Syllabus

Content

Formal Course Description
Learning Goals/Skills
Grading
Quizzes
Coding Assignments
Autograding Policy
Getting Help & Extensions
Absences
Use of Large Language Models, Code Assistants, and generative AI Models
Academic Integrity
Land Acknowledgment Statement
Sexual Misconduct Policy and Reporting Statement
Diversity Statement
Mental Health Statement
CS Cares Statement

Formal Course Description

This course is an introduction to System Programming. System Programming refers to writing code that prioritizes operating system support for programmers. A computer needs an operating system to manage its resources and provide support for common functions, such as accessing peripherals. There are two categories of "customers" that an operating system must support.

The first category is the community of users. We have all used computers, and you may recognize operating systems' functions such as creating folders (directories) and moving files around. These are examples of operating system support for users. User support is not the objective of this course.

The second category of users is programmers. This course addresses this category. When you write a program, it may have to interact with physical hardware (memory, flash storage, screen, network, etc.). For example, you may want to get input from a keyboard or mouse; you may want to read some configuration file stored on disk; you may want to output data to a screen or printer; or you may want to access a remote server across a network.

The operating system presents common interfaces for programmers to perform these functions. It also provides useful abstractions such as "tasks" (also called processes), "threads", and "semaphores". You can make the computer multi-task by creating new tasks or new threads. You can make these tasks coordinate and synchronize by using semaphores. You can tell the computer the order in which you want tasks to be executed by using a scheduling policy. Finally, you can manage computer memory by calling on the function for memory management.

Learning Goals/Skills

- Identify the basic components of an operating system, describe their purpose, and explain how they function.
- Write, compile, debug, and execute C programs that correctly use system interfaces provided by UNIX or a UNIX-like operating system.
- Be familiar with important UNIX system calls and invoke them correctly from within C programs.
- Describe the difference between programs, processes, and threads.
- Write a memory allocator or malloc (/malloc_hall_of_fame)

- Explain the meaning and purpose of process control blocks and other mechanisms that the operating system uses to implement the process and thread abstractions.
- Write, compile, debug, and execute C programs that create, manage and terminate processes and threads on UNIX.
- Define concurrency and explain the problems that may arise because of concurrent execution of multiple processes or threads. Explain how these problems can be avoided. Write code that avoids these problems.
- Define semaphores, mutexes, and other synchronization primitives. Also, explain their purpose, and describe their internal implementation.
- Describe possible problems that arise from improper use of synchronization primitives (such as deadlocks) and present their solutions.
- Write, compile, debug, and execute C programs that use UNIX synchronization primitives.
- Describe operating system scheduling and use UNIX interfaces to set and modify scheduling policy parameters.
- Define UNIX signals and signal handlers, and describe their use.
- Write, compile, debug, and execute C programs with processes and threads that interact by invoking and catching signals.
- Describe the concepts of I/O devices, files, directories.
- Explain the internal implementation of files systems and operating system I/O.
- Write, compile, debug, and execute C programs that use files and I/O on UNIX.
- Describe the machine memory hierarchy, describe its components such as caches and virtual memory, and explain memory management mechanisms pertaining to these components such as paging and segmentation.
- Write, compile, debug, and execute C programs that make use of memory management functions.
- Describe the protocols (such as TCP and IP) and interfaces (such as sockets) used for communication among different computers.
- Write distributed applications that communicate across a network.
- Understands and uses system security mechanisms to build secure programs.
- By the end of this course, you should be proficient at writing programs that take full advantage of operating system support.
- Can analyze how a specific security error (e.g. buffer overflow, file access control, page access control) impacts the Confidentiality, Integrity and/or Availability of data or service.
- Can identify multiple development practices (e.g. design reviews, code reviews, testing) as important practices to build secure programs.
- Can briefly describe well-known security case studies (e.g. network protocol implementation errors, CPU side channel attacks) and how they comprise the Confidentiality, Integrity and/or Availability of data or service.

Grading

The following is subject to minor changes:

```
Final Exam : 20 - 23% (see notes)
Quizzes: 15%

MP Programming Assignments : 45%
Lab Programming Assignments: 17%

Lab Attendance & other items: see below
```

We publish the following thresholds:

Points	Minimum Grade
[90 - 100]	A-
[80 - 90)	B-

Points	Minimum Grade
[70 - 80)	C-

All lab programming assignments are equally weighted. MP programming assignments are weighted by the time given to complete them. This means that three week MPs are worth triple one week. For grading, we will drop your lowest lab score, and two lab attendance grades. Some examples: you slept in late; your dog ate your homework; you destroyed the internet.

The final is take-home, 3 hours and will be available online. The exam can be completed at any time over a 3 day period during exam week. More details will be published at the end of the semester. Early exams will not be offered. The exam will be available on the first Sunday after reading day and closes 3 days later, Wednesday 11 pm.

Opportunities for a small amount of extra credit may be offered and will be annouced during the semester.

Grading issues should be raised with your TA during section or by email. Missing scores need to be reported within 3 days of being published.

Regrades

At the end of the semester there will be a last chance regrade option for three weeks of machine problem or lab grades. To be able to take advantage of this opportunity you will need to have a "perfect" (we define) attendance grade after the drops.

When will this start? For Fall semesters, it will start around the Thanksgiving week. for Spring semesters, it will be around the middle of April.

Can we use the autograder at this time? Yes! During the regrade period, the autograder will open for every assignment. You will be able to use one pre-deadline autograder run per day on each assignment (i.e. you can run the autograder on different assignments on the same day). Running the autograder on an assignment does not use a regrade.

How do I specify what assignments I want regraded? Regrades are determined by a file released during this period where you indicate what assignments you want to use your regrades on. We trigger deadline autograder runs for only these assignments, which determines the final grade.

More info? We will announce the regrade policy on the course forum later in the semester.

Please note that the score for each regrade will be capped at 90%. For the assignment's final grade, we will **automatically** use your higher score between the regrade score and the original score.

Lab Attendance

Chicago students will have their own online lab (arranged with the TA). Urbana-Champaign students have in-person labs in Siebel. Labs are required and there is also a small reward for in-person attendance: Your final exam contribution (23%) can be reduced to 20% by attending all labs. Partial attendance of in-person labs will be prorated. For example, if after allowing for 2 drops, you attended 6 of 9 in-person labs (11 labs in total after drops **in real**) then your final exam will be worth 21%, and you will have 2% of full course credit.

What this means: If you choose to complete this course virtually, or are sick and unable to attend labs, or do not wish to be in close contact with other students you can still succeed and earn a high grade.

For those that need it: There is still a small carrot to encourage you to leave your dorm and make it to Siebel! But don't get anxious about loosing points if you can't attend.

Quizzes

Each quiz covers approximately 2 weeks of content. They can be completed multiple times (highest score is used) at any time. They are due by last day of instruction (i.e. before reading day). No quizes are dropped; you should complete all 7 of them.

Quizzes will be online using Prairie Learn (https://prairielearn.org). Please see the Quiz topics page for more information.

Coding Assignments

There are two different types of coding assignments in this course labs and machine problems.

Labs are primarily teaching exercises designed either to prepare you for the machine problems or to allow you to explore a topic of systems in a hands on manner. Labs may be done in a collaborative manner. You are allowed to work with each other so as to learn in the best manner for you. This can include sharing code, debugging each other's code, and discussing the assignment at any level. You still may not publicly publish either your solutions or our code. Finally you should have a comment block with any students or other resources you used in the completion of your lab assignments. This block will not be used for grading but is needed the same way that citing your sources is required when writing a paper.

Machine Problems are different. While they are a key learning experience MPs are also a key assessment of your skills. These are solo exercises and you must work alone. For MPs, you may not share code in any way. This includes show, share, email, or debug each others code. You may have high level discussions with each other on the ideas in the assignments but that is it. You are responsible for keeping the code for you machine problems private this includes not letting people view your code. You may not publish your code on any open website.

No late submissions will be accepted.

Autograding Policy

You walk into the investor meeting ready to show your demo. You ship your code ready for a million Internet of things. You deploy your code to the Internet backbone. It had better compile and be functional.

Forgot to commit or your committed code that does not compile? Zero. The basic headline is that you're not in Kansas anymore (to quote Dorothy). Don't leave it until the last minute.

There are two kinds of autograder runs:

- Pre-deadline runs: **You are responsible for starting these.** Our team has worked really hard to improve our grading system and make it more reliable and flexible. Now, you can schedule your pre-deadline autograder runs using the on-demand grading system, that you can find on the `Assignments` page! You have to log in with your GitHub Enterprise account. Assignments will become visible on the web app as we release them. You will get one AG run a day which you can trigger at your disposal. Please be careful in using these. **These pre-deadline runs do not roll over.** We recommend that you develop and work on your assignments every day to make the best of this system. Once you click "Grade Now", your code will start getting tested on our grading machines. You can expect to see feedback in your CS 341 repository's `_feedback` branch in a few minutes. In rare circumstances, the grading process might fail (if your code made our Docker containers crash). In this scenario, there will be no visible feedback. You should make a private post in the course forums and we will deal with this on a per-student basis. Use these runs for feedback as you work on the assignment. **Students are 100% responsible for the pre-deadline runs and they have no effect on your grade for the assignment.**
- Deadline runs: These will be triggered by us and the grades you get on these will be counted towards your final grades. The results will show up in the `_feedback` branch as usual.

Labs:

- Released every Wednesday
- Pre-deadline runs: Available every day from Wednesday to Wednesday (you will get to start these!)
- Deadline AG run on Wednesday at 11:59 pm

MPs:

- Released on Mondays (for multi-week MPs, the entire assignment is released at once)
- Pre-deadline runs: Available every day from Tuesday to Monday (you will get to start these!)
- Deadline run on Monday at 11:59 pm

We will test your code on a multi-core machine; testing on your own laptop is insufficient. Don't be surprised if race conditions that go undetected on a different machine cause your code to fail. We encourage you to develop and test your code on your CS 341 VM, which is near-identical to the grading machine. We will attempt to give you some partial credit if your code passes the tests.

If you have a question about your personal autograder results after the final autograde run, then feel free to make a private post in the course forums titled "<assignment name> Autograde Question" with the folders/tags/labels `<MP or Lab>` and `<assignment name>` selected.

- It will take time to go through autograder questions, so please do not expect an immediate (or even same day or same week) response. We will try to answer you as quickly as possible.
- *You must show us your test cases first. If they are not close to exhaustive, we reserve to right to not answer your question.*
- We will not tell you the details of specific tests, beyond what the test description already says.
- These questions should be for "I have exhaustive test cases for *X*, so how am I failing *Y*?"
- Please mention your NetID in the post, so we can look up your code if needed.

Getting Help & Extensions

It's important to say physically and mentally healthy. Please see the links below for mental health resources.

- Office hours and peer mentoring for course staff will be posted in the forums. Office hours will start the second full week of the semester
- For unusual administrative items (e.g. sickness preventing you from working, DRES, 1% issues, problems with your TA) then please email (cs341admin@cs.illinois.edu (mailto:cs341admin@cs.illinois.edu)) and explain your scenario.

Absences

If you are in an exceptional situation – i.e. family emergency, sickness, please email (cs341admin@cs.illinois.edu (mailto:cs341admin@cs.illinois.edu)) and we will discuss with your situation on a case-by-case basis via the course admin (cs341admin@cs.illinois.edu (mailto:cs341admin@cs.illinois.edu)). For illness-related excuses, you will need a doctor's note of some kind verifying your illness. *No* illness-related excuses will be accepted without a dated Emergency Dean's note (http://odos.illinois.edu/community-of-care/student-assistance-center/) stating that you contacted the Emergency Dean.

Lab attendance credit will *not* be given unless you are physically present in lab (there is no point asking the admin for this if you are sick or away). Lab attendance reduces the fraction contribution from your final exam by a small amount. This is a small reward for attending.

Use of Large Language Models, Code Assistants, and generative AI Models

You may use Large Language AI Models (e.g. Github Copilot, ChatGPT, and other models) in this class with the following conditions-

- You must document their use in your code. Clearly describe which parts of your submitted code were developed with the help of generative AI.
- Document when you used an AI model as part of your development process and how well it worked. Specifically, in your code comments describe if you used an AI model to assist in your initial design, initial implementation, developing test code, debugging code or other software development process.
- Unless explicitly allowed by the assignment or exam, you may NOT use generative AI to complete (large portions of) assignments for you. Please see the Academic Integrity section below.

Academic Integrity

CS 341 is considered a critical step in your ability to create useful programs for your later classes and beyond. Unfortunately for grading purposes, a minority of students submit code that was created by others. Cheating is taken very seriously, and all cases of cheating will be brought to the University, your department, and your college. You should understand how academic integrity

(https://wiki.illinois.edu/wiki/display/undergradProg/Honor+Code) applies to Computer Science undergrad.

Rule of Thumb: If at any point you submit an assignment that does not reflect your understanding of the material, then you have probably cheated.

In the cases of labs, you are allowed to collaborate with others in the class, which includes detailed debugging and code sharing. All you need to do is put your partners' netid at the top.

This does *not* mean that will you share the same grade for the assignment, and each group member must submit a solution to receive credit.

EVERY MACHINE PROBLEM IS A SOLO ASSIGNMENT IN THIS CLASS!

This means you are not allowed to split the work with a partner. You are, however, allowed to discuss the assignments at a very high level. You can even share testing scripts! If you are found to have shared code work on any machine problem, you will receive a zero on that assignment and a 10% sanction in the course for each infraction where you are found to have used material that is not yours. Intentional obfuscation of code (e.g., adding dead code, no-op functions, non-standard formatting, and deceptive naming conventions) also constitutes a violation of the course policy.

You may not publish your solutions or leave them in "plain view", thereby leaving your programs open to copying, which constitutes cheating. If your code (or a variation of it) is found publicly accessible, then you will receive a letter grade reduction in the class for each infraction. Do not put your code anywhere besides your private course repository and take measures to ensure that nobody can copy your code, so that you are not charged with a violation.

In the case of quizzes in the CBTF, it is a violation of our course policy to access or provide access to the quiz material outside your registered window. Cheating at CBTF may also result in immediate failure of the course and further action by the college of engineering. If you are found to have done so you will receive a zero on the quiz and a 10% sanction in the course. This includes seeking descriptions of the questions from students who have taken the quiz, as well as any other method that would give you access to the quiz outside your scheduled time. If there is prep material provided in lecture or on the course forums, you are welcome to share that material freely.

We want you to get the most out your education, and cheating not only affects your peers, but also your level of knowledge and ability.

You may use AI and other code-assist tools - see AI statement above.

Land Acknowledgment Statement

Please see our important Diversity Statement here (/statements#LandAcknowledgementStatement)

Sexual Misconduct Policy and Reporting Statement

Please see our important Diversity Statement here (/statements#SexualMisconductPolicyStatement)

Diversity Statement

Please see our important Diversity Statement here (/statements#InclusivityStatement)

Mental Health Statement

Please see our important Mental Health Statement here (/statements#MentalHealthStatement)

CS Cares Statement

Please see our important CS Cares Statement here (/statements#CSCaresStatement)