**Ex No: 5 CPU SCHEDULING**

**Date : 01.04.2021**

**Aim**

To schedule the processes queued according to FCFS, SJF, PRIORITY, ROUNDROBIN scheduling.

**1.** **FCFS Scheduling.**

**Algorithm**

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 (n)

3. Obtain btime for each process.

4. The wtime for first process is 0.

5. Compute wtime and ttime for each process as:

a. wtime i+1 = wtimei + btimei

b. ttime i = wtimei + btimei

6. Compute average waiting time awat and average turnaround time atur.

7. Display the btime, ttime and wtime for each process.

8. Display awat time and atur

9. Stop

**Program**

#include <stdio.h>

struct process

{

int pid;

int btime;

int wtime;

int ttime;

} p[10];

main()

{

int i,j,k,n,ttur,twat;

float awat,atur;

printf("Enter no. of process : ");

scanf("%d", &n);

for(i=0; i<n; i++)

{

printf("Burst time for process P%d (in ms) : ",(i+1));

scanf("%d", &p[i].btime);

p[i].pid = i+1;

}

p[0].wtime = 0;

for(i=0; i<n; i++)

{

p[i+1].wtime = p[i].wtime + p[i].btime;

p[i].ttime = p[i].wtime + p[i].btime;

}

ttur = twat = 0;

for(i=0; i<n; i++)

{

ttur += p[i].ttime;

twat += p[i].wtime;

}

awat = (float)twat / n;

atur = (float)ttur / n;

printf("\n FCFS Scheduling\n\n");

for(i=0; i<28; i++)

printf("-");

printf("\nProcess B-Time T-Time W-Time\n");

for(i=0; i<28; i++)

printf("-");

for(i=0; i<n; i++)

printf("\n P%d\t%4d\t%3d\t%2d",

p[i].pid,p[i].btime,p[i].ttime,p[i].wtime);

printf("\n");

for(i=0; i<28; i++)

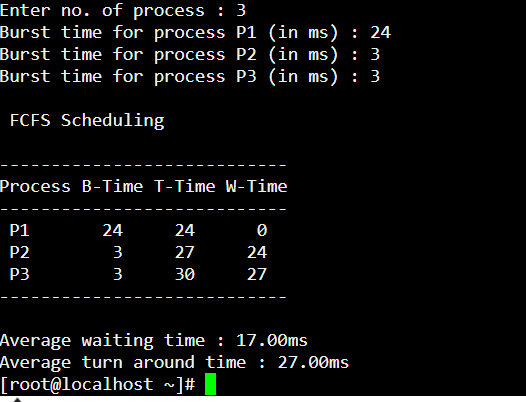
printf("-");

printf("\n\nAverage waiting time : %5.2fms", awat);

printf("\nAverage turn around time : %5.2fms\n", atur);

}

**Output**



**2.SJF Scheduling.**

**Algorithm**

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 (n)

3. Obtain btime for each process.

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

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

5. The wtime for first process is 0.

6. Compute wtime and ttime for each process as:

a. wtimei+1 = wtimei + btimei

b. ttimei = wtimei + btimei

7. Compute average waiting time awat and average turn around time atur.

8. Display btime, ttime and wtime for each process.

9. Display awat and atur

10. Stop

**Program**

#include <stdio.h>

struct process

{

int pid;

int btime;

int wtime;

int ttime;

} p[10], temp;

main()

{

int i,j,k,n,ttur,twat;

float awat,atur;

printf("Enter no. of process : ");

scanf("%d", &n);

for(i=0; i<n; i++)

{

printf("Burst time for process P%d (in ms) : ",(i+1));

scanf("%d", &p[i].btime);

p[i].pid = i+1;

}

for(i=0; i<n-1; i++)

{

for(j=i+1; j<n; j++)

{

if((p[i].btime > p[j].btime) ||

(p[i].btime == p[j].btime && p[i].pid > p[j].pid))

{

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

p[0].wtime = 0;

for(i=0; i<n; i++)

{

p[i+1].wtime = p[i].wtime + p[i].btime;

p[i].ttime = p[i].wtime + p[i].btime;

}

ttur = twat = 0;

42

for(i=0; i<n; i++)

{

ttur += p[i].ttime;

twat += p[i].wtime;

}

awat = (float)twat / n;

atur = (float)ttur / n;

printf("\n SJF Scheduling\n\n");

for(i=0; i<28; i++)

printf("-");

printf("\nProcess B-Time T-Time W-Time\n");

for(i=0; i<28; i++)

printf("-");

for(i=0; i<n; i++)

printf("\n P%-4d\t%4d\t%3d\t%2d",

p[i].pid,p[i].btime,p[i].ttime,p[i].wtime);

printf("\n");

for(i=0; i<28; i++)

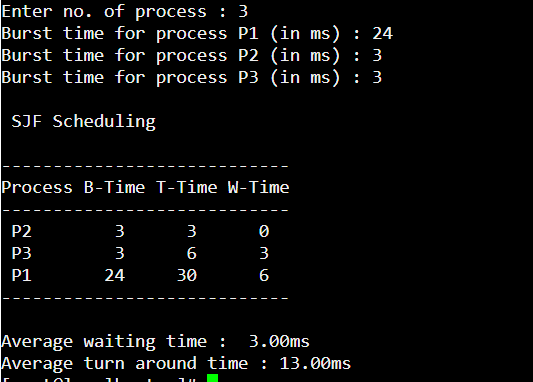
printf("-");

printf("\n\nAverage waiting time : %5.2fms", awat);

printf("\nAverage turn around time : %5.2fms\n", atur);

}

**Output**



**3.PRIORITY Scheduling.**

**Algorithm**

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

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

3. Obtain btime 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 wtime for first process is 0.

6. Compute wtime and ttime for each process as:

a. wtimei+1 = wtimei + btimei

b. ttimei = wtimei + btimei

7. Compute average waiting time awat and average turn around time atur

8. Display the btime, pri, ttime and wtime for each process.

9. Display awat and atur

10. Stop

**Program**

#include <stdio.h>

struct process

{

int pid;

int btime;

int pri;

int wtime;

int ttime;

} p[10], temp;

main()

{

int i,j,k,n,ttur,twat;

float awat,atur;

printf("Enter no. of process : ");

scanf("%d", &n);

for(i=0; i<n; i++)

{

printf("Burst time for process P%d (in ms) : ", (i+1));

scanf("%d", &p[i].btime);

printf("Priority for process P%d : ", (i+1));

scanf("%d", &p[i].pri);

p[i].pid = i+1;

}

for(i=0; i<n-1; i++)

{

for(j=i+1; j<n; j++)

{

if((p[i].pri > p[j].pri) ||

(p[i].pri == p[j].pri && p[i].pid > p[j].pid) )

{

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

p[0].wtime = 0;

for(i=0; i<n; i++)

{

p[i+1].wtime = p[i].wtime + p[i].btime;

p[i].ttime = p[i].wtime + p[i].btime;

}

ttur = twat = 0;

for(i=0; i<n; i++)

{

ttur += p[i].ttime;

twat += p[i].wtime;

}

awat = (float)twat / n;

atur = (float)ttur / n;

printf("\n\t Priority Scheduling\n\n");

for(i=0; i<38; i++)

printf("-");

printf("\nProcess B-Time Priority T-Time W-Time\n");

for(i=0; i<38; i++)

printf("-");

for (i=0; i<n; i++)

printf("\n P%-4d\t%4d\t%3d\t%4d\t%4d",

p[i].pid,p[i].btime,p[i].pri,p[i].ttime,p[i].wtime);

printf("\n");

for(i=0; i<38; i++)

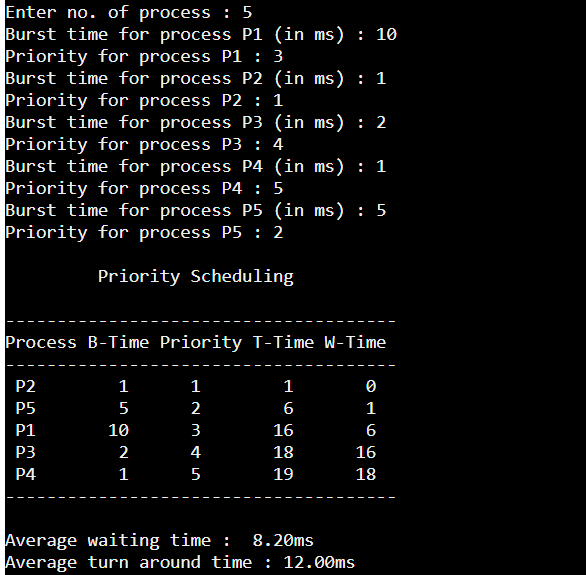
printf("-");

printf("\n\nAverage waiting time : %5.2fms", awat);

printf("\nAverage turn around time : %5.2fms\n", atur);

}

**Output**



**4.ROUND ROBIN Scheduling.**

**Algorithm**

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

2. Obtain Burst time and arrival 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 burst time, arrival time , turnaround time and wait time for each process (in order of rounds they were processed).

9. Display average wait time and turnaround time

10. Stop

**Program**

#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\t\tBurst Time\t Turnaround Time\t Waiting 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\t%d\t\t %d\t\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;

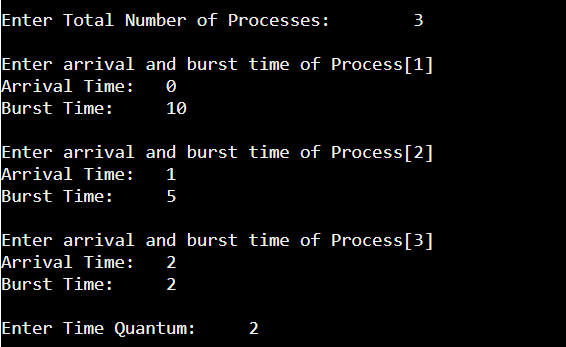
printf("\n\nAverage Waiting Time:\t%f", average\_wait\_time);

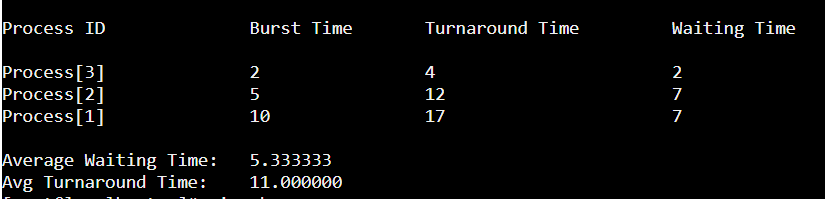
printf("\nAvg Turnaround Time:\t%f\n", average\_turnaround\_time);

return 0;

}

**Output**





|  |  |
| --- | --- |
| **Observation(20)** |  |
| **Record(5)** |  |
| **Total(25)** |  |
| **Initial** |  |

**Result:**

Thus waiting time and turnaround time for processes based on fcfs , sjf , priority ,Round robin scheduling was computed and the average waiting time and average turnaroundtime was determined.