# **ASG 07**

**Name:** Yatharth Thakare

**Roll No:** 51

**PRN:** 12111403

**Subject:** OS

**Title: Scheduling Algorithms**

1. **FCFS**

Code:

#include<stdio.h>

#include<stdlib.h>

#define max 30

void fcfs(){

int i,j, n, bt[max], wt[max], tat[max], at[max],pid[max];

float wtavg, tatavg;

printf("\nEnter the number of processes (MAX 30)-- ");

scanf("%d", &n);

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

printf("\n Enter the arrival time of process %d -- ", i);

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

pid[i] = i;

}

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

printf("\n Enter the burst time of process %d -- ", i);

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

}

wt[0] = wtavg = 0;

tat[0] = tatavg = bt[0];

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

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

if(at[j] > at[j+1]){

// Swap arrival time

int temp = at[j];

at[j] = at[j+1];

at[j+1] = temp;

// Swap burst time

temp = bt[j];

bt[j] = bt[j+1];

bt[j+1] = temp;

// Swap process id

temp = pid[j];

pid[j] = pid[j+1];

pid[j+1] = temp;

}

}

}

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

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

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

wtavg = wtavg + wt[i];

tatavg = tatavg + tat[i];

}

printf("\n\t PROCESS \t ARRIVAL TIME \t BURST TIME \t WAITING TIME \t TURNAROUND TIME\n");

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

printf("\n\t P[%d] \t\t %d \t\t %d \t\t %d \t\t %d", pid[i], at[i], bt[i], wt[i], tat[i]);

}

printf("\nAverage Waiting Time -- %f", wtavg/n);

printf("\nAverage Turnaround Time -- %f", tatavg/n);

printf("\nGantt Chart:\n");

printf(" ");

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

printf("-----------");

}

printf("\n|");

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

printf(" P[%d] |", pid[i]);

}

printf("\n ");

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

printf("-----------");

}

printf("\n0");

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

printf(" %d", tat[i]);

}

}

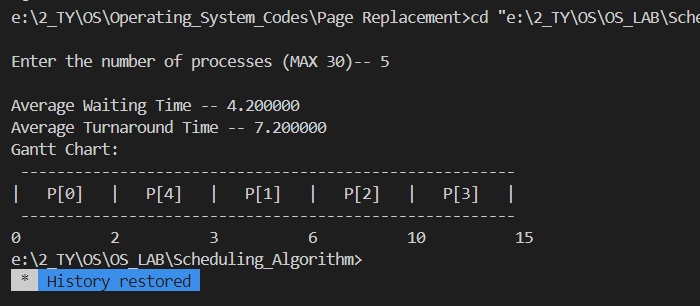
int main(){

fcfs();

return 0;

}

Output :



1. **SJF**

Code:

#include<stdio.h>

#include<stdlib.h>

#define max 30

void sjf(){

int i,j, n, bt[max], wt[max], tat[max], at[max],pid[max],temp;

float wtavg, tatavg;

printf("\nEnter the number of processes (MAX 30)-- ");

scanf("%d", &n);

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

printf("\n Enter the burst time of process %d -- ", i);

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

pid[i] = i;

}

wt[0] = wtavg = 0;

tat[0] = tatavg = bt[0];

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

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

if(bt[j] > bt[j+1]){

// Swap burst time

temp = bt[j];

bt[j] = bt[j+1];

bt[j+1] = temp;

// Swap process id

temp = pid[j];

pid[j] = pid[j+1];

pid[j+1] = temp;

}

}

}

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

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

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

wtavg = wtavg + wt[i];

tatavg = tatavg + tat[i];

}

printf("\n\t PROCESS \t BURST TIME \t WAITING TIME \t TURNAROUND TIME\n");

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

printf("\n\t P[%d] \t\t %d \t\t %d \t\t %d", pid[i], bt[i], wt[i], tat[i]);

}

printf("\nAverage Waiting Time -- %f", wtavg/n);

printf("\nAverage Turnaround Time -- %f", tatavg/n);

printf("\nGantt Chart:\n");

printf(" ");

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

printf("-----------");

}

printf("\n|");

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

printf(" P[%d] |", pid[i]);

}

printf("\n ");

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

printf("-----------");

}

printf("\n0");

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

printf(" %d", tat[i]);

}

printf("\n");

}

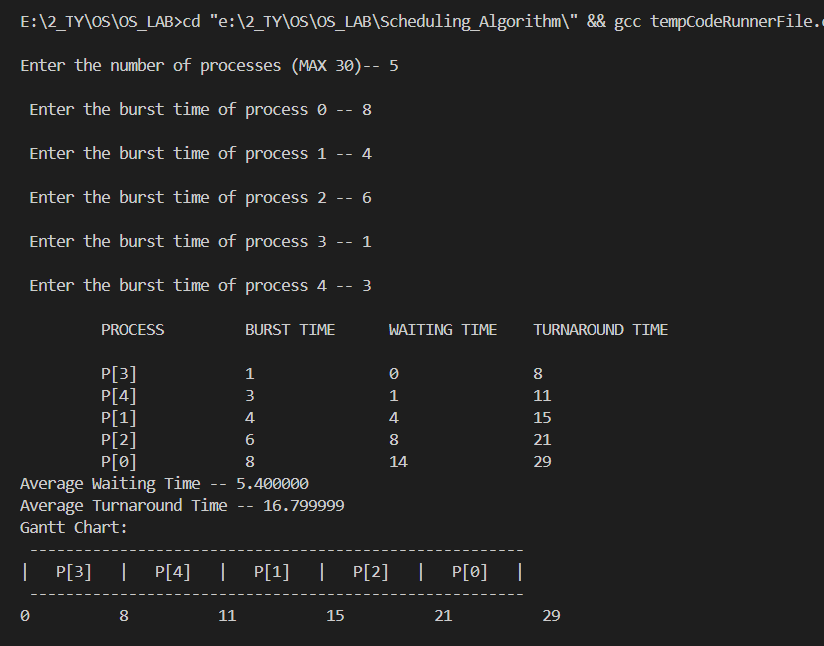
int main(){

sjf();

return 0;

}

Output:



1. **Round Robin**

**Code:**

#include <stdio.h>

#include <stdlib.h>

typedef struct

{

int pid;

int bt;

int at;

int rt;

int ft;

int tat;

int wt;

} Process;

// Sort acc to arrival time, if arrival time same then compare burst time.

void sort(int n, Process p[n])

{

int i, j;

Process temp;

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

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

if ((p[j].at > p[j + 1].at) || ((p[j].at == p[j + 1].at) && (p[j].bt > p[j + 1].bt)))

{

temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

}

}

// t = t % tq

int countIteration(int n, Process p[n], int tq)

{

int i, j = 0;

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

{

if (p[i].bt % tq != 0)

j = j + (p[i].bt / tq) + 1;

else

j = j + (p[i].bt / tq);

}

return j;

}

void print\_chart(int n, Process p[n])

{

int i, j;

printf("\n\t+");

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

{

for (j = 1; j <= p[i].bt; j++)

printf("---");

printf("+");

}

printf("\n\t|");

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

{

printf(" P%d", p[i].pid);

for (j = 1; j < p[i].bt; j++)

printf(" ");

printf("|");

}

printf("\n\t+");

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

{

for (j = 1; j <= p[i].bt; j++)

printf("---");

printf("+");

}

int check = 0;

printf("\n\t%d", p[0].at);

if (p[0].at > 9)

check = 1;

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

{

for (j = 1; j <= p[i].bt; j++)

if (check == 1)

{

printf(" ");

check = 0;

}

else

printf(" ");

printf("%d", p[i].ft);

if (p[i].ft > 9)

check = 1;

}

}

//main rr function

int execute(int n, Process p[n], int o, Process exec[o], int t, int tq)

{

int i, j;

j = 0;

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

{

if (i == 0)

t = p[i].at;

if (p[j].rt > 0)

{

exec[i] = p[j];

if (p[j].rt < tq)

{

t = t + p[j].rt;

exec[i].bt = p[j].rt;

p[j].rt = 0;

p[j].ft = t;

}

else

{

t = t + tq;

p[j].rt = p[j].rt - tq;

if (p[j].rt == 0)

p[j].ft = t;

exec[i].bt = tq;

}

exec[i].ft = t;

exec[i].rt = p[j].rt;

}

else

i--;

j++;

if (j == n)

j = 0;

}

return t;

}

void print\_table(int n, Process p[n])

{

int i, avgw = 0, avgt = 0;

printf("\n\n\tpid\tAt\tBt\tFt\tTAt\tWt\n");

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

{

p[i].tat = p[i].ft - p[i].at;

p[i].wt = p[i].tat - p[i].bt;

avgw = avgw + p[i].wt;

avgt = avgt + p[i].tat;

if (p[i].pid != -1)

printf("\n\t%d\t%d\t%d\t%d\t %d\t%d\n", p[i].pid, p[i].at, p[i].bt, p[i].ft, p[i].tat, p[i].wt);

}

avgw = (avgw / n);

avgt = (avgt / n);

printf("\n\n\tAverage waiting time: %d units\n\tAverage turn-around time: %d units", avgw, avgt);

}

void main()

{

int n, i, j, t, tq;

printf("\n\t\t\t\tRound Robin\n\n\t Enter the total number of processes: ");

scanf("%d", &n);

Process p[n];

printf("\n\t Enter the time quantum: ");

scanf("%d", &tq);

printf("\n\t Enter the following data for each process: ");

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

{

printf("\n\t Process %d:\n\tArrival time: ", i + 1);

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

printf("\tBurst time: ");

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

p[i].rt = p[i].bt;

p[i].pid = i + 1;

}

sort(n, p);

int u = countIteration(n, p, tq);

// printf("\n Divides into %d iterations.",o);

Process exec[u];

t = execute(n, p, u, exec, t, tq);

print\_table(n, p);

printf("\n\n\t Gnatt chart\n");

print\_chart(u, exec);

// printf("\n t: %d\n",t);

int upbt = 4, lowbt = 1, rand\_at, rand\_bt;

int s = t + 2;

int m;

printf("\n\n\n\t\t\tRandom CPU Burst\n\n\t Enter the total number of processes: ");

scanf("%d", &m);

Process q[m];

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

{

rand\_at = s;

s++;

rand\_bt = (rand() % (upbt - lowbt + 1)) + lowbt;

q[i].at = rand\_at;

q[i].bt = rand\_bt;

q[i].rt = q[i].bt;

q[i].pid = i + 1;

}

sort(m, q);

int v = countIteration(m, q, tq);

// printf("\n Divides into %d iterations.",v);

Process exek[v];

t = execute(m, q, v, exek, t, tq);

print\_table(m, q);

printf("\n\n\t Gnatt chart\n");

print\_chart(v, exek);

printf("\n\n\n\t Combined Gnatt chart\n");

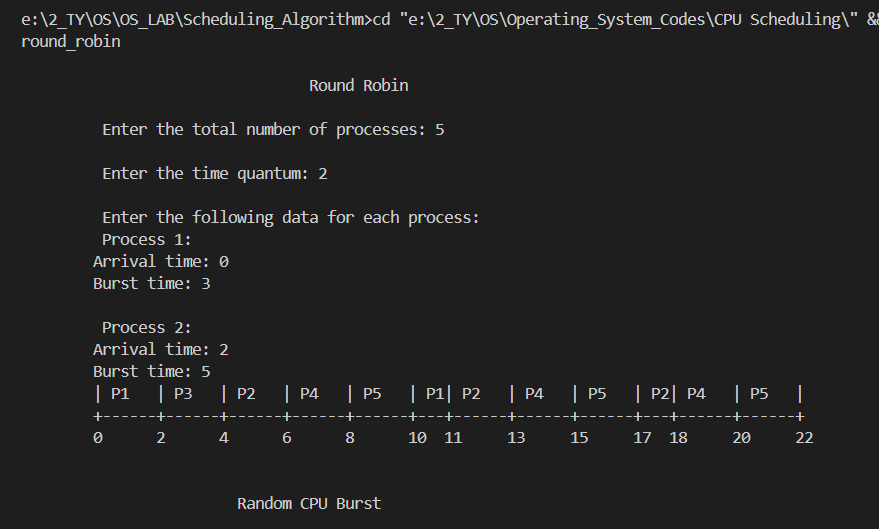
print\_chart(u, exec);

printf("\n\n\t\t--IO of 2 units--\n");

print\_chart(v, exek);

}

Output:



1. **Priority Non preemptive**

**Code:**

#include <stdio.h>

// Function to swap two variables

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int n;

printf("Enter Number of Processes: ");

scanf("%d", &n);

int b[n], p[n], index[n];

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

printf("Enter Burst Time and Priority Value for Process %d: ", i + 1);

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

index[i] = i + 1;

}

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

p[i] = n - p[i] + 1;

}

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

int a = p[i], m = i;

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

if (p[j] > a) {

a = p[j];

m = j;

}

}

swap(&p[i], &p[m]);

swap(&b[i], &b[m]);

swap(&index[i], &index[m]);

}

int t = 0;

printf("Order of Process Execution is\n");

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

printf("P%d is executed from %d to %d\n", index[i], t, t + b[i]);

t += b[i];

}

printf("\n");

printf("PID Burst Wait Turnaround\n");

int wait\_time = 0;

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

printf("P%d %d %d %d\n", index[i], b[i], wait\_time, wait\_time + b[i]);

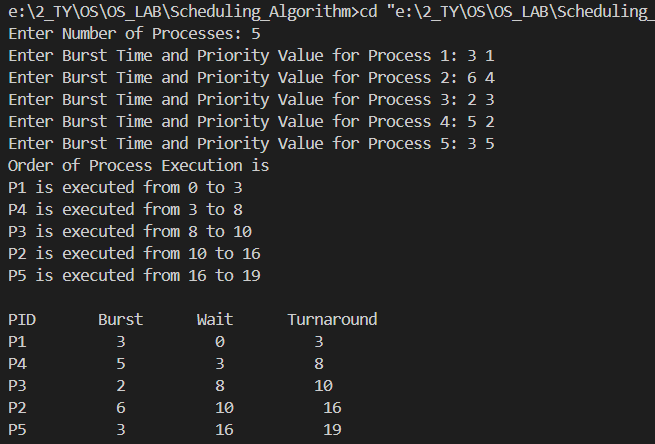
wait\_time += b[i];

}

return 0;

}

Output:



1. **Priority Preemptive**

**Code:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

// vector<int> at={0,1,2,3,4};

// vector<int> bt ={3,4,6,4,2};

// vector<int> priority = {3,2,4,6,10};

// vector<int> ct ={0,0,0,0,0};

// vector<int> tat ={0,0,0,0,0};

// vector<int> wt ={0,0,0,0,0};

// vector<int> temp = bt;

// Time Quantam is 1 by default.

vector<int> at;

vector<int> bt;

vector<int> temp;

vector<int> ct;

vector<int> tat;

vector<int> wt;

vector<int> priority;

vector<int> seq;

vector<int> premptime;

int n;

cout << "Enter the number of processes: ";

cin >> n;

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

{

cout << "Enter the arrival time of process " << i + 1 << ": ";

int arrival\_time;

cin >> arrival\_time;

at.push\_back(arrival\_time);

cout << "Enter the burst time of process " << i + 1 << ": ";

int burst\_time;

cin >> burst\_time;

bt.push\_back(burst\_time);

temp.push\_back(burst\_time);

ct.push\_back(0);

tat.push\_back(0);

wt.push\_back(0);

cout << "Enter the priority of process " << i + 1 << ": ";

int p;

cin >> p;

priority.push\_back(p);

}

int time = 0;

int completed = 0;

int current\_process = -1;

while (completed < n)

{

int min\_priority = INT\_MAX;

int min\_index = -1;

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

{

if (at[i] <= time && priority[i] < min\_priority && bt[i] > 0)

{

min\_priority = priority[i];

min\_index = i;

}

}

// Preempt

if (min\_index != -1 && min\_index != current\_process)

{

if (current\_process != -1 && bt[current\_process] > 0)

{

wt[current\_process]++;

}

seq.push\_back(min\_index);

premptime.push\_back(time);

current\_process = min\_index;

}

// process completed

if (current\_process != -1 && --bt[current\_process] == 0)

{

min\_priority = INT\_MAX;

time++;

ct[current\_process] = time;

tat[current\_process] = ct[current\_process] - at[current\_process];

wt[current\_process] = tat[current\_process] - temp[current\_process];

completed++;

current\_process = -1;

}

else

{

time++;

}

}

cout << "Process\tAT\tBT\tCT\tTAT\tWT\n";

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

{

cout << i + 1 << "\t" << at[i] << "\t" << temp[i] << "\t" << ct[i] << "\t" << tat[i] << "\t" << wt[i] << endl;

}

cout << "Gantt Chart:" << endl;

for(int i=0;i<seq.size();i++)

{

cout<<"| P"<<seq[i]+1<<" ";

}

cout << "|" << endl;

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

{

cout << premptime[i] << "\t";

}

cout << ct[n - 1] << endl;

float avgTAT = 0;

float avgWT = 0;

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

{

avgTAT += tat[i];

avgWT += wt[i];

}

avgTAT /= n;

avgWT /= n;

cout << "Average TAT: " << avgTAT << endl;

cout << "Average WT: " << avgWT << endl;

}

Output:

