**KIET**

**Group**

**of**

**Institutions, Ghaziabad**

**COMPUTER**

**SCIENCE**

**AND**

**INFORMATION**

**TECHNOLOGY**

**PROJECT**

**BASED**

**LEARNING**

**on**

**CPU Scheduling**

**SUBJECT**

**:**

**Operating System**

**(**

**KCS**

**-**

**451)**

**Submitted By:**

**ABHINAV**

**SHARMA**

**-**

**2100290110004**

**(**

**CSIT**

**3**

**A**

**)**

**GYANVI**

**ASHISH**

**CHAUDHARY**

**-**

**2100290110057**

**CSIT**

**(**

**3**

**A**

**)**



# ACKNOWLEDGMENT

I’ve got this golden opportunity to express my kind gratitude and sincere thanks to my subject faculty “Mr. Vinay Kumar”, Computer Science and Information Technology Department, KIET GROUP OF INSTITUTIONS for their kind support and necessary counselling in the preparation of this project report. I’m also indebted to each and every person responsible for the making up of this project directly or indirectly. I must also acknowledge or deep debt of gratitude each one of my colleague who led this project come out in the way it is. It’s my hard work and untiring sincere efforts and mutual cooperation to bring out the project work. Last but not the least, I would like to thank my parents for their sound counselling and cheerful support. They have always inspired us and kept our spirit up.

# PROJECT ABSTRACT

**CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended**.

The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line.

It is compiled in online compiler.

# INDEX

1. **INTRODUCTION**

1. **REQUIREMENT ANALYSIS**

1. **DESIGN**

1. **CODE IMPLEMENTATION**

1. **RESULT/OUTPUT**

1. **REFERENCES**

**INTRODUCTION**

# PROBLEM STATEMENT:-

TO IMPLEMENT CPU SCHEDULING USING C LANGUAGE

Central Processing Unit (CPU) scheduling plays a deep-seated role by switching the CPU among various processes. As processor is the important resource, CPU scheduling becomes very important in accomplishing the operating system (OS) design goals. The intention of the OS should allow the process as many as possible running at all the time in order to make best use of CPU. The high efficient CPU scheduler depends on design of the high quality scheduling algorithms which suits the scheduling goals. In this paper, we proposed an algorithm which can handle all types of process with optimum scheduling criteria

**REQUIREMENT ANALYSIS**

**THIS PROJECT WILL REQUIRE:-**

* C COMPILER

* DATA STRUCTURE(QUEUE**)**

# DESIGN

**FCFS SCHEDULING:**

#include <stdio.h>

int main()

{

int pid[15];

int bt[15];

int n;

printf("Enter the number of processes: ");

scanf("%d",&n);

printf("Enter process id of all the processes: ");

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

{

scanf("%d",&pid[i]);

}

printf("Enter burst time of all the processes: ");

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

{

scanf("%d",&bt[i]);

}

int i, wt[n];

wt[0]=0;

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

{

wt[i]= bt[i-1]+ wt[i-1];

}

printf("Process ID Burst Time Waiting Time TurnAround Time\n");

float twt=0.0;

float tat= 0.0;

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

{

printf("%d\t\t", pid[i]);

printf("%d\t\t", bt[i]);

printf("%d\t\t", wt[i]);

printf("%d\t\t", bt[i]+wt[i]);

printf("\n");

twt += wt[i];

tat += (wt[i]+bt[i]);

}

float att,awt;

awt = twt/n;

att = tat/n;

printf("Avg. waiting time= %f\n",awt);

printf("Avg. turnaround time= %f",att);

}

**SJF SCHEDULING:**

#include <stdio.h>

int main()

{

int A[100][4];

int i, j, n, total = 0, index, temp;

float avg\_wt, avg\_tat;

printf("Enter number of process: ");

scanf("%d", &n);

printf("Enter Burst Time:\n");

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

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

scanf("%d", &A[i][1]);

A[i][0] = i + 1;

}

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

index = i;

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

if (A[j][1] < A[index][1])

index = j;

temp = A[i][1];

A[i][1] = A[index][1];

A[index][1] = temp;

temp = A[i][0];

A[i][0] = A[index][0];

A[index][0] = temp;

}

A[0][2] = 0;

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

A[i][2] = 0;

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

A[i][2] += A[j][1];

total += A[i][2];

}

avg\_wt = (float)total / n;

total = 0;

printf("P BT WT TAT\n");

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

A[i][3] = A[i][1] + A[i][2];

total += A[i][3];

printf("P%d %d %d %d\n", A[i][0],

A[i][1], A[i][2], A[i][3]);

}

avg\_tat = (float)total / n;

printf("Average Waiting Time= %f", avg\_wt);

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

}

**PRIORITY SCHEDULING**

#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;

}

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;

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;

total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

tat[i]=bt[i]+wt[i];

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;

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

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

return 0;

}

**ROUNDROBIN SCHEDULING**

#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 IDttBurst Timet Turnaround Timet Waiting Timen");

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]tt%dtt %dttt %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("nnAverage Waiting Time:t%f", average\_wait\_time);

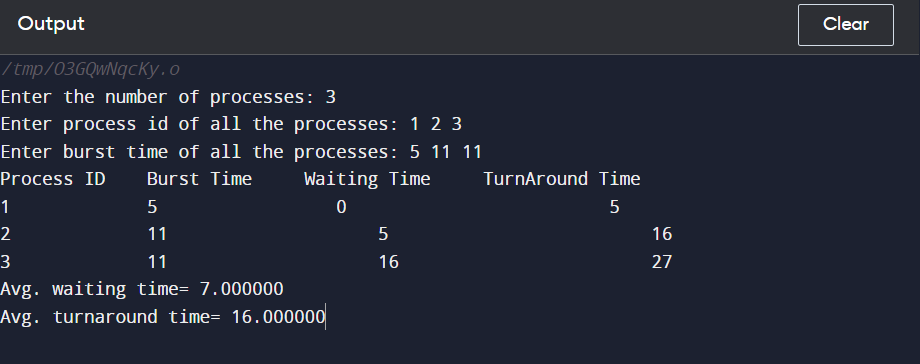
printf("nAvg Turnaround Time:t%fn", average\_turnaround\_time);

return 0;

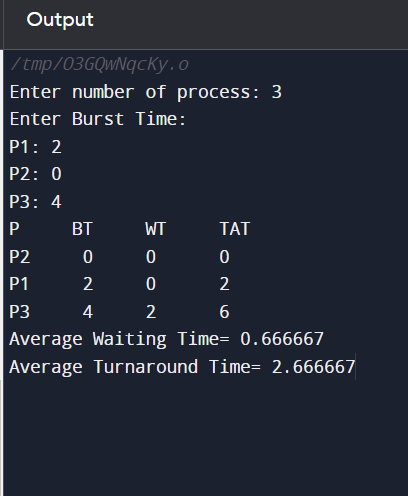
}

# RESULT/OUTPUT

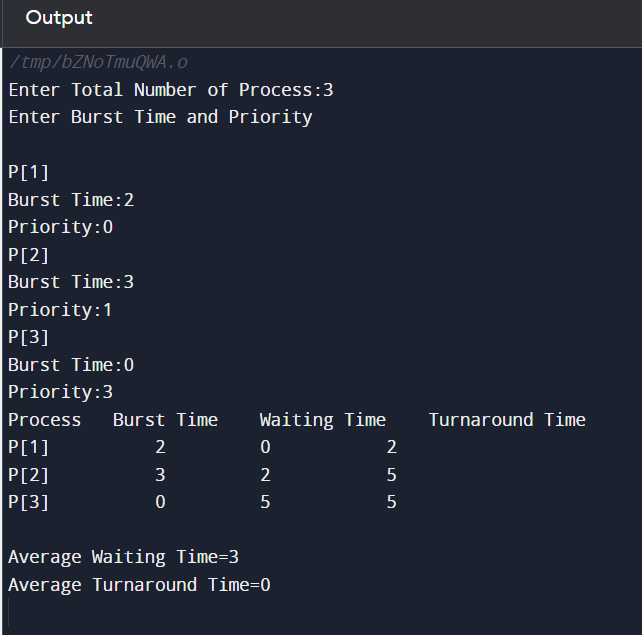
**FCFS SCHEDULING**



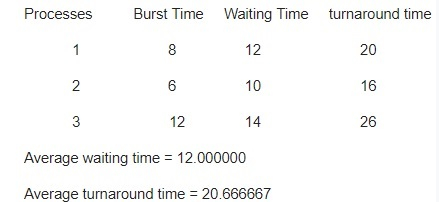
SJF SCHEDULING



PRIORITY SCHEDULING



ROUND ROBIN SCHEDULING



**REFERENCES**

1. [Priority Scheduling Program in C and C++ (thecrazyprogrammer.com)](https://www.thecrazyprogrammer.com/2014/11/c-cpp-program-for-priority-scheduling-algorithm.html)
2. [Round Robin Scheduling in C Programming | Edureka](https://www.edureka.co/blog/round-robin-scheduling-in-c/)
3. [Shortest Job First Scheduling in C Programming | Edureka](https://www.edureka.co/blog/sjf-scheduling-in-c/)

# GITHUB LINK