**Operating Systems: Lab-5**

CPU Scheduling

1. **First Come First Serve**

**Code:**

#include<stdio.h>

int main()

{

int arr[5], burst[5], wait[5], turn[5];

int bur = 0, cmpl\_T;

float aWait, aTurn, Total = 0;

printf("Enter arrival and burst times: \n");

for(int i=0; i<5; i++) {

printf("p%d ", (i+1));

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

}

for(int i=0; i<5; i++)

{

if(i == 0)

wait[i] = arr[i];

else

wait[i] = bur - arr[i];

bur += burst[i];

Total += wait[i];

}

aWait = Total/5;

cmpl\_T = 0;

Total = 0;

for(int i=0; i<5; i++)

{

cmpl\_T += burst[i];

turn[i] = cmpl\_T - arr[i];

Total += turn[i];

}

aTurn = Total/5;

printf("Process\t Wait time\t Turnaround time\n");

for(int i=0; i<5; i++)

printf("%d\t %d\t\t %d\n", i+1, wait[i], turn[i]);

printf("Average waiting time is: %f\n", aWait);

printf("Average turnaround time is: %f\n", aTurn);

return 0;

}

**Output:**

Text

Description automatically generated

1. **Round Robin**

**Code:**

#include<stdio.h>

struct process {

int wait, turn, burst, arr;

};

struct process p[5];

int queue[100];

int front = -1;

int rear = -1;

void insert(int n) {

if(front == -1)

front = 0;

rear++;

queue[rear] = n;

}

int delete() {

int n;

n = queue[front];

front++;

return n;

}

int main() {

int temp[5], gantt[100], exist[5] = {0};

int t = 0;

float tWait = 0, tTurn = 0;

float aWait, aTurn;

int tBurst = 0;

int q, idx;

printf("Enter quantum: ");

scanf("%d", &q);

printf("Enter arrival and burst times: \n");

for(int i=0; i<5; i++) {

printf("p%d ", (i+1));

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

temp[i] = p[i].burst;

tBurst += p[i].burst;

}

insert(0);

exist[0] = 1;

while(front <= rear) {

idx = delete();

if(p[idx].burst >= q) {

p[idx].burst -= q;

t += q;

}

else {

t += p[idx].burst;

p[idx].burst = 0;

}

for(int i=0; i<5; i++) {

if(exist[i]==0 && p[idx].arr<=t) {

insert(i);

exist[i] = 1;

}

}

if(p[idx].burst == 0) {

p[idx].turn = t - p[idx].arr;

p[idx].wait = t - p[idx].arr - temp[idx];

tTurn += p[idx].turn;

tWait += p[idx].wait;

}

else {

insert(idx);

}

}

aWait = tWait/5;

aTurn = tTurn/5;

printf("Process\t Wait time\t Turnaround time\n");

for(int i=0; i<5; i++)

printf("%d\t %d\t\t %d\n", i+1, p[i].wait, p[i].turn);

printf("Average waiting time is: %f\n", aWait);

printf("Average turnaround time is: %f\n", aTurn);

return 0;

}

**Output:**

**Text

Description automatically generated**

1. **Shortest Job First – without Preemption**

**Code:**

#include<stdio.h>

struct process {

int id, wait, arr, burst, turn;

};

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

int tem;

tem = \*c;

\*c = \*b;

\*b = tem;

}

struct process p[6];

int main() {

int check\_ar = 0;

int Cmp\_time = 0;

float aWait, aTurn;

float tWait = 0, tTurn = 0;

printf("Enter arrival and burst times: \n");

for(int i=0; i<5; i++) {

printf("p%d ", (i+1));

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

p[i].id = i+1;

if(i == 0)

check\_ar = p[i].arr;

if(check\_ar != p[i].arr)

check\_ar = 1;

}

if(check\_ar != 0)

{

for(int i=0; i<5; i++)

{

for(int j=0;j<5-i-1;j++)

{

if(p[j].arr > p[j+1].arr)

{

swap(&p[j].id, &p[j+1].id);

swap(&p[j].arr, &p[j+1].arr);

swap(&p[j].burst, &p[j+1].burst);

}

}

}

}

if(check\_ar != 0)

{

p[0].wait = p[0].arr;

p[0].turn = p[0].burst - p[0].arr;

Cmp\_time = p[0].turn;

tWait += p[0].wait;

tTurn += p[0].turn;

for(int i=1; i<5; i++)

{

int min = p[i].burst;

for(int j=i+1; j<5; j++)

{

if(min>p[j].burst && p[j].arr<=Cmp\_time)

{

min = p[j].burst;

swap(&p[i].id, &p[j].id);

swap(&p[i].arr, &p[j].arr);

swap(&p[i].burst, &p[j].burst);

}

}

p[i].wait = Cmp\_time - p[i].arr;

tWait += p[i].wait;

// completion time of the process

Cmp\_time += p[i].burst;

// Turn Around Time of the process

// compl-Arival

p[i].turn = Cmp\_time - p[i].arr;

tTurn += p[i].turn;

}

}

else

{

for(int i=0; i<5; i++)

{

int min = p[i].burst;

for(int j=i+1; j<5; j++)

{

if(min>p[j].burst && p[j].arr<=Cmp\_time)

{

min = p[j].burst;

swap(&p[i].id, &p[j].id);

swap(&p[i].arr, &p[j].arr);

swap(&p[i].burst, &p[j].burst);

}

}

p[i].wait = Cmp\_time - p[i].arr;

// completion time of the process

Cmp\_time += p[i].burst;

// Turn Around Time of the process

// compl-Arrival

p[i].turn = Cmp\_time - p[i].arr;

tWait += p[i].wait;

tTurn += p[i].turn;

}

}

aWait = tWait/5;

aTurn = tTurn/5;

printf("Process\t Wait time\t Turnaround time\n");

for(int i=0; i<5; i++)

printf("%d\t %d\t\t %d\n", i+1, p[i].wait, p[i].turn);

printf("Average waiting time is: %f\n", aWait);

printf("Average turnaround time is: %f\n", aTurn);

return 0;

}

**Output:**

**Text

Description automatically generated**

1. **Shortest Job First – with Preemption**

**Code:**

#include<stdio.h>

struct process {

int wait, arr, burst, turn;

};

struct process p[6];

int main() {

int temp[5], gantt[100];

int count = 0;

int t = 0;

int sBurst;

float tWait = 0, tTurn = 0;

float aWait, aTurn;

int tBurst = 0;

printf("Enter arrival and burst times: \n");

for(int i=0; i<5; i++) {

printf("p%d ", (i+1));

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

temp[i] = p[i].burst;

tBurst += p[i].burst;

}

p[5].burst = 10000;

for(t=0; count!=5; t++) {

sBurst = 5;

for(int i=0; i<5; i++) {

if(p[i].burst<p[sBurst].burst && (p[i].arr<=t && p[i].burst>0))

sBurst = i;

}

p[sBurst].burst--;

gantt[t] = sBurst;

if(p[sBurst].burst == 0) {

count++;

p[sBurst].wait = t + 1 - p[sBurst].arr - temp[sBurst];

p[sBurst].turn = t + 1 - p[sBurst].arr;

tWait += p[sBurst].wait;

tTurn += p[sBurst].turn;

}

}

aWait = tWait/5;

aTurn = tTurn/5;

printf("Process\t Wait time\t Turnaround time\n");

for(int i=0; i<5; i++)

printf("%d\t %d\t\t %d\n", i+1, p[i].wait, p[i].turn);

printf("Average waiting time is: %f\n", aWait);

printf("Average turnaround time is: %f\n", aTurn);

printf("Gantt chart:\n");

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

printf("p%d ", gantt[i]+1);

printf("\n");

return 0;

}

**Output:**

**Text

Description automatically generated**

1. **Priority Based – without Preemption**

**Code:**

#include<stdio.h>

struct process

{

int id, wait, burst, arr, turn, P;

};

struct process p[6];

void swap(int \*b,int \*c)

{

int tem;

tem=\*c;

\*c=\*b;

\*b=tem;

}

int main()

{

int check\_ar = 0;

int Cmp\_time = 0;

float tWait = 0, tTurn = 0;

float aWait, aTurn;

printf("Enter arrival time, burst time and priority: \n");

for(int i=0; i<5; i++) {

printf("p%d ", (i+1));

scanf("%d %d %d", &p[i].arr, &p[i].burst, &p[i].P);

p[i].id = i+1;

if(i==0)

check\_ar = p[i].arr;

if(check\_ar != p[i].arr)

check\_ar = 1;

}

if(check\_ar != 0)

{

for(int i=0; i<5; i++)

{

for(int j=0; j<5-i-1; j++)

{

if(p[j].arr > p[j+1].arr)

{

swap(&p[j].id, &p[j+1].id);

swap(&p[j].arr, &p[j+1].arr);

swap(&p[j].burst, &p[j+1].burst);

swap(&p[j].P, &p[j+1].P);

}

}

}

}

if(check\_ar != 0)

{

p[0].wait = p[0].arr;

p[0].turn = p[0].burst - p[0].arr;

Cmp\_time = p[0].turn;

tWait += p[0].wait;

tTurn += p[0].turn;

for(int i=1; i<5; i++)

{

int min = p[i].P;

for(int j=i+1; j<5; j++)

{

if(min>p[j].P && p[j].arr<=Cmp\_time)

{

min = p[j].P;

swap(&p[i].id, &p[j].id);

swap(&p[i].arr, &p[j].arr);

swap(&p[i].burst, &p[j].burst);

swap(&p[i].P, &p[j].P);

}

}

p[i].wait = Cmp\_time - p[i].arr;

tWait += p[i].wait;

Cmp\_time += p[i].burst;

p[i].turn = Cmp\_time - p[i].arr;

tTurn += p[i].turn;

}

}

else

{

for(int i=0; i<5; i++)

{

int min = p[i].P;

for(int j=i+1; j<5; j++)

{

if(min>p[j].P && p[j].arr<=Cmp\_time)

{

min = p[j].P;

swap(&p[i].id, &p[j].id);

swap(&p[i].arr, &p[j].arr);

swap(&p[i].burst, &p[j].burst);

swap(&p[i].P, &p[j].P);

}

}

p[i].wait = Cmp\_time - p[i].arr;

Cmp\_time+= p[i].burst;

p[i].turn = Cmp\_time - p[i].arr;

tWait += p[i].wait;

tTurn += p[i].turn;

}

}

aWait = tWait/5;

aTurn = tTurn/5;

printf("Process\t Wait time\t Turnaround time\n");

for(int i=0; i<5; i++)

printf("%d\t %d\t\t %d\n", i+1, p[i].wait, p[i].turn);

printf("Average waiting time is: %f\n", aWait);

printf("Average turnaround time is: %f\n", aTurn);

return 0;

}

**Output:**

**Text

Description automatically generated**

1. **Priority Based – with Preemption**

**Code:**

#include<stdio.h>

struct process

{

int wait, turn, arr, burst, P;

};

struct process p[6];

int main()

{

int temp[5], gantt[100], t, count = 0, short\_p;

float tWait = 0, tTurn = 0;

float aWait, aTurn;

int tBurst = 0;

printf("Enter arrival time, burst time and priority: \n");

for(int i=0; i<5; i++) {

printf("p%d ", (i+1));

scanf("%d %d %d", &p[i].arr, &p[i].burst, &p[i].P);

temp[i] = p[i].burst;

tBurst += p[i].burst;

}

p[5].P = 10000;

for(t=0; count!=5; t++)

{

short\_p = 5;

for(int i=0; i<5; i++)

{

if(p[short\_p].P>p[i].P && p[i].arr<=t && p[i].burst>0)

{

short\_p = i;

}

}

p[short\_p].burst--;

gantt[t] = short\_p;

if(p[short\_p].burst == 0)

{

count++;

p[short\_p].wait = t + 1 - p[short\_p].arr - temp[short\_p];

p[short\_p].turn = t + 1 - p[short\_p].arr;

tWait += p[short\_p].wait;

tTurn += p[short\_p].turn;

}

}

aWait = tWait/5;

aTurn = tTurn/5;

printf("Process\t Wait time\t Turnaround time\n");

for(int i=0; i<5; i++)

printf("%d\t %d\t\t %d\n", i+1, p[i].wait, p[i].turn);

printf("Average waiting time is: %f\n", aWait);

printf("Average turnaround time is: %f\n", aTurn);

printf("Gantt chart:\n");

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

printf("p%d ", gantt[i]+1);

printf("\n");

return 0;

}

**Output:**

**Text

Description automatically generated**