

# CMSI 387

## OPERATING SYSTEMS

Spring 2009

### Final Exam Review Sheet

The final exam will take place on May 7, at 11:00am. It will be open everything: book, notes, handouts, and computer (which means that either we must have the test in the Keck lab, or everyone has a laptop that they can use). This guide should help you to prepare for the final properly.

### Covered Material

The final exam covers the following areas, including all handouts and sample code that have been distributed in support of this content:

- In detail — SGG Chapters 1–11
- In general — assorted operating system papers by Dijkstra, Brinch Hansen, Hoare, Lamport, etc.
- Working knowledge of how to configure and build a Linux kernel
- Working knowledge of an assortment of prominent operating systems (i.e., the ability to access the process, memory, I/O, and storage functions of these platforms)
- Working knowledge of C and the POSIX APIs that were covered in class
- Ability to express assorted operating system-related data structures and algorithms in code

### Sample Tasks and Questions

In addition to the possible questions and tasks for the midterm, you may also be asked to accomplish one or more of the following:

- Perform some analysis, critique, or evaluation of an operating system concept presented in class (e.g., context switches, round-robin scheduling, the critical section problem, deadlock prevention, main memory fragmentation, virtual memory, file system interface/implementation, and many many more...)
- Provide a Gantt chart and other relevant metrics for process execution based on some set of processes and one or more scheduling policies
- Manually perform some operating system algorithm; for example:
  - A critical section solution for some problem
  - Banker's algorithm

- Contiguous memory allocation for a given set of process
- Logical to physical address translation
- Page replacement for a given page reference string and physical memory space
- Answer questions pertaining to storage devices and file systems, including but not limited to common volume structures, mounting/unmounting, file system drivers, file allocation schemes (contiguous, linked list, indexed [e.g., inodes]), and directory structures
- Navigate or interpret an instance of a given file system (e.g., *ext2* at a low level, FAT or some other hypothetical implementation in general)
- Given multiple approaches to a particular operating system issue, compare their relative advantages and disadvantages, possibly making a definitive choice for a particular situation
- Answer a “big picture” operating system question — something that tests your understanding of an operating system's overall functionality as well as relevant issues regarding an operating system's design and implementation