CS 26 — COMPUTER SYSTEMS II
COURSE INFORMATION

Be sure to read all of this document!

1 The topics

This course is the second of a two-course series (the first of which is CS 16 — Computer Systems I). These courses examine how the component of a computer system are structured, how they interact, and how the problems of making those components function are solved. Each component can (and should) be broken into layers. Moreover, each layer is the result of combining operations and capabilities from the layer below, resulting in qualitatively different operations and capabilities. Somehow, by building one layer on top of another, complete, complex systems can be built.

In the first course, we examined the layering and function of hardware of a computer systems, particularly the CPU and memory components. We examined how basic software can be written and executed on a hardware CPU. Finally, we addressed the basic structure of the memory hierarchy that supports the CPU.

In this course, we will examine the fundamental structure and function of an *operating system*, which is the system that controls and manages the computer's hardware for all other software. We will also examine compilers and how they provide higher-level capabilities for directing a computing system. Finally, as time permits, we will examine more advanced CPU designs. Ultimately, we hope to complete a view of how a total hardware and software structure can provide a general-purpose computing structure on which applications can be built and run. Along the way, we will confront and solve the difficult structural and algorithmic problems that we encounter.

Below is a brief list of **many** (but not all) of the topics that we will cover, in **roughly** the order that we will cover them. Unlike the first course that built "upwards" from the simplest hardware components towards complete CPU and memory devices, providing the basis of simple software structures, this course will begin at a more abstract layer and proceed "downward."

- **Compilers:**
  - Grammars and parsing
  - Code generation
- **Operating Systems:**
  - Booting
  - CPU scheduling
  - Virtual memory
  - File systems
- **CPU's:**
  - Pipelines

- Multi-core processors

This course will be project-intensive. You will work in pairs or small groups to build parts of a basic compiler. You will then use that compiler to create critical parts of an OS that will run on a simulated CPU. By building the critical parts of a working system, you will get a deeper sense of how the layers are created, why they are organized as they are, how some of the interesting algorithmic problems are solved, and how the pieces interact.

This course should be fun, as there is a great deal of hands-on experience with the material. It is also a great demystifying course, as you will have a much better understanding of the operation and principles behind the computers that surround us. Note, however, that it is a course with a great many details, as well as a course that is exceedingly cumulative. It will be critical that you stay on top of lectures and labs at all times.

2 Lectures and labs

This class meets on **Monday, Wednesday, and Friday** of each week, from **11:00 am to 11:50 am**. Lectures will be held in **Seeley Mudd 204**, while labs will be in **Seeley Mudd 007** (the UNIX lab). Although any given day may be used for lecture or lab, typically we will hold lectures on Mondays and Wednesdays, and labs on Fridays. I will announce, for any given class day, where you are expected to be.

You are expected to be present for **all of the lectures and labs**, and so missing either is strongly discouraged. I will not teach material twice, so if you miss a lecture or a lab, you're on your own. If you must miss lecture or lab due to an illness or other emergency situation, contact me and we will arrange to handle the situation. **If you have a conflict** with a lecture or lab—for an athletic event, performance, or other extra-curricular activity, or to depart early for or arrive late from a vacation, or 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** want to know *why* nor even *that* you are missing class. You have elected, voluntarily, not to attend, and you must be prepared to obtain and learn the material that you missed on your own. I, of course, 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 make accommodations for your academic demands.

I expect you not only to attend lectures and labs, but also to be attentive for them. The time will be best spent if it is interactive, and that requires that you be up-to-date on the class material, and that you be alert and prepared to participate.

3 Texts and materials

The texts for this course are optional. You may obtain one or both of them if you seek additional reinforcement or reference material. The texts are:

- *Operating Systems, 3/e* by *Deitel, Deitel, and Choffnes*, ISBN #978-0131828278.
- *Computer Organization and Design: The Hardware/Software Interface* by *Patterson and Hennessy*, ISBN #978-0123744937.

I have not ordered these texts at a local bookstore—you should be able to find copies, if you want them, for yourselves. Notice that this course will spend more time on the operating systems material than the CPU material, but both books are good references.

Any other essential material will be posted to the course web pages.

4 Assignments, deadlines, and extensions

There will be a number of labs, projects, and problem sets. The deadline for each will be stated clearly on the assignment, **down to the minute**. The assignment will also state the manner in which you are expected to submit or show your work. **Late submissions will 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.

5 Grading

Your final grade will be determined by a formula roughly like the one below:

- 60% labs/projects
- 30% final exam
- 5% participation
- 5% free bonus for staying up late and finishing the big projects

The final exam will be a scheduled, 3-hour final exam given during exam period. The participation portion of your grade may seem small, but it can be the difference between one grade and another.

6 Academic dishonesty

You will be expected to do your own work on all assignments and exams in this course except where explicitly noted on group assignments. 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.

7 The big picture

There will be so many details to remember and understand in this course that it is easy to miss the forest for the trees. Don't forget, now and then, to consider the bigger picture: that from one simple level of capabilities, you can create another, fundamentally different, more complex level. As you repeat this process, you develop something so complex from components that are so simple that it seems nearly impossible that the former could arise from the latter. Only because we have seen the progression, step by step, do we see how it is done.