CS330 –ASSIGNMENT2

N SAI HARSHA - 14408 ABHISHEK GUNDA - 14257 BHARGAV REDDY - 14468

- First we estimate of the CPU burst lengths in the testloop.c program. This is done by running batch1 with algorithm-1 and by taking the average of these burst lengths By this we get
  - A = 124.3
- Q1 = 31 , Q2 = 62 , Q3 = 93

- Next, we have to find out the minimum value of the quantum that offers the maximum achievable CPU utilization for testloop.

- We generated a set of values for CPU utilization by varying the values of Q4 in a preemptive round robin fashion order for testloop.c(Batch1).On observing the values of the data obtained we came to know that the lower the value of Q4 gets the higher CPU utilization it generated.

  As we go on to decrease the value of Q4 there came a point below which the system is getting crashed , the value of Q4 at that point came to be (11) because it is given that a quanta of 10 is utilised in a context switch which implies that Q4 cannot be less than 10.

Q4	CPU Utilization
11	70.602497%
20	65.962597%
25	64.18507%
30	62.792934%
35	62.722934%
40	61.836236%
50	61.239949%
55	61.052582%
60	60.513588%
65	60.541324%

Explanation :

CPU Utilization depends mainly on the time spent in running combined with the time for doing context switches. As we know that if the value of Q4 is less then the context switches occuring in the process must be high.So in our problem as the testloop.c running time is same for all processes irrespective of Q4 the utilization mainly depends on the number of context switches occured during the process which are high if the value of Q4 is less and as the minimum quanta required for a context switch is 10 implies that Q4 =11 has the highest CPU utilization.

Theoritically it can be said that the maximum utilization of CPU occurs at (Q4 = 11) but it is given that the Q4 should be atleast 20 which made us to choose the value 20 for the maximum utilization.

- Q4 = 20 (As the minimum allowed value is 20)

PART – I

Batch1

Algo	CPU Utilization	Average Waiting Time
Non-preemptive default NachOS scheduling	56.283318	24409.300000
Non-preemp shortest next CPU burst 1st algorithm	56.283318	24409.300000
Round-robin with Quanta = 31	62.755076	90656.600000
Round-robin with Quanta = 62	60.540901	76483.500000
Round-robin with Quanta = 93	59.530872	72893.300000
Round-robin with Quanta = 20	65.962597	105936.400000
UNIX scheduler with Quanta = 31	63.036352	90084.500000
UNIX scheduler with Quanta = 62	60.318333	72874.600000
UNIX scheduler with Quanta = 93	59.597858	75328.600000
UNIX scheduler with Quanta = 20	65.962597	105937.400000

Batch2 :

Algo	CPU Utilization	Average Waiting Time
Non-preemptive default NachOS scheduling	82.974849	24355.300000
Non-preemp shortest next CPU burst 1st algorithm	82.974849	24355.300000
Round-robin with Quanta = 31	87.785536	85087.300000
Round-robin with Quanta = 62	87.629778	77378.800000
Round-robin with Quanta = 93	87.136490	72447.800000
Round-robin with Quanta = 20	88.807711	97254.200000
UNIX scheduler with Quanta = 31	89.187155	90463.300000
UNIX scheduler with Quanta = 62	87.327813	72244.500000
UNIX scheduler with Quanta = 93	86.744032	72621.800000
UNIX scheduler with Quanta = 20	88.386110	97055.500000

Batch3 :

Algo	CPU Utilization	Average Waiting Time
Non-preemptive default NachOS scheduling	94.850203	24355.300000
Non-preemp shortest next CPU burst 1st algorithm	94.850203	24355.300000
Round-robin with Quanta = 31	98.614251	85091.500000
Round-robin with Quanta = 62	99.214724	77183.600000
Round-robin with Quanta = 93	98.411224	71500.800000
Round-robin with Quanta = 20	98.142846	97159.500000
UNIX scheduler with Quanta = 31	99.438372	90140.000000
UNIX scheduler with Quanta = 62	98.974869	76219.700000
UNIX scheduler with Quanta = 93	98.764200	72248.000000
UNIX scheduler with Quanta = 20	97.932761	97067.200000

Batch4 :

Algo	CPU Utilization	Average Waiting Time
Non-preemptive default NachOS scheduling	100.000000	36577.000000
Non-preemp shortest next CPU burst 1st algorithm	100.000000	36577.000000
Round-robin with Quanta = 31	100.000000	87417.300000
Round-robin with Quanta = 62	100.000000	76356.600000
Round-robin with Quanta = 93	100.000000	71115.300000
Round-robin with Quanta = 20	100.000000	109588.000000
UNIX scheduler with Quanta = 31	100.000000	87592.300000
UNIX scheduler with Quanta = 62	100.000000	76808.600000
UNIX scheduler with Quanta = 93	100.000000	71720.600000
UNIX scheduler with Quanta = 20	100.000000	109597.000000

PART – II

- Evaluating the difference between the two non-preemptive algorithms, using testloop4 and testloop5 programs using batch 5.
- Report the average waiting time in the ready queue for the two non-preemptive algorithms.

Algorithm Used	CPU Utilization	Average Waiting Time
Algo - 1	100.000000	55500.000000
Algo - 2	100.000000	40195.000000

- Each testloop4 and testloop5 yeild after a loop
- Algo -1 will run the threads by FCFS(non-premptive) and Algo-2 uses Shortest next CPU burst first algorithm
- Algo-2 uses Exponential Averaging technique to find the next CPU burst, which ensures that after some iterations it would predict the next CPU burst and schedule accordingly based on the Estimated CPU bursts.
- Since the Algo-1 runs the all 5 testloop4 threads , then all the 5 testloop5 threads till yeilding and then all 5 testloop4 threads and goes on in this way, thus the order is fixed for the threads.This increases the waiting time by not allowing the shorter bursts come earlier as the order is fixed
- But in the 2nd case, shorter bursts come earlier and the waiting time is reduced .

PART – III

Overall estimation error in CPU bursts for the second non-preemptive algorithm

Batch	**OUTER_BOUND = 4**	**OUTER_BOUND = 10**
Batch1	0.861970	0.387911
Batch2	0.913513	0.395738
Batch3	0.810605	0.347336
Batch4	1.00000	1.00000
Batch5	0.694081	0.364632

- As we increase the OUTER_BOUND, the number of loops increases.Each of the loop yields
- As we can see that from the data obtained the error decreases gradually on increasing the OUTER_BOUND, the prediction of the next CPU burst would be more accurate due to the increased number of iterations thus the error decreases
- But for the testloop3 there is no yielding, the whole thread runs in a single cpu burst, thus leaving no error.

PART – IV

Difference between round-robin and the UNIX scheduling Algorithms

	Round Robin	**Unix Scheduler**
Min Completion Time	153592	157269
Max Completion Time	160742	160899
Avg Completion Time	159323.000000	159524.000000

- The round robin algorithm doesnt take the priorities into account when scheduling the processes and runs with a fixed quantum which is 100 provided in this case
- Unix scheduler calculates and reassigns the prorities for every timer interrupt and then schedule the processes according to their computed priorities
- If a program has a very high priority the chance of selecting this program is very high in unix scheduler and if this process takes a very long time then it would leave a lasting impact on the waiting time of all the other processes
- In contrary the round robin algorithm treats all the processes with equal priority and the cpu remains consitent with not very large waiting time than the unix scheduler