6.11

a) CPU utilization vs response time: To increase CPU utilization you would need to reduce the time it spends idle, so reduce the number of context switches which in turn would minimize the response time. According to the book, that when it comes to interactive systems like desktop minimizing the variance between response time is generally better than minimizing the average response time that way response times are predictable.

b) Average turnaround time and maximum waiting time: According to the book both of these should generally be minimized. But in order to minimize the turnaround time you would need to implement the scheduler as SJF. If the scheduler is SJF then long jobs will have a long waiting time generally thus the maximum waiting time will increase and so these two criterias conflict with each other.

c) I/O device utilization and CPU utilization: To maximize CPU utilization, you would need the scheduler to focus on jobs that have larger CPU bursts so there is less time wasted by context switches, but if the scheduler is scheduling CPU events to maximize CPU utilization, I/O device utilization will decrease because the scheduler is only picking CPU bound tasks.

6.14

a) α = 0 τ = 100ms

=0+0+…+0+… = = 100ms so previous burst lengths have no effect on the prediction of the next burst length

b) α = .99 τ = 10ms = which means that the previous burst do have an effect on the next cpu burst prediction.

6.16

a)

b)

turnaround time = completion time – arrival time

FCFS

|  |  |
| --- | --- |
| Process | Turnaround time |
| P1 | 2 |
| P2 | 3 |
| P3 | 11 |
| P4 | 15 |
| P5 | 20 |

SJF

|  |  |
| --- | --- |
| Process | Turnaround time |
| P1 | 3 |
| P2 | 1 |
| P3 | 20 |
| P4 | 7 |
| P5 | 12 |

Nonpreemptive priority

|  |  |
| --- | --- |
| Process | Turnaround time |
| P1 | 19 |
| P2 | 20 |
| P3 | 8 |
| P4 | 17 |
| P5 | 13 |

Round Robin

|  |  |
| --- | --- |
| Process | Turnaround time |
| P1 | 2 |
| P2 | 3 |
| P3 | 20 |
| P4 | 13 |
| P5 | 18 |

c)

FCFS

Waiting time = service time – arrival time

|  |  |
| --- | --- |
| Process | Waiting time |
| P1 | 0 |
| P2 | 2 |
| P3 | 3 |
| P4 | 11 |
| P5 | 15 |

SJF

|  |  |
| --- | --- |
| Process | Waiting time |
| P1 | 1 |
| P2 | 0 |
| P3 | 12 |
| P4 | 3 |
| P5 | 7 |

Nonpreemptive priority

|  |  |
| --- | --- |
| Process | Waiting time |
| P1 | 17 |
| P2 | 19 |
| P3 | 0 |
| P4 | 13 |
| P5 | 8 |

Round Robin

|  |  |
| --- | --- |
| Process | Waiting time |
| P1 | 0 |
| P2 | 2 |
| P3 | 12 |
| P4 | 9 |
| P5 | 13 |

d)

average waiting time = (sumofwaitingtimes)/#process :

FCFS =( 0+2+3+11+15)/5 = 6.2

SJF = (1+0+12+3+7)/5 = 4.6

Nonpreemp = (17+19+0+13+8)/5 = 11.4

Round Robin = (0+2+12+9+13)/5 = 7.2

Shortest job first has the smallest average waiting time with 4.6 bursts.

6.17

b)

Turnaround time = completion time – arrival time

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) waiting time = service time – arrival time

p1 = 0-0 = 0

p2 = (25 -25) + (55 -45) +(80-60) = 30

p3 = (35-30) +(55-45) + (80 -60) = 35

p4 = 60 – 60 =0

p5 = (100 – 100) + (115 – 105) = 10

p6 = 105-105 = 0

d)cpu utilization = ((max turnaround – idle time)/max turnaround time)) x 100

= ((120 -15)/120) \* 100 = 87.5%

6.19

SJF can cause starvation because if short jobs keep coming in long jobs will indefinitely be push back in the queue and therefore be starved. Lets say theres a job with a cpu burst time of 7 and before it gets to execute tasks with cpu burst of 6 or lower keep coming in, those new jobs will keep pushing that task with a cpu burst of 7 back indefinitely and starve it.

Priority scheduling can also result in starvation for a similar reason. Jobs with high priority will always be executed first so a job with a low priority can be starved if jobs with higher priority are constantly being added to the scheduler before the low priority task can be executed. A way to fix this is increasing a jobs priority based on how much time its spent in the queue.

6.21

We can define cpu utilization as the time it would take to complete the task divided by the actual time it took do to context switches and then multiply the result by 100

1. For a time quantum of 1ms the job should only take 1ms but the actual time is 1 + .1 ms for because of the context switch so we get 1/1.1 \* 100 which is about 90.909%
2. For a time quantum of 10ms the i/o and cpu task should take 20ms total but it will actually take 10.1 + (10 x 1.1)ms which is 10.1ms for the cpu task and context switch and 11ms for io tasks with contexts switches which comes out to a total of 21.1ms. so we get 20/21.1 \* 100 = 94.786%