PROCESS SCHEDULING

- ❖ CPU scheduling is used in multiprogrammed operating systems.
- ❖ By switching CPU among processes, efficiency of the system can be improved.
- ❖ Some scheduling algorithms are FCFS, SJF, Priority, Round-Robin, etc.
- ❖ Gantt chart provides a way of visualizing CPU scheduling and enables to understand better.

FIRST COME FIRST SERVE (FCFS):

- ❖ Process that comes first is processed first
- ❖ FCFS scheduling is non-preemptive
- ❖ Not efficient as it results in long average waiting time.
- ❖ Can result in starvation, if processes at beginning of the queue have long bursts.

SHORTEST JOB FIRST (SJF):

- ❖ Process that requires smallest burst time is processed first.
- ❖ SJF can be preemptive or non–preemptive
- ❖ When two processes require same amount of CPU utilization, FCFS is used to break the tie.
- ❖ Generally efficient as it results in minimal average waiting time.
- ❖ Can result in starvation, since long critical processes may not be processed.

PRIORITY:

- ❖ Process that has higher priority is processed first.
- ❖ Prioirty can be preemptive or non–preemptive
- ❖ When two processes have same priority, FCFS is used to break the tie.
- ❖ Can result in starvation, since low priority processes may not be processed.

ROUND ROBIN:

- ❖ All processes are processed one by one as they have arrived, but in rounds.
- ❖ Each process cannot take more than the time slice per round.
- ❖ Round robin is a fair preemptive scheduling algorithm.
- ❖ A process that is yet to complete in a round is preempted after the time slice and put at the end of thequeue.
- ❖ When a process is completely processed, it is removed from the queue.

EX.NO: 1a   **FCFS SHEDULING**

DATE: 03.02.2023

AIM:

   To schedule snapshot of processes queued according to FCFS (First Come First Serve) scheduling.

ALGORITHM:

1. Define an array of structure process with members process , burst time, waiting time and ttime.

2. Get length of the ready queue, i.e., number of process (say n)

3. Obtain burst time for each process.

4. The waiting time for first process is 0.

5. Compute waiting time and turnaround time for each process as:

a. waiting time $(i+1)$ = waiting time (i) + burst time (i)

b. turnaround time (i) = waiting time (i) + burst time(i)

6. Compute average waiting time and average turnaround time

7. Display the burst time, turnaround time and waiting time for each process.

8. Display GANTT chart for the above scheduling

9. Display average waiting time and average turnaround time

10. Stop

CODE:

```c
#include<stdio.h>
#include<conio.h>
//global variables
int n,wt[10],bt[10],tat[10];
//Calculating waiting time for each process
void waitingtime(){
        int i;
        wt[0]=0;
        for(i=1;i<n;i++)
                wt[i]=bt[i-1]+wt[i-1];
}
//Calculating turnaround time for each process
void turnaroundtime(){
        int i;
        for(i=0;i<n;i++)
                tat[i]=wt[i]+bt[i];
}
```

```c
//Calculating the average waiting time and turnaround time
void average(){
        int twt=0,ttat=0;
        float awt,atat;
        int i;
        waitingtime();
        turnaroundtime();
        printf("Wating time and turnaround time for each process: \n");
        for(i=0;i<n;i++){
                printf("Process %d\t%d\t%d\n",i+1,wt[i],tat[i]);
                twt+= wt[i];
                ttat+=tat[i];
        }
        awt=(float)twt/n;
        atat=(float)ttat/n;
        printf("The average waiting time: %f\n",awt);
        printf("The average turnaround time: %f\n",atat);
}

//Dispalying the gantt chart
void ganttchart(){
        int i;
        printf("The Gantt Chart \n");
        printf("| ");
        for(i=0;i<n;i++)
                printf("P%d | ",i+1);
        printf("\n0   ");
        for(i=0;i<n;i++)
                printf("%d   ",tat[i]);
}

//Main function
int main(){
        int i;
        printf("\t\t\tFCFS Scheduling\n");
        printf("Enter the number of process : ");
        scanf("%d",&n);
        printf("Enter the burst time of the each process: \n");
        for(i=0;i<n;i++){
                printf("Process %d :",i+1);
                scanf("%d",&bt[i]);
        }
        average();
        ganttchart();
        return 0;
}
```

OUTPUT:

```
                    FCFS Scheduling
Enter the number of process : 3
Enter the burst time of the each process:
Process 1 :24
Process 2 :3
Process 3 :3
Wating time and turnaround time for each process:
Process 1         0         24
Process 2        24         27
Process 3        27         30
The average waiting time: 17.000000
The average turnaround time: 27.000000
The Gantt Chart
| P1 | P2 | P3 |
0    24   27   30
----------------------------------
Process exited after 8.082 seconds with return value 0
Press any key to continue . . .|
```

RESULT:

      Thus Waiting time and Turnaround time for processes based on FCFS scheduling was computed and the average waiting time was determined.

EX NO: 1b   **SJF SCHEDULING**

DATE:03.02.2023

<u>**AIM:**</u>

      To schedule snapshot of processes queued according to SJF (Shortest Job First) scheduling.

<u>**ALGORITHM:**</u>

1. Define an array of structure process with members pid, btime, wtime & ttime.

2. Get length of the ready queue, i.e., number of process (say n)

3. Obtain btime for each process.

4. Sort the processes according to their btime in ascending order.

      a. If two process have same burst time, then FCFS is used to resolve the tie.

5. The waiting time for first process is 0.

6. Compute waiting time and turnaround time for each process as:

      a. Waiting time $(i+1)$ = Waiting time (i) + burst time (i)

      b. Turnaround time (i) = Waiting time (i) + burst time (i)

7. Compute average waiting time and average turnaround time.

8. Display burst time, turnaround time and waiting time for each process.

9. Display GANTT chart for the above scheduling

10. Display average waiting time and average turnaround time.

11. Stop

<u>**CODE:**</u>

```
#include<stdio.h>
#include<conio.h>

//global variable
int n,process[10],bt[10],wt[10],tat[10];

// Sort the burst time -- selection sort technique is used
void sortbt(){
        int i,j,pos,temp;
        for(i=0;i<n;i++){
                pos=i;
                for(j=i+1;j<n;j++){
                        if(bt[j]<bt[pos])
                                pos=j;
```

```
            }
            //burst time is swapped
            temp=bt[pos];
            bt[pos]=bt[i];
            bt[i]=temp;
            //Process number is swapped
            temp=process[pos];
            process[pos]=process[i];
            process[i]=temp;
        }
}

//Calculating the waiting time
void waitingtime(){
        int i;
        wt[0]=0;
        for(i=1;i<n;i++)
                wt[i]=bt[i-1]+wt[i-1];
}

//Calculating the turn around time
void turnaroundtime(){
        int i;
        for(i=0;i<n;i++)
                tat[i]=bt[i]+wt[i];
}

//Calculating the average waiting time and turnaround time
void average(){
        float awt,atat;
        int twt=0,ttat=0,i;
        sortbt();
        waitingtime();
        turnaroundtime();
        printf("Waiting time and Turn around time for each process: \n");
        for(i=0;i<n;i++){
                printf("Process %d:\t%d\t%d\n",process[i],wt[i],tat[i]);
                ttat+=tat[i];
                twt+=wt[i];
        }
        awt=(float)twt/n;
        atat=(float)ttat/n;
        printf("The average of the waiting time: %f\n",awt);
        printf("The average of the turnaround time : %f\n",atat);
}
//Displaying the gantt chart
void ganttchart(){
        int i;
        printf("| ");
        for(i=0;i<n;i++)
                printf("P%d | ",process[i]);
```

```
        printf("\n0    ");
        for(i=0;i<n;i++)
                printf("%d    ",tat[i]);
}

//Main function
void main(){
        int i;
        clrscr();
        printf("\t\t\tSJF\n");
        printf("Enter the number of the process: ");
        scanf("%d",&n);
        printf("Enter the burst time of the each process: \n");
        for(i=0;i<n;i++){
                printf("Process%d : ",i+1);
                scanf("%d",&bt[i]);
                process[i]=i+1;
        }
        average();
        ganttchart();
        getch();
}
```

OUTPUT:

```
                        SJF
Enter the number of the process: 3
Enter the burst time of the each process:
Process1 : 24
Process2 : 3
Process3 : 3
Waiting time and Turn around time for each process:
Process 2:       0        3
Process 3:       3        6
Process 1:       6        30
The average of the waiting time: 3.000000
The average of the turnaround time : 13.000000
| P2 | P3 | P1 |
0    3    6    30
```

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

RESULT:

     Thus Waiting time and Turnaround time for processes based on SJF scheduling was computed and the average waiting time was determined.