questions, independent thought or interaction between the students. Instructionism tends to turn OS classes into extremely abstract ones, compromising the understanding of the concepts.

An alternative is to change the focus from teacher dominated to student-centered using a Constructivism approach. Constructivism is not a new concept, but in CSE, only recently has it been discussed [19]. This theory claims that knowledge is actively constructed by the student, not passively absorved from textbooks and lectures. Students are encouraged to ask their own questions, implement labs, make analogies and come to conclusions on their own. However, introducing Constructivism is not an easy task. Teachers are not prepared for paradigm shifting, and also, there is a lack of tools to help them in this change [21].

A simple way to introduce these ideas in OS classes is by adopting the theory of Constructionism [20]. In Constructionism, through the computer, students build on their own knowledge to develop projects which are presented Prior Work section, i.e. an IPC program, source code modification or build a simulation.

SOsim allows teachers to introduce the ideas of Constructivism in traditional classes. A great advantage of using such a tool is to build a hybrid environment, where Behaviorism and Constructivism can be mixed together. The following topics represent some characteristics of a constructivist tool [22], such as SOsim:

- Engages students in experiences that challenge conceptions of their existing knowledge.
- Encourages the spirit of questioning, discussion among students, and teamwork.
- Allows students to explore new situations and learn from their own mistakes.
- Encourages student autonomy and initiative.
- Doesn't separate knowing from the process of finding out.
- Allows students to learn based on case studies.
- Allows problem identification, definition and solution.

PROPOSED MODEL

SOsim´s main goals are to present the concepts and techniques found in modern OS [23]-[25], such as multiprogramming, process, scheduling and memory management. Some implemented algorithms can be seen in commercial OS, such as HP OpenVMS, Microsoft (MS) Windows NT, Windows 2000 and Windows XP.

Multiprogramming

Multiprogramming is an old concept, but it is still a fundamental technique in OS. Multiprogramming allows computer resources to be shared by many users (process) in a controlled and safe way. SOsim permits to visualize CPU and main memory sharing in a dynamic and animated way.

Process

Process is one of the most important concepts in modern OS. The notion of process is the basis for the understanding of other important concepts, such as scheduling and virtual memory.

The simulator implements processes as Process Control Blocks (PCB), where process information is kept. It is possible to visualize the hardware context, software context and virtual address space (Process Page Table) for each process created. It is not necessary to write a program to create a process. There are different templates that can be used to simplify process creation.

As shown in Figure 1, processes follow the classic three-state-model (ready, running or waiting) and it is possible to notice all process transitions (ready-to-run, run-to-ready, run-to-wait and wait-to-ready). Processes can be suspended, resumed and deleted, like in a real OS.

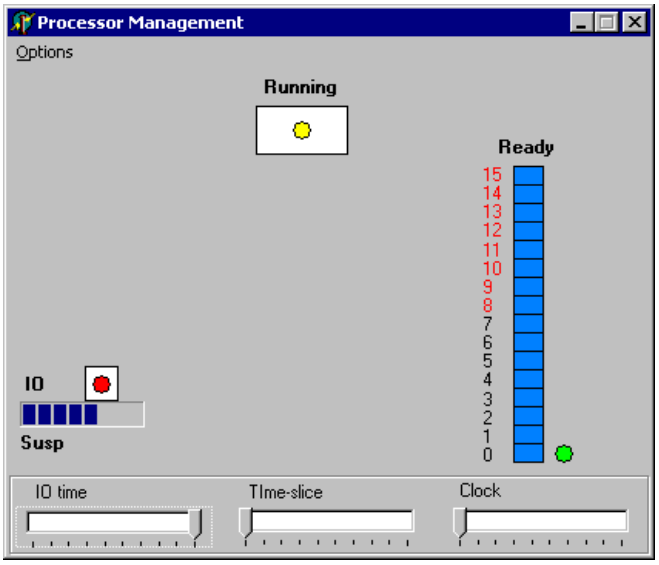

FIGURE. 1
PROCESSOR MANAGEMENT WINDOW.

Scheduling

Scheduling is the task to choose which process will gain the CPU. SOsim implements different scheduling algorithms, starting with preemptive and non-preemptive algorithms. Preemptive scheduling can be done by using time and priority. Also, it is possible to select if the priority scheduling is static or dynamic. Dynamic priority is based on process type (CPU-bound and IO-bound), defined when process is created.

Memory Management

All modern OS´s make use of virtual memory. Virtual memory (VM) is a powerful and sophisticated memory management technique, which expands main memory limits. SOsim implements paging as VM management.

As shown in Figure 2, the main memory is divided in 100 page frames and each process can allocate up to five frames. Processes have their own page table that can be observed while changes take place. Teachers can use the simulator to show and explain several VM details and policies step-by-step, starting from program loading to page replacement.

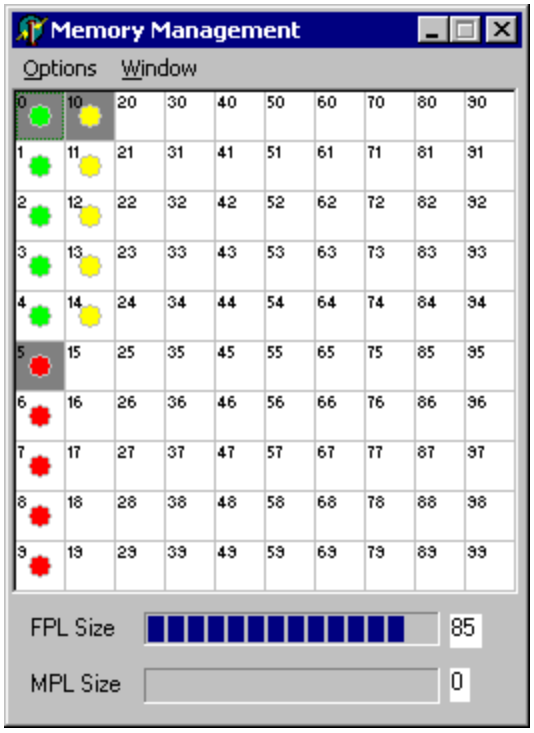

FIGURE. 2
MEMORY MANAGEMENT WINDOW.

It is possible to select from two types of page fetch policy: demand paging or pre-paging. When a process is created, it is possible to define how many pages will be allocated in main memory. Thereafter, the system will keep a static page allocation policy. We chose FIFO with two page buffers (free page list and modified page list) as page replacement policy. If there is not enough main memory space, swapping is started, using a page/swap file.

Additional Features

SOsim offers an on-line log that can be used to follow the simulation steps. The same log is stored on disk for future analysis. Moreover, SOsim offers module statistics, allowing performance evaluation. Metrics, such as throughput, CPU utilization, page fault rate and queue wait time, can be followed and analyzed.

ARCHITECTURE AND IMPLEMENTATION

Simulators create dynamics and simple real-world models. In the CSE, simulators have been used for a long time in some disciplines, allowing teachers and students to understand and analyze concepts and mechanisms. However, to build a simulator is not a simple task. It takes time and can be very expensive.

One of the most important points in building a visual education tool is how to show dynamic events in a simple and comprehensive way. To reach this goal, the simulator tries to simplify the real model as long as it does not mislead OS concepts and mechanisms.

The SOsim project was implemented based on object oriented (OO) programming, using Borland Delphi as the programming tool. There are several advantages to using OO for developing complex software such as a simulator; for example, code reutilization, extensibility and maintainability [26]. Borland Delphi was selected as a programming tool because Pascal is a well-known programming language. Students with this knowledge can study and modify the simulator source code, having a hands-on lab experience. Also, Delphi is a RAD (Rapid Application Development) tool, making programming easier and faster.

SOsim was developed to run under MS-Windows because it is very popular and easy to use. To run the simulator, it is not necessary to have any knowledge of programming, just an inexpensive Personal Computer and some contact with MS-Windows. Users just have to download the software from the Internet and run it.

BENEFITS

SOsim is a simple graphic interface which serves as an effective support tool for the better teaching and learning of the concepts and techniques in modern OS, rendering the whole process more efficient. Teachers and students should consider the following benefits:

- It allows users to visualize complex concepts in a dynamic and animated way, making classes more interesting than traditional ones.
- It improves communication between teachers and students, and students themselves, allowing debates and case studies.

- Simulations can be performed not only in ordinary classes. Students can use the software in labs or at home to review concepts or produce their own simulations.
- It is possible to use the simulator as a tool in a distance education course.
- The software is easy to run. Compilers, linkers, interpreters, scripts, libraries, or anything else, are not necessary.
- The software is easy to use. There is no command line interface, only a graphical interface. Besides that, there is full documentation available for download, showing how to execute the main tasks [6].
- It is possible to implement Problem-Based Learning [27].
- It is totally free of charge.

A SPECIFIC ASSIGNMENT

The simulator is currently being used in OS classes at the undergraduate level at PUC-Rio. It has been implemented in two different assignments. In the first assignment, concepts related to scheduling are explored, while in the second one, virtual memory. Below we present only the first assignment.

The assignment is executed just after the theoretical part has been presented in class. Students are then taken to a PC lab and split into groups of two. For twenty minutes, educational software and its basic functions are presented to the students. After that, they have to simulate and analyze four questions and present their conclusions in forty minutes. All the questions are related to one another, and in this way, allows the students to compare the different scheduling policy. At the end, the students and the teacher discuss the results of their simulations and share their experiences.

In the first question, each group simulates round-robin scheduling without priority with two processes: one CPU-bound and the other IO-bound. After verifying the distribution of processor time, the student is asked how the modification of time-slicing affects the behavior of the two processes.

In the second question, each group simulates round-robin scheduling with priority and creates two processes: one CPU-bound with priority four, and the other, IO-bound with priority three. The student has to compare the processor distribution to the first exercise and verify how the change of IO's time influences the behavior of the processes.

In the third question, the group utilizes the same late scheduling policy, but produces a starvation, or rather, a high priority CPU-bound process that does not let the IO-bound process be executed. In this exercise the student has to think about the criteria that should be utilized to define process priority in a multiprogramming environment. All in the same exercise, it is possible to create a zombie process just by deleting a process in starvation.

In the last exercise, the groups use round-robin scheduling with dynamic priority and compare it to the scheduling with static priority presented in the second question. In the same exercise the students have to analyze the advantages of utilizing processes with different boost priorities in this kind of scheduling.

RESULTS

After finishing the assignment, the student has to fill out a unanimous survey. There are seven questions which correspond to thirty students on average. The first five questions can be answered by choosing one of the following options: totally disagree, partially disagree, no opinion, partially agree, and totally agree. Two of the last questions are open ended and can be answered freely. The questions are the following:

1. The simulator makes learning theoretical concepts more satisfying.
2. The simulator elicits interest in the subject.
3. The simulator is easy to use and its interface is clear.
4. Using the simulator helps in the understanding and assimilation of theoretical concepts.
5. It is interesting to use the simulator to reproduce OS real situations.
6. In regards to the simulator, which aspects did you find most interesting in the learning of operational systems?
7. What is your opinion on the use of the simulator?

The results to the student survey can be observed in Table I. Since there were not any students who answered "totally disagree," the column was omitted. In conclusion, the survey shows that the majority of the students found that learning with the simulator was more enjoyable, got them interested in the subject, helped in the understanding of the concepts, and let them reproduce real life situations. The answers to the third question showed that the simulator interface has to be improved.

TABLE I
RESULTS OF STUDENT SURVEY

Question	Partially Disagree	No Opinion	Partially Agree	Totally Agree
Q1			15,79%	84,21%
Q2		15,79%	42,11%	42,11%
Q3	10,53%	21,05%	52,63%	15,79%
Q4		5,26%	15,79%	78,95%
Q5		5,26%	52,63%	42,11%

In question six, the students answered, in general, that the software helped them to visualize OS concepts and problems, bringing together theory and practice. Following are some of the comments from the survey:

- "The simulator helps the student move beyond theory."
- "The visualization of problems, such as starvation."
- "The visualization of concepts given in the classroom."

In question seven the majority of the students supported and praised the software initiative. Some students asked for more labs and others suggested software improvements.

Following are some more of the comments given in response to the survey.

- "Very good. I've learnt a lot."
- "There should be more labs because the simulator helps a lot in the assimilation of concepts."
- "I thought it was great, however, other tools should be included."
- "The idea is good, but the software has to be improved."

Beyond the positive results taken from the research, it was possible to create new situations which were not predicted in the proposed assignment; for example, the visualization of the zombie processes came from one of the groups. This situation had never been explored before, by even the authors.

CONCLUSION

The SOsim is a simulator with visual facilities to serve as an effective support tool for the better teaching and learning of the concepts and techniques in modern operating systems, serving as a way to render the whole process more efficient.

Teachers can explain dynamic events and their relationships using a simple-visual-friendly interface. In this way, students can understand complex concepts observing how the mechanics really work. It is especially important for the students who do not have a solid background. It is also important for short courses that do not have the time to cover the entire OS curriculum.

It is possible to extend this project and improve the benefits brought about by the simulator. Some modules are incomplete and others not yet implemented. We intend to render the software a more collaborative tool and create a viable construction of knowledge based upon projects and working groups.

SOsim has been used at the CS department of PUC-Rio, Brazil. The number of new users is growing and they have started to send their comments and suggestions. From this point on, we expect to create a group of users interested in improving and participating in the development of SOsim.

ACKNOWLEDGMENT

We would like to thank Prof. Francis Berenger Machado from the CS department of Pontifical Catholic University of Rio de Janeiro (PUC-Rio) for his support in this project.

REFERENCES

[1] L. P. Maia, "SOsim: A Simulator Supporting Lectures on Operating Systems," M.S. thesis, IM/NCE, Federal Univ. of Rio de Janeiro, Rio de Janeiro, Brazil, 2001.

[2] F. B. Machado and L. P. Maia, *Introduction to Operating Systems Architecture*. Brazil, LTC, 1992.

[3] F. B. Machado and L. P. Maia, *Operating Systems Architecture*. 2nd ed., Brazil, LTC, 1997.

[4] F. B. Machado and L. P. Maia, *Operating Systems Architecture*. 3rd ed., Brazil, LTC, 2002.

[5] F. B. Machado and L. P. Maia, "Operating Systems Architecture book website". http://www.training.com.br/aso, 2001.

[6] L. P. Maia, "SOsim website". http://www.training.com.br/sosim, 2002.

[7] A. Fekete and A. Greening, "Designing closed laboratories for a computer science course," in *Proc. 27th ACM SIGCSE*, 1996.

[8] A. B. Downey, "Teaching experimental design in an operating systems class," in *Proc. 30th ACM SIGCSE*, 1999.

[9] T. D. Wagner and E. K. Ressler, "A practical approach to reinforcing concepts in introductory operating systems," in *Proc. 28th ACM SIGCSE*, 1997.

[10] A. Pérez-Dávila, "OS bridge between academia and reality," in *Proc. 26th ACM SIGCSE*, 1995.

[11] S. Ramakrishman and A. M. Lancaster, "Operating system projects: linking theory, practice, and use," in *Proc. 24th ACM SIGCSE*, 1993.

[12] A. S. Tanenbaum and A. S. Woodhull, *Operating Systems: Design and Implementation*, 2ed., Prentice-Hall, 1997.

[13] D. Comer, *Operating System Design – The XINU Approach*. Prentice-Hall, 1984.

[14] B. Bynum and T. Camp, "After you, Alfonse: a mutual exclusion toolkit - an introdution to BASI". http://www.mines.edu/fs_home/tcamp/baci/, 1999.

[15] T. E. Anderson, W. A. Christopher and S. J. Procter, "The Nachos instructional operating system". http://www.cs.washington.edu/homes/tom/nachos/, 1999.

[16] M. Kifer and S. Smolka, *OSP: An Environment for Operating Systems (Instructor Version)*. Addison-Wesley, 1991.

[17] D. Jones and A. Newman, "RCOS.java: a Simulated Operating System with Animations". http://cq-pan.cqu.edu.au/david-jones/Publications/ Papers_and_Books/RCOS.java/, 2001.

[18] P. Denning, D. Comer, D. Gries, M. Mulder, A. Tucker, A. Turner, and P. Young, "Computing as discipline," *Communications of ACM*, vol. 32, no. 1, Jan. 1989.

[19] M. Ben-Ari, "Constructivism in computer science education," in *Proc. 29th ACM SIGCSE*, 1998.

[20] J.A.Valente, "Computers in education: shifting the pedagogical paradigm form instructionism to constructionism," Logo Exchange, 12(2), 39-42, 1993.

[21] S. Hanley, "On constructivism," Maryland Collaborative for Teacher Preparation. http://www.inform.umd.edu/ UMS+State/UMD-projects/MCTP/Essays/Constructivism.txt, 1994.

[22] D. Jonassen, "Designing constructivist learning environments,". http://www.coe.missouri.edu/~jonassen/courses/CLE/index.html, 2003.

[23] A. Silberschatz, *Operating System Concepts*. 6th ed., Addison-Wesley, 2001.

[24] W. Stallings, *Operating Systems Internal and Design Principles*. 4th ed., Prentice-Hall, 2000.

[25] A. S. Tanenbaum, *Modern Operating Systems*. 2nd ed., Prentice-Hall, 2001.

[26] G. Booch, *Object-Oriented Analysis and Design with Applications*. 2nd ed., Addison-Wesley, 1994.

[27] S. Pak, "Situated cognition and Problem-Based Learning: an implication of instruction,". http://chd.gse.gmu.edu/immersion/ttac/fall2000/portfolios/sunghye/Edit704/Problem-based%20learning.htm, 2003.