# Syllabus

## Content



## Course Introduction

This course is designed to challenge you as a programmer and new computer scientist at the University of Illinois. Rather than the sand-boxed, contained, and simple problems of your previous courses that used significant scaffolding and pre-built libraries, you will be interacting with a much more complex environment: the entire system and even computing networks.

Further, you will need to fully understand how memory is allocated, used, and re-used within a process. You will also need to know how input and output can be optionally buffered between processes and files. In short, it is time to remove the training wheels of and instead fling open the doors, welcoming you to the big, wide world of computing.

Oh, and did we mention the challenge of concurrency and solving asynchronous problems, so that your program can take advantage of the multi-core CPU inside each machine?

## 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.

malloc
- Write a memory allocator or (/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.

# Resources

**this page**
**If you added late, check (/late_add) to get caught up.**

Coursebook
CS 241 Crowd-Sourced Wikibook (https://github.com/angrave/SystemProgramming/wiki) (https://github cs241/coursel
An introduction to system programming is Angrave's We also have
That provides html, pdf, and wiki versions. Angrave's mini searchable video-introduction and playful *system programming-in-the-browser*
mini lectures
(http://cs-
environment is at: (Firefox and Chrome recommended).

No formal textbook is required, but if you really want to buy a physical book, we recommend the following custom book Angrave put together in 2007:

```
Introduction to Systems Concepts and Systems Programming
University of Illinois Custom Edition
Copyright 2007 by Pearson Custom Publishing
ISBN 0-536-48928-9
```

# Tutorials

- Debugging Tutorial (/coursebook/Background#debugging-and-environments)
- Development Tutorial (/tutorials/development)
- Emacs Tutorial (/tutorials/emacs)
- SSHFS Workflow Tutorial (/tutorials/sshfs)

Shell Tutorial
- (/coursebook/Appendix#shell)

# Grading

The following is subject to minor changes:

```
Final Exam : 25%
Quizzes/CBTF: 25%

MP Programming Assignments : 35%
Lab Programming Assignments: 12%

Lab Attendance & other items: 3%
```

We publish the following thresholds:

| Points | Minimum Grade |
|---|---|
| [92 - 100] | A- |
| [82 - 92) | B- |
| [72 - 82) | 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 quiz score, lowest lab score, and two lab attendance grades. Some examples: you slept in late; your dog ate your homework; you destroyed the internet. At the end of the semester there will be a last chance regrade option for two weeks of machine problem grades. To be able to take advantage of this opportunity you will need to have a perfect attendance grade after the drops.

The 3-hour handwritten final exam is comprehensive and will test all CS 241 topics, including programming topics covered in the MP and Labs. Do not book your flights until the exam date is known. Early exams will not be offered. Conflict final exams will be offered if you have three exams in a 24 hour period, or you have an exam in another, smaller enrollment course at the same time.

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

# Exams

The quizzes are approximately every other week. Quizzes include a mix of multiple choice questions and at least one short coding problem. In the coding problem, you will be asked to create programs similar to the MPs and labs using a standard Linux machine with local tools ( `gedit vim gcc man make bash` (https://linux.die.net/man/1/bash) ). Runnable tests will be provided.

This course uses the College of Engineering Computer-Based Testing Facility (CBTF) for its quizzes (https://cbtf.engr.illinois.edu/sched/).

The policies of the CBTF are the policies of this course, and academic integrity infractions related to the CBTF are infractions in this course.

If you have accommodations identified by the Division of Rehabilitation-Education Services (DRES) (http://www.disability.illinois.edu/), please take your Letter of Accommodation (LOA) to the CBTF proctors in person before you make your first exam reservation. The proctors will advise you as to whether the CBTF provides your accommodations or whether you will need to make other arrangements with your instructor.

Any problem with testing in the CBTF **must** be reported to CBTF staff at the time the problem occurs. If you do not inform a proctor of a problem during the test then you **forfeit** all rights to redress.

The complete and authoritative CBTF detailed schedule is available at the CBTF website (https://cbtf.engr.illinois.edu/). Be careful: *start and end dates change from week to week*

Note the first quiz runs a day late because Jan 21 is Martin Luther King day.

| Quiz | First day | Last day |
|---|---|---|

| Quiz | First day | Last day |
|---|---|---|
| Quiz 1 | Tue, Jan 22 | *Thu*, Jan 24 |
| Quiz 2 | Mon, Jan 28 | Wed, Jan 30 |
| Quiz 3 | Mon, Feb 11 | Wed, Feb 13 |
| Quiz 4 | Mon, Feb 25 | Wed, Feb 27 |
| Quiz 5 | Mon, Mar 11 | Wed, Mar 13 |
| Quiz 6 | Mon, Apr 1 | Wed, Apr 3 |
| Quiz 7 | Mon, Apr 15 | Wed, Apr 17 |
| Quiz 8 | Mon, Apr 29 | Wed, May 1 |

The final exam will be scantron and paper-based and during the finals exam period. The date and time will be published by the university the final exam schedule The topics page when the exam schedule has been finalized. Check (https://courses.illinois.edu/schedule/DEFAULT/DEFAULT) (The topics of the quiz are listed at /quiz_topics)

## 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.

## Absences

If you are in an exceptional situation – i.e. family emergency, sickness, please email ( cs241admin@illinois.edu (mailto:cs241admin@illinois.edu) ) and we will discuss with your situation on a case-by-case basis via the course admin ( cs241admin@illinois.edu (mailto:cs241admin@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 (http://odos.illinois.edu/community-of-care/student-assistance-center/) note stating that you contacted the Emergency Dean.

## How to Succeed

Is this course hard? Yes, but you are bright. You're taking computer science at UIUC. Schedule the time to do it. The two big changes from CS 225:

1. Your code is now much smaller than the complexity of the system around it.
2. No, we will not debug your code for you.

With lecture content, one lab, one MP, and one Quiz/Midterm every week or two, it can get easy to fall behind.

How to fail: some students do not take the time to learn how to debug and reason about system code and then end up complaining that office hours is too busy before deadlines. If you can't write correct solutions, you need to learn *exactly* how C works, the *details* of the system calls you are using, learn *better debugging* skills and *reason* behind synchronization. Only then can you spot and fix mistakes. Hard? Yes. Impossible? No.

There are no shortcuts to mastery, but we can help you get there. We recommend the Feynman technique (https://www.youtube.com/watch?v=tkm0TNFc1eg) for learning. Remember, simply recognizing some text in a past exam or in the coursebook is not mastery of those concepts! Find ways to deeply engage your brain with the ideas by working actively with those ideas. Yes, this requires effort. Start the assignments early; expect to get stuck. Write code *slowly*; reason about every line of code you write. Experiment with your own mind hacks so that you have fun spending "time on task" with these materials.

# 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.

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 241 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 Piazza post titled "<assignment name> Autograde Question" with the folders/tags/labels `autograder` 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.

**NOTE:** If your Git author is not correctly set to your netid@illinois.edu, *your code will not be graded*. There are **no** exceptions. If you aren't sure if its set correctly, please make a private Piazza post with your net ID and someone will happily help.

# Academic Integrity

CS 241 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 applies to Computer Science courses (https://wiki.illinois.edu/wiki/display/undergrad/Academic+Integrity).

**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. All you need to do is put your partners netid at the top. This can include detailed debugging and code sharing.

**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% penalty in the course for each incident where you are found to have used material that is not yours.

Additionally, 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 assignment. For example, if we find your code on GitHub for one MP then you will receive 10% penalty in the course if your code is there for two you will get a 20% penalty. You also may not publish our code anywhere publicly and you will receive the same penalty for sharing our code. If you are confused on what it means to be "publicly accessible", then 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% penalty 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 Piazza, 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.

## Diversity Statement

UIUC is committed to equal opportunity for all persons, regardless of race, ethnicity, religion, sex, gender identity or expression, creed, age, ancestry, national origin, handicap, sexual orientation, political affiliation, marital status, developmental disability, or arrest or conviction record. We value diversity in all of its definitions, including who we are, how we think, and what we do. We cultivate an accessible, inclusive, and equitable culture where everyone can pursue their passions and reach their potential in an intellectually stimulating and respectful environment. We will continue to create an inclusive campus culture where different perspectives are respected and individuals feel valued.