**Experiment No: 8**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar 19CSU185 |
| **Semester /Section:** V/FS-A-1 |
| **Link to Code:** https://github.com/NamitKumar16/OS |
| **Date:** 13th October 2021 |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective: Objective**  To familiarize the students about CPU scheduling Algorithms |
| **Program Outcome**  The students will understand the Round Robin Algorithm. |
| **Problem Statement:**  Implement the Round Robin Algorithm. |
| **Background Study:**   * In Round Robin each process is assigned a fixed time slot in a cyclic way and this is preemptive. It has a disadvantage of context switch and have quantum time |
| **Question Bank:**   1. What is Preemptive and Non- Preemptive CPU scheduling? Explain with examples. 2. Explain the difference between short term, long term and medium term scheduling. 3. Explain the function of Dispatcher and Context Switch mechanism. 4. What are the advantages and disadvantages of Round robin? 5. Give the application are of Robin Robin. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Q1** – Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute. Examples of preemptive scheduling are Round Robin and Shortest Remaining Time First.

**Q2** –

|  |  |  |
| --- | --- | --- |
| **Long Term** | **Short Term** | **Medium Term** |
| It is a job scheduler. | It is a CPU scheduler. | It is swapping. |
| Speed is less than short term scheduler. | Speed is very fast. | Speed is in between both |
| It controls the degree of multiprogramming | Less control over the degree of multiprogramming. | Reduce the degree of multiprogramming. |
| It selects processes from the pool and load them into memory for execution. | It selects from among the processes that are ready to execute. | Process can be reintroduced into the meat and its execution can be continued. |

**Q3** – Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

A dispatcher is a special program which comes into play after the scheduler. When the scheduler completes its job of selecting a process, it is the dispatcher which takes that process to the desired state/queue.

**Q4** – Advantages -

* Each process is served by the CPU for a fixed time quantum, so all processes are given the same priority.
* Starvation doesn't occur because, for each round robin cycle, every process is given a fixed time to execute. No process is left behind.

Disadvantages -

* The throughput in RR largely depends on the choice of the length of the time quantum. If time quantum is too large it behaves as FCFS. If time quantum is too short much of the time is spent in process switching and hence low throughput.
* Also here one cannot assign priority to any process which can be a drawback.

**Q5** - Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks.

#include<stdio.h>

#include<conio.h>

void main()

{

int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];

float avg\_wt, avg\_tat;

printf(" Total number of process in the system: ");

scanf("%d", &NOP);

y = NOP;

for(i=0; i<NOP; i++)

{

printf("\n Enter the Arrival and Burst time of the Process%d\n", i+1);

printf(" Arrival time is: \t");

scanf("%d", &at[i]);

printf(" \nBurst time is: \t");

scanf("%d", &bt[i]);

temp[i] = bt[i];

}

printf("Enter the Time Quantum for the process: \t");

scanf("%d", &quant);

printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");

for(sum=0, i = 0; y!=0; )

{

if(temp[i] <= quant && temp[i] > 0)

{

sum = sum + temp[i];

temp[i] = 0;

count=1;

}

else if(temp[i] > 0)

{

temp[i] = temp[i] - quant;

sum = sum + quant;

}

if(temp[i]==0 && count==1)

{

y--;

printf("\n Process %d \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);

wt = wt+sum-at[i]-bt[i];

tat = tat+sum-at[i];

count =0;

}

if(i==NOP-1)

{

i=0;

}

else if(at[i+1]<=sum)

{

i++;

}

else

{

i=0;

}

}

avg\_wt = wt \* 1.0/NOP;

avg\_tat = tat \* 1.0/NOP;

printf("\n Average Turn Around Time: \t%f", avg\_wt);

printf("\n Average Waiting Time: \t%f", avg\_tat);

getch();

}

Text

Description automatically generated