CS 140: Operating Systems
Syllabus

Winter 2014

Instructor

Name:	Nathan Dautenhahn
Office:	4307A Siebel Center, 201 N Goodwin Ave, Urbana, IL 61801
Office Hours:	MW: 9:00a - 10:00a or by appointment
Phone:	(xxx) xxx-xxx
Email:	dautenh1@illinois.edu

General Course Information

Website:	CS140.com
Lecture:	MW 4:15pm-5:30pm, Gates B01
Section:	F 3:15pm-4:05pm Gates B01
Credits:	3 undergraduate hours
Education Requirement:	This course does not fulfill any general requirements
Syllabus Flexibility:	This syllabus is provided as a starting point for the semester. Specific topics and duration of coverage may change as the semester continues.

Course Overview and Purpose

We love operating systems and want to show you why you should too! This course will express why we believe operating systems are fascinating and worth pursuing not only for course credit but also experimenting with on your own. In this course we will explore the core principles of operating systems design and implementation, including basic operating system structure; process and thread synchronization and concurrency; file systems and storage servers; memory management techniques; process scheduling and resource management; virtualization; and security. In our study of operating systems we will encounter the following themes:

Resource sharing: concurrent threads of execution **[enables]** multitasking
Programmability: abstractions and library support **[enables]** easier application programming

In this course we will explore operating systems design via the combination of readings, lectures, homework assignments, collaborative programming projects, in-class participation, and online interactions.

Readings and lecture content are the primary means of learning definitional concepts of operating systems design.

Homework assignments reiterate relationships and refine knowledge on core operating systems concepts, and also provide the primary means for test preparation.

Programming projects address the fact that operating systems are **complex**. They are large and impractical to understand without first-hand knowledge of underlying mechanisms. Therefore, to move beyond rote memorization of operating systems concepts we will implement pivotal operating system features in a real world operating system.

Collaborative projects address the fact that operating systems are complex and **large**, so large that it requires immense investment from multiple people to implement the entire system. Therefore, a core point of emphasis in this course is working in teams to learn how to collaborate on large software projects (sharing code), communicate effectively, coordinate design and implementation in a timely fashion, and write code that is sharable. Collaboration is a key component to all but one programming project and integrates operating systems concepts with team building characteristics.

Class participation and online interaction further asserts the necessity of collaboration when investigating operating systems design.

Upon completion of this course you will be prepared to contribute to the next generation operating system or enhance existing operating systems such as Windows, Mac OSX, Linux, *BSD, and many more. You will also be able to integrate your efforts into existing software development environments and be a contributing team member.

Prerequisites

This course assumes that you have a working knowledge of computer systems organization (UIUC ECE 290) and the basics of UNIX systems programming (UIUC CS 241). This course relies heavily on a working knowledge of the C programming language. Therefore, students should have prior experience in C. It is possible to learn C at the outset of the course, but it will be challenging to do so.

Course Themes and Objectives

We will take a hands-on approach to developing an intuitive and practical understanding of operating systems design. We will emphasize programming exercises that connect course readings and lectures to practical experience. We will also emphasize techniques for managing common issues associated with large scale software development. As a student in this class you will have the opportunity to learn to:

- Explicitly define and intuitively describe why operating systems virtualize hardware and how the operating system both makes it possible for many applications to share resources and to make programming easier for user space applications.

- Describe the differences between user and kernel level operating modes and implement communications interfaces in the kernel to support user level services, i.e., system calls.
- Describe and explain synchronization algorithms and utilize kernel synchronization mechanisms for traditional kernel operations.
- Describe and implement thread context management and switching using kernel synchronization and scheduling algorithms (i.e., priority scheduling and multilevel feedback queue).
- Design and implement a functional virtual memory system including virtual memory paging and resource sharing functionality.
- Design and implement persistent and correct file system mechanisms including functionality that utilizes a buffer cache, enables extensible files, and enables a hierarchical file name space (e.g,. subdirectories).
- Collaboratively develop large complex programs in C utilizing standard operating systems development toolchains including a debugger (GDB), build system (GNU Make), and a version management system to both track and share code modifications (Git).
- Evaluate and analyze multiple competing approaches to common operating systems design choices using micro and macro benchmarks.
- Read, interpret, and examine code that you did not write.

Materials

Required Text: Silberschatz, Abraham, Peter B. Galvin, and Greg Gagne. Operating System Concepts. Vol. 8. Wiley, 2013.

Course Communications: We will use Piazza for communicating on course topics. Programming assignments will be posted on Piazza and on the course website. Piazza is an excellent resource for facilitating discussions and question and answer sessions. We anticipate that many questions about any course issues will be brought up on Piazza so that your fellow students can benefit from what you are learning.

Coding Resources:

- Kernighan, Brian W., and Dennis M. Ritchie. The C Programming Language. Vol. 2. Englewood Cliffs: prentice-Hall, 1988.
- Stevens, W. Richard, and Stephen A. Rago. Advanced Programming in the UNIX Environment. Pearson Education, 2013.

Git Version Control: http://git-scm.com/

Lab Equipment: You will find the institutional (UIUC CS) lab equipment to be prepared for completing assigned programming projects. Each programming assignment will be completed in a simulated x86 environment (Qemu or Bochs).

Class Sessions

Class sessions will be comprised of traditional lecture materials, one to two short ungraded quizzes (individual and group), short discussions on the trade-offs of competing solutions to common operating system tasks, and a small amount of reserved time to answer questions about programming projects. You can expect that class discussions will cover material not explicitly in the book or

other materials, of which you will be responsible for on the exams. Developing a culture of questioning, analyzing, and discussing operating systems is a key goal of structuring lectures in this way.

Assignments

Time Expectations

The programming projects involve developing practical understanding of operating systems concepts and are extremely demanding! These assignments should be started the day of being handed out and will take approximately 15 to 20 hours each. Do not underestimate the time required to complete these assignments. In total, previous course students reported spending approximately 12 to 15 hours a week on this course including lecture attendance.

Programming Projects

Projects are the core mechanism for learning in this course. Operating systems are complex and large pieces of software. Transferring knowledge from theoretical or conceptual models to real understanding requires hands-on experience. You will be evaluated on your ability to convert the conceptual knowledge from course readings and lectures to operating systems implementations that mirror existing real world functionality. In these programming projects you will be implementing parts of the Pintos operating system.

Projects will be completed in teams of up to 3 people. The teams will be assigned by the instructors at their discretion and will differ for each project. In this way you will have the opportunity to work with several different people and perspectives during the course of the semester. We suggest that teams attempt to use pair programming [http://en.wikipedia.org/wiki/Pair_Programming] in an effort to improve coding efforts; however, you will not be graded on this suggestion.

Project grading will be focused on functionality of the code when tested by our test suite. The tests are designed so that they isolate as much program functionality as possible, thus, implementation efforts can be done in small chunks with measurable outcomes. Projects will be graded based upon functional correctness, design documentation and coding style, performance evaluation on real programs, and peer reviews of the code. Each of these are detailed as follows. Additionally, we describe at a high level each project.

Code Functional Correctness Against Grading Tests

Functional correctness will account for 50% of the project grade.

Design Document

Each project will require a design document to be turned in with the code. We provide a design template. The design document will require information about the following aspects of the project: data structures, algorithms, synchronization, and rationale for the design. The last section of the

design document will survey you about the project in general in an effort to obtain feedback on the real challenges associated with the project. The design document is worth 25% of the project grade.

Performance Evaluation of Implemented Code

A critical aspect of operating systems is to empirically compare differing design choices as it relates to performance criteria. As such each programming assignment will require the team to execute an evaluation of their implementation and present results in both written and graphical form. The evaluation will account for 15% of the project grade.

Code Peer Review

Practical operating systems development environments require extensive ability to both document your own code and read others' code. Therefore, each project submission will be reviewed individually by at least two non-team members in the class. Peer reviewers will be required to fill out a set of rankings against a rubric that we pass out. The peer reviews will also include a two to three paragraph description commenting on two of the following: design decisions and problems, coding errors, comment support, and performance expectation from implementation. The peer review will contribute 5% to the project grade of the code being reviewed as well as be graded by the instructors for an individual 5% of your individual project grade.

Project 1: Threads and Synchronization

Project 1 requires you to enhance a minimally functioning threading system. The goal of this project is to gain a deeper understanding of synchronization, scheduling algorithms, and interrupt handling. This will be the only project accomplished without a team.

Project 2: Enabling User Level Programs

This project requires you to develop the system interface to user applications, namely system calls. It requires you to develop several system calls to enable user programs to interface with the file system. The system call implementation requires you to also learn how the calling convention work for x86 system calls so that you can perform parameter passing to the system.

Project 3: Virtual Memory

This project requires you to enable the memory system by enabling virtual memory, including adding paging support, stack growth, memory mapped file support, and protect user level pages while in use by the kernel.

Project 4: File Systems

This project requires you to enhance the file system support in Pintos. Namely, you will implement a buffer cache and integrate it into the existing file system, add support for extensible files and subdirectories, and manage synchronization to keep everything correct.

Exams

There will be two exams throughout the course. The first will be a one hour exam in the middle of the semester and the second will be a 2 hour comprehensive exam during the university specified final examination time. The midterm will be worth 15% of the grade and the final exam will be worth 25% of your final grade. Exams are worth a total of 40% of your final grade.

Homeworks

There will be two homeworks throughout the course, each worth 2.5% and count up to a total of 5% to the final grade. The primary goal of the homework is to provide questions relevant to exam material; therefore, the homeworks will **not** be graded for correctness but rather for completion.

Grading

Homeworks	5%
Programming Projects	55%
Exams	40%
Total	100%

Grades will be given based upon a criterion-reference score according to the following scale:

$A+$	97%	A	93%	$A-$	90%
$B+$	87%	B	83%	$B-$	80%
$C+$	77%	C	73%	$C-$	70%
$D+$	67%	D	63%	$D-$	60%

Note that the minimum grade per range may be lowered but never raised.

Course Policies

We will follow all policies in the Student Code (http://admin.illinois.edu/policy/code/).

Professional Etiquette

We expect all of your interactions to be positive and never derogatory to anyone. We anticipate personal differences, but as you interact with others on the discussion boards, within your projects, and in-class, we expect common courtesy and never condone offensive behaviors.

Attendance and Online Interaction

You and your peers will benefit from your presence at lectures. Lectures may present material (via questions and discussions) that will not be presented any where else, and they will also create unique learning opportunities due to in-class activities; however, attendance will not be tracked. Online interactions will similarly not be graded, but provide a method for continuing discussion outside of class.

Accommodations

If you require any special accommodations please contact the instructor immediately. All accommodations will follow the procedures as stated in Article 1-110 of the Student Code (http://admin.illinois.edu/policy/code/article1_part1_1-110.html).

Academic Integrity

The integrity of **your** work is a precious commodity. Any violations will addressed according to Articles 1-401 through 1-406 of the Student Code (beginning at http://admin.illinois.edu/policy/code/article1_part4_1-401.html). Unless otherwise noted assume that you are to work alone, copying or sharing code is not permissible between groups. Group work will clearly state members who contributed to the effort. Discussion of specific challenges and or design issues are encouraged and will even be facilitated in-class; however, code must be isolated from other student projects. If you obtain information from another source in any way you must cite it.

Late Assignments

Assignments can be turned in late at the cost of 1% of the final grade for each hour late.

Course Schedule

Week of	Monday	Wednesday	Friday
Jan 6 - Jan 10	Intro Text: Ch. 1-2	Threads & Processes Text: Ch. 3-4	HW 1 Section Threads Due: Lab 0 (setup)
Jan 13 - Jan 17	Concurrency Text: Ch. 6-7, Birrell	Scheduling Text: Ch. 5	no section
Jan 20 - Jan 24	MLK Day	Advanced scheduling	HW 2 Section Userprog Due: HW 1 (Threads) Add/drop deadline (5pm)
Jan 27 - Jan 31	Virtual Memory Hardware Text: Ch. 8	Virtual Memory OS techniques Text: Ch. 9	no section
Feb 3 - Feb 7	Synchronization	Memory allocation	Midterm Review Section Due: HW 2 (Userprog)
Feb 10 - Feb 14	Midterm Quiz	Linking	HW 3 Section VM
Feb 17 - Feb 21	Presidents' Day	I/O & Disks Text: Ch. 12-13	no section
Feb 24 - Feb 28	File Systems Text: Ch. 10-11	Advanced File Systems	HW 4 Section Filesys Due: HW 3 (VM) Course withdrawal deadline
Mar 3 - Mar 7	Networking Text: Ch. 16	Protection Text: Ch. 14	no section
Mar 10 - Mar 14	Security Text: Ch. 15	Virtual Machines	Final Review Section Due: HW 4 (File system)

Final Exam Friday, March 21 12:15pm - 3:15pm