Midterm Solution

1 Multiple Choice Questions

1. Only (c) is correct. Traps and interrupts are not the same: traps are "software-initiated", i.e. as a result of the execution of a program (divide-by-zero, system call); interrupts are "hardware-initiated" (I/O completion, clock interrupt).

2. (b), (d) are correct.

3. (a), (d) are not equivalent. The differences between condition variables and semaphores were examined in Homework 3. Semaphores (i.e. counting semaphores, the most general type) and *binary semaphores* are equivalent, but a lock is more restrictive than a binary semaphore (value must be initialized to 1).

4. (b), (c) are correct.

5. Only (a) is correct. Process IDs are unique to each process, so they cannot be inherited. A newly-forked process has no children, it does not inherit its parent's children. The new process gets a copy of its parent's address space, but a new page table (i.e. it does not share the address space with its parent).

6. Only (c) is correct. There are situations where reducing the time quantum can *increase* the average turnaround time for a set of processes.

7. Only (a) is correct. Any correct `find()` will allow a safe sequence to be discovered; however, no safe sequence does not imply a deadlock exists.

8. (a), (b), (c) are correct. Killing one process should be interpreted as also releasing the resources it holds.

2 Process Synchronization

(a) 35, 51, 52. n.b. The statements `x = x+1`, `x = y+1` are not atomic!

(b) 51, 52

(c) 52

(d) 35, 51, 52

(e) There is also a possibility of deadlock. If no deadlock occurs, then 51, 52.

3 Processes and Threads

1. For the PCB, some acceptable answers were: process ID, list of threads belonging to process, list of child processes, list of open files, address space state (i.e. page table pointers, etc). For the TCB, thread ID, program counter, stack pointer, register state were acceptable (the pointer to the process was already mentioned and so was not acceptable).

2. Kernel-level threads are more appropriate. With user-level threads, the entire process will block if one thread does a blocking I/O operation; since kernel threads are scheduled by the kernel, they can block independently.

4 Scheduling

1. We were looking for the simplest scheduling algorithm to solve a particular problem, e.g. asserting that (a) could be solved with priorities and the right settings is more complex than necessary, and doesn't ensure regular feedback to the user.

   (a) Round-robin, since there are interactive jobs in the system.

   (b) FCFS, since there is no requirement mentioned for anything more sophisticated.

   (c) Priorities with preemption.

   (d) Nonpreemptive SJF, since it provably minimizes waiting time in this situation. Preemptive SJF was disallowed, there is no mention in the question that preemption would be advantageous: the arrival times aren't mentioned, and it is not stated if preemption of jobs is even possible.

2. (a) We'll give triples of $(\text{start}, \text{end}, \text{process-name})$:
   (0, 2, P1), (2, 4, P2), (4, 6, P3), (6, 8, P4), (8, 10, P1), (10, 12, P2), (12, 14, P4), (14, 16, P1), (16, 18, P2), (18, 20, P4), (20, 22, P1), (22, 24, P2), (24, 25, P4), (25, 28, P1).

   (b) (0, 11, P1), (11, 13, P3), (13, 20, P4), (20, 28, P2).

   (c) Remember to subtract the arrival time from the completion time!
   For round-robin: $(28 + 23 + 5 + 22) \div 4 = 19.5$.
   For SJF: $(11 + 27 + 12 + 17) \div 4 = 16.75$.
   It follows that SJF has the shorter average turnaround time.

5 Deadlocks

1. (a) No deadlock: one process cannot be in a deadlock with itself (no circular wait).

   (b) No deadlock: there is starvation, but that is not the same as deadlock.

   (c) No deadlock: there is no circular wait. One of the two processes will get 3 printers and be able to terminate.

   (d) Deadlock possibility: Each process could acquire one printer, and then try and get an additional two printers.

   (e) No deadlock: hold-and-wait cannot be satisfied, since it requires that a process hold one resource and wait for another, but each process needs only one printer.

2. False, a cycle does not necessarily imply a deadlock if resources have multiple instances.