

Using iPodLinux in an Introductory OS Course

Barry Lawson
blawson@richmond.edu

Lewis Barnett
lbarnett@richmond.edu

Department of Mathematics and Computer Science
University of Richmond
Richmond, VA 23173-0001

ABSTRACT

This paper describes a proof of concept for introducing iPods and iPodLinux into a one-semester introductory undergraduate operating systems course. iPodLinux is a version of the Linux operating system modified to run on iPods. We added a project to our course in which the students modified the iPodLinux kernel, and we supplemented lectures by discussing specifics of the Linux implementation as they relate to general operating systems concepts. We feel the course was much improved by these additions, with no substantive omission of regular material. Student response was very enthusiastic, and we feel the new material enhanced their course experience by providing a component that was empowering and helped to further improve their knowledge and skills.

Categories and Subject Descriptors

D.4.7 [Operating Systems]: Organization and Design;
K.3.2 [Computer and Information Science Education]:
Computer Science Education

General Terms

Design, Experimentation

Keywords

Operating systems, iPodLinux, Linux, education

1. INTRODUCTION

This paper describes our experience introducing iPods and iPodLinux [11] into a one-semester introductory undergraduate OS course. iPodLinux is a version of the open-source Linux operating system modified to run on iPods. We supplemented our existing OS course, in which we typically use one warm-up project in C and three standard Nachos [6] projects, by including an additional project in which the students modified the stock iPodLinux kernel. We also supple-

mented lecture material by discussing specifics of the Linux implementation as they relate to general OS concepts.

Thanks to a grant offered by our university, each student in the course was loaned a fifth-generation iPod to use for the entire semester and the instructor was given an iPod to keep. Because the fifth-generation iPod is not yet officially supported by iPodLinux and because the iPodLinux documentation is not complete, there was a significant time investment required during the semester to realize the project. However, the time commitment required for a future offering by us, or other interested instructors, will be significantly less because of this investment.

The main contribution of this approach is that it provides a proof of concept for successfully using in the same course a project based on modifying iPodLinux subsequent to projects in an instructional operating system. By starting the projects earlier in the semester, we were able to add the new material with no substantial omissions from our typical material. We feel that the lectures were improved by adding concrete examples from a real-world operating system that the students use every day. We also feel that the students benefited in terms of improved programming skills, increased confidence in their abilities, and exposure to a real-world system. The progression from simulated operating system to real-world operating system allowed students to hone their skills in a carefully controlled environment before progressing to the more challenging real-world environment. For the students, it was empowering to transition into the Linux kernel and work with OS components similar to ones they had already implemented in Nachos. Feedback from students was very positive, and we plan to use this material again in our next offering of the course.

2. RELATED WORK

This paper focuses on the nature of the projects used to support an introductory undergraduate operating systems course and the effect that this choice has on the course as a whole. One choice is to have students write user space programs that make use of operating system services. Many textbooks supply suggested projects of this sort (see, e.g., Nutt's textbook [19]). Another option is to have students implement all or part of an OS themselves. There are many variations on this theme. One prominent example is Nachos [6], which supports implementation of projects in a simulated instructional operating system. OS/161 [7] also takes this approach. Tanenbaum's Minix operating system [20] exemplifies the approach of providing a fairly complete instructional OS for students to modify running directly on

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCSE'08, March 12–15, 2008, Portland, Oregon, USA.

Copyright 2008 ACM 978-1-59593-947-0/08/0003 ...\$5.00.

real hardware. GeekOS [8] also falls into this category, though it can also be run on the Bochs IA-32 emulator [5]. As a final possibility, students can be asked to modify a “real” operating system running on real hardware or on a virtual machine. Nutt’s projects for Linux [18] are an example. There has recently been interest in instructional systems that model mobile or embedded operating systems. PortOS [3] and BabyOS [15] are examples. Anderson and Nguyen [2] provide a relatively recent survey of instructional OS systems as well as an informal survey of their use among U.S. colleges and universities. We describe a course that incorporates both module construction in a simulated OS and kernel modification of a real operating system running on real hardware.

3. MOTIVATION

The University of Richmond offers an iPod grant as part of the University’s Mobile Learning Initiative. In a program inspired by the Duke University First Year iPod Experience [4], iPods are awarded on a competitive basis to faculty who submit proposals detailing plans for use of iPods in a specific class. Each student in the class is loaned an iPod to use for the semester and the faculty member is given an iPod to keep. Use of the iPods, both at our institution and at Duke, typically center on flexible delivery of audio/video course materials or use of the iPods as a field recording device. The availability of an open-source replacement operating system for the device presents interesting opportunities for the computer science curriculum.

We took advantage of this opportunity in our Operating Systems course. Students modified iPodLinux [11], a version of the open-source Linux operating system modified to run on iPods, to provide additional functionality on the device. Use of a special-purpose laboratory (see, e.g., [17]) or virtual machines for this type of project are also an option. However, the iPod/iPodLinux combination is attractive for use in the OS course for several reasons. First, students are able to modify a real-world OS running *directly* on hardware, moving beyond emulated hardware and/or pedagogical OSs. Second, each student has a dedicated machine, accessible at any time — competition for resources is not an issue. Third, the iPods are easily restored to their original Apple-OS state using iTunes [14], so that installing and modifying iPodLinux poses no threat to the iPod itself. Finally, using iPods rather than desktops or virtual machines for Linux kernel modifications makes the project more interesting to a wider audience. Although members of the university community are generally not familiar with Linux or virtual machines, we are able to easily promote our project because of the ubiquity of and familiarity with iPods.

4. COURSE STRUCTURE

In our course, we were able to expose students both to the controlled environment of emulator-based projects and to a hardware-based project using a real-world operating system. From our experience, using an instructional operating system like Nachos [6] is invaluable because it provides students the opportunity to develop operating systems constructs from the ground up in a carefully controlled environment. For this reason, we allowed the students to develop an understanding of operating system internals using Nachos projects before assigning an iPodLinux project. Fur-

thermore, students entering our course generally do not have sufficient experience in C to jump directly into an iPodLinux project. We use C++-based Nachos projects, preceded by a warm-up project in C, to allow them to develop the necessary C skills.

More specifically, the course centered around a sequence of five programming projects:

- a warm-up project in C, assigned the first day of class, to familiarize the students with threads, inter-process communication, and concurrency (e.g., using `fork()`, `execve()`, `pipe()`, etc. system calls available in C);
- three successive commonly-used Nachos assignments (thread system, multiprogramming, and virtual memory) in C++; and
- a final project in which the students had to modify the iPodLinux kernel by introducing new system calls.

Students worked individually on the first project, in teams on each of the three Nachos projects, and individually on the iPodLinux project.

Our academic calendar operates on a standard 14-week semester system, and the course is four credit hours including a one-hour laboratory. Lectures in the course typically focused on one of three areas: introducing general operating systems concepts found in any OS textbook; describing Nachos details necessary for students to tackle the Nachos projects; or describing Linux kernel details, relating implementations back to Nachos and general concepts.

5. SOFTWARE ENVIRONMENT

The basic iPodLinux installation allows the user to choose between Apple’s OS and iPodLinux at boot time — in this way, students can continue to use the iPod as the familiar music player as well. The initial install of iPodLinux can be accomplished using Linux, Windows, or Mac platforms. We chose to use a Windows-based laboratory for iPodLinux installation (so that students could also use the iPods as music players compatible with their own Windows machines), though a simple installation program is also available under Linux. The process required installing and using iTunes on a Windows machine to format the iPod file system to Windows (FAT) rather than Mac or Linux format, followed by installing and executing the iPodLinux installer software. We chose the Loader2 boot loader for the iPod (available as an option in the iPodLinux installer) so that changing the iPod’s kernel requires no firmware modifications — the iPodLinux kernel binary can simply be copied to the iPod as a regular file, which will be recognized upon subsequent reboot of the iPod. During the installation process, we recommend avoiding any tempting but unnecessary additional modules because they will slow the iPod’s boot process, and many reboots will be necessary as modifications are made to the kernel.

iPodLinux [11] is currently based on version 2.4 of the Linux kernel, appropriately modified (i.e., patched) for the iPod’s ARM architecture. In order to modify the kernel (or create new user-level programs), the source must be cross-compiled using an appropriate toolchain [13]. Similar to installation, the toolchain can be installed on a variety of platforms (e.g., Linux, Mac, or within Cygwin for Windows) — we used machines in our department’s Linux-based cluster

of workstations. Assuming Loader2 is present, the resulting kernel binary (as well as any user-program executables) can simply be pushed onto the iPod for execution.

Because we were new to Linux kernel modification, getting the software environment configured to support the project required a substantial investment in time. Because the hardware was not released to us until the beginning of the semester, the development needed to be done concurrently with running the course. Most of the difficulties arose from dealing with our “officially” unsupported version of the iPod hardware and with sketchy iPodLinux documentation. Regardless, the commitment required for a future offering of this course (by us or other interested instructors) will be significantly less because of this work.

The setup work involved two main avenues: understanding Linux implementations in general and implementing particulars in iPodLinux. For the former, we primarily used two sources as references for Linux. We found Robert Love’s text [16] to be a well-written compendium on Linux, although the currently-available second edition describes version 2.6 of the Linux kernel, rather than version 2.4 on which iPodLinux is based. Accordingly, we also made frequent use of Tigran Aivazian’s collection of Web pages [1], which discuss specifics of the 2.4 kernel. For the latter, we relied heavily on the iPodLinux Kernel Building page [10]. Unfortunately, the version of the kernel (2.4.24) referred to on that page would not work for us despite considerable effort. As of this writing, iPodLinux is officially supported only through the third generation of iPods; our grant provided fifth-generation iPods. Modifying and cross-compiling the kernel for fifth-generation iPods requires use of an experimental kernel (version 2.4.32) [9] and an appropriate cross-compiler (version 3.4.3 of the toolchain) [13]. In addition, although the documentation indicated that the shell would not work on our hardware, the File Browser module under podzilla (the GUI for iPodLinux) worked reasonably well for our purposes of executing user-level programs.

We should also mention that, because iPodLinux is not officially supported for fifth-generation iPods, using iPodLinux on those iPods is unreliable. In our experience, the iPod froze on several occasions during execution of iPodLinux. Some of these freezes may have been due to an error introduced in the modification of the kernel, while other times it appears to be a result of a kernel that is not fully fleshed out for the specific architecture. This can make debugging difficult, but our students’ experience with C/C++ during the Nachos projects proved especially valuable in this regard.

6. PROJECT DESCRIPTION

Our assigned project was loosely based on a system call project by Gary Nutt [18]. Students were required to add at least one new system call to iPodLinux. The purpose of the required system call was to return to a user-level C program information similar to that given by the typical process status (`ps`) command — a list displaying process ID, owner, group, name, and start time of all running processes. Students were also required to write a simple user-level C program that invoked and exhibited correctness of their system call implementation.

This project required students to become familiar with basic Linux kernel data structures, allocating and freeing memory in the kernel, and passing information from kernel space to user space. More specifically, to successfully com-

plete the projects, students needed to:

- add an appropriate `SYMBOL_NAME` and syscall offset for the new system call;
- add their system call function to either a new source file, with appropriate `Makefile` modifications, or to an existing file containing system calls (e.g., in the `kernel/` directory);
- use appropriate kernel routines for dynamically allocating and deallocating memory in the kernel as necessary (e.g., using `vmalloc()` and `vfree()`);
- use appropriate kernel routines for copying from user to kernel and from kernel to user spaces as appropriate (e.g., using `access_ok()` followed by `copy_to_user()`);
- understand basics of the `task_struct` data structure containing much of the information of interest;
- access all current processes using either a doubly-linked list or hash table of processes, available via pointers in `task_struct` through the current process;
- use appropriate locking mechanisms to ensure proper access control (e.g., using `read_lock()`);
- ensure appropriate matching of data types between kernel and user space;
- write a user-level C program to invoke their new system call; and
- cross-compile their modified kernel source and user-level program, copy to the iPod, and thoroughly test.

Students interested in enhancing their project beyond the basic requirements were also encouraged to:

- display elapsed runtime of each process (e.g., using the `jiffies` variable);
- display meaningful process information available from virtual filesystem mechanisms (e.g., using `d_path()`; and/or
- display user and group names rather than IDs (note, however, that only one user, `root`, executes under normal iPodLinux operation).

A complete assignment description for the project is available from the lead author’s Web site [12].

7. EVALUATION

Because we have only been able to use the iPodLinux project in one offering of our OS course, the results presented in this section are admittedly anecdotal. However, we have demonstrated the feasibility of incorporating a “live” kernel modification exercise into an introductory operating systems course using familiar and readily-available hardware devices and open-source iPodLinux software. We believe strongly that the project benefits the students and instructor and makes for a more exciting and improved OS course.

We feel that including iPodLinux in the course had a positive impact on lectures and class discussions. Much as we feel it necessary to discuss Nachos details in certain of the lectures, so too we feel it necessary to discuss specifics of

the iPodLinux implementation. In our experience with this course, lectures involving high-level concepts presented in isolation are not as exciting for the instructor or as motivating for the students as are lectures pairing the concepts with details of implementations with which the students are currently working. Indeed, those lectures centered on Nachos or iPodLinux details sparked more interested discussion from the students than did concept-only lectures. Adding iPodLinux to the mix improved this aspect of the course, providing real-world examples directly accessible to the students.

We also feel that our process of first presenting a general concept, followed by discussing the concept in the context of Nachos and then in the context of iPodLinux was valuable. By revisiting concepts, each time in a more concrete setting, we feel that the students have a better opportunity to fully understand the concepts. In particular, because iPodLinux is a version of a real-world operating system that the students use every day, studying the concepts through iPodLinux gives students a better appreciation for the importance of those concepts.

The process of incorporating iPodLinux into the course is also very beneficial for the instructor. Because neither of us are experts in OS, we had little prior experience modifying the Linux kernel. The process of understanding the implementations sufficiently well to develop and assign a project and discuss it with students improved our knowledge of the subject.

Student feedback was very positive. As mentioned above, in-class discussions about iPods and iPodLinux were lively. Students' confidence in their abilities appeared to improve as they began to realize that kernel modification is not as daunting as it originally seemed — their experience in Nachos was, in our opinion, critical for this. The students were able to interest other students on campus about their project, which is unusual for an upper-level computer science course. Furthermore, at the end of the semester, three of the students expressed interest in working on a larger-scale iPodLinux project to contribute to the iPodLinux community. Unfortunately, because of the iPod loan agreement and those students' summer research schedules, completing such a project was not feasible (though may be considered for a future independent study project).

Following are written comments related to iPodLinux provided by students in end-of-semester evaluations. Aside from the iPod/iPodLinux pair being unreliable, all of the students' comments about the new material was positive.

- *A special software and iPods were incorporated into the curriculum and although this didn't always work smoothly because of technical glitches, I feel that it was the best part of the course because they provided some "real world" insight into the course topics.*
- *This was a great course because Nachos helped me learn a lot about OS's and C++ programming, and iPodLinux showed me how OS concepts are used in a "real-world" OS.*
- *This class has been more work than I've ever done in any other class. In most cases, this would be described as excessive work, but I really think it was justified, and really DID help us learn the material. It wasn't busy work, and I enjoyed it.*

- *The course is a lot of work, but it's worth it. It teaches a lot of the design and decisions that have been made over the years, as well as teaching some important skills with regards to team programming.*
- *The concepts in operating systems are simple and clear, though the solutions are pretty theoretical. Thus I really like the emphasis of the course on projects. We basically implement parts of an OS in each project.*
- *I think we should've gotten started a little earlier in digging into a real OS (iPodLinux)...*

In addition, the response from the campus community has been extremely positive, helping to bring more exposure to our computer science program. The iPod grant review committee and the Center for Teaching, Learning, and Technology expressed keen interest in the project because of its uniqueness compared to the proposals typically received. During the semester, we used the project in a presentation to newly accepted students, advertising opportunities in the computer science major. In addition, our project was chosen as the faculty teaching highlight for the inaugural Arts and Sciences Newsletter, which debuted in the Fall 2007 semester.

8. CONCLUSIONS AND FUTURE WORK

We feel that the inclusion of iPods and iPodLinux into our one-semester introductory OS course was very successful and required little substantive omission of usual material. Obviously, including any sort of new material into a course will require omitting some existing lecture material. Fortunately, the iPodLinux material was easy to add to our undergraduate OS course without compromising important pedagogical aspects. We omitted OS history from class discussion, simply assigning it as reading. Also, standard OS textbooks typically provide some real-world examples. Again, we assigned these as reading, focusing in class on the Nachos and iPodLinux examples to which the students could directly relate. Finally, several of the OS solutions are theoretically interesting but not practical or implementable in practice. We did not focus on these as heavily as we did in the past, assigning the topics as reading and homework, preferring instead to spend class time discussing solutions that can be implemented in practice.

The biggest payoff was simply that students had the opportunity to modify a real Linux kernel that ran directly on hardware, further improving their knowledge, skills, and confidence in their own abilities. Moreover, despite the additional workload, more of the students in this offering of the course completed optional assignments than in our previous (no iPodLinux) offerings. It was empowering for the students to know that, with respect to iPodLinux, they were but a few steps behind, and sometimes on par with, the instructor in tackling the new material.

Grading the projects was straightforward. We cross-compiled each student's modified kernel source and user-level test program, and then pushed onto our iPod and tested. We also carefully reviewed each source submission, ensuring that students wrote clean, efficient, well-documented code that followed the appropriate steps (e.g., appropriately allocated and deleted memory within the kernel).

We also realize there are some limitations. This type of project is naturally limited to instructors who have iPods

readily available. However, older generation iPods (which may be fully supported by iPodLinux) can be purchased at a more reasonable cost than investing in new iPods — none of the iPod functionality required for this type of project would be compromised. Also, because many students already own iPods, and because the iPod can dual-boot iPodLinux and Apple's OS with no ill effects (though backing up existing files prior to installation is highly recommended), the opportunity exists to use personal iPods for this sort of project. Furthermore, not all generations of iPods are officially supported by iPodLinux, but the success of our project indicates that projects on other generations can still be made to work with moderate effort.

Based on the success of this course, we plan to use iPods and iPodLinux again in our next offering of the OS course. We expect the experience to be even better next time because much of the necessary setup work is already accomplished. (Perhaps fifth-generation iPods will be supported by iPodLinux by the time of our next offering.) We will be able to focus more on additional interesting project ideas without having to learn Linux kernel modification anew. More specifically, some OS courses using Nachos include a fourth project on the Nachos filesystem. We are interested in an iPodLinux project in which the students work with and/or modify the Linux filesystem, specifically understanding the virtual filesystem, inodes, the `dentry` data structure, superblocks, etc. (This more ambitious project may require supplanting the final exam with a final iPodLinux project.) We are also interested in having the students give short presentations throughout the semester on various related aspects of the iPod (e.g., specifics of the ARM architecture — a nice segue from the computer organization course) and iPodLinux.

9. ACKNOWLEDGMENTS

We thank the University of Richmond Center for Teaching, Learning and Technology for their support through the iPod loan program, part of the University of Richmond Mobile Learning Initiative. We also thank the students in our OS course, most notably Andy White for his additional assistance with multi-platform iPodLinux installation.

10. REFERENCES

- [1] T. Aivazian. Linux kernel 2.4 internals. http://www.faqs.org/docs/kernel_2_4/1ki.html.
- [2] C. L. Anderson and M. Nguyen. A survey of contemporary instructional operating systems for use in undergraduate courses. *J. Comput. Small Coll.*, 21(1):183–190, 2005.
- [3] Benjamin Atkin and E. G. Sirer. PortOS: an educational operating system for the post-PC environment. In *SIGCSE '02: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, pages 116–120, New York, NY, USA, 2002. ACM Press.
- [4] Y. Berlinger. Duke University iPod first year experience final evaluation report. http://cit.duke.edu/pdf/reports/ipod_initiative_04_05.pdf, June 2005.
- [5] Bochs, the cross platform IA-32 emulator. <http://bochs.sourceforge.net>, 2007.
- [6] W. Christopher, S. Procter, and T. Anderson. The Nachos instructional operating system. In *Proceedings of the 1993 Winter USENIX Conference*, pages 481–489, January 1993.
- [7] D. Holland, A. Lim, and M. Seltzer. A new instructional operating system. In *SIGCSE '02: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, pages 315–319, New York, NY, USA, 2002. ACM Press.
- [8] D. Hovemeyer, J. K. Hollingsworth, and B. Bhattacharjee. Running on the bare metal with GeekOS. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, pages 315–319, New York, NY, USA, 2004. ACM Press.
- [9] iPodLinux experimental kernel. http://ipodlinux.org/Experimental_Kernel.
- [10] iPodLinux kernel building. http://ipodlinux.org/Kernel_Building.
- [11] iPodLinux project home page. <http://ipodlinux.org/>.
- [12] iPodLinux system call programming assignment. http://www.mathcs.richmond.edu/~blawson/OS_S07_iPodLinuxProject.pdf.
- [13] iPodLinux toolchain. <http://ipodlinux.org/Toolchain>.
- [14] Apple iTunes home page. <http://www.apple.com/itunes/>.
- [15] H. Liu, X. Chen, and Y. Gong. BabyOS: a fresh start. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 566–570, New York, NY, USA, 2007. ACM Press.
- [16] R. Love. *Linux Kernel Development*. Novell Press / Pearson Education, Inc., Indianapolis, IN, 2nd edition, 2005.
- [17] J. Mayo and P. Kearns. A secure unrestricted advanced systems laboratory. In *SIGCSE '99: Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education*, pages 165–169, New York, NY, USA, 1999. ACM Press.
- [18] G. Nutt. *Kernel Projects for Linux*. Addison Wesley Longman, Inc., Boston, MA, 2001.
- [19] G. Nutt. *Operating Systems*. Pearson Education, 3 edition, 2004.
- [20] A. Tanenbaum and A. Woodhull. *Operating Systems Design and Implementation (The MINIX Book)*. Pearson Education, Inc., Upper Saddle River, NJ, 3rd edition, 2006.