The Journal of Open Source Education

An Active Learning Laboratory Manual for Teaching a Contemporary Undergraduate Operating System Course

Saverio Perugini[1], **Zhongmei Yao**[1], **Phu H. Phung**[1], **Andrew Rettig**[2], **Adam R. Bryant**[4], **Rusty O. Baldwin**[1], **David J. Wright**[3], **and John C. Gallagher**[4]

1 Department of Computer Science, University of Dayton, 300 College Park, Dayton, Ohio 45469–0280 USA **2** Department of Geology and Environmental Geosciences, University of Dayton, 300 College Park, Dayton, Ohio 45469–2364 USA **3** Learning Teaching Center, University of Dayton, 300 College Park, Dayton, Ohio 45469–1302 USA **4** Department of Computer Science and Engineering, Wright State University, 3640 Colonel Glenn Hwy, Dayton, Ohio 45435 USA

DOI: 10.21105/jose.00162

Software

- Review
- Repository
- Archive

Submitted: 12 November 2021
Published: 14 December 2022

Summary

We present a laboratory manual for use in teaching an undergraduate course in operating systems. The manual contains a set of labs (i.e., active-learning exercises), which are categorized into modules, that can be completed individually or in a team. Instructors can make use of these labs as they see fit in their teaching activities. The plug-and-play nature of the labs within each module allows instructors the ability to craft a tailor-made operating systems course. The manual offers significant value especially since developing lab and project plans is quite time consuming in computer science education. Moreover, the open-source access to the LaTeX source code of the entire manual allows instructors to customize the labs to suit the needs of their students and program, contribute additional labs to the modules, and/or develop new modules. This paper provides an introduction to the laboratory manual. Through the brief exposition of its modules in this paper, we hope to inspire both adoption and adaptation of this laboratory manual by instructors of operating system courses.

Statement of Need

A course in operating systems (OS) plays a central role in the curriculum of undergraduate degree programs in computer science. Unfortunately, many OS courses are out-of-date and in need of time-consuming revision since innovations in course content and related pedagogy are typically focused, understandably enough, on the introductory computer science sequence. The OS laboratory manual discussed here resolves issues of misalignment found between existing OS courses and employee professional skills and knowledge requirements. Instructors can make use of the labs in the manual as they see fit in their teaching activities. The plug-and-play nature of the labs within each module allows instructors the ability to craft a tailor-made operating systems course. The manual offers significant value especially since developing lab and project plans is quite time consuming in computer science education. Thus, this laboratory manual fills a much-needed void in the operating systems education community.

Introduction

The laboratory manual constitutes a contemporary operating system curriculum that involves four progressive modules (and a setup module):

- Setup Module
- Module 0: Fundamentals (Linux and C)
- Module I: Mobile OSs and Internet of Things
- Module II: Concurrency and Synchronization
- Module III: Cloud computing and Big Data Processing

The labs in these modules are active learning exercises that can be completed in a team within a learning environment (e.g., an OS studio) that can be reconfigured based upon the content and equipment needs of the module (e.g., *Raspberry Pi* or *BeagleBone*), thereby blending the boundaries of a lecture and laboratory.

To facilitate adoption of this laboratory manual, the preface contains a mapping from the chapters of a widely used (undergraduate) OS textbook (Silberschatz et al., 2018) (i.e., a traditional approach) to the labs in the manual. The open-source nature of the manual not only grants immediate access to the shared module content therein, but also enables instructors to adapt the labs to suit the needs of their students and program, contribute additional labs to the modules, and/or develop new modules.

This laboratory manual is, to the best of our knowledge, the first and only of its kind for a contemporary operating systems course that is, one that is aligned with the current landscape of computing practices and OS knowledge requirements companies have for their prospective employees.

Our use of this lab manual and the student data we have collected suggests this approach improves student engagement. This laboratory manual has the potential to be widely used both nationally and internationally. Promotion of this laboratory manual through avenues such as the *Journal of Open Source Education* will help attract attention to it. It may also inspire computer science faculty to realign their OS course content and develop similar approaches in their teaching activities and, thus, foster continued community engagement. In this context, its adoption has the potential to advance the ongoing development of innovative teaching practices as well as improve student retention.

Our lab manual was inspired by the SEED security labs https://seedsecuritylabs.org/ [Du (2011); SEEDbook].

Modules of the Lab Manual

Setup Module

The setup modules contains labs detailing how to setup the software and equipment used in the lab manual. These labs include a RaspberryPi setup lab, a VirtualMachine setup lab, and a secure copy lab (i.e., `scp`).

Module 0: Fundamentals (Linux and C)

This preliminary module involves labs that help students acquire the knowledge of Linux and C programming requisite to working through the labs in Modules I–III.

Module I: Mobile Devices and Internet of Things

This module contains four Internet of Things labs: one involving camera sensors and three involving moisture sensors. This module also contains a project that entails developing a simulation of a variety of the job/process scheduling algorithms and semaphore-processing activities of a time-shared OS. A GUI has been overlayed as a front end to the simulation and can be used by students to explore the job/process scheduling algorithms as well as the semaphore processing activities of a preemptive OS. The simulation can also be used by instructors to demonstrate these algorithms and activities in class. The graphical front end of the simulator is available for use at https://cpudemo.azurewebsites.net/#/ and described in detail in (Buck & Perugini, 2019a, 2019b, 2019c).

Module II: Concurrency and Synchronization

Module II involves a suite of labs that explore concurrency, threads, critical sections, race conditions, and the Communicating Sequential Processes (CSP) and Actor models of concurrency and synchronization. CSP is explored in Go and the Actor model is used in Elixir.

Module III: Cloud Computing and Big Data Processing

Module III involves a suite of labs that explore cloud computing and big data processing using Amazon Web Services (AWS).

Compiling the Laboratory Manual

Compiling the Entire Composite Manual

The laboratory manual is written in LaTeX and BibTeX and can be compiled with the standard packages that ship with a LaTeX installation (e.g., the The MacTeX-2021 Distribution available at https://www.tug.org/mactex/).

There is a `Makefile` in the repository for building the entire manual. A manual can be built using the following commands in sequence: `make clean; make`.

```
$ make clean
$ make
...
...
...
$ open OSlabManual.pdf
```

Compiling Individual Labs Separately

Each lab in the manual can also be compiled/built individually. Simply navigate to the directory of the lab you desire to build and enter `pdflatex main`. This will produce the file `main.pdf`, which will contain only the contains of the particular lab. For instance:

```
$ pwd
operating-systems-lab-manual
```

```
$ cd module1mobile/IoTcamera/
$ pwd
operating-systems-lab-manual/module1mobile/IoTcamera
$ pdflatex main
...
...
...
$ open main.pdf
```

The last three labs of Module I, which constitute the *BigSenseWorkbook* (version 2), involve the use of moisture sensors on a BeagleBone. Compiling these labs requires the installation of some extra LaTeX packages. In particular, these labs require the LaTeX `minted` package and the Python syntax highlighter Pygments available at https://pygments.org/docs/cmdline/ as well as the `pygmentize` script. Compiling these labs also requires passing the `-shell-escape` flag to `pdflatex`.

```
$ pwd
operating-systems-lab-manual
$ cd module1mobile/BigSenseWorkbookV2
$ ls
lab1/ lab2/ lab3/
$ cd lab1
$ pdflatex -shell-escape main
...
...
...
```

Adding New Labs to the Manual

Users can add new labs (and/or modules) to the laboratory manual. There is a template for creating a new lab in the directory `ADDING_NEW_LABS`. The process of getting started adding a new lab involves the following series of commands:

```
$ pwd
operating-systems-lab-manual
$ cp -r ADDING_NEW_LABS/lab_template new_lab_name
$ cd new_lab_name
$ ls
GUIDE      lab.tex    main.idx   main.out   pristine/  solutions/
README     main.aux   main.log   main.tex   rubric/    src/
$ vi lab.tex #  to develop the contents of the new lab
...
...
...
```

Larger Scope

This laboratory manual is part of a larger effort to foster innovation in the teaching of operating systems at the undergraduate level as part of a three-year NSF-funded IUSE (Improving Undergraduate STEM Education) project titled "Engaged Student Learning: Re-conceptualizing and Evaluating a Core Computer Science Course for Active Learning

and STEM Student Success" (2017–21). The project website is at https://sites.udayton.edu/operatingsystems/. Through the dissemination of this OS laboratory manual we intend to foster a community of practice among computer science faculty at multiple institutions who adopt or adapt the use of the manual, or elements thereof, for their own students and programs. A GitHub site for the community is available at https://saverioperugini.github.io/Teaching-Operating-Systems-Community-of-Practice/ and a GitHub repository for sharing OS learning materials (e.g., exercises and exams) is available at https://github.com/saverioperugini/Teaching-Operating-Systems-Community-of-Practice.[1] We anticipate the evolution of the GitHub site into a socially-engaging hub for computer science educators, specifically tailored for OS course content and discourse, where teaching experiences and best practices can be shared.

Related Work

Perugini has presented work associated with this laboratory manual in a variety of venues: conference tutorials (Perugini & Wright, 2018, 2019b), a demonstration (Buck & Perugini, 2019a, 2019b, 2019c), and a birds-of-a-feather session (Perugini & Wright, 2019a).

Acknowledgments

We thank Sumit Khanna for helping with the last three labs in Module I that together constitute the *BigSenseWorkbook* (version 2). We thank Paul Talaga at the University of Indianapolis for providing an external perspective on and evaluation of the labs in Module III. Multiple University of Dayton computer science students assisted in the development of this lab manual, including Kevin Brown, Daniel J. Illg, Patrick Marsee, and Matthew Weiler.

This material is based upon work supported by the National Science Foundation under Grant Numbers 1712406 and 1712404. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Buck, J. W., & Perugini, S. (2019a). An interactive, graphical simulator for teaching operating systems. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE)*, 1290. https://doi.org/10.1145/3287324.3293756

Buck, J. W., & Perugini, S. (2019c). *An interactive, graphical CPU scheduling simulator for teaching operating systems* (arXiv:1812.05160 [cs.OH]). Cornell University Library: Computing Research Repository (CoRR). https://doi.org/10.48550/arXiv.1812.05160

Buck, J. W., & Perugini, S. (2019b). An interactive, graphical CPU scheduling simulator for teaching operating systems. *Journal of Computing Sciences in Colleges*, *35*(5), 76–87. https://dl.acm.org/doi/abs/10.5555/3381613.3381622

Du, W. (2011). The SEED project: Providing hands-on lab exercises for computer security education. *IEEE Security and Privacy*, *9*(5), 70–73. https://doi.org/10.1109/MSP.2011.139

[1] Contributions to the laboratory manual must be made in the Git repository hosted in BitBucket (https://bitbucket.org/saverioperugini/operating-systems-lab-manual/) and not in the Git repository for sharing OS learning materials hosted in GitHub (https://github.com/saverioperugini/Teaching-Operating-Systems-Community-of-Practice/).

Perugini, S., & Wright, D. J. (2019a). Developing a contemporary and innovative operating systems course. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE)*, 1248. https://doi.org/10.1145/3287324.3293734

Perugini, S., & Wright, D. J. (2018). Developing a contemporary operating systems course. *Journal of Computing Sciences in Colleges*, *34*(1), 155–158.

Perugini, S., & Wright, D. J. (2019b). Concurrent programming with the Actor model in Elixir. *Journal of Computing Sciences in Colleges*, *35*(5), 108–111. https://dl.acm.org/doi/abs/10.5555/3381613.3381626

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating system concepts* (Tenth). John Wiley; Sons, Inc.