Title: C Program for Scheduling

Experiment No: 05a

Experiment Name: Program for Shortest Job First (or SJF) scheduling

---

Algorithm:

1. **Sort all the processes in increasing order according to burst time.**
2. **Input the processes along with their burst time (bt).**
3. **Find waiting time (wt) for all processes.**
4. **As first process that comes need not to wait so waiting time for process 1 will be 0 i.e. wt[0] = 0.**
5. **Find waiting time for all other processes i.e. for process i ->**
   wt[i] = bt[i-1] + wt[i-1] .
6. **Find turnaround time = waiting_time + burst_time for all processes.**
7. **Find average waiting time =**
   total_waiting_time / no_of_processes.
8. **Similarly, find average turnaround time = total_turn_around_time /no_of_processes.**

Source Code:

```
#include<stdio.h>
0
void main()
{
   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
   float avg_wt,avg_tat;
   printf("Enter number of process:");
   scanf("%d",&n);

   printf("\nEnter Burst Time:\n");
   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;          //contains process number
```

```
}

//sorting burst time in ascending order using selection sort
for(i=0;i<n;i++)
{
   pos=i;
   for(j=i+1;j<n;j++)
   {
      if(bt[j]<bt[pos])
         pos=j;
   }



   temp=bt[i];
   bt[i]=bt[pos];
   bt[pos]=temp;

   temp=p[i];
   p[i]=p[pos];
   p[pos]=temp;
}

wt[0]=0;          //waiting time for first process will be zero

//calculate waiting time
for(i=1;i<n;i++)
{
   wt[i]=0;
   for(j=0;j<i;j++)
      wt[i]+=bt[j];

   total+=wt[i];
}

avg_wt=(float)total/n;      //average waiting time
total=0;

printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
   tat[i]=bt[i]+wt[i];     //calculate turnaround time
   total+=tat[i];
   printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
}
```

```
    avg_tat=(float)total/n;     //average turnaround time
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
dani@RICH:~/Desktop$ ./a.out
Enter number of process:6

Enter Burst Time:
p1:3
p2:2
p3:10
p4:5
p5:4
p6:7

Process     Burst Time          Waiting Time    Turnaround Time
p2                2                 0                  2
p1                3                 2                  5
p5                4                 5                  9
p4                5                 9                  14
p6                7                 14                 21
p3                10                21                 31

Average Waiting Time=8.500000
Average Turnaround Time=13.666667
dani@RICH:~/Desktop$
```

Title: C Program for Scheduling

Experiment No: 5b

Experiment Name: Program for First Come First Served (FCFS) scheduling

---

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 other processes i.e. for process i ->**
   wt[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 turnaround time = total_turn_around_time /no_of_processes.**

Source Code:

```
#include<stdio.h>

int main()
```

```
{
   int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
   printf("Enter total number of processes(maximum 20):");
   scanf("%d",&n);

   printf("\nEnter Process Burst Time\n");
   for(i=0;i<n;i++)
   {
      printf("P[%d]:",i+1);
      scanf("%d",&bt[i]);
   }

   wt[0]=0;    //waiting time for first process is 0

   //calculating waiting time
   for(i=1;i<n;i++)
   {
      wt[i]=0;
      for(j=0;j<i;j++)
         wt[i]+=bt[j];
   }

   printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
    //calculating turnaround time
   for(i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];
      avwt+=wt[i];
      avtat+=tat[i];
      printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
   }
    avwt/=i;
   avtat/=i;
   printf("\n\nAverage Waiting Time:%d",avwt);
   printf("\nAverage Turnaround Time:%d",avtat);

   return 0;
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
dani@RICH:~/Desktop$ ./a.out
Enter total number of processes(maximum 20):6

Enter Process Burst Time
P[1]:5
P[2]:6
P[3]:2
P[4]:3
P[5]:1
P[6]:4

Process         Burst Time      Waiting Time    Turnaround Time
```

Title: C Program for Scheduling

Experiment No:5c

Experiment Name: Program for Priority (Non Pre-emptive) for Scheduling.

Algorithm:

- **Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.**
- **Each process is assigned a priority. Process with highest priority is to be executed first and so on.**
- **Processes with same priority are executed on first come first served basis.**
- **Priority can be decided based on memory requirements, time requirements or any other resource requirement.**

Source Code:

```
#include<stdio.h>

int main()
{
   int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
   printf("Enter Total Number of Process:");
   scanf("%d",&n);

   printf("\nEnter Burst Time and Priority\n");
   for(i=0;i<n;i++)
   {
      printf("\nP[%d]\n",i+1);
      printf("Burst Time:");
      scanf("%d",&bt[i]);
      printf("Priority:");
      scanf("%d",&pr[i]);
      p[i]=i+1;
   }


   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(pr[j]<pr[pos])
            pos=j;
      }

      temp=pr[i];
      pr[i]=pr[pos];
      pr[pos]=temp;

      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;

      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }

   wt[0]=0;
```

```
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=total/n;
    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\nP[%d]\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=total/n;
    printf("\n\nAverage Waiting Time=%d",avg_wt);
    printf("\nAverage Turnaround Time=%d\n",avg_tat);

        return 0;
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
sachin@sachin-VirtualBox:~/Desktop/programs$ cc priority.c
sachin@sachin-VirtualBox:~/Desktop/programs$ ./a.out
Enter Total Number of Process:3

Enter Burst Time and Priority

P[1]
Burst Time:8
Priority:2

P[2]
Burst Time:5
Priority:1

P[3]
Burst Time:12
Priority:3

Process     Burst Time          Waiting Time    Turnaround Time
P[2]              5                 0                  5
P[1]              8                 5                  13
P[3]              12                13                 25

Average Waiting Time=6
Average Turnaround Time=14
sachin@sachin-VirtualBox:~/Desktop/programs$
```

Title: C Program for Scheduling

Experiment No: 5d

Experiment Name: Program for Round Robin for Scheduling.

Algorithm:

1. The queue structure in ready queue is of First In First Out (FIFO) type.
2. A fixed time is allotted to every process that arrives in the queue. This fixed time is known as time slice or time quantum.
3. The first process that arrives is selected and sent to the processor for execution. If it is not able to complete its execution within the time quantum provided, then an interrupt is generated using an automated timer.
4. The process is then stopped and is sent back at the end of the queue. However, the state is saved and context is thereby stored in memory. This helps the process to resume from the point where it was interrupted.
5. The scheduler selects another process from the ready queue and dispatches it to the processor for its execution. It is executed until the time Quantum does not exceed.
6. The same steps are repeated until all the process are finished.

Source Code:

```
int main()
{

  int count,j,n,time,remain,flag=0,time_quantum;
  int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
  printf("Enter Total Process:\t ");
  scanf("%d",&n);
  remain=n;
  for(count=0;count<n;count++)
  {
        printf("Enter Arrival Time and Burst Time for Process Process Number %d
:",count+1);
    scanf("%d",&at[count]);
    scanf("%d",&bt[count]);
    rt[count]=bt[count];
```

```
  }
  printf("Enter Time Quantum:\t");
  scanf("%d",&time_quantum);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(time=0,count=0;remain!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];
      rt[count]=0;
      flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
      wait_time+=time-at[count]-bt[count];
      turnaround_time+=time-at[count];
      flag=0;
    }
    if(count==n-1)
      count=0;
    else if(at[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
sachin@sachin-VirtualBox:~/Desktop/programs$ cc rr.c
sachin@sachin-VirtualBox:~/Desktop/programs$ ./a.out
Enter Total Process:     5
Enter Arrival Time and Burst Time for Process Process Number 1 :0
8
Enter Arrival Time and Burst Time for Process Process Number 2 :1
1
Enter Arrival Time and Burst Time for Process Process Number 3 :3
2
Enter Arrival Time and Burst Time for Process Process Number 4 :4
1
Enter Arrival Time and Burst Time for Process Process Number 5 :2
5
Enter Time Quantum:     4


Process |Turnaround Time|Waiting Time

P[2]    |       4       |       3
P[3]    |       4       |       2
P[4]    |       4       |       3
P[1]    |       16      |       8
P[5]    |       15      |       10

Average Waiting Time= 5.200000
Avg Turnaround Time = 8.600000sachin@sachin-VirtualBox:~/Desktop/programs$
```