1986

The Costs and Benefits of a Teaching Laboratory for the Operating Systems Course

Douglas E. Comer
Purdue University, comer@cs.purdue.edu

Report Number:
86-589

THE COSTS AND BENEFITS OF A TEACHING LABORATORY FOR THE OPERATING SYSTEMS COURSE

Douglas Comer

CSD-TR-589
March 1986

The Costs and Benefits of a Teaching Laboratory for the Operating Systems Course

Douglas Comer

Computer Science Department
Purdue University
West Lafayette, IN 47907

TR-CS-589
March, 1986

During the 1985-86 academic year, the graduate-level operating systems course at Purdue University witnessed a dramatic shift in emphasis and content when we augmented the classroom lectures with supervised, hands-on laboratory work. The lectures continue to present concepts, but focus has moved from the traditional survey of topics in concurrent computing to topics more closely related to system design. Meanwhile, the laboratory fills two important needs: it gives the student a concrete example of the principles discussed in class, and it introduces tools and techniques used to design, build, and manage complex systems.

Results from the first two offerings of the laboraatory show that it is an outstanding success. Students enjoy the opportunity to learn and use good programming tools, and to see how a large, complex system fits together. They often spend time outside the required laboratory hours extending assignments, and experimenting with the system. Surprisingly, students who learn concrete system details in lab seem to understand concepts better than students from previous semesters who concentrated all their efforts on studying abstractions.

This report gives a short history of the move to laboratories, and summarizes our experiences with the operating systems lab. It discusses the assignments, laboratory staff, and facilities, and assesses costs including the hardware, software, and space used.

1. Introduction

In fall, 1985, the Computer Science Department at Purdue University moved into a newly remodeled building that had been carefully planned to encourage experimental research and encourage increased emphasis on experimentation in courses. In the new building, 22% of the floor space was devoted to laboratories that are being used for teaching and research.

Laboratory space in the new building was not an accident, but part of a conscious plan to change the department. The shift to experimental computer science began over five years earilier when we realized that both our courses and research suffered because faculty (and students) were deprived of adequate experimental facilities. The few faculty who managed to acquire equipment had (literally) no floor space to use it. Most realized that without adequate space, power and cooling, or professional staff, equipment was useless and experiments hopelessly difficult. So when the opportunity came to completely redesign the interior of a building, the department chose to allocate significant portions of the space to laboratories for teaching and research.

While the new building was still a dream, we proposed new experimental research programs, and contemplated how undergraduate and graduate courses could use laboratory facilities. We presented the administration with a plan to make programming labs a required part of the undergraduate core courses, and outlined the need for instructional laboratory space. We requested equipment grants from vendors and made good use of the resulting donations. We started to offer graduate seminars in the areas of networks and operating systems, allowing a few advanced Ph.D. students to use the meager facilities.

Students, who were eager for hands-on experience and willing to test their ability to put knowledge to use on real software, made the early systems seminars exciting and productive. It became apparent that we would soon have more students interested in taking them than we could accommodate comfortably.

2. The Xinu Project

One of the early systems seminars focused on the design and implementation of a hierarchically structured operating system. Digital Equipment Corporation had donated five LSI 11 microcomputers to the department, and these became the basis for a prototype. We set the goal of producing a system capable of supporting concurrent processing and network connections among the machines. The real goal, of course, was to understand operating system components, and to find an organization that was both useful and interesting.

Surprisingly, living with the constraints of microcomputers proved to be valuable because it made us consider the design carefully. Small memory size forced economy of space; slow processor speed forced economy of code. Early in the project, we decided to follow a hierarchical design in which successive levels of functionality are built using primitives provided by lower levels; in the ideal case, no function would be duplicated. The resulting system, called *Xinu*, provides impressive functionality in very little space. It is both useful and elegant.

The *Xinu* project worked well because it provided a faculty member with the opportunity to explore operating systems design and students an opportunity to become

directly involved in the project. In the first fews seminars, students built cross-development tools like compilers, loaders, and linkers, as well as downloaders and debugging aids. They tracked down hardware details and conducted experiments to solve small problems. Other students read design notes and code carefully (even before a compiler was available). Within eight weeks we had successfully downloaded and executed the basic parts of an operating system; by the end of the first semester we demonstrated a ring of LSI 11s sending packets.

3. Need for a Text

During the time Xinu was built, the Department began to discuss introducing more experimentation into conventional courses. Although extremely successful, systems seminars involved only between 6 and 12 of the most advanced students each semester. We talked about how to provide more students with the same experience. For example, Xinu technology provided a convenient environment for learning about operating systems and for testing new ideas. But the standard operating system course contained much material, and it was not obvious how to squeeze in experimentation as well, especially if students had no background.

The lack of a textbook posed the most serious problem. Few of the existing texts discussed operating system design at all. Most surveyed topics related to operating systems, dealing only with abstractions or languages for concurrent programming. Students were left with little or no information about the internals of a system, or worse, the impression that a special language or compiler was needed to build an operating system.

4. The Four Pieces Fall Into Place

One: A Text. During the 1982-83 academic year, while on sabbatical at Bell Labs, the author wrote a textbook on operating system design using the Xinu system as an example [COME84]. The text provided one of the essential ingredients for a laboratory-oriented operating systems course, a source of information on system structure and design details.

Two: Student Background A second key ingredient occurred when faculty teaching the compiler course switched to a UNIX-based environment, and began using C the implementation language. Having written a compiler in C, students were prepared to read and write C code, the implementation language for Xinu.

Three: Facilities. The third ingredient needed for a laboratory was computing facilities. During the 83-84 academic year, we hooked our LSI 11s to an instructional UNIX system, and had students cross-compile and download Xinu as part of their programming assignments. Although high machine loads made the environment less than ideal, our early experiences with experimental facilities reinforced our belief that they would make a significant improvement in the way students learn. Our efforts convinced the university that we had made a serious commitment to laboratories, and showed that we desperately needed better facilities.

We submitted proposals to Digital Equipment Corporation, and by late spring, 1985, we had acquired a VAX 11/785, 8 LSI 11/23 systems, and two Microvax I machines for the lab. The LSI 11/23s were faster than the early LSI 11/2s, and had Ethernet connections, providing an environment that supported network communication as well as

conventional operating system development. The Microvax I systems provided an environment with paging hardware, allowing students to experiment with virtual memory. More important, the VAX 11/785 provided the computing power needed to have multiple students in a lab working concurrently on their own operating system.

Four: The Space. The final requirement for a lab is one of the most difficult in most universities – space. In our case, the department was preparing to move, and building rennovations had been planned while we were acquiring equipment and planning labs. In fact, a Xinu lab had been included in some of the earliest building plans. Thus, the lab was in use shortly after the move.

5. The Xinu Laboratory

Current Facilities. The Xinu lab currently consists of 986 square feet of floor space partitioned into two areas by a semipermanent, sound-absorbing wall 7 feet high. The outer area consists of 15 work areas, each with a 3X5 table and desk area, CRT terminal as well as a white board which is used for instruction. Behind the wall are 7 machine racks that house 17 Digital Equipment Corporation LSI 11s, 5 Intel Corporation 8086 machines, 2 Microvax I systems, and miscellaneous other systems. The main computing engine is a dedicated VAX 11/785, located in an adjacent machine room.

Purpose. The Xinu lab supports both instruction and research. Students in the operating systems course are required to enroll in one of the 3-hour sections where they go each week to work on assigned problems. The lab problems involve modifying, testing, measuring, or extending the operating system. Additional lab hours are provided to allow students to access the facilities outside their assigned formal lab session. The lab also supports advanced students in a graduate systems seminar, as well as several independent study projects each semester. Finally, the lab is available to Ph.D. students working in the general area of operating systems.

Supervision. A key part of the instructional laboratory concept is supervision. The lab is not merely a computing facility or a quiet place to work; it is a teaching facility where students work under the supervision of lab instructors. We want students to learn good habits, to use tools well, to understand the concepts underlying design, and to learn how to construct and debug systems systematically. Thus, the focus shifts from an end product (i.e., the program) to a process (i.e., the design and implementation).

Personnel. At present, two half-time Teaching Assistants assigned to the operating systems course supervise the lab (in addition to their usual duties in the course). They give a short background lecture at the beginning of the lab and explain the problem to be solved. During the lab, they visit each station, monitoring students, offering advice, and answering questions. At the end of the lab period, they check each student's progress and assign a grade of superior, satisfactory, or unsatisfactory. They invent supplemental software and make sure the hardware and software are operating correctly. The assistants have taken an active role in suggesting problems and preparing small pieces of software for the assignments.

Modus Operandi. in the lab, students use the 11/785 and UNIX* Timesharing system for software development activities. They prepare a new version of their operating

* UNIX is a trademark of Bell Labs.

system and then download it into one of the LSI 11 machines. Once resident, their operating system executes independently, and the software we supply automatically connects the student's terminal to the console of the LSI 11 to observe results. If the operating system crashes, the student can restart it, abandon it, or upload a copy of the LSI 11 memory to the VAX. Once a copy of the memory has been uploaded, the student can use a post mortem program to pick through the image and print out information on the activity of each process.

6. Student Reaction

Student reaction to the hands-on laboratory experience has been almost unanimous and positive. In the beginning, we worried that the best students would complete the assigned problem and leave, but that has not been a problem. Students with previous systems experience are eager to extend the assignments, and to explore additional parts of the operating system. Even the students who find the assignments difficult indicate that the lab is a positive learning experience.

Because the operating system course is not required, enrollment gives a good indication of student interest. At the beginning of Spring semster 1986, the course was filled to capacity (40), and an additional 16 students were waiting for admission. The systems seminar was also flooded with 15 people participating. Such enrollments are especially significant in our environment, because students often seek advice on courses from their predecessors and unpopular seminars vanish quickly. Thus, one generation of students is encouraging the next generation.

Another measure of student interest is their willingness to tolerate inconvenience. Before fall of 1987, laboratories were not part of the official registration, so students had to adjust their schedules to squeeze in a laboratory on the first day of class. In Spring, 1986, when it became apparent that we could not accommodate the entire class during the working week, the students themselves suggested an evening lab.

Finally, we have had increased interest in teaching assistantships. Assistants enjoy the opportunity to devise interesting exercises, and often spend extra hours helping students with both the concepts and system details.

7. Lessons Learned.

In general, the laboratory has been a positive and stimulating experience. Students show genuine excitement about taking concepts out of the classroom and use them to make real systems run. Beyond that endorsement, we can relate several specific lessons learned from the labs.

There is no replacement for experience. We have found that students who work on a real system appreciate the intellectual content of the course much more than those who do not. Faced with the difficult task of modifying or extending a system, they comprehend the design issues, and seek principles that will help them solve problems. We use their inquiries to point them to fundamentals, relieving us from constantly trying to motivate them to read and study.

Robust hardware and software is needed. Because students push our systems to the limit, having a reliable, robust environment is essential. This many seem absurd in the context of a lab that allows students to write operating systems that completely control a

machine. In fact, the students' code often causes the microcomputer to halt in a catastrophic way. If any part of the support environment is unreliable, however, students tend to blame it for their problems. Thus, we need to be able to depend on every peice we give them so we can confidently point to their work as the problem. They need to know, for example, that our compiler and downloader produce a reliable, correct translation of their program; that our library procedures are correct and robust; and that our hardware is reliable and fault tolerant.

Having a design-oriented text is essential. Our lab requires students to apply concepts to practical, working systems. The lab concentrates on system details, tools, and the problem to be solved; students depend heavily on the text for background information. Texts that discuss only abstract concepts leave them unable to see how concepts apply to real systems, whereas texts that present case studies concentrate on details and fail to associate concepts with the code. Students need a text that bridges the gap and shows principles of design that translate concepts into working systems.

Labs should be integrated with the course. Early laboratory assignments grew out of conventional programming exercises. They used laboratory work to reinforce ideas in the weeks following their presentation in class. We found by trial and error that close coordination between lab and class works much better. We now try to coordinate assignments so that laboratory work parallels topics in class, extending and motivating the classroom presentation. As a result, labs deepen and broaden understanding.

Experience With Real Machines is Helpful. The *Xinu* lab grew from work on real machines, and in the beginning we wondered whether students would learn more from a simulated environment. In particular, we were worried that giving students a raw machine would make their systems too difficult to debug. From previous experience with a simulated environment [Comer76] we knew that they could provide detailed diagnostics and reproductability, making it much easier to write and debug systems. As it turns out, having a real machine has worked well in three ways. First, students have adapted to real machines better than expected. Although using real machines makes debugging difficult, they have accepted the challenge and worked hard to make their code correct. Second, dealing with hardware has added a sense of realism to the lab. For many students, the lab is their first experience with machines that they can touch and control. They find it satisfying and exciting to know that their operating system is not just a toy. They take pride in making their systems work correctly. Third, having real hardware has helped us expand and try new projects. Because it is possible to attach new devices, or to interconnect machines, students have been able to experiment with new disks, virtual memory, serial line interconnection among systems, and Ethernets.

Operating System Design Is Best Understood Through Experience. Perhaps the most important lesson we have learned is about teaching operating systems. Over two decades of basic research has taught us much about operating systems. We know much about individual system components, and we even know good ways to organize them. We understand how to build reliable, useful systems, and how to reason about them. It is time to shift emphases from courses that talk about operating systems to courses that teach students how to build operating systems. There are two aspects to system building;abstract concepts and practical knowledge. The combination of a course that emphasis concepts and a lab that emphasizes practical systems building satisfies both. The lab pushes students closer to the design process. They appreciate the difficulties of

design and learn to assess the consequences of design decisions. They see, first hand, how scientific principles guide design, and learn that concurrency is both subtle and deep. They learn that a real operating system is among the most difficult programs to conceive, understand, or modify. They grasp the importance of concepts and learn tools and techniques needed for system design.

References

[Comer76] Comer, D.,"HAL/411 Simulator: Software Support for a Course in Operating Systems," *Proc. Texas Conf. on Computing Systems*, Austin, Texas, 1976, 152-156.

[Comer84] Comer, D., *Operating System Design: The Xinu Approach*, Prentice-Hall, 1984.