**Solution**

Spring 2021 CSC 332 HW3 (ch4- Process Management)

**Due April 19 11:59 PM.**

**Q1 to be done by people whose first letter of last name is A-M.**

**Q2 to be done by all others.**

Please upload your file as a WORD or Notepad file, rather than typing in the Blackboard system.

Consider process management with RR as discussed in ch 4.

Suppose we did the same shell command that created the 4 processes for ./cpu

Make these assumptions:

1. There is only one cpu.
2. Initially all the 4 user processes are in ready list in the order A, B, C,D.
3. There are only 5 processes in the system—A, B, C, D, and the shell.
4. Some parts of a process may be on disk.
5. The only interrupts in the system are the Hardware Timer and interrupts caused by this program.
6. **The cpu scheduler:**

The cpu scheduler strictly follows the RR algorithm, with time slice = 5 seconds.

When a ready proc gets cpu, it gets a fresh time slice of 5 seconds. This slice has nothing to do with how much time it used last time when it was executing.

1. When a proc terminates (normal termination or runtime abortion), an interrupt is generated and a service routine is called. The service routine will do some work such as updating state of the process etc. and the cpu scheduler gives cpu to another process. Any unused time of the process that just terminated, is gone and is irrelevant.
2. The spin(1) in the code will take roughly 1 second of computation time. Any other computation will take nearly zero time, ex. the initial if statement, checking the loop condition etc.
3. Once a process executes the printf statement in cpu.c, it gets blocked until that output text line has been printed. One print operation of a text line takes 1 second. The output device prints things in strictly First-come-first-served order.
4. When an i/o is finished, the corresponding process gets to ready state.

Q1. Is it possible to get a printout like:

A

C

….

In other words, we would normally expect a ‘B’ after the ‘A’. What might have happened?

**Answer: One possibility of what might have happened to get AC...:**

**A made i/o request to print A and got blocked.**

**Then B started execution. But during its spin(1) some needed piece of logical space was not in physical memory. Ex1. the code of spin function may be on disk. Ex2. the needed stack for the call to spin(1) may be unallocated, i.e., neither in physical memory nor on the disk. So OS issued request to get that piece from disk to physical memory or allocate memory to that logical space, and gave cpu to C.**

**Then C made i/o request for C.**

**So i/o requests were in the order AC, and they were printed in FCFS order.**

**Thus we did not print B yet; only printed AC.**

Q2. Give details of a scenario (sequence of events) where we get the following output; in particular, explain why in your scenario we are not getting two A’s one after the other.

A

B

C

…

**Answer: the scenario is:**

**Proc A gets CPU. It does spin for 1 sec. It requests i/o and gets blocked.**

**Proc B gets CPU. It does spin for 1 sec. It requests i/o and gets blocked.**

**Proc C gets CPU. It does spin for 1 sec. It requests i/o and gets blocked.**

**…**

**Obviously in this scenario we do not get AA. Before process A gets to the second iteration of the loop, it gets blocked and B gets started and prints a B. So AA is not printed.**

**We are not getting AA in this scenario because Proc A gets blocked when it requests its first output of A. Then B will start and does its output. So Proc A does not get a chance to do a second output before others. So the i/o requests are in the order AB and not AA. The printing is in FCFS order so the second A will not be printed before the first B.**