

Introduction to Operating Systems

Prof. Darrell Long
Computer Science Department
Jack Baskin School of Engineering
darrell@cs.ucsc.edu

Winter 2004

What the catalog says ...

Fundamental principles of operating systems: process synchronization, deadlocks, memory management, resource allocation, scheduling, storage systems, and [the] study of several operating systems. A major programming project will be required. Prerequisites: courses CMPS 101 and CMPE 12C.

What we will do ...

The course will cover the basics of uniprocessor operating systems, paying particular attention to the performance aspects. This includes, but is not limited to: processes, interprocess communication, synchronization, scheduling, memory management, swapping, virtual memory, page replacement algorithms, segmentation, file systems, security, protection, input/output, interrupts, device management, resource allocation and deadlocks.

As we go along, we will also discuss more advanced topics such as distributed operating systems, storage systems, networks, communications protocols, synchronization, atomicity, remote procedure call, distributed file systems and fault tolerance.

What you must read ...

The primary source for the course will be the text by Tanenbaum. It covers the main topics of uniprocessor operating systems in the early chapters, and then covers distributed systems and other advanced topics in the later chapters. It is in its second edition, and is one of the most up-to-date undergraduate operating system texts available.

- Andrew S. Tanenbaum, *Modern Operating Systems*, Second Edition, Prentice Hall, 2001.

We will cover all the core operating systems issues, but since we only have ten weeks, we will have to skip some later chapters (for example, the case studies). Additional material, primarily from the research literature, will be presented when appropriate.

What you really ought to read ...

If you have some spare time (and spare money), or just want to learn more about operating systems, then the following texts may be helpful. The first is a standard introduction to operating systems and contains some material not in the Tanenbaum text.

- Abraham Silberschatz, Peter B. Galvin and Greg Gagne, *Operating System Concepts*, Sixth Edition, John Wiley & Sons, 2001.
- Marshall Kirk McKusick, Keith Bostic and Michael J. Karels, *The Design and Implementation of the 4.4BSD Operating System*, Addison-Wesley, 1996.
- E. G. Coffman and P. J. Denning, *Operating Systems Theory*. Prentice-Hall: Englewood Cliffs, 1973 (*out of print*).

Keeping in touch ...

You are expected to read your electronic mail *daily*. It is your primary means of communication with the instructor and with the teaching assistant.

You must read (and *contribute to*) the class mailing list (<http://www.soe.ucsc.edu/mailman/listinfo/cmeps111>). Discussions of the material and announcements to the entire class appear here.

Programs, Examinations & Exercises ...

- There will be weekly exercises worth a total of 10% of your total grade. These questions taken from the primary text and are intended to make sure that you is reading it and to test your understanding of the material from each chapter.
- There will be four programming assignments due approximately every two or two and a half weeks, each worth 10% of your total grade. You will be implementing several components of an operating system kernel running atop a simulated DLX processor.
- There will be a midterm examination at approximately halfway through the quarter that will be worth 20% of your grade.
- There will be a comprehensive final examination worth 30% of your grade.

Note: *You must average above 50% on the programming assignments and on the examinations in order to pass the source. A score lower than 50% on either portion of the class will result in a failing grade regardless of the total score.*

All examinations and quizzes are closed book. You may bring one $8\frac{1}{2}'' \times 11''$ single-sided sheet of notes (the font may *not* be smaller than 10 point) with you. You may *not* share this sheet with you friends and neighbors.

Academic Honesty ...

This really should not be an issue, but recent events have made the following necessary. The bottom line is that you are expected to conduct yourself as a person of integrity.

You are expected to adhere to the highest standards of academic integrity. This means that plagiarism¹ in any form is unacceptable. As a (soon to be) computing professional, I encourage you to consult the code of ethics appropriate to your discipline.²

Plagiarism will be assumed, until disproved, on work that is essentially the same as that of other students. This includes identically incorrect, off-the-wall, and highly unusual duplicate answers where the probability of a sheer coincidence is extremely unlikely. All parties to this unacceptable collaboration will receive the same (*zero*) score. A *zero* score on either examinations or on a segment of the programming project is grounds for failing the course.

Your work must be your own. This refers to examinations, written assignments and programming projects. Should you be found to be cheating, at a *minimum*, you will fail that assignment and a letter will be sent to your Department, the School of Engineering, and to your Provost and academic preceptor. The instructor reserves the right to stronger action should the situation warrant it.

You should bring a picture identification with you to all examinations and be prepared to show it upon request.

You may discuss programming projects with your friends, but you are expected to abide by the *Gilligan's Island*³ rule. The only thing you may bring to such a discussion is you, and no notes may be taken away from the meeting. The copying of code is strictly forbidden.

The bottom line is: *if you cheat, you will fail the course.*

¹ **plagiarize** *vt.* [*< L. plagiarius, kidnaper*] to steal and pass off as one's own (the ideas or words of another) to present as one's own an idea or product derived from an existing source – **plagiarizer** *n.* (source: Webster's New World Dictionary)

² The Association for Computing Machinery is <http://www.acm.org/>, the IEEE is <http://www.ieee.org/> and the IEEE Computer Society is <http://www.computer.org/>.

³ The Gilligan's Island rule states that following a discussion of the project, a break must be taken for at least a half hour before coding. Watching something inane like *Gilligan's Island* on television satisfies this rule. Better yet, read a non-computer book before beginning to code the project: for example, I recently finished *Baudolino* by Umberto Eco, and am working my way through Edward Teller's *Memoirs*.