CSE 421/521 – Operating Systems **Sassi Kesavan**
Fall 2018 - Homework Assignment #1 **UBIT: sassikes**

Problem 1:

Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority number implies higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α; when it is running, its priority changes at a rate β. All processes are given a priority of 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.

a) What is the algorithm that results from $\beta > \alpha > 0$?

b) What is the algorithm that results from $\alpha < \beta < 0$?

Solution a: When the first process enters the ready queue with 0 priority, it gets the CPU immediately and begins to run. When it is running its priority increases at the rate of β. By that time, if any other process comes into the queue with 0 priority it must wait till the process that is currently running is complete. While waiting the priority of the process increases at the rate of α. Since, β has the higher increment rate than α, the process that will be executed will continue as it will have to higher priority and the process which has waited long in the queue will be executed next. This process looks like First come First Serve (FCFS).

Solution B: In this case, since the order of increment is exactly in reverse, the newer process that enters the ready queue will get higher priority than the one that is executing. So, this may probably be the reverse of FIFO which is Last in First Out (LIFO).

Problem 2:

Consider a system running ten 1/O-bounds tasks and one CPU-bound task. Assume that the I/O bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and all processes are long-running tasks. What is the CPU utilization for a round-robin scheduler when:

a) The time quantum is 1 millisecond

b) The time quantum is 10 milliseconds

Solution:

a) *As the scheduler has a 0.1 millisecond context-switching cost for every context-switch, the utilization is*

   *(1/1.1) * 100 = 90.9%*

b) *The I/O-bound tasks has a context switch after using up only 1 millisecond of time quantum. The time required to cycle through all the processes is 10*1.1 + 10.1. The utilization is,*

   *(20/21.1) * 100 = 94.79%*

Problem 3:

Which of the following scheduling algorithms could result in starvation? Explain the reasoning.
a. First-come, first-served
b. Shortest job first
c. Round robin
d. Priority

a) *First Come First Serve: Since, the processes execute in the order of the entry in the ready queue, there is no chance of starvation in this algorithm.*
b) *Shortest job First: Since, the shortest job gets the higher priority, the CPU will continue to execute the shortest job in the ready queue. Due to this, there is chance that the process whose burst time is higher tends to wait in the queue for long time. This results in starvation.*
c) *Round Robin: In this method, since the context switching occurs for every time quantum the response time will be very less. So, there will not be any starvation.*
d) *Priority: Since, the process with the higher priority executes continuously, the process with the lowest priority tends to wait longer resulting in starvation.*

Problem 4:

Consider the deadlock situation that could occur in the dining philosopher's problem when the philosophers obtain the chopsticks one at a time. Discuss how the four necessary conditions for deadlock indeed hold in this setting. Discuss how deadlocks could be avoided by eliminating any one of the four conditions.

Mutual Exclusion: When a philosopher picks up one chopstick, it cannot be shared with others.

Hold and Wait: When the philosopher tries to pick up a chopstick, he only picks up one at a time.

No preemption: Once a philosopher picks up a chopstick, it cannot be taken away from him.

Circular wait: As all the philosophers are sitting in a round table and each philosopher has access to the chopsticks next to them, if a philosopher picks up one chopstick, the philosopher sitting next to him will have to wait until he drops it. The philosopher on that side can also affect the philosopher sitting next to her in the same manner. This holds true all the way around the table.

If any one of the above situations do not exist, it could prevent deadlock.

Problem 5:

Consider the dining-philosophers problem where the chopsticks are placed at the center of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

Solution:

If (number of philosophers with two chopsticks==0 && number of available chopsticks==1)

{

//Do not allow the request.

}

Problem 6:

Show that, if the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated.
Solution:

If two operations are not executed atomically and if two wait operation is executed on the semaphore whose value is 1, they both will go ahead and decrement the value by violating the mutual exclusion.