Operating Systems

Fall 1998
CS 43201 / 53201

Instructor

Dr. Robert A. Walker
walker@mcs.kent.edu

MSB 351, 672-4004 ext. 351
Office hours = Tu 1-3pm, Th 1-3pm, or by appt.

Teaching Assistant

To be determined…

Course Prerequisites

The *1998-1999 Undergraduate Catalog* lists the prerequisites for this course as *CS 33001 Data Structures* and *CS 33003 Computer Organization and Assembly Language*. Equivalent courses taken elsewhere are also acceptable. It is also expected that you are moderately familiar with C++ classes and methods, as you will be writing and modifying C++ code in each of the projects.

Course Overview

The goal of this course is to provide an introduction to the internal operation of modern operating systems. In particular, the course will cover processes and threads, mutual exclusion, CPU scheduling, deadlock, memory management, and file systems. If time permits, we may briefly examine networking and distributed computing. Students will use the Nachos instructional operating system for several programming projects.

Textbook

The required textbook for this course is:

- *Operating Systems Concepts*, 5th edition, Silberschatz and Galvin, Addison-Wesley, 1998.

Other reasonable textbooks that you might want to refer to, if you have access to them, are:

- *Modern Operating Systems*, Tanenbaum, Prentice Hall, 1992.
- *Operating Systems*, 3rd edition, Stallings, Prentice Hall, 1998.

Class Web Page

The web page for this class is **http://www.mcs.kent.edu/~walker/classes/os.f98**. The web page will contain link to the following course materials.

- Current class syllabus and schedule
- Lecture notes (in PostScript and Adobe PDF format)
- Homework and exam solutions
- Homework and programming project assignments, along with:
  - Information about the Nachos operating system simulator
  - Questions and answers about the assignment

Other information may be included as well. You might want to check the web page on a regular basis, in particular when a programming project is outstanding.

Lectures

Students are expected to attend each lecture. I will not take roll, and I understand that it may occasionally be necessary to miss a class, but in general I expect you to attend each lecture.

At each class, I will hand out one sheet of paper containing reduced copies of *at most eight* of my slides for that lecture. If you would like to have reduced copies of *all* of my slides for that lecture, the full version of the lecture notes will be on the class web page before the lecture, and you can print them out. Note that you are not required to either look at or print out these notes; they are provided solely for your convenience should you want them. However, you should ***not*** consider skimming these notes to be an adequate substitute for attending the lecture, as they will contain only the text of my slides, not the comments that I will make in class.

My lecture notes will be drawn from a variety of sources. The required text (OSC) will serve as a primary reference, although some material may be drawn from other books on operating systems. I will also use lecture notes from other professors as a reference, in particular notes by Kathryn McKinley, Bradley Chen, Mendel Rosenblum, Tom Anderson (all based on an earlier set of notes by John Ousterhout), and on notes by Divyakant Agrawal and Paul Farrell / Steve Chapin.

Homework Assignments and Programming Projects

There will be approximately four homework assignments and three programming projects during the semester. The homework assignments will be pencil-and-paper based, while the projects will be based on the Nachos instructional operating system, and will involve reading and writing code. Tentative due dates are shown on the Class Schedule, attached at the end of this syllabus.

Nachos Programming Projects

The Nachos instructional operating system is written in C++ (actually, a subset of C++ that uses classes and methods, but avoids troublesome C++ constructs like inheritance and overloading). If you need quick refresher on C++, see the document "A Quick Introduction to C++" on the class web page.

Late Policies

In general, you will have adequate time to complete each assignment. However, you should begin work on each assignment early so that you will have plenty of time to become familiar with it and with the Nachos code that you must read and/or modify, and so that you will have time to "sleep on" the difficult parts. Waiting until two days before the due date to start the project is a bad idea.

For homework assignments, ***no*** late homeworks will be accepted, unless you make *prior* arrangements with me, or have a *documented* illness (in which case I expect you to contact me as soon as possible).

For programming projects, late projects ***will*** be accepted with a 10% penalty for ***each day or portion thereof*** that the project is late. Other extensions will not be granted, unless you make *prior* arrangements with me, or have a *documented* illness (in which case I expect you to contact me as soon as possible).

Exams

There will be three exams (held during class) and a final exam (held during finals week). The tentative dates for the exams are shown on the Class Schedule, attached at the end of this syllabus. All exams are closed book and closed notes, and must be individual work. It is expected that you take each exam at the scheduled time, unless you make *prior* arrangements with me, or have a *documented* illness (in which case I expect you to contact me as soon as possible).

Academic Integrity

Student-teacher relationships are built on trust. Students must trust that teachers have made appropriate decisions about the structure and content of the courses they teach, and teachers must trust that the assignments which students turn in are their own. Acts which violate this trust undermine the educational process. In this course, the penalty for ***any*** act of academic dishonesty is a final course grade of F.

Cooperation on Homework Assignments and Programming Projects

For both homework assignments and programming projects, I strongly believe that discussion with your peers is an excellent way to learn. If you don't understand something, discussing it with someone who does can be far more productive than beating your head against the wall.

Having advocated discussion, then, I must be about clear what is allowed, and what is not. In general, students are allowed to cooperate as follows: you are allowed to discuss with other students *the assignment*, and *general methods for solving the assignment*. However, you are ***not allowed*** to work with someone else to actually *solve* the assignment, or to *write code* (even pseudocode) for a program, and you are certainly ***not allowed*** to *copy* anyone else's solution; doing any of these things will be considered cheating, and will be grounds for failing the course.

Note that there is a fine line between discussion and cheating. If you are unsure what is allowed and what isn't, feel free to discuss the distinction with me, but if something feels uncomfortable, it's probably not allowed.

Finally, you should be careful not to give others access to your code. This means that you shouldn't keep your program in a publicly-accessible directory, you shouldn't leave your terminal unattended, and you shouldn't forget to pick up your printouts.

Grades

Your final course grade will be broken down as follows:

• Homeworks (approximately 4)	20%	
• Programming projects (approximately 3)	25%	(although all may not be weighted equally)
• Exams (3)	30%	
• Final exam	25%	

The final course grade will be determined with A = 90–100, B = 80–99.99, etc. There will be no curve at the end of the course, although individual exams, homeworks, etc. may occasionally (although rarely) be curved. Thus you should always be able to determine how well you are doing in the course.

Credit for CS 43201 Versus Credit for CS 53201

This course is being offered at both the senior level (CS 43201) and the graduate level (CS 53201). Grades will be determined separately for each level, with those students taking CS 53201 being held to a higher standard of performance.

Students "Sitting In"

Students who want to unofficially "sit in" on the course, either to qualify for admission the CS graduate program, or to prepare for the graduate Qualifying Exam, should contact me as soon as possible. In general, I allow sit-ins, but will not grade any assignments or exams for anyone other than officially-enrolled students. Any requests for graduate program references, etc. should be discussed with me at the beginning of the course.