**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\t{wait\_time[i]}\t\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):

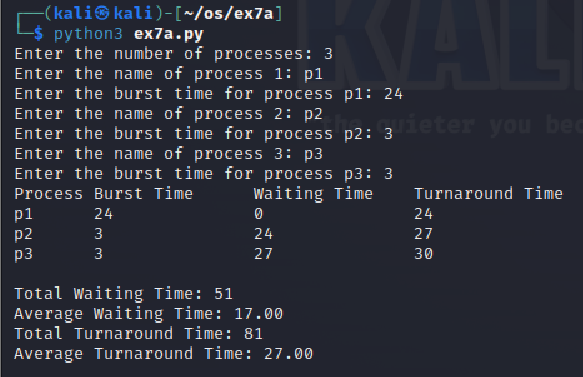
process\_name = input(f"Enter the name of process {i+1}: ")

burst\_time = int(input(f"Enter the burst time for process {process\_name}: "))

processes.append((process\_name, burst\_time))

fcfs\_scheduling(processes)

**Output:**

****

**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,bt[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")

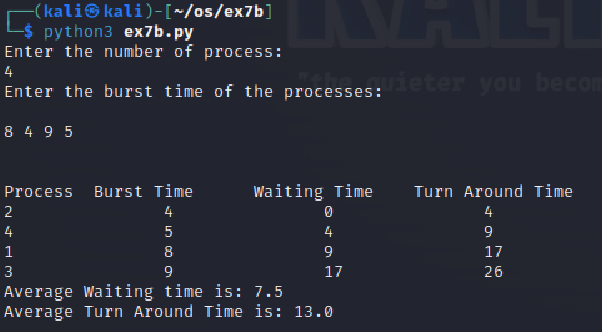
for i in range(0,n):

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:**

****

**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

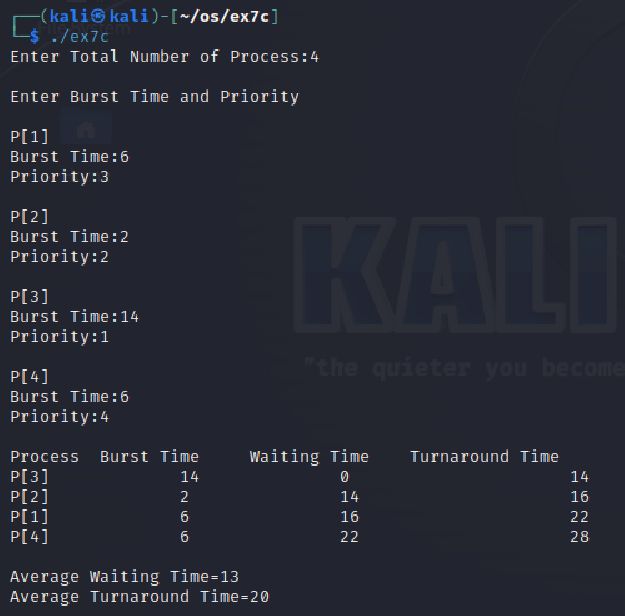
printf("\n\nAverage Waiting Time=%d",avg\_wt);

printf("\nAverage Turnaround Time=%d\n",avg\_tat);

return 0;

}

**Output:**

****

**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\t%d\t\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;

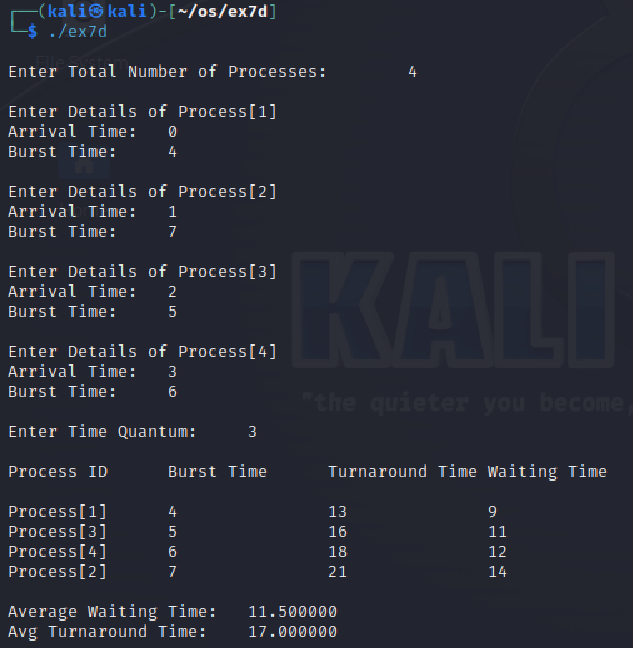
printf("\n\nAverage Waiting Time:\t%f", average\_wait\_time);

printf("\nAvg Turnaround Time:\t%f\n", average\_turnaround\_time);

return 0;

}

**Output:**

****

**Result:**

The above program executed successfully and output got verified.