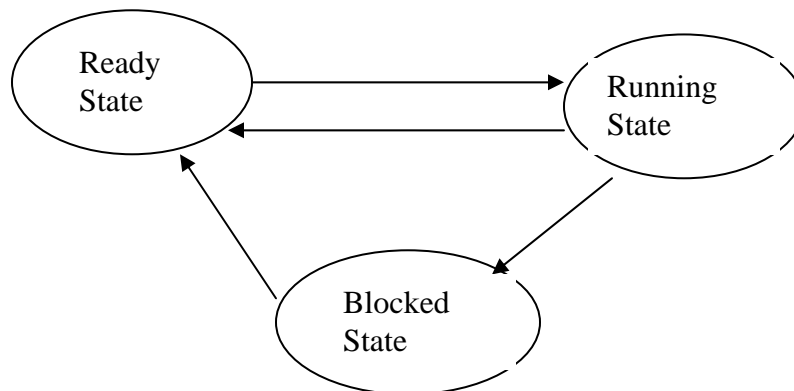


## Shadow Homework 2 – System Programming (Processes, Threads, Synchronization and Scheduling)

Posted February 15, 2007 (NOT GRADED)

Shadow Homework 2 on Processes, Threads, Synchronization and Scheduling

**1. Problem: Draw a state transition diagram with three process states “Ready” “Running” and “Blocked”. Indicate allowed transitions between these states.**



Answer: Process in ready state can only be moved to running state. Process in running state can be preempted either due to time slice/time sharing into the ready state or due to I/O system call/semaphore/signals into the blocked state. Once the blocking condition is removed, the process will be moved from the blocking to the ready state.

**2. Give at least two differences and two similarities between an application that is implemented using multiple processes and one that is implemented using multiple threads. Your answer should emphasize the performance and resource issues in each case.**

Answer: Differences: (1) Application implemented using multiple processes (e.g., producer and consumer processes) will not have implicitly shared memory between the multiple processes, since each process has its own address space. This means that if two processes want to share an information, they either explicitly need to specify share memory segment (specific system calls need to be used) or they use other inter-process communication mechanism such as pipes, sockets, files, etc (we will cover some of them in the second half of the semester) . Application implemented using multiple threads within the same process (e.g., web server threads) will share the memory explicitly among the threads. No specific commands/mechanisms are needed to share the memory among the threads. If threads are implemented in different processes, the threads will have different memory space. (2) Multiple processes have their own signal handlers,

multiple threads in a process have joint signal handlers, since signal handlers are defined per process.

Similarity: (1) each process and each thread have control block that holds program counter; (2) each process and each thread have their own priorities.

**3. List and explain four resources that must be saved and restored during a process context-switch.**

Answer: (1) memory (registers for page tables, ...), (2) disk - I/O files , (3) network connections, (4) process control block that holds all the state information about memory, priorities, program counter, stack, I/O, accounting, etc.

**4. Explain what is the problem with this implementation of the one-writer many-readers problem?**

(Note, “Up” is equivalent to “sem\_post” & “Down” is equivalent to “sem\_wait”)

```

int readcount;           // shared and initialized to 0
Semaphore mutex, wrt;   // shared and initialized to 1;

// Writer :
Down(wrt);
/* Writing performed*/
Up(wrt);

// Readers :
Down(mutex);
readcount := readcount + 1;
if readcount == 1 then Down(wrt);
Up(mutex);
/*reading performed*/
Down(mutex);
readcount := readcount - 1;
if readcount == 0 then Up(wrt);
Up(mutex);

```

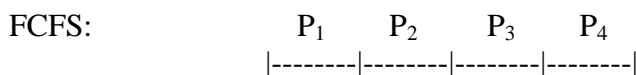
Answer: The code for the one-writer many readers is fine if we assume that the readers have always priority. The problem is that the readers can starve the writer(s) since they may never all leave the critical region, i.e., there is always at least one reader in the critical region, hence the ‘wrt’ semaphore may never be signaled to writers and the writer process does not get access to ‘wrt’ semaphore and writes into the critical region.

**5. Consider the following table about processes:**

Process	Priority*	Burst Time	Arrival Time
P <sub>1</sub>	4	100ms	0ms
P <sub>2</sub>	2	20ms	30ms
P <sub>3</sub>	1	50ms	40ms
P <sub>4</sub>	3	5ms	60ms

\* A smaller priority number means a higher priority process

The schedule for First Come First Served is shown below



0    100    120    170    175

**a) Explain the term Average Waiting Time.**

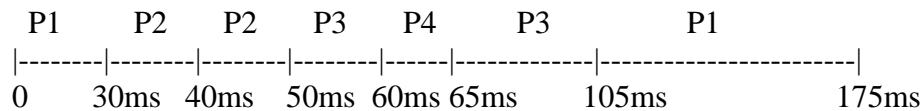
Answer: Average waiting time is a scheduling performance metric that indicates how long on average each process in a considered snapshot waits in the ready queue. In our example (see the table above), the average waiting time will indicate how long on average each of the 4 processes under the specific scheduling policy will wait in the ready queue.

**c) What is the most important characteristic of Shortest Job First scheduling?**

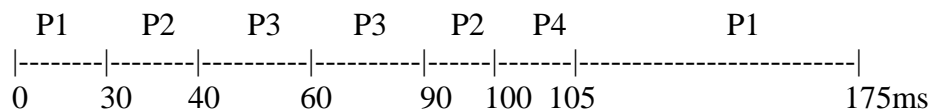
Answer: The most important characteristic of SJF scheduling policy is the burst time of each process, since SJF policy considers process selection based on the burst time of a process. Also, if all processes arrive at the same time into the ready queue, and their burst times are known, SJF policy is optimal in its selection.

**d) Edit the following skeleton schedules for the process table to show the scheduling disciplines:**

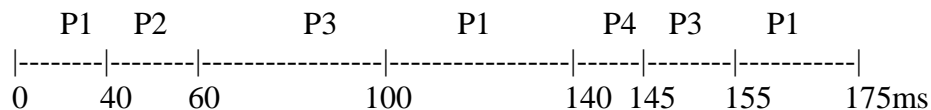
**i) Preemptive Shortest Job First**



**ii) Preemptive Priority**



**iii) Round Robin (with quantum 40ms)**



Note for Round-Robin: At the time 40ms the new arrived process P3 is queued in the ready queue before the preempted process P1 (this is an assumption – it is important that you specify assumptions for any conflict situation in your system – in this case what order you consider for new and old processes that either arrive new or are preempted due to quantum at the same time in ready queue). Note that the process P2 is already in the ready queue at time 30ms of its arrival. So at time 40ms, the queue order is P2, P3, P1. At this point first the process P2 is scheduled for 20 ms. Once P2 is done, P3 is scheduled for 40ms (quantum) and then P1. Keep in mind that at time 60ms, P4 arrives, so the ready

queue process order at time 60 ms is P3, P1, P4. Hence, P3 runs first at time 60ms, then P1 for 40ms and then P4 for 5ms, and then the rest of P3 and P1.