CMSC 23000
Operating Systems
— *Tentative* Syllabus —
Subject to change

Department of Computer Science
University of Chicago

Winter 2011 Quarter

Dates: January 3 through March 19, 2011
Lectures: TuTh 4:00-5:20 in Ryerson 276
Website: `http://www.classes.cs.uchicago.edu/archive/2011/winter/23000-1/`

Lecturer: Borja Sotomayor
E-mail: borja@cs.uchicago.edu
Office: Searle 211-A
Office hours: Open door policy (see page 6)

TA: Allan Espinosa
E-mail: aespinosa@cs.uchicago.edu
Office: Jones 209

Contents of this Document

Course description

This course provides an introduction to the basic concepts and techniques used to implement operating systems. Topics include processes and threads, interprocess communication and synchronization, memory management, segmentation, paging, linking and loading, scheduling, file systems, and input/output. The course will revolve around the implementation of an x86 operating system kernel.

The learning goals of this course are for students to be able to...

1. Use process management and communication primitives to write multiprocess and/or multithreaded code.
2. Use synchronization primitives, such as semaphores and locks, to prevent race coditions and deadlocks in their code.
3. Implement significant components of an operating system, such as system calls, memory management, and file systems.
4. Read and understand source code from other operating systems.
5. Develop software collaboratively through the use of version control tools, code reviews, and project management.

CMSC 15400 and a working knowledge of the C programming language are strict prerequisites of this course. Students who have not taken CMSC 15400 must speak with the instructor to ascertain that they meet the prerequisites for this course.

Course organization

This course revolves around the implementation of an x86 operating system kernel, divided into four separate projects, which accounts for the majority of the grade. Students will develop these projects in pairs. To successfully complete these projects, students must understand fundamental concepts in operating system design and implementation. The class meets two times a week for lectures that provide this conceptual scaffolding. There will be a midterm and a final exam, but no homework assignments. The course calendar, including the contents of each lecture and project deadlines, is shown in Table 1.

Project

In this course, students will implement an x86 operating system kernel. Since implementing such a kernel entirely from scratch is infeasible in a single quarter (and requires an in-depth knowledge of the x86 architecture), we will be using the Pintos instructional kernel. This kernel already implements most of the low-level functionality, allowing the student to concentrate on implementing higher-level operating system functionality, such as thread management, memory management, etc. while still allowing them to peek under the hood. This kernel also has the advantage of having been used in several OS courses at other universities, which means it has been thoroughly tested and documented.

The project is divided into four parts:

1. **Threads**: Students are given a minimally functional thread system, which they will extend to gain a better understanding of synchronization problems.
2. **User Programs**: Pintos already supports loading and running user programs, but no I/O or interactivity is possible. Students will enable programs to interact with the OS via system calls.
3. **Virtual Memory**: Pintos is limited by the machine's main memory size. In this project, students will remove this limitation by implementing a virtual memory management system.
4. **File Systems**: Students will improve Pintos' basic file system.

Although each project has its own deadline (see Table 1), students will have access to all the documentation and code for all four projects from the first day. All the documentation and instructions necessary to develop the projects can be found here:

`http://www.classes.cs.uchicago.edu/archive/2011/winter/23000-1/pintos/doc/html/`

The source code can be found on PhoenixForge:

`https://phoenixforge.cs.uchicago.edu/projects/pintos`

Students will develop the projects in groups of two. Groups must be formed by January 11. Groups can be changed from one project to another, but you must inform the instructor that you intend to do so. If your partner drops out of the course or you feel he/she is not contributing to the group's effort, you should make the instructor aware of this.

Exams

There will be a midterm on February 10, 2011. This midterm will take place in class, and will only occupy the first 40 minutes of the lecture. The final exam is tentatively scheduled for March 18 (during Finals Week). The exact date and time of the exam will be announced by the end of second week.

For the most part, these exams will not quiz students on the contents presented in lectures or, at least, not directly. Instead, most of the questions and problems will be related to the project. Students who have developed the projects on their own (which requires understanding the contents presented in class) should be able to answer these questions with relative ease. There will also be a few questions that are not related to the project, but will be in line with the learning goals outlined above.

Table 1: CMSC 23000 Winter 2011 Calendar

Week	Date	Lecture	OSC	MOS	Project
1	4 January	Introduction to Operating Systems.	1, 2	1	
	6 January	Processes and Threads	3, 4	2.1, 2.2	
2	11 January	Concurrency, Process Synchronization	6	2.3, 2.5	
	13 January	Concurrency, Process Synchronization	6	2.3, 2.5	
3	18 January	Scheduling	5	2.4	
	20 January	Deadlock	7	6	
4	24 January				Project 1 Due
	25 January	Processes revisited			
	27 January	Memory management	8	3.1, 3.2	
5	1 February	Virtual Memory	9	3.4–3.7	
	3 February	Virtual Memory	9	3.4–3.7	
6	7 February				Project 2 Due
	8 February	Virtual Memory	9	3.4–3.7	
	10 February	Midterm. File systems	10, 11	4	
7	15 February	File systems	10, 11	4	
	17 February	Input/Output	12, 13	5	
8	22 February	Input/Output	12, 13	5	
	23 February				Project 3 Due
	24 February	History of OSs. OS Architectures.	1, 23	1.2, 1.7	
9	1 March	Guest lecture TBD (Instructor out of town)			
	3 March	Guest lecture TBD (Instructor out of town)			
10	8 March	Virtual Machines	2.8	1.7.5	
	9 March				Project 4 Due

OSC: Operating System Concepts **MOS**: Modern Operating Systems

Books

This course has two *suggested* texts:

- *Operating system concepts, 8th Ed.*, Abraham Silberschatz, Peter B. Galvin, Greg Gagne, J. Wiley & Sons 2008. Available for purchase from the Seminary Co-op Bookstore.
- *Modern Operating Systems, 3rd Ed.*, Andrew S. Tanenbaum, Pearson Prentice Hall 2008.

The lectures will not be tied to any one text in particular and all the material necessary to implement the project will be delivered in the lectures. Thus, students are not required to purchase any of the above texts (although they can be a valuable reference).

Grading

Grading will be based on the project (60%, each part worth 15%), midterm (10%), and final exam (30%).

Types of grades

Students may take this course for a quality grade (a "letter" grade), a pass/fail grade, or as an auditor. Students will declare on the final exam whether, depending on their final grade, they want to receive a letter grade, a pass/fail grade or withdraw from the course (a *W* grade). For example, students can declare "If my final grade is a C+ or lower, I will take a *P* (Pass) instead of a letter grade and, if my grade is an *F*, I wish to take a *W*".

Students are welcome to audit this course. However, be aware that a student who is auditing the course cannot switch to a letter grade or a pass/fail grade, and will receive an *R* (Registered) grade.

> Note: *Students taking this course to meet general education requirements must take the course for a letter grade.*

Late submissions

The instructors will collect the latest revision each group commits to their SVN repository before the deadline. Any work committed after the deadline is ignored and not collected.

Each group is allowed three 24-hour extensions during the quarter. More than one extension can be applied to a single submission. i.e., a single 24-hour extension on three submissions, or a 72-hour extension on a single submission.

Policy on academic honesty

The University of Chicago has a formal policy on academic honesty which you are expected to adhere to:

`http://www.uchicago.edu/docs/studentmanual/academic_honesty.shtml`

In brief, academic dishonesty (handing in someone else's work as your own, taking existing code and not citing its origin, etc.) will *not* be tolerated in this course. Depending on the severity of the offense, you risk getting a hefty point penalty or being dismissed altogether from the course.

Even so, collaboration between students is certainly allowed (and encouraged) *as long as you don't hand someone else's work as your own.* If you have discussed parts of an assignment with someone else, then make sure to say so. If you consulted other sources, specially the source code of other operating systems, please make sure you cite these sources.

If you have any questions regarding what would or would not be considered academic dishonesty in this course, please don't hesitate to ask the instructor.

Asking questions

This course has an *open door policy* for asking questions. Instead of setting fixed office hours, you are welcome to consult with the instructor at any time. Nonetheless, you should try to give the instructor, whenever possible, some advance warning of your visit (by e-mail) to make sure that he will be in the office at that time.

The preferred form of support for this course is though the *course mailing list*, which can be used to ask questions and share useful information with your classmates. All questions over e-mail must be sent to the mailing list, and not directly to the instructor, as this allows your classmates to join in the discussion and benefit from the replies to your question. The only exception to this rule is if your question requires revealing part of your solution; in that case, please e-mail the instructor off-list and, whenever possible, a reply that omits your solution will be sent to the mailing list.

You can subscribe to the mailing list in the following web page:

`http://mailman.cs.uchicago.edu/mailman/listinfo/cmsc23000`

Acknowledgements

Thanks go to Ben Pfaff, the original author of Pintos while at Stanford, and Godmar Back at Virginia Tech for their help with getting Pintos up and running at UChicago. John Ousterhout from Stanford also provided some valuable pointers on organizing a Pintos-oriented course.