

## Guidelines 340 Operating Systems

### 340

*Catalog description:* Computer system overview (devices, interrupts, memory hierarchy), operating system classification, and the basic concepts of operating systems including processes (scheduling, threads, synchronization, inter-process communication, deadlock), memory management (swapping, virtual memory), I/O subsystems and file systems. Pre-requisites: a C (2.0) or better in the following: CSSE 251, Introduction to Computer Organization; and CSSE 250 Data Structures.

*Course objective:* Achievement of catalog description as well as a solid understanding of how the building blocks of a modern operating system work, including processes, threads, interrupts, I/O, virtual memory, synchronization, etc., -- an understanding necessary for the development of large-scale, real-world applications.

*Pre-requisite knowledge:* CSSE 250 Data Structures (queues, hash tables, etc.)  
CSSE 251 Computer Architecture (basic computer hardware organization -- machine language, physical memory, memory hierarchy, interrupts)

*Topics covered:*

- 1) Computer system hardware (physical memory, memory management hardware, DMA, disk drives, interrupts, busses)
- 2) System calls, interrupt service routines, user mode vs. kernel mode
- 3) Processes, process creation, process state, the process control block, context switching, CPU scheduling, IPC
- 4) Threads, multithreading models
- 5) Critical sections, synchronization, semaphores
- 6) Deadlock, necessary conditions for deadlock, deadlock prevention
- 7) Memory management, memory hierarchies, caching, virtual memory, page tables, page replacement algorithms, address spaces

*Comments:*

- 1) Departmental expectation is that this course include a substantial programming component for projects covering material as specified below.
- 2) Examples of projects include programming with low-level system calls (e.g. Unix open, close, read, write, fork, exec, etc), programming with a multithreading library such as pthreads, programming with synchronization primitives such as semaphores, writing a simulation of one or more operating systems components, modifying a "toy" operating system that runs on a hardware simulator, modifying a real operating system such as Linux, and so on.

*Demonstrable Outcome:* ability to explain the basic functionality of the various operating system components and the alternatives and tradeoffs involved, that is:

- 1) understand the basic terminology of operating system design, in order to be able to read and understand operating systems documentation, and as a foundation for

further study in networking, embedded systems, distributed systems, and other advanced topics.

2) understand and evaluate the tradeoffs between design alternatives (e.g. multiple processes vs. multithreading), in order to make intelligent design decisions.

3) understand, recognize and evaluate common failure modes (e.g. thrashing in a virtual memory system, or synchronization failure leading to inconsistency, or deadlock)

*Assessment:* exams will measure coverage of topics and achievement of course objectives  
Several programming assignments