CS 4284 Syllabus

Overview

CS 4284 “Systems & Networking Capstone” provides an in-depth introduction to the principles and practices of operating systems. Particular emphasis is given to the topics of multiprogramming, process and thread management, memory management, including virtual memory, concurrency, including synchronization and deadlock, resource allocation and management, including scheduling, and storage management and file systems. Additional topics include inter-process communication, networking, and device management.

Rather than learning what an OS looks like from the outside and how to use its facilities, this course will show you what an operating system looks like from the inside. Whereas the prerequisite CS 3214 course looked at systems from the perspective of an application programmer using an OS, this course will look at systems from a system’s designer point of view.

The topics will be accompanied by a series of programming projects that will give you hands-on experience in building significant parts of a real operating system. All projects will be done in groups.

In addition to a series of structured projects, each group will engage in one open-ended project. Each group will create a poster for this project. These posters will be displayed at the VTURCS capstone symposium in April.

Staff Information and Meeting Times

Instructor: Dr Godmar Back

VTKW-2212

1-3046

Office hours:

TBA or by appointment, held in McB 122

Class website:

http://courses.cs.vt.edu/cs4284/spring2022

GTA: There is no GTA for this course.

Email: Send email to gback@cs.vt.edu or godmar@gmail.com

Class Meeting Times:

NCB 110A 12:30pm – 1:45pm T R

Regular class attendance is not enforced, but is strongly recommended. Subjects taught in class closely correspond to the concurrently run projects.

For each project, there will be a mandatory group meeting about 1.5 weeks before the project deadline.

Prerequisites

The formal prerequisite for this class consists of CS 3214: Computer Systems. Each student must prove that they have obtained a grade of C or better in those classes. *I recommend, however, that only students with a B or better enroll in this capstone course.*

Students who fail to meet the prerequisite requirements should drop the class.

The most important informal prerequisite consists of strong C programming skills. In particular, the projects in this class will require a solid understanding of pointer-based data structures.

Objectives

Upon completion of the course, students should be able to

1. Understand the basic structure and organization of a multiprogrammed computer system, including the distinction between user and kernel mode, the use of interrupts and context switches, runtime organization, application-binary interfaces and system calls, program linking and loading.
2. Understand the principles underlying concurrency and know how to use proper synchronization and deadlock avoidance techniques.
3. Understand the principles behind memory management, including user-level memory management, virtual memory management and paging.
4. Understand the principles behind CPU scheduling, including round-robin, priority-based, multi-level feedback queues, and weighted fair queuing based scheduling algorithms.
5. Understand how an OS provides protection to its applications and how it manages and virtualizes resources.
6. Understand how file and storage systems are constructed and what factors influence their performance.

At a higher level, I would like for you to take away an appreciation for the complexity of operating systems, and view this class as an example of how to learn managing complexity.

Format

The course work consists of a mix of lectures, structured programming projects and one open-ended project. The open-ended project's results will be presented in a final presentation. Each group will create a poster on their project.

There will be no other exams or other homeworks in this class.

Projects: There will be 4 structured and one open-ended project, where you will be working in groups of 3. Projects will be submitted electronically and grades will be posted electronically. Instructions will be posted on the class website.

Structured Projects

We will be using Pintos as our project infrastructure, an educational operating system developed by Ben Pfaff at Stanford University. Pintos is used at a number of universities in comparable undergraduate OS courses. We provide a baseline version of Pintos, and you will add various features to Pintos through the course of the semester. Pintos runs on the x86, and could boot on a PC, but we will use a virtual machine simulator to run it on the Linux remote login cluster. We will grade your submissions on those machines.

The projects in this class have a number of unique characteristics. First, they are pretty complex, although some of the work in CS3214 should have been good preparation.. Although a correct solution can be implemented in a total of about 2,500 lines of C code, getting your projects to actually pass the tests will require intense coding and debugging. We will provide you with tools and help.

Second, you will work in a group. Working in a group more closely resembles what you will encounter outside academia, but it requires trust and cooperation among group members. Note that your grade will be determined by the performance of your group. All group members will receive the same score for a project, unless otherwise specified. Do not expect to farm out work and receive credit for a partial solution. For more details, read the collaboration policy outlined later in this document.

Third, you will have to read a substantial amount of code *before* you can start writing the first line of code in your solution. We will guide you to the parts of the code you have to study, but do not expect to start coding right away. We believe this to be beneficial in two ways: first, this mirrors what you encounter outside academia. Second, we believe you will pick up good programming practices by reading well-written and well-documented code.

Fourth, only 50% of your grade in those projects is determined by passing the test cases we provide (all of which are public). 25% comes from a review of your code by me to evaluate its design. The remaining 25% of your score comes from a design document you are required to submit with your solution. Note that passing the tests is a precondition for receiving full credit for documentation – we don't give credit for documentation when a test shows that a solution you describe does not actually work. The design documents this semester will have to be written by **each student individually**. They consist of a free-form description of the overall submission's design (not just the part you worked on). Groups can – and should – discuss the designs, but each group member needs to be able to formulate it in their own words.

The original Pintos series was designed for a 10-week quarter and for students who had not taken a systems course before. We are using it for a 15-week semester course for students who have already learned about computer systems. We will start with projects right away, and use the remaining 5 weeks to complete the implementation of an open-ended project.

Open-ended Project

Each group will work on an open-ended project that extends the Pintos infrastructure in an interesting way. Examples of possible projects include:

- Implementing multi-threading for user processes
- Implementing inter-process communication facilities such as pipe
- Implementing security such as principals and permissions
- Implementing shared memory
- Implementing more advanced virtual memory, including sbrk() and anonymous memory
- Implementing a more advanced file system facility
- Add a device driver for a new device
- Adding 64-bit support
- Adding basic support for networking
- Development of software checking tools
- Experimenting with support for other languages (e.g. Rust)
- Implementing exceptionless system calls (e.g. io_uring)

Each group will create a poster presenting their project.

Late Policy

Accommodations beyond automatic late days (aka "extra late days"):

The automatic late day system is designed so that we do not have to respond to requests for additional late days. However, we will provide additional late days in 2 situations:

DoS/University accommodations: If you have family or other emergencies that prevent you from submitting assignments on time, please contact the Dean of Students Office (https://www.dos.vt.edu). They will make a determination as to what accommodations should be given, and inform the instructors of the classes in which you are enrolled of their decision. Our policy is to provide you with as many additional late days as the note from DoS advises. For reasons of consistency and fairness, we will not make any determinations about emergency accommodations; we will defer all such decisions to the DoS. If you have learning or other disabilities, please also see the section Students With Disabilities below.

Sickness policy: If you cannot complete an assignment due to illness, **you must tell the teaching staff how many days you need to catch up on any work you were not able to do because of sickness.** The deadline for the assignment in question will then be moved by this many days without counting against the late days described above. For group projects, sick days will be granted to the group as a whole for the project in question. **No Doctor's note is required or expected, but the honor code and the university policy on class attendance apply.** If the number of requested sick days is on an assignment is more than 5, or the overall number in the semester is more than 10, we will encourage you to coordinate with the Dean of Students as well.

Late Drop and Incomplete Policy

Less-than-hoped-for performance or realizing you have taken on too much work this semester, are not permissible reasons to grant course withdrawal requests after the drop deadline. (This policy applies only to drop requests that have to be approved by the instructor; in particular, it does not apply to the course withdrawals for six credit hours to which you are entitled according to college policy.)

I will not grant incompletes for this course unless truly extraordinary, unforeseen circumstances outside of your control are to blame.

Grading

I estimate that the contributions of the different portions to your final grade will be as listed below, but I reserve the right to adjust these weights as necessary:

50%	Structured Projects

30%	Open-ended Project
20%	Final Poster & Presentation

We will publish score distributions for the projects to give you an indication of where you are.

Since this class is a capstone/elective, I will not predict where the median grade for the class will be. Nevertheless, to achieve an A or B in this class, you should expect to produce working and well-documented solutions to both the structured projects and the open-ended project.

Auto-Fail Rule

Passing a capstone course is a requirement for graduation. It is important that all students passing the class have shown their ability for project work and their ability to produce actual functioning software artifacts. I will not assign a passing grade to students who cannot produce a fully working kernel by the end of the semester, defined as a kernel that passes 90% of tests for project 2. I will also not assign a passing grade to students who do not make substantial progress towards their open-ended project.

Collaboration Policy and Honor Code

The University's officially required honor code statement applies to this course, which is reproduced below.

The Undergraduate Honor Code pledge that each member of the university community agrees to abide by states:

"As a Hokie, I will conduct myself with honor and integrity at all times. I will not lie, cheat, or steal, nor will I accept the actions of those who do."

Students enrolled in this course are responsible for abiding by the Honor Code. A student who has doubts about how the Honor Code applies to any assignment is responsible for obtaining specific guidance from the course instructor before submitting the assignment for evaluation. Ignorance of the rules does not exclude any member of the University community from the requirements and expectations of the Honor Code. For additional information about the Honor Code, please visit: www.honorsystem.vt.edu.

The tenets of the honor code will be strictly enforced in this course, and all assignments shall be subject to the stipulations of the Undergraduate Honor Code. For more information on the Honor Code, please refer to the Undergraduate Honor System Constitution, located online at http://www.honorsystem.vt.edu/

The following policies regarding collaboration apply in this class.

1. All submitted work is expected to be the original work of the individual student or group unless otherwise directed by the instructor. Note the emphasis on “submitted” work – this includes work that is explicitly graded and work that may not be.
2. Projects are to be the work of the individual student or team as specified. You may discuss general concepts, such as software libraries, Internet resources, or class and text topics, with others outside your team. However, discussion of project solutions, specific code, or detailed report content is an honor code violation. All source material used in project code and reports must be properly cited.
3. For the projects you will team up in groups of typically 3 students. You may switch teams or form new teams, but only between projects. You may work with at most one group on a given project.
   Students must contribute equally to the project within a team. It is not acceptable for students to either not contribute to the project or not to let the other group members contribute equally to the project. Please bring any problems in this regard to the instructor’s attention early on.
4. You are required to read-protect your work on shared file space so students outside of your team will not have access. This includes your git repository on git.cs.vt.edu. Failing to do so is an honor code violation.
5. Borrowing code or hiring someone to perform the work for you is an egregious violation of the honor code. We will use plagiarism detection software such as MOSS to screen out students attempting to do this. We have access to project solutions that were created and submitted outside of Virginia Tech, as well as all solutions that were submitted in past quarters.
6. Discourse rules: you are not allowed to post code that is part of your solution on the forum. An exception is a single line if it causes a compile-time or runtime error. Posting debugging output, including backtraces, is ok.
7. You may not post detailed descriptions of your design or solution on the forum. You may not post answers to design document questions on the forum.
8. Not having read the honor code and its stipulations is no excuse for violating it.
9. If you have any doubt about what is and is not allowed, it is your obligation to ask the instructor beforehand.

Students with Disabilities

If you need accommodations because of a disability (learning disability, attention deficit disorder, psychological, or physical), if you have emergency medical information to share with the instructor, or if you need special arrangements in case the building must be evacuated, please meet with the instructor as soon as possible.