CMPS111 HW3

Jinxuan Jiang

Question 1:

List and describe the necessary conditions for deadlock:

1. **Mutual exclusion**: At least one resource is held in a non-sharable mode that is only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.
2. **No preemption:** Resources cannot be preempted; that is, a resource can only be released voluntarily by the process holding it, after the process has completed its task.
3. **Hold and Wait:** There must exist a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by other processes.
4. **Circuit Wait:** There must exist a set $\{p_0, p_1,.....p_n\}$ of waiting processes such that p_0 is waiting for a resource which is held by p_1, p_1 is waiting for a resource which is held by p_2,..., p_{n-1} is waiting for a resource which is held by p_n and p_n is waiting for a resource which is held by p_0.

Questions 2:

Possibility of deadlock: Assume that process 1 acquires lock A and preempts, process 2 acquires lock B.

Now process 2 is waiting for lock A which is acquired by process1 AND process 1 is waiting for lock B which is acquired by process 2. Here circular waiting occured which leads to dead lock.

To avoid this deadlock problem Pseudo code can be modified to below:

```
Process1 {
        aquire(lockA);
        aquire(lockB);
        modify(resourceA);
        modify(resourceB);
        release(lockB);
        release(lockA);
}
Process2 {
        aquire(lockA); // change from lockB to lockA
        aquire(lockB); // change from lockA to lockB
        modify(resourceB);
        modify(resourceA);
        release(lockA);
        release(lockB);
}
```

If P1 acquires lock A and then preempts P2 can't execute first statement because lock A acquired by process1

If P2 acquires lock A and then preempts P1 can't execute first statement because lock A acquired by process 2

Here in modified pseudo code there is no possibility of circular waiting, No possibility of hold and wait, Hence no possibility of deadlock.

Question 3:

Explain how quantum value and the time taken to perform a context switch affect each other in a round robin process-scheduling algorithm.

The main goal of the CPU scheduling is distribution of the CPU time among the ready processes in a way that at least one of the system efficiency criteria is achieved. These criteria can be listed as below:

- Acceptable response time.
- Performing the task during the user pre-defined time period.
- Increasing the CPU utilization.
- Increasing the utilization of other system resources.
- Decreasing overall overhead.
- Decreasing the user waiting time.
- Decreasing turnaround time.
- Increasing the system throughput.

Round Robin is one the pre-emptive short-term scheduling policies in which, each of the ready processes waiting in the ready queue, get a time slice known as quantum to allocate the processor. If the amount of this quantum is enough for completion of the process, the process will release all allocated resources after termination, otherwise, the operating system re-takes the processor, and the process moves to the end of the ready queue to get dispatched again. So, a context switching occurs and the next ready process allocates the processor for the same quantum of time

The most important parameter in this scheduling algorithm, is the amount of the time quantum. If a small value is assigned to this parameter, the short processes will pass the system very soon, and the overhead of the system will increase due to the increasing the number of context switches. On the other hand, if the amount of the time quantum is larger than the maximum burst time of the ready processes, this policy will downgrade to FCFS scheduling

algorithm. The value of this parameter at least should be considered as a little more than a specific transaction r the time needed for a common conversation

It's obvious that the amount of this parameter, highly affects the value of waiting time, turnaround time and response time of all the processes, and the number of context switches。

Question 4:

Period 1 = 50 ms CPUPeriod 1 = 35ms

Period 2 = 100 ms CPUPeriod 2 = 20ms

Period 3 = 200 ms CPUPeriod 3 = 10ms

Period 4 = 250 ms CPUPeriod 4 = xms

35/50 + 20/100 + 10/200 + x/250 <= 1

X <= 12.5ms

Therefore, the largest value of X is 12.5ms and the scheduling algorithm is Shortest Job First or First-come First-serve, because there is no priority.

Question 5:

TAT = turnaround time

WT = waiting time TAT = CT - AT

AT = arrived time WT = TAT - BT

BT = brust time

a.) First Come First Served

Process	AT	BT	CT	TAT	WT
A	0	10	10	10	0
B	1	6	16	15	9
C	2	2	18	16	14
D	3	4	22	19	15
E	4	8	30	26	18

Average WT = (0+9+14+15+18)/5 = 11.2s

Average TAT = (10+15+16+19+26)/5 = 17.2s

b.) Round Robin

Process	Priority	AT	BT	CT	TAT	WT
A	3	0	10	30	30	20
B	5	1	6	24	23	17
C	2	2	2	10	8	6
D	1	3	4	14	11	7
E	4	4	8	28	24	16

Average TAT = (30+23+8+11+24)/5 = 19.2s

Average WT = (20+17+6+7+16)/5 = 13.2s

c.) Preemptive Priority Scheduling

	BURST	T arrive	Priority
A	10	0	3
B	6	1	5
C	2	2	2
D	4	3	1
E	8	4	4

So the average turnaround time = [(7-3)+(8-2)+(16-0)+(24-4)+(30-1)]/5=15

Each waiting time:

A=(0-0)+（8-2)

B=(24-1)

C=(2-2)+(7-3)

D=(3-3)

E=(16-4)

So the average waiting time = (A+B+C+D+E)/5=9

d) Preemptive shortest job first

	BURST	T arrive
A	10	0
B	6	1
C	2	2
D	4	3
E	8	4

average turnaround time=[(4-2)+(8-3)+(13-1)+(21-4)+(30-0)]/5=13.2

Each waiting time

A=(0-0)+(21-1)

B=(1-1)+(8-2)

C=(2-2)

D=(4-3)

E=(13-4)

Average waiting time=7.2