Delaplaine, Victor
Cpe 453, Section 01
Lab 3
February 3, 2020

Lab 3: Exercise 3

1.)In Minix (or any other Unix), if user 2 links to a file owned by user 1, then user 1 removes the file, what happens when user 2 tries to read the file? (Tanenbaum and Woodhull, Ch. 1, Ex. 15)

Nothing the file still exists, because there were two links in the inode structure. To delete the "file", the link number has to be at one.

(2 - come back to) 2.)Under what circumstances is multiprogramming likely to increase CPU utilization? Why?

Under the case where one process is currently blocked (waiting for io) and the other process is using the CPU. Because one is waiting for I/O and the CPU is free.

3.)Suppose a computer can execute 1 billion instructions/sec and that a system call takes 1000 instructions, including the trap and all the context switching. How many system calls can the computer execute per second and still have half the CPU capacity for running application code? (TW 1-21)

$$\frac{10^9 instr}{2s} * \frac{1 syscall}{1000 instr} = \frac{500000 syscalls}{s}$$

4.)What is a race condition? What are the symptoms of a race condition?(TW 2-9)

When two/more processes are reading/wrtiing some shared resource and the final results depend on who runs. Symptoms: Unexpected ouputs such as overwriting another processes action

5.)Does the busy waiting solution using the turn variable (Fig. 2-10 in TW) work when the two processes are running on a shared-memory multiprocessor, that is, two CPUs, sharing a common memory? (TW, 2-13)

Yes, it does work. Although, it violates condition 3 in TW book that says one process must not block the other when its not in critical region.

(come back) 6.)Describe how an operating system that can disable interrupts could implement semaphores. That is, what steps would have to happen in which order to implement

the semaphore operations safely. (TW, 2-10)

I would implement a semaphore as below:
struct sema$_t Queue < Process > que; intcount; sema$;
And follow the steps for UP and DOWN where interrupts are disabled:
UP: if sema.que.length != 0: wakeup(sema.que.poll()) else: sema.que.count++
DOWN: if sema.count==0: sleep() else: sema.count–;

7.)Round robin schedulers normally maintain a list of all runnable processes, with each process occurring exactly once in the list. What would happen if a process occurred twice in the list? Can you think of any reason for allowing this? (TW, 2-25) (And what is the reason. "Yes" or "no" would not be considered a sufficient answer.)

It would just be executed more than the other processes. Maybe in a timesharing system, user1 and user2 has been promsied 50% of the CPU. Lets say user1 has runnable processes A,B and user2 has runnable process C. So the sequence of scheduling would be: A C B C

8.)Five batch jobs, A through E, arrive at a computer center, in alphabetical order, at almost the same time. They have estimated running times of 10, 3, 4, 7, and 6 seconds respectively. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the time at which each job completes and the mean process turnaround time. Assume a 1 second quantum and ignore process switching overhead. (Modified from TW, 2-28)

(a) Round robin.
(b) Priority scheduling.
(c) First-come, first served (given that they arrive in alphabetical order).
(d) Shortest job first.

for (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b)–(d) assume that only one job at a time runs, and each job runs until it finished. All jobs are completely CPU bound.

(a) Round robin.
A = 30s
B = 12s
C = 17s
D = 27s
E = 25s
AVG = 22.2s
(b) Priority scheduling
B = 3s
E = 3s+6s = 9s

A = 3s+6s+10s=19s
C = 3s+6s+10s+4s=23s
D = 3s+6s+10s+4s+7s=30s
AVG = 16.8s
(c) First-come, first served (given that they arrive in alphabetical order).
A = 10s
B = 10s+3=13s
C = 10s+3+4=17s
D = 10s+3+4+7=24s
E = 10s+3+4+7+6=30s
AVG = 18.8s
(d) Shortest job first
B = 3s
C = 7s
E= 7s+6 = 13s
D=7+13=20s
A=10+20=30s
AVG = 14.6s

9.)Re-do problem 8a with the modification that job D is IO bound. After each 500ms it is allowed to run, it blocks for an IO operation that takes 1s to complete. The IO processing itself doesn't take any noticeable time. Assume that jobs moving from the blocked state to the ready state are placed at the end of the run queue. If a blocked job becomes runnable at the same time a running process's quantum is up, the formerly blocked job is placed back on the queue ahead of the other one.

(a) Round robin.
B=12s
C=17s
D=21s
A=28s
E=24s
AVG = 20.4s

10.)A CPU-bound process running on CTSS needs 30 quanta to complete. How many times must it be swapped in, including the first time (before it has run at all)? Assume that there are always other runnable jobs and that the number of priority classes is unlimited. (TW, 2-29)

5 times