**Assignment #3**

**Chapter 5 Questions**

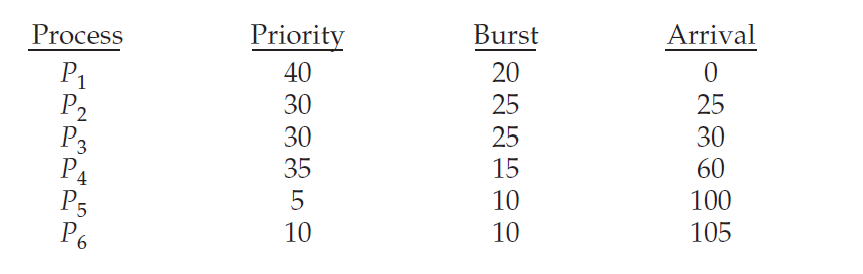
18. A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of n.

n!

19. Explain the difference between preemptive and non-preemptive scheduling.

Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Non-preemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

20. The following processes are being scheduled using a preemptive, round robin scheduling algorithm.



Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an idle task (which consumes no CPU resources and is identified as Pidle). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority  
process, the preempted process is placed at the end of the queue.  
     a. Show the scheduling order of the processes using a Gantt chart.

P1 –– idle –– P2 –– P3 –– P2 –– P3 –– P4 –– P4 –– P2 –– P3 –– idle –– P5 –– P6 –– P5  
0 10 20 25 35 45 55 60 70 75 80 90 100 115 120  
     b. What is the turnaround time for each process?

P1: 20-0=20, P2: 80-25=55, P3: 90-30=60, P4: 75-60=15, P5: 120-100=20, P6: 115-105=10  
     c. What is the waiting time for each process?

P1: 0, P2: 40, P3: 35, P4: 0, P5: 10, P6: 0  
     d. What is the CPU utilization rate?

105/120 = 85%

21. What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

Processes that need more frequent servicing, for instance, interactive processes such as editors, can be in a queue with a small time quantum. Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing, and thus making more efficient use of the computer.

22. Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on. These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?  
     a. Priority and SJF  
The shortest job has the highest priority.  
     b. Multilevel feedback queues and FCFS  
The lowest level of MLFQ is FCFS.  
     c. Priority and FCFS  
FDFS gives the highest priority to the job having been existence the longest.  
    d. RR and SJF  
None.

23. Suppose that a CPU scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

It will favor the I/O-bound programs because of the relatively short CPU burst request by them; however, the CPU-bound programs will not starve because the I/O-bound programs will relinquish the CPU relatively often to do their I/O.

24. The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

Priority = (recent CPU usage / 2) + base

where base = 60 and recent CPU usage refers to a value indicating how often a process has used the CPU since priorities were last recalculated.

Assume that recent CPU usage for process P1 is 40, for process P2 is 18, and for process P3 is 10. What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

The priorities assigned to the processes are (40/2)+60=80, (18/2)+60=69 , and (10/2)+60=65 respectively. The scheduler lowers the relative priority of CPU-bound processes.

25. Using the Windows scheduling algorithm, determine the numeric priority of each of the following threads.  
     a. A thread in the REALTIME PRIORITY CLASS with a relative priority of NORMAL  
24

     b. A thread in the ABOVE NORMAL PRIORITY CLASS with a relative priority of HIGHEST  
12  
     c. A thread in the BELOW NORMAL PRIORITY CLASS with a relative priority of ABOVE NORMAL  
7