Md Shahid Emdad

CSC 33200

11-11-21

**HW 03**

Make these assumptions:

1. There is only one cpu.
2. There are only 5 processes in the system—A, B, C, D, and the shell.
3. Initially all the 4 user processes and the shell are in ready list in the order A, B, C,D, 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 execution of these processes.
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 has already executed before becoming ready.

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, executing a service routine, 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.

**Q2**. Is it possible to get a printout like:

A

A

….(some unknown stuff)

**Answer:**

It is possible to get a printout as provided in the question description. One possible way to get AA is to do the following steps:

First, we assume A made i/o request to pring A and got blocked. Then, B started its execution. However, while its spin(1) happening, some necessary piece of logical space were not present in the physical memory. This might be the reason for the code of spin function not located on the disk or completely unallocated. Therefore, the OS issued the request to get the information from the disk to physical memory or allocated memory to that logical space and giving CPU to the next process C.

Now, assume C and D process also suffered from the same disruption as process B and OS returned CPU back to process A. Following that, inside the second iteration of the loop, we returned back in A which made another i/o request to print A but get blocked and process B execute again.