CS 321 - Operating Systems

Meeting time: 10:30-11:30pm Room 106 Chapman Building University of Alaska Fairbanks	UAF CS F321-F01 #33448 3.0 Credits, Spring 2006 Prerequisite: CS 301 (Asm)	Instructor: Dr. O. Lawlor ffosl@uaf.edu, 474-7678 Office: 210C Chapman Hours: 2-3 MWF or by appointment

Required Textbook: Operating System Concepts, 7th Edition by Silberschatz, Galvin, & Gagne; 2005 Wiley & Sons (bookstore, or Amazon)	ADA Compliance: Will work with Office of Disabilities Services (203 WHIT, 474-7043) to provide reasonable accomodation to students with disabilities.	Course Website (& links to Blackboard): **http://www.cs.uaf.edu/2006/spring/cs321** Machines: ASSERT lab, nanook.uaf.edu, Chapman lab, or Linux CDs available

Course Goals and Requirements

By the end of the course, you will be able to design system-level libraries for a variety of tasks; be familiar with the general abilities and interfaces provided by common operating systems; and understand in a deep way the implementation of modern processor execution, memory, and storage. To understand this, you will need to have experience writing programs in some standard systems programming language (C or C++), with at least some idea of how your code relates to assembly language and how it runs on the real machine.

Calendar

Last day to drop: February 3. Spring break: March 11-19. Last day to withdraw: March 24. Midterm exam will be held at 10:30am on Wednesday, March 8. Final exam will be held at 10:15am on Monday, May 8.

Student Resources

Academic Help: Google, Rasmuson Library, Academic Advising Center (509 Gruening, 474-6396), Math Lab (Chapman Room 305), English Writing Center (801 Gruening Bldg, 478-5246).

Grading

Your work will be evaluated on correctness, rationale, and insight, not on successful regurgitation of random trivia. Grades for each assignment and test may be curved. Your grade is then computed based on four categories of work:

1. **HW:** Homeworks and machine problems, to be distributed through the semester.
2. **PROJ:** A substantial software development project related to operating systems, together with a short presentation of your results. Example projects: build an interesting linux kernel module; write a program that accesses any interesting piece of hardware; write a library that sensibly merges two different interfaces (e.g., Linux and Windows memory-mapping interfaces).
3. **MT:** Midterm Exam.
4. **FINAL:** Final Exam (comprehensive).

The final score is then calculated as:

TOTAL = 20% **HW** + 25% **PROJ** + 25% **MT** + 30% **FINAL**

Letter grades are then assigned at the usual 90/80/70 (etc) cutoffs. At my discretion, I may round your grade up if it is near a grading boundary.

Homeworks are due by 5pm on the day they are due. Late homeworks will receive no credit. At my discretion, I may allow late assignments without penalty when due to circum stances beyond your control. Major assignments that are slightly late may be accepted at a 50% grade penalty (e.g., on-time grade: 86%; late grade: 43%). Everything you turn in must be your own work--violations of the UAF Honor code will result in a minimum penalty equal to THAT ENTIRE SECTION OF YOUR GRADE (e.g., one plagarized homework question will negate an otherwise perfect grade on all homeworks). However, even substantial reuse of other people's work is fine (and not plagarism) if it is clearly cited; you'll be graded on what you've added to others' work. Group work on substantial assignments (not homeworks, not tests) is acceptable if you clearly label who did what work; but I do expect a two-person group project to represent twice as much work as a one-person project. Department policy does not allow tests to be taken early; but in extraordinary circumstances may be taken late.

Course Outline (Tentative)

First section: Time Management

- Event-driven programming
  - DOS-style polling loop
  - Mac/X (or other GUI) event loop
  - Win32 wndProc
- Processes (Ch. 3)
  - Semantics: multiprogramming
  - Creation: UNIX fork, Win32
- Hardware Implementation (Ch. 3.1)
  - Resources: Stack, registers, heap
  - Protected (privileged, supervisor, "ring 0") mode kernel & security
  - System calls, timers, and other hardware interfaces
- Signals & interrupts (Ch. 4.4.3 & 13.2.2)
  - Hardware interrupts
  - UNIX/Win32 signals/handlers
  - Interrupt safety (reentrancy)
- Threads (Ch. 4)
  - Kernel-level: pthreads, win32
  - User-level: coroutines
- Concurrent Interaction (Ch. 6 & 7)
  - Motivation: Race conditions
  - Locks (pthread lock, win32 mutex), semaphores (win32)
  - Deadlock prevention
  - Not covered: deadlock detection & response
- CPU Scheduling (Ch. 5)
  - Starvation, poor utilization
  - Prioritization
  - Priority Inversion
  - Job scheduling: Shortest-Job-First

Second Section: Space Management

- Memory allocation (Ch. 8.3.2)
  - Memory heirarchy & cost-capacity-speed tradeoff
  - Low-level memory allocation: sbrk
  - Mid-level memory allocators
- Virtual memory: uses (Ch. 9)
  - DLL/text page sharing, copy-on-write
  - Memory-mapped files: UNIX mmap, mprotect, SYSV IPC; Win32 MapViewOfFile (Ch. 9.7)
  - Software distributed shared memory
- Virtual memory: implementation (Ch. 8, 9.4)
  - Page table and TLB (presence, permissions, and dirty bits)
  - Demand paging & page replacement strategies
- Filesystem (Ch. 10 & 11)
  - Layouts: File Allocation Table (FAT), inode, b-tree
  - Caching, fragmentation, corruption during crash
- Accounting and security
  - Terminology: Tampering and authentication, secrecy and encryption
  - Common security holes: buffer overflow, unquoted inputs, excess priviledge