Lab 3

Jason Tov

1. The file no longer exists, so user 2 will receive an error when it tries to read the file which no longer exists.
2. Multiprogramming would increase CPU utilization when there are multiple processes running with at least some of them being IO-bound. For example, a terminal process will spend most of its time waiting for user input. While that process is waiting, another process can run in the meantime so that the CPU isn't spending most its time waiting.
3. 500,000,000,000/1,000 = 5,000,000,000 system calls/sec
4. A race condition is any situation where the a program's output depends on the ordering of the parallel processes or threads. Some symptoms of race conditions are non-deterministic behavior where a program run on one machine will give a different result than that same program run on a different machine.
5. Assuming each CPU has its own cache, the busy waiting solution does not work because the `turn` variable gets cached into each CPU. When turn gets cached, without any other special monitoring, each CPU will maintain its own state of `turn` and reads and writes to `turn` will not propagate to the other CPU, meaning that it will deadlock because they cannot talk to each other via `turn`. If the CPU cache is optimized for this situation and correctly propagates changes, then this issue may still occur with the registers if the compiler does not push the register update to the shared memory.
6. To implement semaphores, when a thread calls a semaphore instruction, the thread should be atomic and be the only one running. This means that it cannot block or be interrupted until it finishes the down operation.
   DOWN(): First, control is handed to the OS, interrupts are disabled, then the semaphore value can be checked. If it is available, the semaphore value is downed, interrupts are re-enabled, and control is given back to the thread. If it is not available, then the OS puts the thread to sleep, schedules the next thread to run, re-enables interrupts, then hands control to the next thread.
   UP(): First, control is handed to the OS, interrupts are disabled, then the semaphore value is upped. Any threads that were asleep for that semaphore are woken back up and rescheduled, interrupts are reenabled, and control is handed to the next thread in the queue be it the UP() thread or one of the waiting threads.
7. If a process appears twice in the round robin scheduler list, it would simply be run twice as often as processes that only appear once. This essentially makes that process higher priority, run more often, and finish faster. Yes, there are situations where this would be fine, such as very important background tasks that should be run often.
8. Problem 8

1. Round Robin: ABCDE ABCDE ABCDE ACDE ADE ADE AD A A A
   B finishes after 12 seconds
   C finishes after 17 seconds
   E finishes after 25 seconds
   D finishes after 27 seconds
   A finishes after 30 seconds
   Average turnaround time: 22.2 seconds
2. Priority: BBB EEEEEE AAAAAAAAAA CCCC DDDDDDD
   B finishes after 3 seconds
   E finishes after 9 seconds
   A finishes after 19 seconds
   C finishes after 23 seconds
   D finishes after 30 seconds
   Average turnaround time: 16.8 seconds
3. First Come First Serve: AAAAAAAAAA BBB CCCC DDDDDDD EEEEEE
   A finishes after 10 seconds
   B finishes after 13 seconds
   C finishes after 17 seconds
   D finishes after 24 seconds
   C finishes after 30 seconds
   Average turnaround time: 18.8 seconds
4. Shortest Job First: BBB CCCC EEEEEE DDDDDDD AAAAAAAAAA
   B finishes after 3 seconds
   C finishes after 7 seconds
   E finishes after 13 seconds
   D finishes after 20 seconds
   A finishes after 30 seconds
   Average turnaround time: 14.6 seconds

9. In my notation, (D1.5) means process D runs and has finished 1.5 out of its 7 second run time and has just blocked
   ABC(D0.5)E ABC(D1.0)E ABC(D1.5)E AC(D2.0)E A(D2.5)E A(D3)E A(D3.5) A(D4.0) A(D4.5) A(D5.0) (1)(D5.5) (1) (D6.0) (1) (D6.5) (1) (D7)
   B finishes after 11 seconds
   C finishes after 15.5 seconds
   E finishes after 22 seconds
   A finishes after 28.5 seconds
   D finishes after 35 seconds
   Average turnaround time: 22.4 seconds
10. A CPU bound process will not block, so each time the process begins running, it will run out its quanta. Assuming quanta begins at 1, each time it finishes its quanta, it gets rescheduled

with double the quanta. So its quantas each time it is scheduled is 1, 2, 4, 8, 16. In its 16-quanta run, it completes, so it is called a total of 5 times.