

# CS-343 Operating Systems

## Fall Quarter 2005

Syllabus and class information

### **Administrative Information**

#### **Professor**

Fabián E. Bustamante  
Technological Institute, L465  
2145 Sheridan Road  
+1 847 491-2745  
fabianb@cs.northwestern.edu  
Office Hours: TBD (or by appointment)

#### **TAs**

Stefan Birrer  
Ford Design Center, Room 2-217  
+1 847 491-7060  
sbirrer@cs.northwestern.edu  
Office Hours: TBD

David Choffnes  
Ford Design Center, Room 2-217  
+1 847 491-7060  
drchoffnes@cs.northwestern.edu  
Office Hours: TBD

#### **Final Exam**

TBD

#### **Web Site**

<http://www.cs.northwestern.edu/~fabianb/cs-343.html>

## Location and Time

Tuesdays and Thursdays 2:00-3:20.

Lectures will be held in Technological Institute, Room M345.

## Course Prerequisites

- **CS-311 Data structures and data management.**
- **CS-213 Introduction to Computer Systems** *or*  
**ECE 205 Fundamental of Computer Systems Software** *and*  
**ECE 231 Advanced Programming for Computer Engineers.**
- Familiarity with basic computer architecture concepts and proficiency in C programming in UNIX systems.

## Communication Channels

There are a number of communication channels set up for this class:

- We will use the course web site to post homeworks, projects, course-related announcements, etc. You should check this regularly.
- There will be a newsgroup (cs.343 at news.cs.northwestern.edu). This particular channel is intended to foster communication among you, the students. You'll find that someone else in the class will have thought of the same problem that you have and will perhaps have some valuable insight that will prove helpful. The staff will be monitoring the discussion threads and will step in with guidance when appropriate.
- Finally, there is an email alias (cs-343-ta@cs) for the TA that you should use for questions that would be inappropriate to post on the newsgroup (source code being a good example).

## Textbook and Other Readings

- *Operating Systems Concepts, 7th Ed.*, A. Silberschatz, P. Galvin and G. Gagne, J. Wiley & Sons, New Jersey, 2005 (*Textbook*).
- Supplemental Reading. A set of papers will be made available providing a deeper historical perspective of operating systems, in-depth treatment of some of the topics I can only briefly cover in class, and some useful practical advice in designing and implementing complex systems. You will not be responsible for this material; they are only intended for those specially interested in the topic.
- *Advanced Programming in the Unix Environment*, R. Stevens, Addison-Wesley, 1992. A basic book for anyone writing programs that run under Unix (*Highly Recommended*).
- *The C Programming Language, 2nd Ed.*, B. W. Kernighan and D. M. Ritchie, Prentice Hall, 1988. A must (*Recommended*).

- *The Practice of Programming*, Brian W. Kernighan and Rob Pike. Addison-Wesley, 1999. Programming is more than just writing code - this book is packed with great practical advice on style, design, interfaces, testing and debugging, maintenance, etc.; all issues that programmers must deal with in the real world (*Recommended*).

## Course Description

**Catalogue Description:** Fundamental overview of operating systems. Operating system structures, processes, process synchronization, deadlock, CPU scheduling, and memory management.

**Detailed Description:** Operating systems control all of a computer's resources and present users with the equivalent of virtual machines that are easier to program than their underlying hardware. This course provides an overview of fundamentals of operating systems principles, complemented with discussions of concrete modern systems to help you understand how these principles are applied in real OSs. Topics covered include an overview of the components of an operating system, mutual exclusion and synchronization, implementation of processes, scheduling algorithms, memory management and file systems.

Although the main learning objective of the course is to understand the requirements, design and implementation of modern operating systems, at a higher level the course aims to provide you with a good grasp of basic abstractions employed in system-level software (such as processes, threads, virtual memory, caching, etc.), while uncovering the "magic" that happens inside the box.

The course has a strong project component intended to provide essential experience in designing and implementing complex systems and working in a team. In designing the projects and estimating their required effort/hours, I am assuming you are (1) familiar with basic computer organization and data structures and (2) capable of programming in C in UNIX (or UNIX-like) systems (experienced with pointers, explicit dynamic memory allocation, multi-file projects, etc.).

## Course Organization

The course is organized as a series of lectures, TA sessions, reading, homework, project and exams:

- Lectures - A set of lectures through which I present the core of the material.
- TA Sessions - Discussion sessions held by the TAs to answer questions about the lecture, readings, homework assignments and projects.
- Readings - Textbook reading in preparation for (not substitution of) the lecture and additional reading for those interested in delving further into some topics.
- Homework - Five homework assignments with questions from (or similar to those in) your textbook, aimed at reinforcing the material covered in the reading and the lectures.
- Project - Four programming projects to give you a better understanding of the subject matter and some experience with system level programming including thread-level programming.

- Exams - Two exams, a midterm and a final exam. These exams cover the material presented in the lecture, homework and projects. You'll be able to bring one page of notes to assist you during the exam.

## Homework

There will be two kinds of homework assignments given throughout the class: reading assignments and textbook-style questions. You should have finished the assigned reading before coming to lecture. In addition, there will be a set of written homework assignments that must be done alone and turned in at the end of class on the due date (see course policies below).

## Projects

As you can deduce from the allocation of weights for grading, programming projects make up a major portion of this class. There will be four (4) projects. Except for the first "warm-up" assignment that you will work by yourself, all other projects are to be done by teams of two (2) people. Both partners should work cooperatively on the design, implementation, and testing of their solution.

## Exams

There will be a midterm and a final exam. Exams will be in-class, closed-book (except for one page of notes you are allowed to bring in), and will cover materials from lecture, required readings and projects. The final exam will not be cumulative.

## Grading

I use a criterion-referenced method to assign your grade; in other words, your grade will be based on how well you do relative to predetermined performance levels, instead of in comparison with the rest of the class. Thus, if a test has 100 possible points, anyone with a score of 90 or greater will get an A, those with scores of 80 or greater will get a B, those with scores of 70 or greater will get a C, and so on. Notice that this means that if everyone works hard and gets  $>90$ , everyone gets an A.

Total scores (between 0 and 100) will be determined, roughly, as follows:

- Homeworks: 20%
- Projects: 40%
- Exams (20% each): 40%

A note about *class participation*: while not explicitly included as an item in the previous list, your participation in class will be taken into consideration throughout the quarter and when granting partial and final scores/grades.

## Course Outline and Approximate Dates

*Because one has to be an optimist to begin an ambitious project, it is not surprising that underestimation of completion time is the norm.*

– Fernando J. Corbató, “On Building Systems that Will Fail”, 1990 Turing Award Lecture.

Class	Date	Topic	Textbook/paper†
01	09/20	Introduction	1
02	09/22	OS concepts and structure	1 & 2
03	09/27	Processes	3 (3.6 excluded)
04	09/29	Processes	3 (3.6 excluded)
05	10/04	Threads	4
06	10/06	Scheduling	5
07	10/11	Process Synchronization	6
08	10/13	Process Synchronization	6
08	10/18	Process Synchronization	6
09	10/20	Deadlock and Review	7
10	10/25	<i>Midterm</i>	
11	10/27	Memory Management	8
12	11/01	Memory Management	8
13	11/03	Virtual Memory	9
13	11/08	Virtual Memory	9
14	11/10	File System Interface	10
15	11/15	File System Implementation & Examples	11 (11.9-10 excluded)
16	11/17	Mass Storage & I/O Systems	12 & 13 (13.6 excluded)
17	11/22	Protection & Security	14 & 15
18	11/24	Introduction to Distributed Systems	16
19	11/29	<i>Thanksgiving break</i>	
20	12/01	Research in OS/Review	TBD
22	12/07	<i>Final (Time TBD)</i>	

## Policies

**Late policy:** Unless otherwise indicated, homework assignments and projects are due by midnight on their due date. If you hand in an assignment late, we will take off 10% for each day (or portion thereof) it is late.

**Cheating vs. Collaboration:** Collaboration is a really good thing and we encourage it. On the other hand, cheating is considered a very serious offense. When in doubt, remember that it's OK to meet with colleagues, study exams together, and discuss assignments. However, what you turn in must be your own (or for group projects, your group's own) work. Copying code, solution sets, etc. from other people or from any other sources is strictly prohibited.