Ex No: 2a
Date:06.02.2023

PRIORITY SCHEDULING

Aim

To schedule snapshot of processes queued according to Priority scheduling.

Algorithm

1. Define an array of structure process with members str, burst, pri, wait & turn.
2. Get length of the ready queue, i.e., number of process (say n)
3. Obtain burst and pri for each process.
4. Sort the processes according to their pri in ascending order.
a. If two process have same pri, then FCFS is used to resolve the tie.
5. The wait for first process is 0.
6. Compute wait and turn for each process as:
a. waiti+1 = wait + burst
b. turni = waiti + bursti
7. Compute average waiting time avgwait and average turn around time avgturn
8. Display the burst, pri, turn and wait for each process.
9. Display GANTT chart for the above scheduling
10. Display avgwait and avgturn
11. Stop

Program

```
#include<stdio.h>
void main()
{
   int n;
   printf("enter number of process:");
   scanf("%d",&n);
   char str[n],s[n],ch='A';
   for(int I=0;I<n;I++){
      str[I]=ch;
      s[I]=str[I];
      ch+=1;
   }
   int dummy[n],pri[n],priority[n],burst[n],wait[n],turn[n],index[n];
   for(int I=0;I<n;I++)
   {
      printf("enter burst time of process %d : ",I+1);
      scanf("%d",&burst[I]);
      dummy[I]=burst[I];
      printf("enter priority of process %d :",I+1);
      scanf("%d",&pri[I]);
      priority[I]=pri[I];
   }

   for(int I=0;I<n;I++){
      int min=pri[I],pos=I ;
      for(int j=I+1;j<n;j++){
```

```
            if(min>pri[j]){
                min=pri[j];
                pos=j;
            }
        }
        int temp=pri[I];
        pri[I]=pri[pos];
        pri[pos]=temp;
        char t=s[I];
        s[I]=s[pos];
        s[pos]=t;
    }
    int initial=0,a=0,b=0;
    for(int I=0;I<n;I++){
        wait[s[I]-'A']=initial;
        initial+=burst[s[I]-'A'];
        a+=wait[s[I]-'A'];
    }
    for(int I=0;I<n;I++){
        turn[s[I]-'A']=wait[s[I]-'A']+burst[s[I]-'A'];
        b+=turn[s[I]-'A'];
    }
    float avgwait=((float)a)/n;
    float avgturn=((float)b)/n;

    printf("----------------PRIORITY----------------\n");

    printf("process    BurstTime    Priority    Waiting Time    TurnAround Time\n");
    for(int I=0;I<n;I++){
        printf("%c          %d ms          %d ms          %d ms          %d
ms\n",str[I],dummy[I],priority[I],wait[I],turn[I]);
    }
    printf("\n\nGantt Chart : \n");
    for(int I=0;I<3*n;I++){
        printf("--");
    }
    printf("\n");
    for(int I=0;I<n;I++){
        printf("|  %c  ",s[I]);
    }
    printf("|\n");
    for(int I=0;I<3*n;I++){
        printf("--");
    }
    int sum=0;
    printf("\n0");
    for(int I=0;I<n;I++){
        sum+=burst[s[I]-'A'];
        printf("    %d",sum);
    }
```

```
    printf("\nAverage Waiting Time : ");
    printf("%.2f ms\n",avgwait);
    printf("\nAverage Turn Around Time : ");
    printf("%.2f ms\n",avgturn);
    printf("-------------------------------------");
}
```

Output

```
enter number of process:4
enter burst time of process 1 : 5
enter priority of process 1 :3
enter burst time of process 2 : 4
enter priority of process 2 :1
enter burst time of process 3 : 2
enter priority of process 3 :2
enter burst time of process 4 : 1
enter priority of process 4 :4
----------------PRIORITY-----------------
process      BurstTime      Priority        Waiting Time        TurnAround Time
A            5 ms             3 ms           6 ms                11 ms
B            4 ms             1 ms           0 ms                4 ms
C            2 ms             2 ms           4 ms                6 ms
D            1 ms             4 ms           11 ms                12 ms


Gantt Chart :
------------------------
|  B  |  C  |  A  |  D  |
------------------------
0    4    6    11    12
Average Waiting Time : 5.25 ms

Average Turn Around Time : 8.25 ms
-------------------------------------
```

Observation (20)	
Record (5)	
Total (25)	
Initial	

Result

Thus waiting time & turnaround time for processes based on Priority scheduling was computed and the average waiting time was determined.

Ex No: 2b
Date: 06.02.2023

ROUND ROBIN SCHEDULING

Aim

To schedule snapshot of processes queued according to Round robin scheduling.

Algorithm

1. Get length of the ready queue, i.e., number of process (say n)
2. Obtain Burst time Bi for each processes Pi.
3. Get the time slice per round, say TS
4. Determine the number of rounds for each process.
5. The wait time for first process is 0.
6. If Bi > TS then process takes more than one round. Therefore turnaround and waiting time should
include the time spent for other remaining processes in the same round.
7. Calculate average waiting time and turn around time
8. Display the GANTT chart that includes
a. order in which the processes were processed in progression of rounds
b. Turnaround time Ti for each process in progression of rounds.
9. Display the burst time, turnaround time and wait time for each process (in order of rounds they
were processed).
10. Display average wait time and turnaround time
11. Stop

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n;
printf("enter number of process :");
scanf("%d",&n);
int burst[n],dummy[n],count[n],sum=0,slice,s=0,i,prewait[n],wait[n],turn[n];
char str[n],ch='A';
for(i=0;i<n;i++)
{
      str[i]=ch;
      ch+=1;
      count[i]=0;
}
for(i=0;i<n;i++){
printf("enter burst time of process %d :",i+1);
scanf("%d",&burst[i]);
dummy[i]=burst[i];
sum+=burst[i];
}
printf("enter time slice :");
scanf("%d",&slice);
```

```
char cha[sum];
int j=0,x=0,pro[sum];
pro[j]=0;
while(s<=sum && x<n){
            if(burst[x]!=0)
            {
            if(burst[x]<=slice && burst[x]>0){
                  cha[j++]=str[x];
                  prewait[x]=s;
                  s+=burst[x];
                  pro[j]=s;
                  burst[x]=0;
            }
            else if(burst[x]>0){
                cha[j++]=str[x];
                prewait[x]=s;
                s+=slice;
                pro[j]=s;
                burst[x]-=slice;
                count[x]+=1;
            }
            if(x==n-1){
                     x=0;
            }
            else{
                x++;
            }}
            else{
               x++;
            }
}
printf("\n\n----------ROUND ROBIN----------\n");
printf("\nGANTT CHART:\n");
printf(" ");
for(i=0;i<j*3;i++){
printf("--");
}
printf("\n");
for(i=0;i<j;i++){
printf("|  %c  ",cha[i]);
}
printf(" |\n");
printf(" ");
for(i=0;i<j*3;i++){
printf("--");
}
printf("\n");
for(i=0;i<j;i++){
printf("    %d",pro[i]);
}
```

```
printf("    %d\n",sum);
int a=0,b=0;
for(i=0;i<n;i++){
      wait[i]=prewait[i]-(count[i])*slice;
      a+=wait[i];
}
float avgwait=((float)a)/n;
for(i=0;i<n;i++){
      turn[i]=wait[i]+dummy[i];
       b+=turn[i];
}
float avgturn=((float)b)/n;
printf("\n\nProcess BurstTime WaitingTime TurnaroundTime\n");
for(int I=0;I<n;I++){
   printf("%c       %dms       %dms       %dms\n",str[I],dummy[I],wait[I],turn[I]);
}
printf("\n\nAverage Waiting Time:%.2f ms", avgwait);
printf("\nAverage Turn Around Time:%.2f ms", avgturn);
printf("\n---------------------------------\n");
return 0;
}
```

Output

```
C:\Users\nanda\Desktop\round.exe
enter number of process :5
enter burst time of process 1 :8
enter burst time of process 2 :3
enter burst time of process 3 :4
enter burst time of process 4 :2
enter burst time of process 5 :5
enter time slice :2


------------ROUND ROBIN---------

GANTT CHART:
 ---------------------------------------------------------------------
| A | B | C | D | E | A | B | C | E | A | E | A |
 ---------------------------------------------------------------------
   0   2   4   6   8   10   12   13   15   17   19   20   22


Process BurstTime WaitingTime TurnaroundTime
A        8ms        14ms        22ms
B        3ms        10ms        13ms
C        4ms        11ms        15ms
D        2ms        6ms        8ms
E        5ms        15ms        20ms


Average Waiting Time:11.20 ms
Average Turn Around Time:15.60 ms
---------------------------------

Process returned 0 (0x0)   execution time : 18.489 s
Press any key to continue.
```

Observation (20)	
Record (5)	
Total (25)	
Initial	

Result

Thus waiting time and turnaround time for processes based on Round robin scheduling was computed and the average waiting time was determined.