**Copying content of file B .GNANAPRADEEP**

#include<stdio.h> 192210250

#include<stdlib.h>

int main()

{

FILE \*f1,\*f2;

char filename[100],c;

f1=fopen("D:\DEVC++\test2.c","r");

f2=fopen("D:\DEVC++\test1.txt","w");

c=fgetc(f1);

while(c!=EOF)

{

fputc(c,f2);

c=fgetc(f1);

}

printf("CONTENTS COPIED SUCCESSFULLY....");

fclose(f1);

fclose(f2);

}

**Bankers algorithm**

#include <stdio.h>

int main()

{

// P0, P1, P2, P3, P4 are the Process names here

int n, m, i, j, k;

n = 5; // Number of processes

m = 3; // Number of resources

int alloc[5][3] = { { 0, 1, 0 },

// P0 // Allocation Matrix { 2, 0, 0 }, // P1

533]={ 3, 0, 2 }, // P2

{ 2, 1, 1 }, // P3

{ 0, 0, 2 } }; // P4

int max[5][3] = { { 7, 5, 3 }, // P0 // MAX Matrix

{ 3, 2, 2 }, // P1

{ 9, 0, 2 }, // P2

{ 2, 2, 2 }, // P3

{ 4, 3, 3 } }; // P4

int avail[3] = { 3, 3, 2 }; // Available Resources

int f[n], ans[n], ind = 0;

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

f[k] = 0;

}

int need[n][m];

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

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

need[i][j] = max[i][j] - alloc[i][j];

}

int y = 0;

for (k = 0; k < 5; k++) {

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

if (f[i] == 0) {

int flag = 0;

for (j = 0; j < m; j++) {

if (need[i][j] > avail[j]){

flag = 1;

break;

}

}

if (flag == 0) {

ans[ind++] = i;

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

avail[y] += alloc[i][y];

f[i] = 1;

}

}

}

}

int flag = 1;

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

{

if(f[i]==0)

{

flag=0;

printf("The following system is not safe");

break;

}

}

if(flag==1)

{

printf("Following is the SAFE Sequence\n");

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

printf(" P%d ->", ans[i]);

printf(" P%d", ans[n - 1]);

}

return (0);

**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 calculating waiting time of each process

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

{

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

}

Fcfs

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]);

//calculating and printing turnaround time of each process

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

printf("\n");

//for calculating total waiting time

twt += wt[i];

//for calculating total turnaround time

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

}

float att,awt;

//for calculating average waiting time

awt = twt/n;

//for calculating average turnaround time

att = tat/n;

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

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

}

#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;

}}

**Non premptive sjf**

} 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 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);

printf("\n\nGANTT Chart\n");

printf("-");

for(i=0; i<(p[n-1].ttime + 2\*n); i++)

printf("-");

printf("\n|");

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

{

k = p[i].btime/2;

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

printf(" ");

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

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

printf(" ");

printf("|");

}

printf("\n-");

for(i=0; i<(p[n-1].ttime + 2\*n); i++)

printf("-");

printf("\n0");

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

{ for(j=0; j<p[i].btime; j++)

printf(" ");

printf("%2d",p[i].ttime);

}

}

**Premptive priority**

#include<stdio.h>

struct priority\_scheduling {

char process\_name;

int burst\_time;

int waiting\_time;

int turn\_around\_time;

int priority;

};

int main() {

int number\_of\_process;

int total = 0;

struct priority\_scheduling temp\_process;

int ASCII\_number = 65;

int position;

float average\_waiting\_time;

float average\_turnaround\_time;

printf("Enter the total number of Processes: ");

scanf("%d", & number\_of\_process);

struct priority\_scheduling process[number\_of\_process];

printf("\nPlease Enter the Burst Time and Priority of each process:\n");

for (int i = 0; i < number\_of\_process; i++) {

process[i].process\_name = (char) ASCII\_number;

printf("\nEnter the details of the process %c \n", process[i].process\_name);

printf("Enter the burst time: ");

scanf("%d", & process[i].burst\_time);

printf("Enter the priority: ");

scanf("%d", & process[i].priority);

ASCII\_number++;

}

for (int i = 0; i < number\_of\_process; i++) {

position = i;

for (int j = i + 1; j < number\_of\_process; j++) {

if (process[j].priority > process[position].priority)

position = j;

}

temp\_process = process[i];

process[i] = process[position];

process[position] = temp\_process;

}

process[0].waiting\_time = 0;

for (int i = 1; i < number\_of\_process; i++) {

process[i].waiting\_time = 0;

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

process[i].waiting\_time += process[j].burst\_time;

}

total += process[i].waiting\_time;

}

average\_waiting\_time = (float) total / (float) number\_of\_process;

total = 0;

printf("\n\nProcess\_name \t Burst Time \t Waiting Time \t Turnaround Time\n");

printf("------------------------------------------------------------\n");

for (int i = 0; i < number\_of\_process; i++) {

process[i].turn\_around\_time = process[i].burst\_time + process[i].waiting\_time;

total += process[i].turn\_around\_time;

printf("\t %c \t\t %d \t\t %d \t\t %d", process[i].process\_name, process[i].burst\_time, process[i].waiting\_time, process[i].turn\_around\_time);

printf("\n-----------------------------------------------------------\n");

}

average\_turnaround\_time = (float) total / (float) number\_of\_process;

printf("\n\n Average Waiting Time : %f", average\_waiting\_time);

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

return 0;

}.

**SJF scheduling**

#include <stdio.h>

int main()

{

// Matrix for storing Process Id, Burst

// Time, Average Waiting Time & Average

// Turn Around Time.

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");

// User Input Burst Time and alloting Process Id.

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

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

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

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

}

// Sorting process according to their Burst Time.

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;

// Calculation of Waiting Times

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");

// Calculation of Turn Around Time and printing the

// data.

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);

}

**cpu scheduling(priority)**

#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;

}