CLASS ASSIGNMENT

CSE 316

OPERATING SYSTEM

NAME – Akash yadav

Roll No – 28

Section – K18JC

Reg No - 11802704

Ques. 5. CPU schedules N processes which arrive at different time intervals and each process is allocated the CPU for a specific user input time unit, processes are scheduled using a preemptive round robin scheduling algorithm. Each process must be assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes one task has priority 0. The length of a time quantum is T units, where T is the custom time considered as time quantum for processing. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue. Design a scheduler so that the task with priority 0 does not starve for resources and gets the CPU at some time unit to execute. Also compute waiting time, turn around.

```
#question number 5 solution

#code

#include<stdio.h>

#include<conio.h>

void main()

{

  char p[10][5],temp[5];

  int i,j,pt[10],wt[10],totwt=0,pr[10],temp1,n;

  float avgwt;

 printf("enter no of processes:");

 scanf("%d",&n);

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

  {

  printf("enter process%d name:",i+1);

 scanf("%s",&p[i]);

  printf("enter process time:");

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

  printf("enter priority:");

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

  }

 for(i=0;i<n-1;i++)

 {

 for(j=i+1;j<n;j++)

 {

   if(pr[i]>pr[j])

 {
   temp1=pr[i];
 pr[i]=pr[j];
  pr[j]=temp1;
  temp1=pt[i];
  pt[i]=pt[j];
  pt[j]=temp1;
 strcpy(temp,p[i]);
 strcpy(p[i],p[j]);
  strcpy(p[j],temp);
  }
  }
  }
 wt[0]=0;
  for(i=1;i<n;i++)
  {
   wt[i]=wt[i-1]+et[i-1];
   totwt=totwt+wt[i];
   }
avgwt=(float)totwt/n;
printf("p_name\t p_time\t priority\t w_time\n");
for(i=0;i<n;i++)
{
   printf(" %s\t %d\t %d\t %d\n" ,p[i],pt[i],pr[i],wt[i]);
   }
  printf("total waiting time=%d\n avg waiting time=%f",tot,avg);
  getch();
  }
```

OUTPUT:

```
enter no of processes: 5

enter process1 name: aaa

enter process time: 4

enter priority:5

enter process2 name: bbb

enter process time: 3

enter priority:4

enter process3 name: ccc

enter process time: 2

enter priority:3

enter process4 name: ddd

enter process time: 5

enter priority:2

enter process5 name: eee

enter process time: 1

enter priority:1

p_name P_time priority w_time

eee 1 1 0

ddd 5 2 1

ccc 2 3 6

bbb 3 4 8
```

```
aaa 4 5 11

total waiting time=26

avg waiting time=5.20
```

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

50

51

52

DESCRIPTION :-

CPU Scheduling in Operating Systems

Scheduling of processes/work is done to finish the work on time.

Below are different time with respect to a process.

Arrival Time: *Time at which the process arrives in the ready queue.*
Completion Time: *Time at which process completes its execution.*
Burst Time: *Time required by a process for CPU execution.*
Turn Around Time: *Time Difference between completion time and arrival time.*
Turn Around Time = Completion Time – Arrival Time

Waiting Time(W.T): *Time Difference between turn around time and burst time.*
Waiting Time = Turn Around Time – Burst Time

Why do we need scheduling?
A typical process involves both I/O time and CPU time. In a uni programming system like MS-DOS, time spent waiting for I/O is wasted and CPU is free during this time. In multi programming systems, one process can use CPU while another is waiting for I/O. This is possible only with process scheduling.

Objectives of Process Scheduling Algorithm
Max CPU utilization [Keep CPU as busy as possible]
Fair allocation of CPU.
Max throughput [Number of processes that complete their execution per time unit]
Min turnaround time [Time taken by a process to finish execution]
Min waiting time [Time a process waits in ready queue]
Min response time [Time when a process produces first response]

Different Scheduling Algorithms

First Come First Serve (FCFS): Simplest scheduling algorithm that schedules according to arrival times of processes. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first. It is implemented by using the

FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. FCFS is a non-preemptive scheduling algorithm.
Note: First come first serve suffers from convoy effect.
Shortest Job First (SJF): Process which have the shortest burst time are scheduled first.If two processes have the same bust time then FCFS is used to break the tie. It is a non-preemptive scheduling algorithm.

Longest Job First (LJF): It is similar to SJF scheduling algorithm. But, in this scheduling algorithm, we give priority to the process having the longest burst time. This is non-preemptive in nature i.e., when any process starts executing, can't be interrupted before complete execution.
Shortest Remaining Time First (SRTF): It is preemptive mode of SJF algorithm in which jobs are schedule according to shortest remaining time.
Longest Remaining Time First (LRTF): It is preemptive mode of LJF algorithm in which we give priority to the process having largest burst time remaining.
Round Robin Scheduling: Each process is assigned a fixed time(Time Quantum/Time Slice) in cyclic way.It is designed especially for the time-sharing system. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1-time quantum. To implement Round Robin scheduling, we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1-time quantum, and dispatches the process. One of two things will then happen. The process may have a CPU burst of less than 1-time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1-time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.
Priority Based scheduling (Non-Preemptive): In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is scheduled first. If priorities of two processes match, then schedule according to arrival time. Here starvation of process is possible.
Highest Response Ratio Next (HRRN): In this scheduling, processes with highest response ratio is scheduled. This algorithm avoids starvation.

```
Response Ratio = (Waiting Time + Burst time) / Burst time
```

Multilevel Queue Scheduling: According to the priority of process, processes are placed in the different queues. Generally high priority process are placed in the top level queue. Only after completion of processes from top level queue, lower level queued processes are scheduled. It can suffer from starvation.

Multi level Feedback Queue Scheduling: It allows the process to move in between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it is moved to a lower-priority queue.

Some useful facts about Scheduling Algorithms:

1. FCFS can cause long waiting times, especially when the first job takes too much CPU time.
2. Both SJF and Shortest Remaining time first algorithms may cause starvation. Consider a situation when the long process is there in the ready queue and shorter processes keep coming.
3. If time quantum for Round Robin scheduling is very large, then it behaves same as FCFS scheduling.
4. SJF is optimal in terms of average waiting time for a given set of processes,i.e., average waiting time is minimum with this scheduling, but problems are, how to know/predict the time of next job.

ALGORITHM:-

1-input the processes along with their burst time [bt].

2-find waiting time [wt] for all processes.

3-as first process that comes need not to wait so waiting time for process 1 will be 0 i.e

Wt[0] = 0.

4-find waiting time for all processes i.e for process i->w[i] = bt[i-1]+wt[i-1].

5-find turnaround time = waiting time + burst time for all processes

6-find average waiting time = total waiting time / no of processes

7-similarly find average turn around time = total turn around time/no of processes.