**First Come First Serve (FCFS)**

Experiment No : 3 Date :-

Aim :- To implement First Come First Serve(FCFS) CPU Scheduling Algorithm

Theory :

The **First Come First Served (FCFS)** algorithm, also known as First In First Out (FIFO), is a straightforward non-preemptive scheduling technique. It prioritizes processes based on their request order for CPU time. The earliest arriving process is allocated the CPU initially, following a sequential approach using a queue. Processes are added to the end of the queue upon arrival and removed from the front once their execution is complete.

For a given set of n processes with their respective burst times, the goal is to determine the average waiting time and average turnaround time using the FCFS algorithm. In this method, tasks are scheduled based on their arrival time, with the first arrival gaining CPU access first. It's important to note that this description assumes all processes have an arrival time of 0.

Nonetheless, FCFS scheduling may introduce the issue of starvation if the initial process has the longest burst time among all tasks. Efficient process scheduling is crucial to ensure work completion within deadlines. Processes involve both I/O and CPU time, and optimizing scheduling helps maximize CPU utilization, allocate resources fairly, and enhance throughput.

In the context of process scheduling, several terms hold significance:

**Arrival Time**: The moment a process enters the ready queue.

**Completion** Time: When a process finishes its execution.

**Burst Time**: The time needed for a process to execute on the CPU.

**Turnaround Time**: The interval between completion and arrival times: Turnaround Time = Completion Time – Arrival Time.

**Waiting Time (W.T):** The gap between turnaround and burst times: Waiting Time = Turnaround Time – Burst Time.

Example :

|  |  |
| --- | --- |
| **Processes** | **Burst Time** |
| **P1** | **5** |
| **P2** | **3** |
| **P3** | **4** |
| **P4** | **2** |

**Calculating Waiting Time and Turnaround Time:**

Waiting Time= Burst time of previous process + Waiting Time of previous process

TurnAround time = Waiting time+Burst time (current process)

* For P1,

Waiting time(wt[1])= 0

=0+5

=5

Turnaround Time = 5 + 0

=0

* For P2,

Waiting Time(wt[2]) =Burst time(P1)+Waiting Time(wt[1])

**=**5+0

=5

Turnaround Time = 3 + 5

=8

* For P3,

Waiting Time(wt[3]) =Burst time(P2)+Waiting Time(wt[2])

=3+5

=8

Turnaround Time = 4 + 8

=12

* For P4,

Waiting Time(wt[4]) =Burst time(P3)+Waiting Time(wt[3])

=4+8

=12

Turnaround Time = 2+ 12

=14

|  |  |  |  |
| --- | --- | --- | --- |
| Processes | Burst Time | Waiting Time | Turnaround Time |
| 1 | 5 | 0 | 5 |
| 2 | 3 | 5 | 8 |
| 3 | 4 | 8 | 12 |
| 4 | 2 | 12 | 14 |

**Average Burst Time=3.5 seconds**

**Average Waiting Time=6.25**

**Average Turnaround Time=9.75**

**Gantt Chart**

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P2 | P3 | P4 |

1. 5 8 12 14

Code

#include<bits/stdc++.h>

using namespace std;

void input\_burst\_time(int n ,int burst\_time[]){

cout<<"Enter Burst Time"<<endl;

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

cin>>burst\_time[i];

}}

void calculate\_waiting(int n ,int wt\_time[],int burst\_time[]){

for (int i = 1; i < n; i++){

wt\_time[i]= burst\_time[i-1] + wt\_time[i-1];

}

cout<<endl;

}

void calulate\_turn\_around(int n , int wt\_time[],int burst\_time[],int turn\_around[]){

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

turn\_around[i] = wt\_time[i] + burst\_time[i];

}

cout<<endl;

}

void display(int n,int wt\_time[],int turn\_around[]){

cout<<"Waiting time of processes"<<endl;

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

cout<<wt\_time[i]<<" ";

}

cout<<"\nTurn around time of processes"<<endl;

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

cout<<turn\_around[i]<<" ";

}

cout<<"\n";

}

void gchart(int n ,int burst\_time[]){

cout<<"\nGantt Chart:"<<endl;

int current\_time = 0;

int size=3;

for (int i = 0; i < n\*7; i++){

cout<<'-';

}

cout<<endl;

cout<<"| ";

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

cout<<'p'<<i+1<<" | ";

}

cout<<"\n";

for (int i = 0; i < n\*7; i++){

cout<<'-';

}

cout<<"\n";

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

cout<<current\_time<<" ";

current\_time+= burst\_time[i];

if(i == n-1){

cout<<current\_time;

} }

cout<<"\n";

}

void avg\_tat(int n,int turn\_around[],int wt\_time[]){

float avg\_sum =0.0;

float avg\_wt = 0.0;

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

avg\_sum += turn\_around[i];

avg\_wt += wt\_time[i];

}

cout<< "\nAverage waiting time is "<<(avg\_wt/n)<<endl;

cout<< "\nAverage Turn around time is "<<(avg\_sum/n)<<endl;

}

int main(){

int n;

cout<<"Enter no of processes : "<<endl;

cin>>n;

int burst\_time[n]={0};

input\_burst\_time(n,burst\_time);

int wt\_time[n]={0};

calculate\_waiting(n,wt\_time,burst\_time);

int turn\_around[n]={0};

calulate\_turn\_around(n,wt\_time,burst\_time,turn\_around);

display(n,wt\_time,turn\_around);

cout<<"\n";

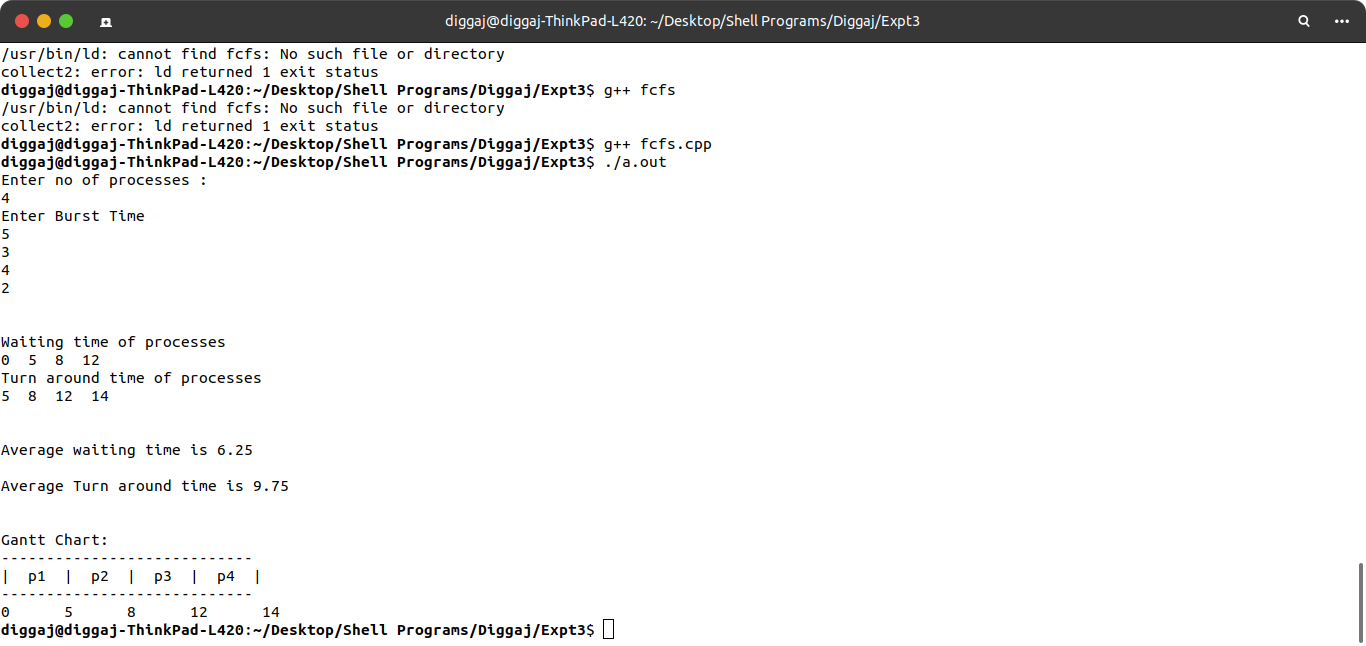
avg\_tat(n,turn\_around,wt\_time);

cout<<"\n";

gchart(n,burst\_time);

}

Output :



**Conclusion** : First Come First Serve (FCFS) CPU Scheduling Algorithm was successfully studied and implemented.