COMP 3500: A Study Guide for Midterm

This is a guide of topics that may be included in the midterm exam 1. Note that questions pertaining to any of these topics may appear on the midterm exam.

1. OS Overview (Ch 1.1-1.6 and Ch 2.1-2.7)
    - 1.1 What is an operating system? Resource manager. It hides details of how underlying machinery operates.
    - 1.2 System view of the OS
        - Computer and Software
        - Resource allocation: CPU, memory, disk, and the like
        - Control programs
        - Resource sharing
    - 1.3 Goals of an OS
        - Convenience for user
        - Efficient operation of computer systems
    - 1.4 Operating System Strategies
        - Batch systems
        - Multiprogrammed batch systems
        - Time-shairng systems
    - 1.5 Monilithic kernel vs. Microkernel

2. Processes (Ch 3.1-3.4)
    - 3.1 Concept
        - Process vs. program
        - Process states
        - Process Control Block (PCB)
    - 3.2 Process Control
        - Creation
        - Termination
        - Process state transitions
    - 3.3 Process states
        - Two-state process model
        - Creation and termination
        - Five-state model
        - Suspended processes

3. Synchronization (Ch 5.1-5.8)
    - 3.1 Motivation: an example
        - The critical-section problem
    - 3.2 Synchronization hardware:
        - TestAndSet
        - Swap instruction

3.3 Semaphores: synchronization tool
- Three operations
- Definition of semaphore primitives
- Mutual exclusion using semaphores
- Solving synchronization problems using semaphores

3.4 Monitors: Concept

3.5 A case study: Cats-Mice Problem

4. Projects: OS/161

4.1 Thread questions
- What happens to a thread when it exits (i.e., calls `thread_exit()`)? What about when it sleeps?
- What function(s) handle(s) a context switch?
- How many thread states are there? What are they?
- What does it mean to turn interrupts off? How is this accomplished? Why is it important to turn off interrupts in the thread subsystem code?
- What happens when a thread wakes up another thread? How does a sleeping thread get to run again?

4.2 Scheduler questions
- How does that function pick the next thread?
- What role does the hardware timer play in scheduling?

4.3 Synchronization Questions
- What is the purpose of the argument passed to `thread_sleep()`?
- Why does the lock API in OS/161 provide `lock_do_i_hold()`, but not `lock_get_holder()`?