

CPSC 457 L04 (Winter 2001)
Assignment 2 (7.5 marks)
CPU Scheduling

Due March 2, 2001

In this assignment, you will simulate CPU scheduling using the Shortest Job First (SJF) algorithm. You can assume that a process is represented by a variable-length record containing: (1) process id, (2) process length (number of CPU and I/O bursts), and (3) alternating CPU bursts and I/O bursts. For instance, the following process record indicates that the process id is 10 and it has 9 bursts. The CPU bursts are bold faced and are of 7, 8, 3, 6, and 1 time units. The I/O bursts has 26, 14, 10 and 6 time units.

10 9 **7** 26 **8** 14 **3** 10 **6** 6 **1**

You will have a file (call it JobQueue) containing 100 processes which will be supplied by your TA. This file simulates the processes that are on the job-queue. So, the job file is sorted by the process number in ascending order. You can assume that each time a process is admitted to the system, it is assigned an id that is higher than the maximum process id in the system. In fact, this simulates FCFS long-term scheduling.

Job (Long-Term) Scheduling

Memory can hold N processes. Initially, your long-term scheduler fills up the memory with N (processes 1 to N). When a process terminates, another process is loaded to memory from the job file (the next process in the file).

CPU (Short-Term) Scheduling

Once a process is loaded into memory, it can be allocated the CPU and start execution. Process execution is simulated as follows. You will maintain one queue, the ready queue (RQ). Once a process is loaded into memory, it is inserted into RQ. If dispatched, CPU execution is simulated by advancing the CPU clock k time units, where k is the length of the CPU burst that the just dispatched process executed. Context-switch time costs 2 time units. For example, if process 10 above has been dispatched, the CPU clock will be 9 (2 for the context-switch + 7 for the first CPU burst). Then, process 10 requests an I/O operation, a context-switch occurs and the CPU clock becomes 11. Process 10 will be waiting for I/O, and another process is switched in. For simplicity, You may assume that there are N I/O processors, so that processes do not need to wait for I/O in a queue (there is a maximum of N processes on RQ.) The CPU clock need not be advanced for I/O operations. The I/O processor interrupts the CPU on I/O completion so that the latter places the process that has been waiting for I/O on RQ. This interrupt costs the CPU 1 time unit. When the

interrupt service is completed, the CPU selects another process for execution. Finally, It costs the CPU 15 time units to load a process from disk (JobQueue) to memory.

1. Write the above program for non-preemptive SJF CPU scheduling. Estimate the next CPU burst using a three point moving average (i.e., the estimated length of the next CPU burst is the average of the previous three CPU bursts of the same process.)

Note that the O.S. does not know for sure the length of the next CPU burst. It can only guess.

2. Repeat part 1 with preemptive SJF CPU scheduling. Say P_i is being added to RQ and P_j is executing on the CPU. If P_i has an estimated next CPU burst of k units and P_j has $l > k$ units remaining in its current CPU burst. P_j must be preempted, and P_i is dispatched.

For each case above, report the following statistics for $N = 5$ and $N = 15$:

- (a) Percentage CPU utilization
- (b) Throughput measured in processes per 100 time units
- (c) Average and standard deviation for all CPU bursts
- (d) Average and standard deviation for all I/O bursts
- (e) The five processes with the highest turnaround time (potentially starving processes)