CS 4400: Computer Systems

Course Details and Objectives
Spring 2023

Course Information

Description of CS 4400

The objective of this course is to help students bridge the gap between high-level programming and actual computer systems: processors, the memory hierarchy, operating systems, compilers, linkers, assemblers, networks, and more. Our basic goal is to understand how a computer works, so that as programmers we can make it work efficiently, correctly, and securely. Thus, this course is an introduction to computer systems from a programmer's point of view.

Instructor

Daniel Kopta, *Email:* `dkopta@cs.utah.edu` *Office:* MEB 3190A

Lectures

Lectures are Mondays and Wednesdays from 11:50am to 1:10pm in room HEB 2004.

Labs

When enrolling in this class, you also enrolled in a lab section. These are smaller, weekly meetings led by the TAs designed to give you practice with the concepts discussed in class and other useful instruction. Attend the lab section you are enrolled in.

Laptop Requirement

Per School of Computing policy, students enrolled in a CS class with a lab/discussion component are required to use their own laptop for the lab. Students are responsible for administering their own laptops, such as installing software and backing up data. For recommendations and further information, please see the official policy:

`https://handbook.cs.utah.edu/2022-2023/CS/Academics/laptop_policy.php`

Class Website

The class website is on Canvas at `https://utah.instructure.com`. It will contain all pertinent course info and materials such as lectures, announcements, updates, corrections, and grades. Students are required to check their email and Canvas regularly until final grades are posted.

Textbook The textbook for this course is *Computer Systems: A Programmer's Perspective* by Randal E. Bryant and David R. O'Hallaron, 3rd edition.

A recommended supplemental C programming book is *The C Programming Language* by Brian W. Kernighan and Dennis M. Ritchie.

Coursework

Grading

Your grade for this course will be determined by the following:

Assignments	45%
Exams	45%
Quizzes	5%
In-class Participation	5%

If X is your overall course score, letter grades will be assigned using the below scale. Scores will *not* be rounded.

	$90 > X \geq 87$ B+	$80 > X \geq 77$ C+	$70 > X \geq 67$ D+	
$100 \geq X \geq 93$ A	$87 > X \geq 83$ B	$77 > X \geq 73$ C	$67 > X \geq 63$ D	$60 > X \geq 0$ E
$93 > X \geq 90$ A-	$83 > X \geq 80$ B-	$73 > X \geq 70$ C-	$63 > X \geq 60$ D-	

Assignments

The assignments make heavy use of C, Unix, command-line tools, and the x86-64 architecture. Students not currently fluent in any of these topics should not panic, as this course will cover them in more detail throughout the semester. However, students should be prepared to learn some of the C programming language on their own, for which the Kernighan and Ritchie supplemental text may be useful. All work must use an x86-64 processor running a Unix OS. Code must be written in C11 standard C — nothing else will work. Unless explicitly noted otherwise, grading of assignments will be done using CADE Lab 1 machines using the default path installed gcc. Students who choose to develop their code on any other machine must verify their solutions work on a CADE Lab 1 machine before turning it in. There will be no credit for programs that do not compile and run on a CADE Lab 1 machine, even if they run somewhere else.

Exams

Midterm exams will be given during the regular class time in the regular class room on Wednesday, Feb 8 and Wednesday, March 15. The final exam will be held on Thursday, April 27 at 10:30am in the regular class room. All exams are written exams.

Quizzes

There is a short quiz on Canvas associated with each lecture. In general, there will be two quizzes due each week except for weeks that do not have two lectures. Quizzes will not be accepted late.

Participation

The class participation grade will be based on attendance and completion of in-class responses using polling software during almost every class session. Half of the credit will be awarded for answering questions at all, and half will be awarded for answering correctly. Bring an internet-connected device to class.

Dropped Scores

The two lowest quiz, and four lowest participation scores will be dropped. The purpose of these dropped scores is to account for illness or other extraordinary circumstances preventing you from completing or attending them. Do not use your dropped scores simply to avoid doing the work. No exam or assignment scores will be dropped.

Getting Help

See the "Getting Help" page on Canvas for information about my office hours, TA help hours, discussion boards, etc.

Course Guidelines

Piazza

Piazza is used for questions and discussions related to the course. Students must use their first and last names (as they appear in Canvas) in their Piazza profile, such that the correct name is visible to the instructor and TAs on posts. Note that students may select to post anonymously, such that their name is not visible to classmates.

Late Work

Late assignment submissions will incur a penalty of 10% of the assignment's max value if submitted within the 24-hour period following the due date. This penalty increases by 10% per 24-hour period, up to three days. Work submitted more than three days late will not receive credit. An assignment is considered late if submitted any amount of time past the deadline, as measured by the submission system. Any delays caused by the submission system or corrupt/lost files is not an excuse for lateness. **Do not risk submitting at the last minute**. Late days apply to programming assignments only; other work, such quizzes, will not be accepted late.

College of Engineering Guidelines

For information on withdrawing from courses, appealing grades, and more, see:
`https://www.coe.utah.edu/semester-guidelines`

Students with Disabilities

The University of Utah seeks to provide equal access to its programs, services, and activities for people with disabilities. If you need accommodations in this class, reasonable prior notice needs to be given to the Center for Disability Services, 162 Olpin Union Building, 581-5020 (V/TDD). CDS will work with you and the instructor to make arrangements for accommodations.

Safety

The University of Utah values the safety of all campus community members. To report suspicious activity or to request a courtesy escort, call campus police at 801-585-COPS (801-585-2677). You will receive important emergency alerts and safety messages regarding campus safety via text message. For more information regarding safety and to view available training resources, including helpful videos, visit safeu.utah.edu.

Violence and harassment based on race, national origin, color, religion, age, disability, sex or gender (which includes sexual orientation and gender identity/expression) is a civil rights offense and will not be tolerated. If you or someone you know has been harassed or assaulted, you are encouraged to report it to the Title IX Coordinator in the Office of Equal Opportunity and Affirmative Action, 135 Park Building, 801-581-8365, or the Office of the Dean of Students, 270 Union Building, 801-581-7066. For support and confidential consultation, contact the Center for Student Wellness, 426 SSB, 801-581-7776.

Positive COVID-19 Tests

Any student who tests positive for COVID-19 must self-report via `coronavirus.utah.edu`

Learning Outcomes

By completing this course, students will be able to:

- explain the objectives and functions of abstraction layers in modern computing systems, including operating systems, programming languages, compilers, and applications
- understand cross-layer communications and how each layer of abstraction is implemented in the next layer of abstraction (such as how C programs are translated into assembly code and how C library allocators are implemented in terms of operating system memory management)
- analyze how the performance characteristics of one layer of abstraction affect the layers above it (such as how caching and services of the operating system affect the performance of C programs)
- construct applications using operating-system concepts (such as processes, threads, signals, virtual memory, I/O)
- synthesize operating-system and networking facilities to build concurrent, communicating applications
- implement reliable concurrent and parallel programs using appropriate synchronization constructs