Review Topics for Exam #3, CS 306 Spring 2015

Questions may be asked about *any* material that was covered in class OR
is in the assigned portions of the texts. See Readings on the course
webpage to determine what portions of the texts and slides to read and
study, on the following topics:

1. Concurrent Computing
2. System Programming: Processes
3. System Programming: IPC & Pipes

The exam will be about 50% short answer and 50% coding questions.

Coding question(s) will be related to mainly Lab #3 topics:

- -- creating subprocesses
- -- collecting subprocesses and determining status
- -- setting up and executing exec calls
- -- inheritance of attributes across fork() and exec calls
- -- pipe creation and usage

Detailed Topic Overview:
---
1) Concurrent Computing

- -- Basic concurrent computing concepts:
  threads of execution, concurrency vs. parallelism, processes vs. OS threads
- -- Terminology: synchronous, asynchronous/asynchrony, synchronization,
  race condition, atomic/atomicity, shared resource, critical section,
  mutual exclusion, busy wait, deadlock, semaphore, mutex (binary semaphore).
- -- Advantages and disadvantages of concurrent programs
- -- Race conditions: must be able to recognize in simple concurrent programs
- -- Producer-consumer problems: basic form and need for synchronization

2) Processes

- -- Key process attributes: PID, PPID, real vs. effective user IDs and group IDs,
  address space, signal disposition, etc.
- -- Parent and child processes: relations and common/inherited process attributes.
- -- System calls: fork(), wait() (plus its macros), waitpid(), exit(), kill().
- -- Standard syntax for creating subprocess: 3 cases, checking fork() return,
  calling different functions/code in parent vs. child, or using exec().
- -- Parent collecting child: reasons for doing so, standard approaches using
  wait()/waitpid(), obtaining child exit status, for synchronization.
- -- Zombies and orphans: meaning, problems with, how to avoid, how resolved.

3) IPC & Pipes

- -- What are IPC mechanisms and what range of properties do they have.
- -- Pipes as IPC mechanisms and relationship to shell pipeline commands.
- -- Basic properties of pipes as IPC mechanism:
  one way communication, related processes only, byte stream, finite size
- -- Pipes vs. FIFOs (named pipes)
- -- System calls: pipe(), mkfifo().
- -- Standard pipe setup and reasons: close() unused ends, required
  to detect closure of other end (e.g., read() end-of-file return),
  may also need to set SIGCHLD to be ignored.
- -- Using read()/write() with pipes/FIFOs.