An Open Source Laboratory for Operating Systems Projects*

Mark Claypool
Computer Science Department
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609, USA
(508) 831-5409
claypool@cs.wpi.edu

David Finkel
Computer Science Department
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609, USA
(508) 831-5416
dfinkel@cs.wpi.edu

Craig Wills
Computer Science Department
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609, USA
(508) 831-5622
cew@cs.wpi.edu

ABSTRACT

Typical undergraduate operating systems projects use services provided by an operating system via system calls or develop code in a simulated operating system. With the increasing popularity of operating systems with open source code such as Linux, there are untapped possibilities for operating systems projects to modify real operating system code. We present the hardware and software configuration of an open source laboratory that promises to provide students that use it with a better understanding of operating system internals than is typically gained in a traditional operating systems course. Our preliminary projects and evaluation suggest that thus far the lab has achieved its primary goal in that students that used the lab feel more knowledgeable in operating systems and more confident in their ability to write and modify operating system code.

1. INTRODUCTION

Worcester Polytechnic Institute (WPI) is a private university with approximately 2800 undergraduate students and over 400 computer science majors. As part of its core curriculum the Computer Science Department offers students an introductory Operating Systems course. The course covers traditional topics in operating systems, such as process management, synchronization and memory management using well-known texts [SG98,Tan92]. The course has been taught using the general purpose Unix computing facilities provided by the Campus Computing Center. Typically, student projects did synchronization among a set of Unix processes or threads, or managed virtual memory using user-level simulations. The nature of the projects means that students get experience with system programming while learning about operating systems concepts. Our systems courses were successful in providing students a practical exposure to systems calls through projects that access the many operating system services from user programs, but had not been able to provide an adequate hands-on experience with operating system internals.

By using a proprietary Unix operating system, the students did not get an opportunity to even study, let alone implement, real operating systems code. We had considered switching to an operating systems simulation environment such as Nachos [CPA93] or others [KS91,GBC+99], but had not moved in that direction because while students implement operating system code, they do so in a simulated operating system. We did not believe this was an improvement over the current approach.

Systems curricula must address the fact that operating systems with open source code such as Linux [Lin] are not only available, but becoming serious competitors to commercial products in the PC market. Using an open source operating system not only provides students with production-quality system source code for study, but also one on which they can experiment with if it is not designated for general use.

Courses supporting open source projects must provide a laboratory allowing 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 (or can get) root (or super-user) permission. Thus, each machine is insecure in that any file on the machine can be compromised, both in terms of privacy and integrity. This means that open source project machines cannot be easily time-shared since they can be unstable and not private. There must be quick ways to selectively repair or re-install the software on a machine in the event that a system is inadvertently compromised. Moreover, a super-user can infiltrate the network, sniffing network packets and sending large amounts of data, intentionally or inadvertently, in a denial-of-service attack. This requires additional protection outside of the machine itself to create a productive, yet protected environment.

Laboratories at three institutions along with the associated curriculum have served as prime influences on our approach. These laboratories are: a Linux-based lab at Auburn University developed with NSF support [CC97]; ALAMODE, a lab for distributed environments at Colorado School of Mines also supported by NSF [CM99]; and an advanced systems lab at

* This work is partially funded by the National Science Foundation Course, Curriculum and Laboratory Improvement Grant DUE9980803.

Michigan Technological University [MK99]. Our approach differs from these other laboratories in that we concentrate on enabling students to modify the operating system in an introductory course.

Our approach builds upon this previous work in operating and distributed systems to enhance our introductory operating systems course. The primary focus is the development of a laboratory of machines, the *Free/Open Source Laboratory* (FOSL, henceforth called the *Fossil lab*), running the Linux operating system along with projects to use these machines. The Fossil lab offers the chance to increase emphasis on a hands-on experience in the design and implementation of a large piece of systems software that is practical to modern computing systems. A Linux machine is assigned entirely to one group of students for privacy and stability, but is configured such that all students can use any machine in "guest" mode to increase productivity. Custom scripts are placed on publicly available bootable CDs to allow quick re-installation of client machines in the event of machine crashes. Lastly, the outside campus is protected from inadvertent network traffic by a firewall (a high-performance Linux server).

A principal theme in developing the course projects is a performance supplement where we examine performance issues of different system designs and implementations. This approach to teaching operating systems is espoused in a book by Dowdy and Lowery, which was commissioned by the ACM SIGMETRICS and the Computer Measurement Group (CMG) [DL93].

We believe the development of this dedicated lab will not only provide students opportunities to use, but also to experiment with a real system for operating systems work. These opportunities are not available in a general-purpose lab where the machines run a commercial operating system. It is our expectation that the work will not only have a strong positive effect on WPI and its students, but also serve as a cost-effective model that can be replicated at other institutions.

2. APPROACH

Our approach involves three steps: 1) configure both the hardware and software in the Fossil lab to support open source experimentation; 2) develop projects which allow practical exploration of core operating system concepts using the Fossil lab; and 3) evaluate the impact of the Lab in order to disseminate results and tune projects for more effective impact.

2.1 Configuration

The Fossil lab has 30 Intel Pentium-III 600 MHz desktop computers and 1 server. We employed a graduate student for 3 months during the summer to assist in setting up the lab and in developing the projects to be used in the courses. This same student then stayed on through the year as a teaching assistant for the OS course, doubling as the Fossil system administrator.

The hardware configuration is designed to provide an acceptable project development environment for each client machine, while still allowing access to the Internet via the server. The server and networking equipment are physically secure. In addition, the clients are configured so that gaining root access to the machine, and therefore allowing access to all source code, is very difficult except for the group assigned to the machine.

The server connects to the clients via an Ethernet link, connecting to two 24-port hubs. The two hubs are kept in a physically secure network closet. To reduce the susceptibility of network snooping for passwords, only secure connections (`ssh`, `slogin`, `scp`) are enabled to the Fossil server. The server is also configured to allow clients to browse the Web as if they were on the main WPI network.

The client machines are physically located in the Fossil lab. The clients are all on a private subnet while the Fossil server is on the main WPI network. Remote access to the Fossil clients from outside the lab (say, from a dorm room) is available by first connecting to the Fossil server, and from there connecting to any of the clients.

The software configuration is designed to provide as much local access to the client machines as possible for the students in order to provide good performance for developing projects and to reduce network load. Each student group is given the root password for one client. Modifications to the kernel are then done directly on their client machine, using root permissions to install the kernel and reboot the machine. Each student receives an account on the server, but uses this account primarily for backups of their project code and access to their Fossil clients from outside the Fossil lab.

Each Fossil client is configured with three accounts. As indicated above, there is a "root" account that is given to each group for the machine to which they are assigned. Each group then creates a user account for each member of the group, giving each user account `sudo` permission to allow running commands as root. In addition, each client machine comes configured with a "guest" account, with the same password for all client machines in the lab and is given out to each student. The guest account allows student to log into any idle machine in the lab, even if they are not assigned to it. This gives the ability to use the machine for browsing the Web or for document preparation. The third account is an "admin" account which allows access as root for the teaching assistants, system administrators and course instructors.

Each student is able to start an X session on their assigned machine from any idle machine, which allows multiple group members to work on their assigned machine from multiple Fossil clients.

Lastly, in order to not allow root access to the machines via a boot floppy, the boot sequence of the machines is changed in bios to first boot from the hard drive, then the CD ROM. This setting is protected via a BIOS password, which only the group assigned to the machine knows.

We developed software scripts to facilitate the initial installation of each Fossil client as well as to support several different methods for repair and re-installation. The scripts are put on a bootable CD ROM that runs a minimal kernel. The script supports installation methods of: *full* which repartitions and reformats the hard-drive and re-installs all software from the initial, default configuration; *linux* which restores the original Linux file system but leaves the home directories alone; and *kernel* which restores just the Linux source code and re-installs the original bootable kernel.

3. PROJECTS

The first offering of our Operating System course had 4 projects utilizing the Fossil lab[1], some based on [Nut01], spread out over the 7-week term. In the projects the students were required to study the current operating system internals, design and implement a solution, evaluate the performance of their solution and answer some short questions regarding extensions to their project. Since the majority of debuggers do not work at the operating system kernel level, students were first briefed on how to debug effectively using `printk()`, a means of printing kernel data to a log file. Students turned in their complete source code so that the modified kernels could be re-compiled and rebooted for grading.

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`, the location of the Linux source code, instructions on re-compiling and rebooting the kernel, and saving work to the Fossil server. The set of commands used was not intended to be exhaustive, but rather was intended to get students formed into groups and introduced to some of the Linux fundamentals required for the later projects.

The second project involved creating a new process scheduling algorithm. Students studied parts of the Linux scheduler in depth to understand how it decides on which process to run. They then modified the Linux scheduler to implement a new scheduling policy, called fair-share scheduling, that allocated CPU time based on number of processes for each user. They evaluated how their new scheduler performed in terms of fairness and overhead and wrote up details on their implementation and evaluation.

The third project, worth twice the points of the other projects, had students implement a new synchronization primitive, called an *event* and write code to use it. Students 1) created new system calls to allow users to access their new events; 2) designed and implemented data structures that provide the functionality required of the events; 3) added source code to the Linux build process; and 4) designed and developed a simple text-based video producer and video consumers to use the events.

The fourth project had students design and implement a new device driver, called an *mbox* (for "mail box" or "message box") for a FIFO device. The driver was "virtual" in the sense that it was not tied to a piece of particular hardware. Rather, it appeared as a device to the operating system and the user by registering itself with the device independent layer of the Linux kernel. Students implemented the device driver as a loadable module, a convenient means of extending Linux functionality.

4. EVALUATION

Thus far, we have used the Fossil lab successfully in one operating systems course. The breakdown of grades for students in this course offering was comparable to the breakdown of grades in previous operating systems courses that did not use the Fossil lab. We informally asked students how much time they spent doing the Fossil lab projects and compared these to the times previous students had spent on projects in operating systems courses without the Fossil lab. In general, the median amount of time spent on projects was about the same for each course offering, but the variance was greater for the Fossil lab projects in that some students required considerably more time when they had system problems or very difficult bugs.

We further evaluated the effectiveness of the Fossil lab for this course by analyzing feedback from the Teaching Assistants and through a survey of the students that used the Fossil lab compared with a survey of students that took a previous, non-Fossilized operating systems class.

The operating systems course was primarily composed of CS majors, but nearly a third of the class was from other disciplines, especially Electrical Engineering. Most students had their own PCs in their dorms, but only 1/3 of them ran Linux on their PCs.

In the primary offering of the course, there were few major difficulties. One machine crashed for several days due to a loose motherboard connection. There were 5 or so *full* re-installations from students that inadvertently compromised their hard-drives. There were nearly a dozen *linux* installations that restored the original Linux kernel after students were unable to get their modified kernel to work.

The class surveys allowed students to provide anonymous responses to about a dozen questions designed to quantitatively measure the impact of the Fossil lab on student understanding of operating systems. Students provided numeric answers to statements based on their agreement, where a '1' indicated they strongly disagree with the statement, '2' indicated they disagree, '3' indicated agreement and '4' indicated strong agreement. 26 of 70 students responded to the survey in a previous course without the Fossil lab and 49 of 70 students responded to the survey in the course with the Fossil lab. Due to space constraints, we only present the results from several key questions below. "Traditional" represents student agreement in the earlier offering of the operating systems course, and "Fossil" represents student agreement in the course that utilized the Fossil lab.

"I think the course material and projects helped me to gain a good understanding of operating systems in general."

Traditional	3.3
Fossil	3.3

From the above results, students feel that both offerings of the course provide equal understanding of general operating system knowledge. This indicates that the addition of the Fossil lab did not detract from the general concepts taught in the class.

"I think the course material and projects helped me to gain a good understanding of operating systems in terms of the services they provide at the system call level."

Traditional	3.0
Fossil	3.3

From the above response, it appears that students using the Fossil lab have a perception of better understanding the system call

[1] Project descriptions online at `http://fossil.wpi.edu`

services that an operating system provides. This is somewhat surprising since both course offerings made about equal use of system calls from the user level. However, the Fossil lab course did have students implement additional system calls so this may have added to their understanding.

"I think the course material and projects helped me to gain a good understanding of operating systems internals."

Traditional	2.9
Fossil	3.3

"I think the course material and projects gave me experience that would help me write or modify portions of an operating system."

Traditional	2.6
Fossil	3.1

From the above two responses, it appears that the Fossil lab achieved a primary goal in helping students in their understanding of operating system internals.

The surveys also provided space for free-form comments, which we examined carefully. Here we provide a small excerpt of some of the comments that we feel summarize some of the opinions voiced in the use of the Fossil lab:

"Making alterations to the Linux kernel taught me far more than any other part of the course."

"When the kernel crashed, we had to manually reboot the system, which took an awful lot of time."

"I have seen people take other OS courses and they did not dive into the material as far as we did because they will not let you modify the OS on any of the school servers."

5. CONCLUSION

The increasing popularity of today's open source operating systems provide the opportunity for students in operating systems courses to design and implement course projects that modify real, production quality operating system code. In this work, we have presented the configuration and evaluation of the Fossil lab, an unrestricted, yet safe environment for students to gain hands-on experience with real operating system code.

Thus far, we have only recently completed configuration of the Fossil lab and have used it to teach one undergraduate operating systems course. Our preliminary evaluation suggests that students using the Fossil lab appear to have a better understanding of operating systems internals and feel more confident in their ability to write and modify operating system code. Further evaluation will prove critical in completely assessing impact as we use the Fossil lab to support additional operating systems courses, including our offering of an undergraduate Distributed Computer Systems course. In particular, we intend to explore other methods of evaluating the impact of the course, including a closer analysis of grades and perhaps even additional tests.

6. ACKNOWLEDGEMENTS

We would like to thank Teaching Assistants Hari Kannan and Jae Chung for their excellent work in setting up and assisting with operating systems in the Fossil lab.

7. REFERENCES

[CC99] R. Chapman and W. H. Carlisle. A Linux-based Lab for Operating Systems and Network Courses. *Linux Journal*, 1997.

[CM99] T. Camp and M. Misra. Alamode: A Laboratory at Mines offering distributed Environments. Colorado School of Mines. NSF DUE-CCLI funded proposal, 1999.

[CPA93] W. Christopher, S. Procter, and T. Anderson. The Nachos Instructional Operating System. *Technical Report CSD-93-739*, University of California at Berkeley, 1993. URL: http://www.cs.berkeley.edu/~tea/nachos/index.html

[DL93] L. Dowdy and C. Lowery. P.S. to Operating Systems. Prentice-Hall, 1993.

[GBC+99] M. Goldweber, J. Barr, T. Camp, J. Grahm, and S. Hartley. A Comparison of Operating Systems Courseware. In *Proceedings of the ACM SIGCSE Conference*, pages 348-349, March 1999.

[KS91] M. Kifer and S. A. Smolka. *OSP: An Environment for Operating Systems Projects*. Addison-Wesley, 1991.

[Lin] The Linux homepage at Linux online. URL: http://www.linux.org/

[MK99] J. Mayo and P. Kearns. A Secure Unrestricted Advanced Systems Laboratory. In *Proceedings of the ACM SIGCSE Conference*, pages 165-169, March 1999.

[Nut01] G. Nutt. *Kernel Projects for Linux*. Addison-Wesley, 2001.

[SG98] A. Silbershatz and P. Galvin. *Operating System Concepts.* Addison Wesley, 5th edition, 1998.

[Tan92] A. Tanenbaum. *Modern Operating Systems.* Prentice-Hall, 1992.