ECE 329 Computer Systems Structures
Fall 2006

Instructor:	Stan Birchfield
Office:	207-A Riggs Hall, 656-5912, stb at Clemson
Office Hours:	9:00-10:00 MWF, or by appointment
Course website:	**http://www.ces.clemson.edu/~stb/ece329**
Text:	A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, Sixth Edition, John Wiley & Sons, 2003.
Prerequisites:	CPSC 102 or 210; CPSC 340 or 212; ECE 272.

Overview: In this course students will learn the basics of operating systems, including the creation, management, and scheduling of threads and processes; process communication and synchronization; memory management; file systems; and protection. Programming assignments connect the theory with practice and enable students to further develop their programming skills.

Objectives: By the end of the course, students should be able to do the following:

- *Fundamental concepts.* Define and explain the basic concepts and terms of operating systems and structures, such as thread, process, starvation, thrashing, spinlock, deadlock, critical region, atomic transaction, mutex, semaphore, the five process states, graceful degradation, frame, fault-tolerant system, remote procedure call, and file table. Explain the difference between hard and soft real-time systems, logical and physical addresses, segmentation and paging, internal and external fragmentation, I/O-bound and CPU-bound process, preemptive and non-preemptive scheduling. Contrast the different types of message passing. Compute the effects of different scheduling algorithms, and apply the Banker's Algorithm.
- *Programming skills.* Write clean, well-documented multithreaded C/C++ code utilizing thread creation, thread suspension, thread cancellation, mutexes, and semaphores.

Topics:	**Lectures**
• introduction	1
• C/C++ programming language and tools	6
• computer and operating system structures	2
• processes and threads	5
• CPU scheduling	5
• process synchronization	6
• deadlocks	5
• memory	3
• file systems	2
• I/O systems	2
• projects	5
• tests	1

TOTAL	43

Grading:

1. *Tests.* There will be one midterm and a final examination, both of which will be closed-book, closed-notes. The instructor should be notified of any conflict that would prevent a student from taking a test at least one week in advance, so that alternative arrangements can be made. Without prior approval, missed tests cannot be made up except in cases of extreme urgency and importance (e.g., sudden illness, death in the immediate family).
2. *Quizzes.* There may be four or five short, unannounced quizzes during class. A student who keeps up with lectures and reading assignments should do well on these quizzes. Unless agreed upon in advance by the instructor, missed quizzes will receive a grade of zero. To receive a passing grade in the course, a student must take at least three quizzes. The grade of the lowest quiz may be dropped, subject to instructor discretion.
3. *Programming Assignments.* There will be five or six programming assignments. Late assignments will be accepted at a penalty of 10 points per day (according to a six-day work week), up to a maximum of 35 penalty points; assignments turned in more than one week late will receive a zero. Although students are encouraged to discuss the assignments with their colleagues, they must turn in their own work. Looking at the written work or code of another student (including former students or students at other universities), or copying that work is strictly prohibited. Similarly, students may seek information on the web to aid their understanding of the assignments, but any such work may not be copied. Students who violate University rules on academic dishonesty will be subject to disciplinary penalties, such as failure in the course and/or dismissal from the University.
4. *Grading.* Grades will be determined by the following formula: midterm (30%), final exam (30%), programming assignments (30%), quizzes (10%).