

Ex. No.: 7 a
Date: 02.04.2024

FIRST COME FIRST SERVE

Aim:

To implement First-come First- serve(FCFS) scheduling technique

Program Code:

```
def fcfs_scheduling(processes):
    n = len(processes)
    wait_time = [0] * n
    turnaround_time = [0] * n

    for i in range(1, n):
        wait_time[i] = processes[i-1][1] + wait_time[i-1]

    for i in range(n):
        turnaround_time[i] = processes[i][1] + wait_time[i]

    total_waiting_time = sum(wait_time)
    total_turnaround_time = sum(turnaround_time)

    average_waiting_time = total_waiting_time / n
    average_turnaround_time = total_turnaround_time / n

    print("Process\tBurst Time\tWaiting Time\tTurnaround Time")
    for i in range(n):
        print(f' {processes[i][0]} \t {processes[i][1]} \t {wait_time[i]} \t {turnaround_time[i]}')

    print(f'\nTotal Waiting Time: {total_waiting_time}')
    print(f'Average Waiting Time: {average_waiting_time:.2f}')
    print(f'Total Turnaround Time: {total_turnaround_time}')
    print(f'Average Turnaround Time: {average_turnaround_time:.2f}')

num_processes = int(input("Enter the number of processes: "))

processes = []
for i in range(num_processes):
    process_name = input(f'Enter the name of process {i+1}: ')
    burst_time = int(input(f'Enter the burst time for process {process_name}: '))
    processes.append((process_name, burst_time))
```

fcfs_scheduling(processes)

Output:

```
(kali㉿kali)-[~/os/ex7a]
$ python3 ex7a.py
Enter the number of processes: 3
Enter the name of process 1: p1
Enter the burst time for process p1: 24
Enter the name of process 2: p2
Enter the burst time for process p2: 3
Enter the name of process 3: p3
Enter the burst time for process p3: 3
Process Burst Time      Waiting Time      Turnaround Time
p1      24              0              24
p2       3             24              27
p3       3             27              30

Total Waiting Time: 51
Average Waiting Time: 17.00
Total Turnaround Time: 81
Average Turnaround Time: 27.00
```

Result:

The above program executed successfully and output got verified.

Ex. No.: 7b
Date: 06.04.2024

SHORTEST JOB FIRST

Aim:

To implement the Shortest Job First(SJF) scheduling technique

Program Code:

```
bt=[]
print("Enter the number of process: ")
n=int(input())
processes=[]
for i in range(0,n):
    processes.insert(i,i+1)
print("Enter the burst time of the processes: \n")
bt=list(map(int, input().split()))
for i in range(0,len(bt)-1):
    for j in range(0,len(bt)-i-1):
        if(bt[j]>bt[j+1]):
            temp=bt[j]
            bt[j]=bt[j+1]
            bt[j+1]=temp
            temp=processes[j]
            processes[j]=processes[j+1]
            processes[j+1]=temp

wt = []
avgwt = 0
tat = []
avgtat = 0
wt.insert(0,0)
tat.insert(0,0)
for i in range(1,len(bt)):
    wt.insert(i,wt[i-1]+bt[i-1])
    tat.insert(i,wt[i]+bt[i])
    avgwt+=wt[i]
    avgtat+=tat[i]
avgwt=float(avgwt)/n
avgtat=float(avgtat)/n
print("\n")
print("Process\t Burst Time\t Waiting Time\t Turn Around Time")
for i in range(0,n):
    print(str(processes[i])+"\t\t"+str(bt[i])+"\t\t"+str(wt[i])+"\t\t"+str(tat[i]))
print("Average Waiting time is: "+str(avgwt))
print("Average Turn Around Time is: "+str(avgtat))
```

Output:

```
(kali㉿kali)-[~/os/ex7b]
$ python3 ex7b.py
Enter the number of process:
4
Enter the burst time of the processes:

8 4 9 5

Process  Burst Time      Waiting Time    Turn Around Time
2         4              0              4
4         5              4              9
1         8              9             17
3         9             17             26
Average Waiting time is: 7.5
Average Turn Around Time is: 13.0
```

Result:

The above program executed successfully and output got verified.

Ex. No.: 7 c
Date: 06.04.2024

PRIORITY SCHEDULING

Aim:

To implement priority scheduling technique

Program 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; //contains process number
    }
    //sorting burst time, priority and process number in ascending order using selection sort
    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; //waiting time for first process is 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=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=total/n; //average turnaround time
printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\nAverage Turnaround Time=%d\n",avg_tat);
return 0;
}

```

Output:

```
(kali㉿kali)-[~/os/ex7c]
$ ./ex7c
Enter Total Number of Process:4
```

```
Enter Burst Time and Priority
```

```
P[1]
Burst Time:6
Priority:3
```

```
P[2]
Burst Time:2
Priority:2
```

```
P[3]
Burst Time:14
Priority:1
```

```
P[4]
Burst Time:6
Priority:4
```

Process	Burst Time	Waiting Time	Turnaround Time
P[3]	14	0	14
P[2]	2	14	16
P[1]	6	16	22
P[4]	6	22	28

```
Average Waiting Time=13
Average Turnaround Time=20
```

Result:

The above program executed successfully and output got verified.

Ex. No.: 7d
Date 08.04.2024

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

Program Code:

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

```
int main() {
    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
    float average_wait_time, average_turnaround_time;

    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    x = limit;

    for (i = 0; i < limit; i++) {
        printf("\nEnter Details of Process[%d]\n", i + 1);
        printf("Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time:\t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }

    printf("\nEnter Time Quantum:\t");
    scanf("%d", &time_quantum);

    printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time\n");

    for (total = 0, i = 0; x != 0;) {
        if (temp[i] <= time_quantum && temp[i] > 0) {
            total = total + temp[i];
            temp[i] = 0;
            counter = 1;
        } else if (temp[i] > 0) {
            temp[i] = temp[i] - time_quantum;
            total = total + time_quantum;
        }

        if (temp[i] == 0 && counter == 1) {
            x--;
            printf("\nProcess[%d]\t%d\t%d\t%d", i + 1, burst_time[i], total - arrival_time[i], total -
```

```

arrival_time[i] - burst_time[i]);
    wait_time = wait_time + total - arrival_time[i] - burst_time[i];
    turnaround_time = turnaround_time + total - arrival_time[i];
    counter = 0;
}

if(i == limit - 1) {
    i = 0;
} else if (arrival_time[i + 1] <= total) {
    i++;
} else {
    i = 0;
}
}

average_wait_time = wait_time * 1.0 / limit;
average_turnaround_time = turnaround_time * 1.0 / limit;

printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);

return 0;
}

```

Output:

```
(kali㉿kali)-[~/os/ex7d]
$ ./ex7d

Enter Total Number of Processes:      4

Enter Details of Process[1]
Arrival Time:    0
Burst Time:      4

Enter Details of Process[2]
Arrival Time:    1
Burst Time:      7

Enter Details of Process[3]
Arrival Time:    2
Burst Time:      5

Enter Details of Process[4]
Arrival Time:    3
Burst Time:      6

Enter Time Quantum:    3

Process ID      Burst Time      Turnaround Time      Waiting Time
Process[1]      4              13                   9
Process[3]      5              16                   11
Process[4]      6              18                   12
Process[2]      7              21                   14

Average Waiting Time:    11.500000
Avg Turnaround Time:    17.000000
```

Result:

The above program executed successfully and output got verified.

