OS Lab #2: Scheduling

Allan Gottlieb

Spring Semester: 2021-22

1 Introduction

In this lab you will simulate the scheduling of n processes $P_1, P_2, ..., P_n$ on a single CPU. You will simulate both FIFO and RR scheduling.

2 Trivial Case $n = 1$

Process P_1, is run alone (i.e., it is the only process). There would be four phases ($R1_1$, $B1_1$, $R2_1$, and $B2_1$) followed by a final "termination phase".

1. P_1 would first run for time $R1_1$ (presumably initializing its data structures, but the reason is not relevant to the lab).
2. P_1 would then block for time $B1_1$ (presumably reading its input).
3. P_1 would then run for time $R2_1$ (presumably computing its results).
4. P_1 would the block for time $B2_1$ (presumably outputting its results).
5. Finally, P_1 would run for time 1 (just to terminate).

3 Creation Time and Breaking Ties

Creation Time For simplicity all processes are created at time 0. No input is required to specify this rule

Breaking Ties Use our standard tie-breaking rule. Since all processes are created at the same time, ties are broken by favoring the alphabetically first process. No input is needed to specify this rule.

4 Input

All time units are milliseconds. Each simulation begins by reading the parameters for the current run, Specifically

n The number of processes.

SA The scheduling algorithm to use. 1 signifies FIFO; 2 signifies RR with quantum q.

q The quantum for RR scheduling. q is present only if SA=2.

$\mathbf{R1_1}, \mathbf{B1_1}, \mathbf{R2_1}, \mathbf{B2_1}, ..., \mathbf{R1_n}, \mathbf{B1_n}, \mathbf{R2_n}, \mathbf{B2_n}$ The four running and blocking times for each process. Note that all four values for P_i are given before any values for P_{i+1}. Remember that each P_i runs a third time, in order to terminate. Since that last run is always for 1 time unit, no input is needed for its specification.

5 Output

At each cycle indicate which processes (if any) are running, which are ready, and which are blocked. See the two sample runs below.

6 Breaking Ties

Use our standard tie-breaking rule. Since all processes have the same creation time, the alphabetically first process wins the tie.

7 Computer Language

All 202 labs must be in the C (or C++) language.

8 A Simple, But Not Trivial, Example

8.1 Input

n=2 There are 2 processes, $P1$ and $P2$.

SA=1 FIFO scheduling (hence, no preemption and q is not present).

Rs and Bs All Rs are 3 and all Bs are 2.

8.2 Input As Would be Specified To Your Program

2 1 3 2 3 2 3 2 3 2
Note that the order is $n, SA, R1_1, B1_1, R2_1, B2_1, R1_2, B1_2, R2_2, B2_2$.

8.3 A Cycle by Cycle Description of Execution

Your format is not required to match what follows (which was done by hand).
Note that cycle n goes from time $n-1$ms to time $n$ms.

Cycle	**P1 State**	**P2 State**	**Comment**
1	Run (1 of 3)	Ready	Both created; P1 wins tiebreak
2	Run (2 of 3)	Ready	
3	Run (3 of 3)	Ready	
4	Blocked (1 of 2)	Run (1 of 3)	
5	Blocked (2 of 2)	Run (2 of 3)	
6	Ready	Run (3 of 3)	
7	Run (1 of 3)	Blocked (1 of 2)	
8	Run (2 of 3)	Blocked (2 of 2)	
9	Run (3 of 3)	Ready	
10	Blocked (1 of 2)	Run (1 of 3)	
11	Blocked (2 of 2)	Run (2 of 3)	
12	Ready	Run (3 of 3)	
13	Terminate	Blocked (1 of 2)	
14		Blocked (2 of 2)	
15		Terminate	

9 An Example With Preemption

9.1 Input

n=2 There are 2 processes, $P1$ and $P2$.

SA=2 RR scheduling.

q=2 RR quantum is 2ms.

Rs and Bs All Rs are 3 and all Bs are 2.

9.2 Input As Specified To Your Program

2 2 2 3 2 3 2 3 2 3 2
Note that the order is $n, SA, q, R1_1, B1_1, R2_1, B2_1, R1_2, B1_2, R2_2, B2_2$.

9.3 A Cycle by Cycle Description of Execution

Your format is not required to match what follows (which was done by hand).
Note that cycle n goes from time $n-1$ms to time $n$ms.

Cycle	P1 State	P2 State	Comment
1	Run (1 of 3)	Ready	Both created; P1 wins tiebreak
2	Run (2 of 3)	Ready	
3	Ready	Run (1 of 3)	P1 preempted
4	Ready	Run (2 of 3)	
5	Run (3 of 3)	Ready	P2 preempted
6	Blocked (1 of 2)	Run (3 of 3)	P1 blocks
7	Blocked (2 of 2)	Blocked (1 of 2)	CPU idle
8	Run (1 of 3)	Blocked (2 of 2)	
9	Run (2 of 3)	Ready	
10	Ready	Run (1 of 3)	P1 preempted
11	Ready	Run (2. of 3)	
12	Run (3 of 3)	Ready	P2 preempted
13	Blocked (1 of 2)	Run (3 of 3)	
14	Blocked (2 of 2)	Blocked (1 of 2)	CPU idle
15	Terminate	Blocked (2 of 2)	
16		Terminate	