**6.10**

I/O-bound programs usually run for short periods while CPU-bound programs run longer. If a CPU-bound program begins, all other I/O-bound programs are stuck waiting for it to finish. Therefore, if the CPU scheduler gives higher priority to I/O-bound programs the virtual run time will eventually be lower.

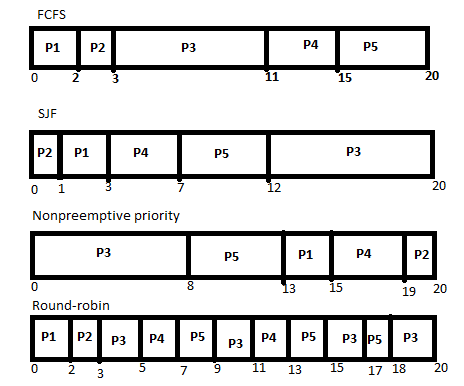
**6.11**

(a) CPU utilization can be sometimes be maximized by minimizing the average response time and performing infrequent context switches. However, inconsistent response times are often less desirable in a system. Therefore, it can be best to minimize the maximum response times, decreasing CPU utilization but maintaining consistent response times.

(b) Using the shortest job first scheduling algorithm minimizes the average turnaround time and the average waiting time. However, this would increase the maximum waiting time of processes that take longer to run.

(c) Improving CPU utilization would mean prioritizing CPU-bound programs and allow them to run non-preemptively which would starve I/O-bound programs. On the other hand, prioritizing I/O-bound programs would increase the number of context switches which would decrease CPU utilization.

**6.16**

1. 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (b) | **P1** | **P2** | **P3** | **P4** | **P5** |
| **FCFS** | 2 | 3 | 11 | 15 | 20 |
| **SJF** | 3 | 1 | 20 | 7 | 12 |
| **NPP** | 15 | 20 | 8 | 19 | 13 |
| **RR** | 2 | 3 | 20 | 13 | 18 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (c) | **P1** | **P2** | **P3** | **P4** | **P5** |
| **FCFS** | 0 | 2 | 3 | 11 | 15 |
| **SJF** | 1 | 0 | 12 | 3 | 7 |
| **NPP** | 13 | 19 | 0 | 15 | 8 |
| **RR** | 0 | 2 | 12 | 9 | 13 |

|  |  |
| --- | --- |
| (d) | Avg Wait |
| **FCFS** | 6.2 |
| **SJF** | 4.6 |
| **NPP** | 11 |
| **RR** | 7.2 |

Shortest job first has the minimum average waiting time over all processes.

**6.19**

(b) Shortest job first– processes with shorter burst times could be continually inserted in front of others

(d) Priority– processes with higher priority could be continually inserted in front of others

**7.13**

If thread\_one calls the mutex lock function on first\_mutex and the CPU scheduler makes a context switch to thread\_two, after thread\_two locks second\_mutex the program will be in a dead lock. Both threads will be forced to wait on each other to release the locked mutex.

**7.16**

1. Safely made under all circumstances
2. Only safe from deadlock if the MAX is less than or equal to AVAILABLE
3. Safe only if MAX is less than or equal to AVAILABLE
4. Safely made under all circumstances
5. Safely made under all circumstances
6. Safely made under all circumstances

**7.17**

There will never be a circular wait for processes. In the worst case, when all resources are assigned to a process, at least one process holds its max and can finish allowing the other two processes to finish.

**7.22**

Need:

P0 = (3,1,0,3)

P1 = (1,0,0,1)

P2 = (0,2,0,0)

P3 = (4,1,0,2)

P4 = (2,1,1,3)

1. Available = (0,3,0,1)

Safe: P2, P1, P3, P4, P0

1. Available = (1,0,0,2)

Unsafe: P1 can finish making Available = (2,0,0,3) and then no other process can finish

**8.9**

Internal fragmentation is unused memory internal to a partition. This happens when breaking physical memory into fixed size blocks and allocation memory in units based on block size. External fragmentation exists when there is enough total memory space to satisfy a request but available spaces are not contiguous – or, there are a lot of small holes.

**8.11**

Memory partition blocks:

300 KB (a), 600 KB (b), 350 KB (c), 200 KB (d), 750 KB(e), 125KB (f)

Process sizes:

115 KB, 500 KB, 358 KB, 200 KB, 375 KB

First-fit:

115 into a (185 left)

500 into b (100 left)

358 into e (392 left)

200 into c (150 left)

375 into e (17 left)

d and f left free

Best-fit

115 into f (10 left)

500 into b (100 left)

358 into e (392 left)

200 into d (0 left)

375 into e (17 left)

a and c left free

Worst-fit

115 into e (635 left)

500 into b (100 left)

358 into e (277 left)

200 into c (150 left)

375 FAIL

In this example, the worst-fit algorithm is worst as it fails to allocate memory for the last process. Both the first and best-fit algorithms are able to allocate space for each process but the first-fit algorithm achieves this much faster. However, the best-fit algorithm leaves larger blocks of memory open for later use for larger processes.