Syllabus

CSCI 3411 - Operating Systems

This course covers the fundamental concepts of operating systems, focusing on ***resource management*** and ***abstraction***. This includes OS structure, processes and thread management, communication with peripherals (I/O), synchronization, deadlocks, memory management, Virtual Machines, cloud infrastructures, and abstractions for cloud computation. The workload for this class is heavy and programming intensive.

Welcome! Great ready for a ***deep dive*** into what's under the water.

Fall 2022 Class Time/Location:

- *Class*: Tuesdays/Thursdays @ 12:45-2:00 ET in Gelman B04. If we have to move class online, we'll use zoom.
- *Labs*: Thursdays @ 2:10-4:00, 4:10-6:00, 6:10-8:00 ET. If we have to move class online, we'll use discord for labs.

Makes sure you're properly preparing and presenting yourself at Office Hours.

Course Overview

Learning Outcomes

Objectives - In completing this class, students will…

- understand key concepts involving system resource management, organization, and abstraction
- understand how an OS manages and interfaces with hardware
- understand fundamental trade-offs integral to system design
- experience both development and experimentation in a real OS

Structure - This class is broken into two main activities: lectures and lab.

- Lectures will discuss the main concepts in systems with reference to implementation details where beneficial. A series of homeworks (all in C) will test student's ability to apply the courses concepts in a set of projects. The midterm and final exams test student's understanding of the lecture's concepts.
- Labs will help you jump into the xv6 kernel and learn how to write and understand code in key subsystems. You will have programming assignments related to lab lectures throughout the semester. In the second half of the class, you will have a large group project involving kernel programming.

Course Topics - The class will cover topics about the design and implementation of operating systems, roughly following this schedule:

- Class Introduction, OS definition and history
- System structure and protection
- System structure and processes
- IPC and threads
- Synchronization
- Deadlocks
- Scheduling
- Real-Time Scheduling
- Memory Management
- Memory Management II
- File Systems APIs and Abstractions
- File Systems Design and Implementation

Direct Instruction and Independent Learning

Each semester, you're expected to spend ***at least***:

- 70 hours in class, and in lab for direct instruction, and
- 150 hours out of class watching lectures, and working on your project and class assignments.

Covid Policy and Actions

This class will run in accordance to GWU's policies around Covid. Please attend to your regular tests, the mask requirements, and actions should you be exposed or symptomatic.

Course Prerequisites and Student Responsibilities

Prerequisites:

- Computer Architecture I, Software Engineering
- Systems Programming is a useful class to have taken, but not necessary. It is mainly useful as

it provides more experience and practice programming in C.

- Assignments for this class will be done in C, and will require kernel-level programming. Though we do not assume that you have kernel experience, if you aren't somewhat comfortable with C, you will have a very difficult time. Thus, a familiarity with C or a willingness to quickly learn and practice it is required. If you are not comfortable with C, at least go through Essential C and the exercises in the Course Material Section.

Responsibilities - Students must

- Conduct your online behavior in accordance with the class' *"Online Social Contract"*.
- Attend all classes unless you are sick or there is an emergency in which cases you must contact the professor via email before class.
- Interact, ask questions, and generally participate in class discussions.
- Attend all labs, and do work assigned therein.
- Complete programming problems assigned in lab, and all written assignments individually.
- Work productively as a group on an extended and difficult project.
- Complete all work in accordance with the academic honesty rules for the university and for the class.
- Work to digest and process all provided material including Piazza content (you ***cannot*** ignore Piazza).
- Contact the Professor if you are worried about completing any responsibilities. We'll work with you to come up with a plan to be successful in the class.

I also need your feedback for parts of the class that aren't working for you. Class evaluations are of limited use since they only are given at the end of the class. Please provide feedback using the linked feedback form in the class organization google doc.

Your role!

Provide feedback. Many aspects of the class are ***new*** and experimental, thus they might require modifications throughout the course of the class. Because of this, all students must take the contents of the section below on "***Professor and Instructional Staff Responsibilities***" seriously: if something isn't working for you, assume that the class needs a tweak, and contact the professor, or provide anonymous feedback. Gabe will take this feedback seriously.

Community and online social contract. As an increased part of the class is online, it is necessary that everyone conduct themselves respectfully and productively online. Please see the class' *"Online Social Contract"*. The class will have a zero-tolerance approach toward disrespectful conduct, or harassment.

Take responsibility for your own education. Much of the work for this class will be more group-oriented than in normal instances of the class. If you burden your partner with most of the work, you are setting yourself up for failure. If you take on your partner's responsibilities, you are setting them up for failure. The homeworks build on each other. The benefit of this is that by the end of the class you'll laugh at how easy some of the earlier homeworks are as you've massively leveled up. The challenge is that if you don't put in a genuine effort on ***each*** homework, you will find later homeworks increasingly intractable. If you don't put effort in for each homework, you'll create dis-proportionally ***more*** work for yourself in later homeworks. If you cannot carry your own weight, you will not be able to achieve the learning objectives for the class.

Also remember that your team members will change throughout the semester, and they happen to be the ***network*** you'll use to help get a job. Not impressing them hurts your future opportunities.

Class Technologies

How you comport yourself online in the class just adhere to our online social contract. For the class we'll use the following technologies for the specified purposes:

- ***Github*** - Assignments are retrieved from github classroom using a link provided on Piazza. They are submitted via `push` es to github. See the details on submission and on how to use github below.
- ***Discord*** - Find the link to the class' Discord server in the Google Drive. Office hours, informal discussion, and other online help will use various rooms associated with the OS Discord. The discussion on Discord is, by default, ***synchronous*** and ***transient***. Synchronous means that you might get an immediate response (if anyone is online), and can have a discussion. Transient means that the conversation will be lost to the brutality of the scroll-bar. This means that no-one in the class (including the instructional staff) are expected to see Discord discussion, in general.

  Note: we ***cannot*** guarantee super-fast responses. In general, we will guarantee that we will reply in a batch, once per day. Additionally, we reserve the right to not answer homework questions the day before it is due – procrastination is not encouraged.
- ***Google Drive*** - All class-private information will be placed into the class' google drive. This includes slides, the `qna` document, and the `class-information` document which includes all links for the class.

Please see the discussion about Academic Integrity below. A rule of thumb is that you should ***not*** post code on any discussion medium with reach beyond your group. We do ***not*** use blackboard.

Absences

If you need to be absent, please send me an email to let me know that you can't make it, and please let me know why. If you know you're going to be absent (religious holiday), try and let me know in advance, otherwise, let me know as soon as you can. I'll follow up and help you plan your involvement in the class (e.g. group projects). If you're sick, keep me updated on how you're feeling if you can.

Office Hours

You must properly prepare for and present yourself at Office Hours.

Course Material

Required reading. You *must* read through the material at the repo for class resources. This includes some intuition as to how to develop C, how to think about debugging and testing, and how to think about generating ***good code***.

Strongly encouraged (but optional) Text. This book is useful to help you understand the concepts that might not be clear from the lectures.

- Operating Systems Concepts, newest edition, by Silberschatz, Galvin, and Gagne. If you have another edition, it is probably OK.

xv6 code documentation. If you have trouble understanding the xv6 source code, or want to better understand it, I recommend the following resources:

- xv6 main webpage
- xv6 book
- The book is paired with the printed code (can be generated with `make print`)

You can fork the xv6 source on github. (This is the same code that is the foundation of most of the homeworks.) When you do, test that you can run it with `make` , and `make qemu` .

In the end, you ***have to get comfortable walking through the code, and understanding it***. Using a code indexing system will make this easier. `ggtags` or `ctags` for emacs does great, but there are corresponding technologies for nearly all editors.

If you're having trouble with C, here is a list of references:

- A good short refresher/introduction is Nick Parlante's Essential C, memory and pointers, linked lists, and linked list problems. These are reproduced from here. The Linked List problem sets are especially useful.
- Beej's Guide to C Programming is a comprehensive guide from simple to complex.
- If you're having trouble understanding what a specific declaration is in C, use the `cdecl` command, or, if it isn't installed, you can use the `cdecl` webpage.
- For those of you who know C, but want to learn how to write better C, one source is the Composite Style Guide.

It is ***your responsibility*** to quickly get up to speed in C, so please use these resources.

Assignment Submission

All of your homeworks for this class will be submitted via `github`. The ***last*** commit that you make to the repo will be graded, and the lateness penalties of that commit (see "**Late Policy**" below) will apply. Please do ***not*** make `push`es to your repo after the version you want graded. Please also note that it is ***necessary*** to issue a `git push` up to the `github` repo for submission. Local commits to your own repo do ***not*** update `github`, thus cannot be counted as submission.

All assignments are submitted on github by doing a push of your commits before the deadline. There are a huge number of errors that one can make, so I highly suggest that you do the following procedure. When you submit an assignment, I suggest that you follow this procedure:

1. `git status` to make sure that you have `git add`ed all of the intended files.
2. `git diff` and `git diff --stat` to make sure that you're committing what you think you are. Of note, check to see if you're committing any `printf`s that are intended for debugging, not for program output.
3. `git commit` and `git push` to upload your code to `github`.
4. Go to the associated webpage for the repository, and make sure that the most recent commit is present in your repository.
5. Use `git clone ...` to get a ***fresh*** version of the repository, build it, and run your tests. Students who omit this step ***get bad grades*** for very ***avoidable reasons***. Always make sure that your submission matches what you ***think you're submitting***.

Git Usage on Group Projects

We want to be able to use your `git` commit logs and diffs, along with partner feedback to assess work done in groups. We don't expect the work to be exactly equal between team members, but we do expect each group member to pull their own weight. Thus, we expect you to follow some simple rules when using `git`:

1. ***Don't rebase.*** Don't rebase your commits to clean up your commit history. I wanted to see your entire commit history, thus avoid squashing commits. If you don't know what this is, then you likely don't need to worry.
2. ***Commit with your GWU email.*** Set your git email address to your GWU email address. Failure to do this will mean that we see you doing ***no*** commits!
3. ***When pair programming, swap who's driving, and commit often.*** If you program together in your group, that's fine. However, you need to swap back and forth regularly. Set a timer.
4. ***Notate commits that don't compile.*** If you make a commit when the code doesn't compile,

include `DOES NOT COMPILE` in the commit message body. Many commits made when pair programming will use this.

Your commit logs ***may*** affect your grade in the class.

Grading

Only talk about your grades with Tim. Other instructional staff do not have access to your grades.

Grades will be assigned with the following proportions:

Category	Percentage
Homeworks	55%
Tests/Quizzes	30%
Participation	15%

- Homeworks include programming assignments, written responses, and a long final programming project. Each of these components ***can*** involve individual ***interviews*** about your code and design.
- There will be a midterm and a series of quizzes in the second half of the semester (instead of a final).
- For full credit, you must acheive 2 participation points each week. You will receive 1 participation point each week for attending Lab. Another point can be earned by actively participating in class or on discord.

If you cannot make it to any aspect of the class, or cannot meet any deadline, please let me know as soon as possible.

Homework Grade Modifications:

The following modifiers apply to your grade on ***all*** homeworks (but not the project).

- ***-50 points***: not removing debugging/logging/informational prints from your code (in industry, you ***cannot*** commit code with printouts).
- ***-50 points***: not following the naming requirements, or not structuring code into specific files as required in the specification. If you rename functions, and that causes your code to not compile ***with our test code***, see the next bullet.
- ***0 points*** (i.e., no credit): code does not compile as submitted. You must ensure it compiles with ***no warnings*** (using the `xv6` warning flags, or `-Wall -Wextra -Werror` if not in

`xv6`) as we will make compilations fail (with `-Werror`) if there are warnings. Forgetting to `git add` files is not an excuse. "A small change makes it compile" is not an excuse. You should always do a fresh `git clone` of your submitted repo, to make sure that it compiles and your tests work. Make sure it works in the VM using the version of the compiler we use (i.e. Ubuntu 20.04). If you don't test in this specific environment, ***your code will likely fail compilation***.

Your grade can be minimum ***0%***, but your maximum grade can be higher than ***100%*** (if there is extra credit). There is one exception:

- *-200 points*: If you cheat on a homework. You cannot get credit for work done by cheating, so you will get a zero on the assignment. In addition, you get this penalty. Instead of cheating, please come and talk to me if you're feeling desperate. Talk to me ***before*** you do anything rash that sets you up for failure. You ***cannot*** resubmit work that you cheated on (see "Late Policy" below). You cannot drop this homework grade.

I cannot emphasize this point enough: **The *only* way people have failed this class is by not trying or by cheating**.

Late Policy:

You are allowed **one** 48 hour extension which can be used on any homework deadline, but not homework resubmissions (see below) or the project. To request an extension you must complete this form within 24 hours of the due date. Note that for group assignments, all students must submit the form. If you do not use your extension by the end of the semester, then it will be converted to bonus participation points.

You get ***0*** credit for late work that is unexcused. However, once the grades are released, you can ***resubmit*** your code within ***seven days***, rounded up to the closest midnight. For example, if the grades are released at 2:30pm on the 8th, resubmissions are due at midnight on the 15th (i.e. `8 + 7 = 15`). You will get ***0*** credit for submissions after this date.

We will grade the last commit before the resubmission deadline, and you will the average of your original and resubmitssion grade. For example, if you got a ***43%*** on the initial deadline, and resubmit and get a ***68%***, your final grade (`g`) will be `g = ceil( (68 + 43)/2 ) = 56` .

Do ***not*** come to rely on the resubmission process. Homeworks are due every week or every other week, so it is very harmful to get behind on them, or to add ***more*** work to your schedule. Thus, our late policy is quite strict to discourage procrastination.

Please note that in terms of ***academic honesty***, you ***cannot*** share your code with your peers even ***after*** it is due.

Testing your implementation:

When you are required to test your code, please note that our test cases will test ***every*** edge case in the implementation. If you do not write code to test the edge-cases of the specifications, then you ***will*** lose credit. This is the largest reason why students get lower grades than they believe they deserve. If you believe that an error was made in grading, then please explicitly address why you believe your test cases are sufficient to test all cases. See the discussion about testing and debugging below.

Style and Code Craftsmanship:

Any homework graded by demo (i.e. the project) must adhere to style guidelines, and you must ***curate*** your code. It must be simple, readable, and clean. See this post. For some thoughts on how to make your code more readable, see the Composite Style Guide.

Academic Honesty

Just as you can do a google search for code online, it is trivial for us to do the same. We have caught numerous people cheating in the past in this way. If you feel pressured about an assignment, please come see me instead of cheating.

You are not allowed to collaborate on the homeworks and the lab assignments. The group projects require collaboration within each group, but no collaboration between teams is permitted. Please refer to the academic integrity policy linked from the course web page. This policy will be strictly enforced. If you're having significant trouble with an assignment, please contact me. ***If you have not signed the academic integrity statement for homework 0, you will not receive a passing grade for the class.***

Summary: I want to be very clear:

- If you see the code of another student (who is not in your group), you are cheating.
- If you ***share*** your code with any other student (not in your group), you are cheating. This applies even when you are no longer taking the class.

In the past, the only way that people have failed this class is to either not try (not do homeworks), or to cheat. There is almost ***no reason*** to cheat. Come to me, and we can discuss where you're at, and what you have to do to make it through this difficult class!

Justification: Cheating hurts your ***ability to compete for CS jobs***. Everyone came to college to learn. You learn by ***doing*** work to reinforce class ideas. Understanding ideas from class is nowhere near as effective at learning as ***applying*** those ideas in an implementation. Thus, you ***need*** to do your ***own*** implementation to be a competitive CS student in the real-world!

Cheating ***hurts your peers***. Courses that are graded on a curve have the side-effect that if someone inflates their grade not by hard work, but by cheating, your good grade (should the

Professor not catch you) will hurt the grade of all of your colleagues that put the work in.

Cheating also hurts the ***quality of the OS class*** in the long term, thus hurts ***all future generations of GW CS undergrads***. If you cheat, especially using previous year student's work, to the point where the Professor believes that they must develop new assignments every year, then those assignments will not be as strong. They will not meet learning objectives in a course like this. All future CS students will be less prepared for competing for the best jobs.

Please see additional information about university policies.

University Policies

Please find details about university policies regarding

- the use and reuse of class materials (e.g. videos),
- details on academic integrity,
- details about observing religious holidays,
- Disability Support Services accommodations, and
- information on counseling availability.