CS3500: Operating Systems

Lab 5

30-August-2019

State transition	**How triggered**
Created	After G creates W, it signals S and shares the process details (m, p, t, n) on a named shared memory
Created → Ready	After W starts, it signals S
Ready → Running	The ready state is immediately changed to Running (we will see later the difference)
Running → Waiting	In the function `io_emulation`() when W signals S before sleeping, running state is changed to Waiting
Waiting → Ready	In the function `io_emulation`() when W signals S after sleeping, running state is changed to Ready
Running → Complete	Before exiting W signals S

1. (4 points) In the above table (which you have done in the tutorial so far), the Ready state immediately moved to the Running state. We will now emulate a situation, where W does not necessarily run when in the Ready state. To achieve this, W must voluntarily yield at specific points of the code, but be ready to execute if signalled by S. This exact behavior is provided by the pause() syscall. W must pause at two specific situations:

   (a) When it is created and it signals S for the first time.

   (b) When it is back from a sleep in `io_emulate` operation.

   In both these cases, W voluntarily yields. It is now the duty of S to decide when to move W from Ready to Running. In this question, we will use a simple rule: Whenever S changes the state of W to Ready, it waits for 1 second and then signals W and changes its state to Running. Upon receiving the signal W would continue. (Now S is emulating some semblance of scheduling.)

   Modify S, W, and G to enable the above.

2. (4 points) If W has no IO operation, then its state always remains 'Running' until it completes. We would like to empower S to 'preempt' W whenever it chooses to. We can do this with the special signal SIGSTOP (check the man page). To resume operation the corresponding signal SIGCONT can be used. To simulate such preemption, add an additional rule in S that if W is in the Running state for 1 second, then a SIGSTOP is sent to it and the corresponding SIGCONT is sent after another 1 second. (Now we have a scheduler which can preempt.)

3. (2 points) Given m = 65,536, p = 0.1, t = 1s, n = 1,000. Create W and schedule it with S to run in a time-triggered schedule such that it is only allowed to execute in fixed time slots given in seconds as: [t, t+1], [t+3, t+4], [t+6, t+7], [t+9, t+10], ..., where t is the start time of the first execution of W. In S measure and report the turnaround time (= completion time - start time) for the execution of W.