Re-Pintos: Revitalizing an instructional operating system

Levente Kurusa
Imperial College London
levente.kurusa15@imperial.ac.uk

Abstract

Traditionally, teaching operating systems has been considered hard for a number of reasons. Most commonly the fact that operating systems require a hefty amount of knowledge in various disciplines of computer science that undergraduates usually have not yet acquired. However, over the recent years, various forms of specialized operating system kernels have been introduced, dubbed "instructional operating systems", that exist solely for the reason of teaching students how critical parts of these systems work. In this abstract, we look at one such system, namely pintos, and discuss what can be done to bring it up to date with recent advancements in hardware and make it more engaging for undergraduate and maybe, even for graduate students.

CCS Concepts • **Social and professional topics** → **Computer science education**; • **Software and its engineering** → **Operating systems**

Keywords education, operating systems, pintos, geekos, os161

1. Introduction

Teaching operating systems is hard. The vast amount of knowledge required to understand some concepts is generally lacking in computer science undergraduates. Most instructional operating systems have used different methodologies, with varying success, to hide unimportant intricacies of the underlying hardware and instead made sure that the task focuses on the important educational values that lie within these systems.

Modern operating systems also have to cope with the ever-changing field of hardware, however, instructional operating systems have to be kept simple and are thus often left behind. For example, there are only a few instructional operating systems that are designed for 64-bit architectures or are in a Symmetric Multiprocessing environment. This is in stark contrast with the fact that most modern operating systems have been running on such systems for years now. This abstract briefly explores the current field of instructional operating systems, presents preliminary feedback from prospective students and goes in-depth into our motivation.

2. Motivation

We believe that hardware has recently undergone multiple rounds of extensive change in the recent years, however, instructional operating systems have largely remained intact. On the other hand, we have also conducted a survey of second-year computer science undergraduates who have just finished their first operating systems class that included the pintos project.

Despite the goal of the pintos project being as close as possible to the role of a real-world kernel, *most* of our respondents have said that they felt somehow disconnected from the real role of the operating system, the only thing they've run in pintos were the tests, citing lack of opportunity to run proper userspace applications. They also wrote that running such programs would have increased their interest and engagement in the course and the project. Interestingly, when asked about prospective new tasks, 70% found implementing the UNIX fork system call and copy-on-write mechanics for the VM an interesting task that they would have loved to work on, despite the fact that *all* of our respondents consistently rated the VM project the least favorable of all.

To our surprise, the second most popular "new" prospective task was implementing inter-process communication using unidirectional pipes [6] and POSIX-like message queues [5]. As part of the "Re-Pintos" project, we have chosen to implement the necessary groundwork required for this task, complete with a test suite to stress test the students' implementation on multicore systems.

An important reason we chose to improve upon pintos was to relieve course staff from the difficulties involved in moving to another OS project while keeping the practical part of the course itself up-to-date with recent changes in commodity hardware. Our work builds upon the original pintos kernel and keeps its extensively praised test-suite,

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. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SOSP'17 October 28–31, 2017, Shanghai, China

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-nnnn-nnnn-n/yy/mm. . . $15.00

DOI: http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

which has been cited by staff members as "saving countless hours of staff time". During the course of our work, we intend to keep the test-suite and update it with more tests to cover both the existing, expected behavior and to further test for various interesting behaviors that will arise from the work described in this abstract.

Given the results of the survey and the recent changes in the field of hardware, it is clear that *revitalizing* an instructional operating system has value worthy to be explored.

3. State of the work

We've begun with porting pintos [7] to the 64-bit version of its native architecture, x86_64. This involved auditing the entire codebase to check whether a pointer was cast to a type that is no longer suitable to hold the full width of the pointer. To our delight, the existing pintos codebase apart from the low-level components required little to none changes, the code was very well designed. During this audit, we also maintained consistency in extracting all architecture-specific code into an `arch/` directory to maintain backwards compatibility with the i686 architecture, and forwards compatibility with our planned ARM (specifically, the Raspberry Pi [2], since our institution has a large number of these readily available and students are already familiar with it after their first-year C course) port. The extra benefit of moving this code into a specific folder is that students do not need to know about the exact architecture the OS is running on. In our view, this emphasizes better design decisions rather than the creation of a design that is suitable for only one architecture. Having student code run on multiple architectures also has the benefit of teaching students that while a pointer may fit into a 32-bit variable (for instance, `uint32_t`) on one architecture (32-bit x86), it may not fit into such a variable on another (x86_64).

At the time of submission as an abstract, the implementation of the "inter-process communication" task is also in the works. Currently, a basic prototype pipe (specification of such is closely related to the POSIX pipes mentioned earlier) implementation is available but is intended to be used as the sample solution by instructors of the course. As it is expected, we endeavour to keep the test coverage of this sample solution high and we are also introducing new tests for existing tasks and features of the pintos operating system kernel.

4. Related work

Today's most well-known instructional operating systems are the pintos kernel (which we explore in this abstract), OS/161 [3] and GeekOS [4]. However, to the best of our knowledge, most of them are not updated to cope with modern commodity hardware that is usually 64-bit and multicore-aware. For example, GeekOS still uses a segmented memory approach, which *none* of the major operating systems do anymore and OS/161 doesn't run on real hardware - only in an emulated context. However, OS/161 is unique in the sense that it was updated in 2015 to support multicore systems in a symmetric manner, unfortunately the issue we see with OS/161 is that it is running on its own CPU architecture that has been designed specifically for it, System/161, thus avoiding much of the "aha!" moment that follows when an operating system a student worked on, boots on their own machines. Pintos, and by extension, Re-Pintos runs on recent x86 hardware and can be used to create a disk image that is also bootable via popular bootloaders readily available for Linux-based systems, for instance via GRUB [1].

5. Conclusion

In this abstract, we described our goals of refreshing an instructional operating system, pintos, originally developed in the early 2000s for 32-bit. single-core, i386 compatible PCs. Our work is focused on bringing this previous work to the present, by embracing 64-bit and introducing multicore support. We hope that refreshing an existing instructional operating system to run on modern hardware will enhance the student experience and increase the educational value of the undergraduate operating systems class, all while preserving years of experience teaching with the project. As this abstract describes ongoing research, it is possible that adjustments are made, however, our goal of enhancing the value and student engagement is assumed to remain intact.

References

[1] `GNU GRUB`. *https://www.gnu.org/software/grub/*.

[2] `Raspberry PI`. *https://raspberrypi.org/*.

[3] HOLLAND, D. A., LIM, A. T., AND SELTZER, M. I. A new instructional operating system. *ACM SIGCSE Bulletin 34*, 1 (2002), 111.

[4] HOVEMEYER, D. GeekOS: An Instructional Operating System for Real Hardware. *Science* (2001), 1–13.

[5] KERRISK, M. Manual page of mq_overview(7). *http://man7.org/linux/man-pages/man7/mq_overview.7.html*.

[6] KERRISK, M. Manual page of pipe(7). *http://man7.org/linux/man-pages/man7/pipe.7.html*.

[7] PFAFF, B., ROMANO, A., AND BACK, G. The pintos instructional operating system kernel. *ACM SIGCSE Bulletin 41*, 1 (2009), 453.