COMPUTER SYSTEMS
FALL 2019
COURSE INFORMATION
PROF. KAPLAN

Be sure to read **all** of this document!

1 Introduction

When you learn how to program, you use a complex set of existing tools to compile and run the code that you write. What are those tools? How does the code you write get carried out? Even the simplest of programs (e.g., *Hello World*) triggers a number of complex mechanisms to complete its task.

This course seeks to reveal how a general-purpose computer system can execute the programs you write (and those that others write). The design and implementation of this type of system requires a thoughtful structuring of a number of *layers of abstraction*, each of which simplifies programming and handles complex problems automatically. The goal of a computer system is to *enable programmers to solve their problems*. That is, the programmer should be able to concentrate on the problem at hand, and not have to worry neither about the details of what the underlying system is doing, nor how it is doing it.

We will start at a low level—individual *processor instructions*—and build our way, layer by layer, to full individual systems, and then modern *distributed* and *cloud* layers that allow programs to use the massive computational and memory capacity of many computers. Along the way, we will see repeated problems and themes that recur at many levels. We will also address the problem of *measuring and comparing* systems for various characteristics (e.g., speed). By the course's end, although you will hardly know every detail of a computer system, you will have enough experience to learn about and reason through the details of any system and its various layers.

2 The topics

The basic topics are given below, roughly in the order that will we cover them. If you have no idea what some (or many) of them are, don't be alarmed—that is why you're taking this course, after all.

- **Instruction set architectures**: Programming to the processor's circuits.
- **Caching and the memory hierarchy**: How are programs and their data stored to maximize performance?
- **Allocators and garbage collectors**: What happens when a program needs more memory? And then doesn't?
- **Operating systems and runtime libraries**: Programming to large collections of pre-made code.

- **Virtual memory**: How can multiple programs share the computer's memory without clobbering one another?
- **File systems**: How can data be stored for later?
- **Virtual machines**: Wheels within wheels...
- **Distributed computing**: Clusters, map/reduce, and other ways of spreading your computation.

These topics are what we directly will be covering, but underlying it all will be the concepts of *interface*, *abstraction*, and *resource allocation*. Don't loose sight of that bigger picture.

This course will be project-intensive. Much of the material will seem easy enough to comprehend when presented in class, but the only way to understand this material thoroughly is to use it. In this case, *using* these ideas requires that you understand an existing implementation of a layer, and then modify or enhance it. Your projects will require you to understand existing code before you then write your own.

3 Lectures, labs, and help

Lectures and labs: This class will meet on **MWF** of each week, from **12:00 pm to 12:50 pm** in **SCCE A131**. We will occassionally use our class time as a lab to work on projects; be sure to bring your laptop for such days (which will be announced).

You are expected to be present for **all of the lectures and labs**. I will not teach material twice, so if you miss a class meeting, then you're on your own for whatever material was covered that day. If you must miss lecture or lab due to an illness, a curricular conflict (e.g., a Geology field trip), or an emergency situation, contact me and I will arrange to handle the problem. **If you have a extra-curricular conflict** with a lecture or lab—for an athletic event, for a (non-curricular) musical or theatrical performance, to depart early for or arrive late from a vacation, or for any other non-emergency—then **the choice is yours to miss or to attend**. If you choose to miss the class meeting, I do **not** need to know **why** nor even **that** you will be absent. You have elected, voluntarily, not to attend, and you must be prepared to obtain and learn on your own the material that you missed. I recommend that you choose to attend the class meeting when these conflicts arise. Do not underestimate the willingness of those who run extra-curricular programs to support and to accommodate your academic priorities.

Office hours and meetings: If you seek assistance, reinforcement, review, or other opportunities to discuss the course material or assignments, you should see me. There is a link on the course web page for scheduling a time to meet. Please use them; chatting with me outside of class is one of the reasons you came to a small college.

TA help sessions: In addition to the TA's who will be present during many of our Friday lab sessions during class time, there will be **evening TA help sessions**. Specifically, these will be held on **Tuesdays** from **7:00 pm to 9:00 pm**, in **SCCE C101**.

Email: Many questions simply do not need an in-person meeting, at least not initially. You should certainly feel free to send email to me with your questions or concerns. Be forewarned, however, that I do not typically respond to email quickly, so do not expect a quick turnaround.

4 Texts and materials

The textbook for this course is, *Computer Systems: A Programmer's Perspective, 3rd edition*, by Byrant and O'Hallaron.[1] All other tools for this course—all of the software and documentation—will be provided.

5 Assignments, deadlines, and extensions

There will be a number of programming projects. The deadline for each will be stated clearly on the assignment. **Late submissions may receive failing grades**. Turn in what you have, and do so on time.

An extension for any assignment **must be requested, in writing** (email counts as *writing*), **at least 48 hours prior to the deadline**. The determination as to whether or not a particular situation merits an extension will be made on a case-by-case basis. Scheduled events are **not** sufficient reason to warrant an extension. Rather, extensions are intended for unusual circumstances that prevent you from planning your time well in order to meet the deadline. Note that a sudden onset of illness or other emergency situation that occurs less than 48 hours before a deadline will be treated as a special case.

6 Exams

There will be **one mid-term exam** given during a regular class lecture hour; there will also be a **comprehensive final exam** given during the final exam period at the semester's end. The mid-term exam will be given during week 7 of the semester (with the specific day of that week to be announced); the final exam will be a 3-hour, scheduled exam. Its time and location will be announced when the Registrar's office posts the final exam schedule.

7 Grading

Your final grade will be chosen by my evaluation of how well you have mastered the course material at the semester's end. All of the work that you submit, as well as your participation in class, contributes to my impression of that mastery.

[1] ISBN-13: 978-0134092669

8 Academic dishonesty

You will be expected to do your own work on all assignments and exams in this course. While I encourage you to interact with your classmates and discuss the material and assignments, there is a limit to the specificity of such discussions. I seek to make that limit clear here.

It is acceptable to discuss any assignment for the class with a classmate. You may even discuss your approach to a particular problem, or review relevant material for a problem with another person. However, you **may not show another student your work, nor see another student's work. If in doubt, *ask me***. If you are unusure whether or not a particular kind of communication would rise to the level of academic dishonesty, then you should contact me immediately and find out.