Operating Systems

Course Syllabus

Instructor:

Baochun Li, Professor, Department of Electrical and Computer Engineering, University of Toronto

Lecture hours (6 hours per week):

Mondays, Thursdays, and Fridays, 8am — 9:35am

Required Textbook:

Operating Systems: Three Easy Pieces, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, available for free download at **http://www.ostep.org**.

Recommended Textbooks:

Operating System Concepts, 8th Edition, A. Silberschatz, P. Galvin, G. Gagne, ISBN 978-1-118-11273-1, John Wiley & Sons, Inc.

Principles of Computer System Design, **An Introduction**, J. H. Saltzer, M. F. Kaashoek, ISBN 978-12374957-4, Morgan Kaufmann

Required reading: The BLITZ Documentation on the official course website

Official course web site:

https://oscourse.org (for posting lecture notes, videos, and lab handouts)

Overview

This course focuses on fundamentals of operating systems, including operating systems structures, processes and threads, CPU scheduling, process synchronization, memory management, file systems, I/O systems, and security. It has a rather substantial lab component, which involves simplified components of a toy operating system, called BLITZ. Lab assignments in this course assume that you have a working knowledge in a basic

programming language such as C, some basic skills of debugging, and using command-line tools in the Linux operating system (a UNIX variant).

Labs

The lab assignments in this course consist of a series of **four** programming assignments using the BLITZ system, each spanning one or two weeks (three hours of lab time per week). The first lab will not be marked. A handout describing each assignment will be available on-line in the official course website as we progress.

Depending on your programming skills, you may find that some of the labs are rather challenging, and be prepared to spend a substantial amount of time in these labs. The lab assignments do ramp up in their difficulty levels, and are designed with a difficulty level that can be challenging to an average student in the class, with a considerable amount of work per week. If you do find them challenging, you should try to spend more time accordingly.

It is worth noting that many of the lab assignments need a significant amount of time reading the BLITZ documentation and the code provided with the assignments, which is required for this course. Each assignment specifies the code that needs to be read and understood. This preparation must be done well before the lab due date, or else there will not be enough time to complete the lab. For most labs, it will be best to write most of the code beforehand and then use the lab to test and submit your code and, if needed, get help from the teaching assistants (TAs) or the online discussion forum. It is very important to start as early as possible on your assignments and avoid procrastination.

It is your responsibility to correctly identify and meet the deadline for each assignment. Assignments will not be accepted after their due dates, without prior arrangement with the course instructor. In general no deadline extensions will be granted to individuals except in severe circumstances.

Although the instructor will try his best to avoid using the material that has not yet been covered in the lectures, a lab assignment may sometimes use some knowledge, terminologies, or abstraction that have not yet been introduced. When this happens, you are encouraged to be a bit more independent and adventurous: visit the future chapters in the

textbook to find out the approximate meaning of those unfamiliar terms, and enjoy the learning process while gaining hands-on experience.

You may also use your home or laptop computer to work on assignments. We will be providing you with a **container image** that contains a BLITZ installation for you to get started. This container image relieves you from the burden of installing BLITZ directly on your computer, but you do need to install the **Docker Community Edition** on your operating system, by following instructions at:

https://www.docker.com/community-edition

The labs in this course are to be completed with *individual* work. Work submitted for credit must be your own work. It is one thing to discuss and compare approaches to a problem, but quite another to rely on some other student's work, or open-source code downloaded from the Internet, to obtain credit for a lab assignment. It is an offence to knowingly allow a copy of your work to be submitted by another person for credit. It is also an offence not to put in place protections to prevent your code from being copied without your knowledge.

Labs are worth 60% of the course marks, while the course project is worth 40%.

The BLITZ system is an integrated part of this course. Both midterm and final examinations will cover the material from both the lectures and the BLITZ system. Knowledge about the BLITZ system is mainly acquired by doing the labs, but will also be covered in the lectures.

Course outline

The following is a list of the topics, and the corresponding required reading in the **Three Easy Pieces** textbook. More detailed lists of specific sections will be given in the lecture notes, typically as the last slide.

Topic	**Required reading in the textbook**
Introduction to operating system concepts	Three Easy Pieces: Chapter 2
Processes	Three Easy Pieces: Chapter 4 and 5

Threads	Three Easy Pieces: Chapter 26 and 27
Thread synchronization	Three Easy Pieces: Chapter 28-33; OS Concepts: Chapter 6; Principles of Computer System Design: Chapter 5.5 and 5.6
CPU scheduling	Three Easy Pieces: Chapter 7-10
Main memory	Three Easy Pieces: Chapter 13-14, 17
Virtual memory	Three Easy Pieces: Chapter 15, 16, 18-23
File systems	Three Easy Pieces: Chapter 39-44
I/O systems	Three Easy Pieces: Chapter 36
Virtual machine monitors	Appendix B
Security (time permitting)	Lecture notes

Note: Coverage of material in this course sometimes go beyond the material in the suggested textbooks. All lecture material, including slides and lecture videos, will be provided in the course website.