

COMP SCI 452 Operating Systems - Spring 2010
Course web site: [<http://www.uwgb.edu/shayw/courses/452>]
Meeting times: MW 8:00-9:20 in MAC 122

Professor: Bill Shay
Phone: 465-2316/5038

Office: MAC C308
email: shayw@uwgb.edu

Office Hours: MTW: 9:30-10:30; F: 8:30-9:30; or by appointment.

Prerequisites: COMP SCI 370; Recommended COMP SCI 353.

Text: Applied Operating System Concepts with Java, by Silberschatz, Galvin, and Gagne, 7th edition

On reserve: Interprocess Communications in Linux by Gray, QA76.76.O63 G7288

Advanced Programming In the UNIX Environment by W. Richard Stevens, QA76.76.O63 S754

UNIX Systems Programming: Communication, Concurrency and Threads by Robbins & Robbins,
QA76.76.O63 R6215

Course: The course in operating systems should change how you look at a computer system and, in particular, the programs you write. If you are like most people with the minimum prerequisites your perceptions likely center on the programs you write and the computer as a means to get their tasks done. If you've taken the recommended course in computer organization you have some idea of the various hardware components that comprise a computer. In this course we center on the software of a computer system. Through this course you begin to realize that the programs you write are just one of hundreds (or even thousands or more) submitted to a system, each requiring service. From the perspective of a computer system, keeping track of that many different programs and their needs is a daunting task.

One may simplistically and generically define an operating system as a manager. A good manager organizes to make sure that all tasks are completed correctly and in timely fashion. The complexity of the manager's job depends in part on the number of activities that must be managed. Most people need look no further than their own personal lives to recognize that managing their daily activities increases dramatically with the demands made of them.

Most computing environments have large numbers of programs, which are running concurrently. Each is doing computations, sending and retrieving information from any number of external devices, and periodically communicating with one another. The operating system must make sure that each gets what it needs to do its work and that the gigabytes of information moving around reach the appropriate destinations.

The goals of this course include dealing with the following issues:

- Compilers generate machine language commands that make memory references, but have no idea which memory locations a program's data will occupy. How can we design a system in which a program will run independent of where it resides in memory?
- At a given instant, the sum of memory needs by all active programs usually exceeds the memory capacity of a computer system. How can an operating system deal with this efficiently and effectively?
- The number of active programs is usually well above the number of processors. This means that programs compete for processor use and that the operating system must define some type of scheduling strategy so that each program gets some processor time.
- For the most part programs must run independent of one another but there is occasionally the need for two or more otherwise independent programs to cooperate on some activity. We need to discover ways in which they can share information and to communicate with one another. We will look at a historical development of this issue and examine (and compare) several approaches.
- When programs request input and output the result may be gigabytes of information traveling among a variety of I/O devices and programs. Managing this huge amount of traffic so that it all gets to where it needs to go without running into each other is no small task.

There will be two semester exams and one final. There will also be one or two programs and programming project, which I will give to you in several phases. The project will involve the creation of multiple processes capable of running concurrently with an occasional need to communicate. You'll get a solid introduction to Linux operating system commands and a good deal of experience with them. I will also periodically divide the class up into groups for discussion purposes. I'll assign questions to groups and each group will have the responsibility to formulate a response. The intent is to have group members share their views on a variety of topics and to get people talking about various operating systems issues.

Syllabus (give or take a week)

Week	Topics
1	Computer system overview; instruction cycle; types, objectives, and functions of an operating system; the process concept. Parts of chapters 1-2. (I'll indicate in class what you can skip).
2	Concurrent processes: process creation, parent/child relationships, zombie processes, asynchronous activity, race conditions, interprocess communication, sending and catching signals. Sections 3.1-3.3, UNIX books on reserve, class demos.
3-4	Memory management: contiguous allocation, logical vs. physical address space, paged and segmented systems, security, virtual memory. Chapters 8-9
4-5	Working set models, page replacement strategies, thrashing, process efficiency, setting up multiple processes for shared memory access, UNIX shared memory primitives. Chapter 9, Notes and class demos, UNIX books on reserve.
5-6	Process management: Scheduling objectives, process states, state transition diagrams, POSIX threads, scheduling criteria and strategies. Chapters 3-5 and class demos.
7	Mutual exclusion and critical sections, historical attempts at software solutions; Dekker's algorithm. Chapter 6
8	Java and C++ threads and synchronization. Chapter 6 and class demos.
9	Semaphores, classic synchronization problems: producer-consumer relationship, bounded buffer problem, readers and writers problem. Chapter 6
10-11	UNIX semaphore operations and their application to the classic synchronization problems; class demos, UNIX books on reserve
12	Concurrent programming languages, monitors, the Ada language, synchronization of Ada processes via the Ada Rendezvous. [http://www.acm.org/sigada], [http://www.adahome.com/Ammo/cpp2ada.html] and [http://www.infres.enst.fr/~pautet/Ada95/a95list.htm]. Java implementation of monitors.
13	Deadlock: Prevention, avoidance, detection, recovery; deadlock detection. Chapter 7
14	Auxiliary storage management: file and directory structures, access methods, mounting a Linux file system, FAT, FAT32, and NTFS, disk access scheduling. UNIX file structures and inodes, security. Selected topics from Chapters 10-12 and parts of Chapters 21 and 22

GRADING:

Project/assignments: 30%; Group discussion questions: 10%; First and second exams: 20% each;
Final exam (20%)

The final exam is scheduled for Friday May 14, 2010 from 8:00 - 10:00. **PLEASE NOTE:** This date is known many months in advance and must be taken as scheduled. Any activities such as work schedules, vacations, trips, etc. must be planned around this date.