16. To implement the FCFS Algorithm

#include<stdio.h>

#include<conio.h>

int a[10][6],cawt=0,ctat=0,n;

main()

{

int i,c=0;

clrscr();

printf("enter the no of processors:");

scanf("%d",&n);

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

{

printf("enter the process");

scanf("%d",&a[i][0]);

printf("enter the burst time");

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

printf("enter arrival time:");

scanf("%d",&a[i][2]);

//a[i][3]=a[i][4]=0;

a[i][5]=c;

c+=a[i][1];

}

awt();

tat();

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

{

printf("\n%d\t%d\t%d\t%d\t%d\t%d",a[i][0],a[i][1],a[i][2],a[i][3],a[i][4],a[i][5]);

}

printf("average waiting time %f\n",(float)cawt);

printf("\naverage turnaround time %f\n",(float)ctat);

getch();

}

awt()

{

int i;

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

{ if(i==0)

a[i][4]=0;

a[i][3]=a[i][5]-a[i][2];

cawt+=a[i][3];

}

}

tat()

{

int i;

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

{

a[i][4]=a[i][3]+a[i][1];

ctat+=a[i][4];

}

}

17. To implement the Shortest Job First Algorithm

#include<stdio.h>

#include<conio.h>

int a[10][6],n;

float cawt=0,atat=0;

void sort(int k)

{

int i,j,t1,t2,t3;

clrscr();

t1=a[0][0];a[0][0]=a[k][0];a[k][0]=t1;

t2=a[0][1];a[0][1]=a[k][1];a[k][1]=t2;

t3=a[0][2];a[0][2]=a[k][2];a[k][2]=t3;

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

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

{

if(a[i][1]>a[j][1])

{

t1=a[i][0]; a[i][0]=a[j][0];a[j][0]=t1;

t2=a[i][1];a[i][1]=a[j][1];a[j][1]=t2;

t3=a[i][2];a[i][2]=a[j][2];a[j][2]=t3;

}

}

}

void awt()

{

int i;

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

{

if(i==0)

a[i][3]=0;

else

a[i][3]=a[i][4]-a[i][1];

}

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

cawt=cawt+a[i][3];

}

void tat()

{

int i;

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

{

a[i][4]=a[i][5]-a[i][2];

}

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

atat=atat+a[i][4];

}

void main()

{

int i,c=0,k;

clrscr();

printf("enter no. of processes:");

scanf("%d",&n);

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

{

printf("enter process:");

scanf("%d",&a[i][0]);

printf("enter burst time:");

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

printf("enter arrival time:");

scanf("%d",&a[i][2]);

}

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

{

if(a[i][2]==0)

{

k=i;

break;

}

}

sort(k);

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

{

if (i==0)

{

a[i][5]=c=a[i][1]+a[i][2];

}

else

{

c=c+a[i][1];

a[i][5]=c;

}

}

tat();

awt();

printf("PID\tBT\tAT\tWT\tTAT\tCBT\n");

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

{

printf("%d\t%d\t%d\t%d\t%d\t%d",a[i][0],a[i][1],a[i][2],a[i][3],a[i][4],a[i][5]);

printf("\n");

}

printf("average waiting time:%f",cawt/n);

printf("\naverage turn around time:%f",atat/n);

getch();

}

18. To implement Priority Algorithm

#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;

clrscr();

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=(float)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=(float)total/n;

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

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

getch();

}

19. To implement the round robin Algorithm

#include<stdio.h>

#include<conio.h>

int a[10][8],awt=0,twt=0,tat=0,n,time\_slace,ttat=0;

void wait\_time()

{

int i;

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

{

a[i][4]=a[i][3]-a[i][1];

twt+=a[i][4];

}

}

void turn\_around\_time()

{

int flag=n,f=0,i;

while(flag>0)

{

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

{

f=0;

if((a[i][5]>0) && (a[i][5]>time\_slace))

{

f++;

a[i][5]=a[i][5]-time\_slace;

tat=tat+time\_slace;

a[i][3]=tat;

printf("process %d is sliced %d seconds %d bbb\n",a[i][0],time\_slace,tat);

}

else if(a[i][5]>0 && a[i][5]<=time\_slace)

{

f++;

tat=tat+a[i][5];

a[i][3]=tat;

printf("process %d is sliced %d seconds %d \n",a[i][0],a[i][5],tat);

a[i][5]=0;

flag--;

}

else if(f==0)

continue;

}

}

}

void sort()

{

int i,j,temp;

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

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

if(a[j][2]>a[j+1][2])

{

temp=a[j][0];a[j][0]=a[j+1][0];a[j+1][0]=temp;

temp=a[j][1];a[j][1]=a[j+1][1];a[j+1][1]=temp;

temp=a[j][2];a[j][2]=a[j+1][2];a[j+1][2]=temp;

temp=a[j][3];a[j][3]=a[j+1][3];a[j+1][3]=temp;

temp=a[j][4];a[j][4]=a[j+1][4];a[j+1][4]=temp;

temp=a[j][5];a[j][5]=a[j+1][5];a[j+1][5]=temp;

temp=a[j][6];a[j][6]=a[j+1][6];a[j+1][6]=temp;

}

}

void total\_tat()

{

int i=0;

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

{

ttat+=a[i][3];

}

}

void main()

{

int i;

clrscr();

printf("enter number of processes:");

scanf("%d",&n);

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

{

printf("enter process id %d:",i+1);

scanf("%d",&a[i][0]);

printf("enter process %d burst time:",i+1);

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

printf("enter process %d arrival time:",i+1);

scanf("%d",&a[i][2]);

a[i][4]=a[i][3]=0;

a[i][5]=a[i][1];

}

printf("enter time slice :");

scanf("%d",&time\_slace);

printf("you entered process details:");

printf("id\tbt\tat\tpr\n\n");

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

printf("%d\t%d\t%d\t%d\t%d\t%d\n",a[i][0],a[i][1],a[i][2],a[i][3],a[i] [4],a[i][5]);

sort();

turn\_around\_time();

wait\_time();

total\_tat();

printf("after round robin algorithm\n");

printf("ID\tBT\tAT\tTRT\tWT\tCBT\n");

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

printf("%d\t%d\t%d\t%d\t%d\t%d\n",a[i][0],a[i][1],a[i][2],a[i][3],a[i][4],a[i][5]);

printf("average waiting time=%f\n",(float)twt/n);

printf("average turn around time=%f\n",(float)ttat/n);

getch();

}

20. To implement the FIFO page replacement Algorithm

#include<stdio.h>

#include<conio.h>

void main()

{

int sec[20],pri[10],smp,pmf,i,j,hits=0;

int misses=0,flag,rear=0;

clrscr();

printf("enter the no.of pages:");

scanf("%d",&smp);

printf("enter the no.of frames:");

scanf("%d",&pmf);

printf("the pages are;");

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

{

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

}

printf("the frames are:");

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

{

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

}

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

{

flag=0;

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

{

if(sec[i]==pri[j])

{

flag=1;

hits++;

break;

}

}

if(flag==0)

{

pri[rear]=sec[i];

rear=(rear+1)%pmf;

misses++;

}

}

printf("the hits are %d\n",hits);

printf("the misses are %d",misses);

getch();

}

21. To implement LRU page replacement Algorithm

#include<stdio.h>

#include<conio.h>

void main()

{

int sec[20],pri[20],smp,pmf,i,j,recent[20];

int hits=0,misses=0,pos,count=0;

clrscr();

printf("enter no of pages:");

scanf("%d",&smp);

printf("enter no of frames are:");

scanf("%d",&pmf);

printf("pages are:");

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

{

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

}

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

{

pri[i]=-1;

recent[i]=0;

}

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

{

int found=0;

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

{

if(sec[i]==pri[j])

{

found=1;

hits++;

recent[j]=++count;

break;

}

}

if(!found)

{

pos=0;

for(j=1;j<pmf;j++)

{

if(recent[j]<recent[pos])

{

pos=j;

}

}

pri[pos]=sec[i];

recent[pos]=++count;

misses++;

}

}

printf("the hits are %d\n",hits);

printf("the misses are %d\n",misses);

getch();

}

24. To implement Sequential File Organization

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Employee {

int id;

char name[50];

float salary;

};

void add();

void display();

void search();

int main() {

int choice;

printf("1. Add\n");

printf("2. Display \n");

printf("3. Search by ID\n");

printf("4. Exit\n");

while(1)

{

printf("Enter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1: add(); break;

case 2: display(); break;

case 3: search();

break;

case 4: exit(0);

default:printf("invalid choice");

}

}

return 0;

}

void add() {

struct Employee emp;

FILE \*file = fopen("employee.dat", "a");

if (!file) {

printf("Unable to open file");

return;

}

printf("Enter ID: ");

scanf("%d", &emp.id);

printf("Enter Name: ");

scanf(" %s", emp.name);

printf("Enter Salary: ");

scanf("%f", &emp.salary);

fwrite(&emp, sizeof(emp), 1, file);

fclose(file);

printf("Record added successfully.\n");

}

void display()

{

struct Employee emp;

FILE \*file = fopen("employee.dat", "r");

if (!file)

{

printf("Unable to open file");

return;

}

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

while (fread(&emp, sizeof(emp), 1, file)) {

printf("ID: %d, Name: %s, Salary:%.2f\n", emp.id, emp.name, emp.salary);

}

fclose(file);

}

void search() {

struct Employee emp;

int found = 0;

int searchid;

FILE \*file = fopen("employee.dat", "r");

if (!file) {

printf("Unable to open file");

return;

}

printf("enter search id:");

scanf("%d",&searchid);

while (fread(&emp, sizeof(emp), 1, file)) {

if (emp.id == searchid) {

printf("Record Found - ID: %d, Name: %s, salary: %.2f\n", emp.id, emp.name, emp.salary);

found = 1;

break;

}

}

if (!found) {

printf("Record with ID %d not found.\n", searchid);

}

fclose(file);

}

25. To implement Random File Organization.

#include <stdio.h>

#include<conio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 5

struct Employee {

int id;

char name[50];

float salary;

int isFilled;

};

int hash(int id) {

return id % SIZE;

}

void initializeFile() {

int i;

FILE \*file = fopen("emp.dat", "wb");

struct Employee empty = {0, "", 0.0, 0};

if(!file)

{

printf("file not found");

return;

}

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

fwrite(&empty, sizeof(struct Employee), 1, file);

}

fclose(file);

}

void add() {

struct Employee e;

int pos;

FILE \*file = fopen("emp.dat", "rb+");

struct Employee temp;

printf("Enter ID: ");

scanf("%d", &e.id);

printf("Enter Name: ");

scanf("%s", e.name);

printf("Enter Salary: ");

scanf("%f", &e.salary);

e.isFilled = 1;

pos = hash(e.id);

if(!file)

{

printf("file not found");

return;

}

fseek(file, pos \* sizeof(struct Employee),SEEK\_SET);

fread(&temp, sizeof(struct Employee), 1, file);

if (temp.isFilled) {

printf("Slot already filled. Cannot add employee with ID %d.\n", e.id);

} else

{

fseek(file, pos \* sizeof(struct Employee),SEEK\_SET);

fwrite(&e, sizeof(struct Employee), 1, file);

printf("Record added at position %d.\n", pos);

}

fclose(file);

}

void display() {

FILE \*file = fopen("emp.dat", "rb");

struct Employee e;

int i;

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

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

fseek(file, i \* sizeof(struct Employee),SEEK\_SET);

fread(&e, sizeof(struct Employee), 1, file);

if (e.isFilled) {

printf("Position %d: ID=%d, Name=%s, Salary=%.2f\n", i, e.id, e.name, e.salary);

}

}

fclose(file);

}

void search() {

int id;

int pos;

FILE \*file = fopen("emp.dat", "rb");

struct Employee e;

printf("Enter ID to search: ");

scanf("%d", &id);

pos = hash(id);

if(!file)

{

printf("file not found");

return ;

}

fseek(file, pos \* sizeof(struct Employee),SEEK\_SET);

fread(&e, sizeof(struct Employee), 1, file);

if (e.isFilled && e.id == id) {

printf("Found: ID=%d, Name=%s, Salary=%.2f\n", e.id, e.name, e.salary);

} else {

printf("Record not found at position %d.\n", pos);

}

fclose(file);

}

int main() {

int choice;

clrscr();

printf("Initializing file (only once)...\n");

initializeFile();

printf("\n1. Add\n2. Display\n3. Search\n4. Exit\n");

while(1)

{

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: add(); break;

case 2: display(); break;

case 3: search(); break;

case 4: exit(0);

default: printf("Invalid choice.\n");

}

}

return 0;

}