# EXPERIMENT NO.1

**CPU SCHEDULINGALGORITHMS**

**A). FIRST COME FIRST SERVE:**SOURCE CODE:

# #include <stdio.h>

# int main()

# {

# int bt[20], wt[20], tat[20], i, n;

# float wtavg, tatavg;

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

# scanf("%d", &n);

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

# {

# printf("\nEnter Burst Time for Process %d -- ", i);

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

# }

# wt[0] = wtavg = 0;

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

# 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("\t PROCESS \tBURST TIME \t WAITING TIME\t COMPLETION TIME\n");

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

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

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

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

# printf("\n\n codes executed by …….");

# return 0;

# }

B). SHORTEST JOB FIRST:

# SOURCE CODE :

#include <stdio.h>

int main()

{

int p[20], bt[20], wt[20], tat[20], i, k, n, temp;

float wtavg, tatavg;

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

scanf("%d", &n);

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

{

p[i] = i;

printf("Enter Burst Time for Process %d -- ", i);

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

}

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

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

if (bt[i] > bt[k])

{

temp = bt[i];

bt[i] = bt[k];

bt[k] = temp;

temp = p[i];

p[i] = p[k];

p[k] = temp;

}

wt[0] = wtavg = 0;

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

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 \tBURST TIME \t WAITING TIME\t COMPLETION TIME\n");

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

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

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

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

printf("\n\n codes executed by ..........");

getch();

}

# C). ROUND ROBIN:

# SOURCE CODE

#include <stdio.h>

int main()

{

int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;

float awt = 0, att = 0, temp = 0;

printf("Enter the no of processes -- ");

scanf("%d", &n);

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

{

printf("\nEnter Burst Time for process %d -- ", i + 1);

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

ct[i] = bu[i];

}

printf("\nEnter the size of time slice -- ");

scanf("%d", &t);

max = bu[0];

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

if (max < bu[i])

max = bu[i];

for (j = 0; j < (max / t) + 1; j++)

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

if (bu[i] != 0)

if (bu[i] <= t)

{

tat[i] = temp + bu[i];

temp = temp + bu[i];

bu[i] = 0;

}

else

{

bu[i] = bu[i] - t;

temp = temp + t;

}

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

{

wa[i] = tat[i] -

ct[i];

att += tat[i];

awt += wa[i];

}

printf("\nThe Average Turnaround time is -- %f", att / n);

printf("\nThe Average Waiting time is -- %f ", awt / n);

printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");

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

printf("\t%d \t %d \t\t %d \t\t %d \n", i + 1, ct[i], wa[i], tat[i]);

printf("\n\n codes executed by ..........");

getch();

}

# D). PRIORITY:

# SOURCE CODE:

#include <stdio.h>

int main()

{

int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;

float wtavg, tatavg;

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

scanf("%d", &n);

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

{

p[i] = i;

printf("Enter the Burst Time & Priority of Process %d --- ", i); scanf("%d %d",&bt[i], &pri[i]);

}

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

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

if (pri[i] > pri[k])

{

temp = p[i];

p[i] = p[k];

p[k] = temp;

temp = bt[i];

bt[i] = bt[k];

bt[k] = temp;

temp = pri[i];

pri[i] = pri[k];

pri[k] = temp;

}

wtavg = wt[0] = 0;

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

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("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND TIME");

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

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

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

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

printf("\n\n codes executed by ..........");

getch();

}

# EXPERIMENT.NO 2

Write a C program to simulate producer-consumer problem using semaphores.

# PROGRAM

#include<stdio.> void main()

{

int buffer[10], bufsize, in, out, produce, consume, choice=0; in = 0;

out = 0;

bufsize = 10;

while(choice !=3)

{

printf(“\n1. Produce \t 2. Consume \t3. Exit”); printf(“\nEnter your choice: ”);

scanf(“%d”,&choice);

switch(choice) {

case 1: if((in+1)%bufsize==out)

printf(“\nBuffer is Full”);

else

{

}

break;;;

printf(“\nEnter the value: “); scanf(“%d”, &produce); buffer[in] = produce;

in = (in+1)%bufsize;

case 2: if(in == out)

printf(“\nBuffer is Empty”);

} } }

else

{

consume = buffer[out];

printf(“\nThe consumed value is %d”, consume);

out = (out+1)%bufsize;

}

break;

# EXPERIMENT.NO 3

Write a C program to simulate the concept of Dining-Philosophers problem.

# PROGRAM

int tph, philname[20], status[20], howhung, hu[20], cho; main()

{

int i; clrscr();

printf("\n\nDINING PHILOSOPHER PROBLEM");

printf("\nEnter the total no. of philosophers: "); scanf("%d",&tph);

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

{

philname[i]=(i+1); status[i]=1;

}

printf("How many are hungry : "); scanf("%d", &howhung); if(howhung==tph)

{

printf(“\n All are hungry..\nDead lock stage will occur”);

printf(\n”Exiting\n”); else{ for(i=0;i<howhung;i++){

printf(“Enterphilosopher%dposition:”,(i+1)); scanf(“%d”,&hu[i]);

status[hu[i]]=2;

}

do

{

printf("1.One can eat at a time\t2.Two can eat at a time

\t3.Exit\nEnter your choice:"); scanf("%d", &cho); switch(cho)

{

case 1: one();

case 2: two();

break; case 3: exit(0);

break;

}

one()

{

}

}while(1);

default: printf("\nInvalid option..");

int pos=0, x, i;

printf("\nAllow one philosopher to eat at any time\n"); for(i=0;i<howhung; i++, pos++)

{

printf("\nP %d is granted to eat", philname[hu[pos]]); for(x=pos;x<howhung;x++)

printf("\nP %d is waiting", philname[hu[x]]);

}

two()

{

}

int i, j, s=0, t, r, x;

printf("\n Allow two philosophers to eat at same time\n"); for(i=0;i<howhung;i++)

{

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

{

if(abs(hu[i]-hu[j])>=1&& abs(hu[i]-hu[j])!=4)

{

printf("\n\ncombination %d \n", (s+1)); t=hu[i];

r=hu[j]; s++;

printf("\nP %d and P %d are granted to eat", philname[hu[i]], philname[hu[j]]);

for(x=0;x<howhung;x++)

{

if((hu[x]!=t)&&(hu[x]!=r))

printf("\nP %d is waiting", philname[hu[x]]);

}

}

}

}

}

# EXPERIMENT.NO 4 MEMORY MANAGEMENT

**A). MEMORY MANAGEMENT WITH FIXED PARTITIONING TECHNIQUE (MFT)**

# SOURCE CODE :

#include<stdio.h>

#include<conio.h> main()

{

int ms, bs, nob, ef,n, mp[10],tif=0; int i,p=0;

clrscr();

printf("Enter the total memory available (in Bytes) -- "); scanf("%d",&ms);

printf("Enter the block size (in Bytes) -- "); scanf("%d", &bs);

nob=ms/bs; ef=ms - nob\*bs;

printf("\nEnter the number of processes -- "); scanf("%d",&n);

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

{

printf("Enter memory required for process %d (in Bytes)-- ",i+1); scanf("%d",&mp[i]);

}

printf("\nNo. of Blocks available in memory--%d",nob); printf("\n\nPROCESS\tMEMORYREQUIRED\tALLOCATED\tINTERNAL FRAGMENTATION");

for(i=0;i<n && p<nob;i++)

{

printf("\n %d\t\t%d",i+1,mp[i]); if(mp[i] > bs)

printf("\t\tNO\t\t---"); else

{

printf("\t\tYES\t%d",bs-mp[i]); tif = tif + bs-mp[i];

p++;

}

}

if(i<n)

printf("\nMemory is Full, Remaining Processes cannot be accomodated"); printf("\n\nTotal Internal Fragmentation is %d",tif);

printf("\nTotal External Fragmentation is %d",ef); getch();

}

# MEMORY VARIABLE PARTIONING TYPE (MVT)

**AIM:** To write a program to simulate the MVT algorithm

# SOURCE CODE:

#include<stdio.h>

#include<conio.h> main()

{

int ms,mp[10],i, temp,n=0; char ch = 'y'; clrscr();

printf("\nEnter the total memory available (in Bytes)-- "); scanf("%d",&ms);

temp=ms; for(i=0;ch=='y';i++,n++)

{

printf("\nEnter memory required for process %d (in Bytes) -- ",i+1); scanf("%d",&mp[i]);

if(mp[i]<=temp)

{

printf("\nMemory is allocated for Process %d ",i+1); temp = temp - mp[i];

}

else

{

printf("\nMemory is Full"); break;

}

printf("\nDo you want to continue(y/n) -- "); scanf(" %c", &ch);

}

printf("\n\nTotal Memory Available -- %d", ms); printf("\n\n\tPROCESS\t\t MEMORY ALLOCATED "); for(i=0;i<n;i++)

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

printf("\n\nTotal Memory Allocated is %d",ms-temp); printf("\nTotal External Fragmentation is %d",temp);

getch();

}

# EXPERIMENT.NO 5

**MEMORY ALLOCATION TECHNIQUES**

Write a C program to simulate the following contiguous memory allocation techniques

* + 1. Worst-fit b) Best-fit c) First-fit

# PROGRAM

## WORST-FIT

#include<stdio.h>

#include<conio.h>

#define max 25 void main()

{

int frag[max],b[max],f[max],i,j,nb,nf,t emp; static int bf[max],ff[max]; clrscr();

printf("\n\tMemory Management Scheme - First Fit"); printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of files:"); scanf("%d",&nf);

printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++)

{

printf("Block %d:",i);

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

}

printf("Enter the size of the files :-\n"); for(i=1;i<=nf;i++)

{

printf("File %d:",i);

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

}

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

{

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

{

if(bf[j]!=1)

{

temp=b[j]-f[i]; if(temp>=0)

{

ff[i]=j; break;

}

}

}

frag[i]=temp; bf[ff[i]]=1;

}

## BEST-FIT

#include<stdio.h>

#include<conio.h>

#define max 25 void main()

{

int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000; static int bf[max],ff[max];

clrscr();

printf("\nEnter the number of blocks:"); scanf("%d",&nb);

printf("Enter the number of files:"); scanf("%d",&nf);

printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++)

printf("Block %d:",i);

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

printf("Enter the size of the files :-\n"); for(i=1;i<=nf;i++)

{

printf("File %d:",i);

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

}

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

{

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

{

if(bf[j]!=1)

{

temp=b[j]-f[i]; if(temp>=0)

if(lowest>temp)

{

ff[i]=j; lowest=temp;

}

}}

frag[i]=lowest; bf[ff[i]]=1; lowest=10000;

}

printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment"); for(i=1;i<=nf && ff[i]!=0;i++)

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]); getch();

}

## FIRST-FIT

#include<stdio.h>

#include<conio.h>

#define max 25 void main()

{

int frag[max],b[max],f[max],i,j,nb,nf,temp,highes t=0; static int bf[max],ff[max];

clrscr();

printf("\n\tMemory Management Scheme - Worst Fit"); printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of files:"); scanf("%d",&nf);

printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++)

{

printf("Block %d:",i);

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

}

printf("Enter the size of the files :-\n"); for(i=1;i<=nf;i++)

{

printf("File %d:",i);

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

}

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

{

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

{

if(bf[j]!=1) //if bf[j] is not allocated

{

temp=b[j]-f[i]; if(temp>=0)

if(highest<temp)

{

}

}

frag[i]=highest; bf[ff[i]]=1; highest=0;

}

ff[i]=j; highest=temp;

}

printf("\nFile\_no:\tFile\_size:\tBlock\_no:\tBlock\_size:\tFragement"); for(i=1;i<=nf;i++)

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]); getch();

}

# EXPERIMENT NO.6

**PAGE REPLACEMENT ALGORITHMS**

Implement FIFO page replacement technique.

# a) FIFO b) LRU c) OPTIMAL

# FIRST IN FIRST OUT SOURCE CODE :

#include<stdio.h>

#include<conio.h> int fr[3]; void main()

{

void display();

int i,j,page[12]={2,3,2,1,5,2,4,5,3,2,5,2};

int flag1=0,flag2=0,pf=0,frsize=3,top=0; clrscr();

for(i=0;i<3;i++)

{

fr[i]=-1;

}

for(j=0;j<12;j++)

{

flag1=0; flag2=0; for(i=0;i<12;i++)

{

if(fr[i]==page[j])

{

flag1=1; flag2=1; break;

}

}

if(flag1==0)

{

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

{

if(fr[i]==-1)

{

fr[i]=page[j]; flag2=1; break;

}

}

}

if(flag2==0)

{

fr[top]=page[j]; top++;

pf++; if(top>=frsize) top=0;

}

display();

}

printf("Number of page faults : %d ",pf+frsize); getch();

}

void display()

{

int i; printf("**\n**"); for(i=0;i<3;i++) printf("%d**\t**",fr[i]);

}

# LEAST RECENTLY USED

**I**mplement LRU page replacement technique.

# SOURCE CODE :

#include<stdio.h>

#include<conio.h> int fr[3];

void main()

{

void display();

int p[12]={2,3,2,1,5,2,4,5,3,2,5,2},i,j,fs[3];

int index,k,l,flag1=0,flag2=0,pf=0,frsize=3; clrscr();

for(i=0;i<3;i++)

{

fr[i]=-1;

}

for(j=0;j<12;j++)

{

flag1=0,flag2=0; for(i=0;i<3;i++)

{

if(fr[i]==p[j])

{

flag1=1; flag2=1; break;

}

}

if(flag1==0)

{

for(i=0;i<3;i++)

{

if(fr[i]==-1)

{

fr[i]=p[j]; flag2=1; break;

}

}

}

if(flag2==0)

{

for(i=0;i<3;i++) fs[i]=0;

for(k=j-1,l=1;l<=frsize-1;l++,k--)

{

for(i=0;i<3;i++)

{

if(fr[i]==p[k]) fs[i]=1;

}}

for(i=0;i<3;i++)

{

if(fs[i]==0) index=i;

}

fr[index]=p[j]; pf++;

}

display();

}

printf("**\n** no of page faults :%d",pf+frsize); getch();

}

void display()

{

int i; printf("**\n**"); for(i=0;i<3;i++) printf("**\t**%d",fr[i]);

}

# OPTIMAL

# SOURCE CODE:

/\* Program to simulate optimal page replacement \*/

#include<stdio.h>

#include<conio.h> int fr[3], n, m; void

display(); void main()

{

int i,j,page[20],fs[10]; int

max,found=0,lg[3],index,k,l,flag1=0,flag2=0,pf=0; float pr;

clrscr();

printf("Enter length of the reference string: "); scanf("%d",&n);

printf("Enter the reference string: "); for(i=0;i<n;i++)

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

printf("Enter no of frames: "); scanf("%d",&m);

for(i=0;i<m;i++) fr[i]=-1; pf=m;

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

{

flag1=0; flag2=0; for(i=0;i<m;i++)

{

if(fr[i]==page[j])

{

flag1=1; flag2=1; break;

}

}

if(flag1==0)

{

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

{

if(fr[i]==-1)

{

fr[i]=page[j]; flag2=1; break;

}

}

}

if(flag2==0)

{

for(i=0;i<m;i++) lg[i]=0; for(i=0;i<m;i++)

{

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

{

if(fr[i]==page[k])

{

lg[i]=k-j; break;

}

}

}

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

{

if(lg[i]==0)

{

index=i; found = 1;

break;

}

}

if(found==0)

{

max=lg[0]; index=0; for(i=0;i<m;i++)

{

if(max<lg[i])

{

max=lg[i]; index=i;

}

}

}

fr[index]=page[j]; pf++;

}

display();

}

printf("Number of page faults : %d\n", pf); pr=(float)pf/n\*100;

printf("Page fault rate = %f \n", pr); getch();

}

void display()

{

int i; for(i=0;i<m;i++) printf("%d\t",fr[i]); printf("\n");

}

# EXPERIMENT NO. 7

**FILE ORGANIZATION TECHNIQUES**

1. **SINGLE LEVEL DIRECTORY:**

Program to simulate Single level directory file organization technique.

# SOURCE CODE :

#include<stdio.h> struct

{

char dname[10],fname[10][10]; int fcnt;

}dir;

void main()

{

int i,ch; char f[30]; clrscr(); dir.fcnt = 0;

printf("\nEnter name of directory -- "); scanf("%s", dir.dname);

while(1)

{

printf("\n\n1. Create File\t2. Delete File\t3. Search File \n

4. Display Files\t5. Exit\nEnter your choice -- "); scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter the name of the file -- "); scanf("%s",dir.fname[dir.fcnt]);

dir.fcnt++; break;

case 2: printf("\nEnter the name of the file -- "); scanf("%s",f);

for(i=0;i<dir.fcnt;i++)

{

if(strcmp(f, dir.fname[i])==0)

{

printf("File %s is deleted ",f); strcpy(dir.fname[i],dir.fname[dir.fcnt-1]); break;

}

}

if(i==dir.fcnt)

printf("File %s not found",f);

else

dir.fcnt--; break;

printf("\nEnter the name of the file -- "); scanf("%s",f);

for(i=0;i<dir.fcnt;i++)

{

if(strcmp(f, dir.fname[i])==0)

{

printf("File %s is found ", f); break;

}

}

if(i==dir.fcnt)

printf("File %s not found",f); break;

if(dir.fcnt==0) printf("\nDirectory Empty"); else

{

printf("\nThe Files are -- "); for(i=0;i<dir.fcnt;i++) printf("\t%s",dir.fname[i]);

}

break;

}

getch();}

# TWO LEVEL DIRECTORY

Program to simulate two level file organization technique

# SOURCE CODE :

#include<stdio.h> struct

{

char dname[10],fname[10][10]; int fcnt;

}dir[10];

void main()

{

int i,ch,dcnt,k; char f[30], d[30]; clrscr(); dcnt=0;

while(1)

{

printf("\n\n1. Create Directory\t2. Create File\t3. Delete File"); printf("\n4. Search File\t\t5. Display\t6. Exit\t Enter your choice --"); scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter name of directory -- "); scanf("%s", dir[dcnt].dname); dir[dcnt].fcnt=0;

dcnt++;

printf("Directory created"); break;

case 2: printf("\nEnter name of the directory -- "); scanf("%s",d);

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

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter name of the file -- "); scanf("%s",dir[i].fname[dir[i].fcnt]);

dir[i].fcnt++; printf("File created");

}

if(i==dcnt)

printf("Directory %s not found",d); break;

case 3: printf("\nEnter name of the directory -- "); scanf("%s",d);

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

{

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter name of the file -- "); scanf("%s",f); for(k=0;k<dir[i].fcnt;k++)

{

if(strcmp(f, dir[i].fname[k])==0)

{

printf("File %s is deleted ",f); dir[i].fcnt--;

strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]); goto jmp;

}

}

printf("File %s not found",f); goto jmp;

}

}

printf("Directory %s not found",d); jmp : break;

case 4: printf("\nEnter name of the directory -- "); scanf("%s",d);

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

{

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter the name of the file -- "); scanf("%s",f); for(k=0;k<dir[i].fcnt;k++)

{

if(strcmp(f, dir[i].fname[k])==0)

{

printf("File %s is found ",f); goto jmp1;

}

}

printf("File %s not found",f); goto jmp1;

}

}

Page 44

printf("Directory %s not found",d); jmp1: break; case 5: if(dcnt==0)

printf("\nNo Directory's ");

else

{

printf("\nDirectory\tFiles"); for(i=0;i<dcnt;i++)

{

}

getch();

}

}

}

break;

default:exit(0);

}

printf("\n%s\t\t",dir[i].dname); for(k=0;k<dir[i].fcnt;k++) printf("\t%s",dir[i].fname[k]);

# EXPERIMENT.NO.8

**FILE ALLOCATION STRATEGIES**

1. **SEQUENTIAL:**

write a C program for implementing sequential file allocation method

# SOURCE CODE :

#include<stdio.h> main()

{

int f[50],i,st,j,len,c,k; clrscr(); for(i=0;i<50;i++) f[i]=0;

X:

printf("\n Enter the starting block & length of file"); scanf("%d%d",&st,&len);

for(j=st;j<(st+len);j++) if(f[j]==0)

{

f[j]=1

;

printf("\n%d->%d",j,f[j]);

}

else

{

printf("Block already allocated"); break;

}

if(j==(st+len))

printf("\n the file is allocated to disk");

printf("\n if u want to enter more files?(y-1/n-0)"); scanf("%d",&c);

if(c==1) goto X; else exit();

getch();

}

# INDEXED:

Implement allocation method using chained method

SOURCE CODE :

#include<stdio.h>

int f[50],i,k,j,inde[50],n,c,count=0,p; main()

{

clrscr(); for(i=0;i<50;i++) f[i]=0;

x: printf("enter index block\t"); scanf("%d",&p);

if(f[p]==0)

{ f[p]=1;

printf("enter no of files on index\t"); scanf("%d",&n);

}

else

{

printf("Block already allocated\n"); goto x;

}

for(i=0;i<n;i++) scanf("%d",&inde[i]); for(i=0;i<n;i++) if(f[inde[i]]==1)

{

printf("Block already allocated"); goto x;

}

for(j=0;j<n;j++) f[inde[j]]=1;

printf("\n allocated"); printf("\n file indexed"); for(k=0;k<n;k++)

printf("\n %d->%d:%d",p,inde[k],f[inde[k]]);

printf(" Enter 1 to enter more files and 0 to exit\t"); scanf("%d",&c);

if(c==1) goto x; else exit();

getch();

}

# LINKED:

# SOURCE CODE :

#include<stdio.h> main()

{

int f[50],p,i,j,k,a,st,len,n,c; clrscr();

for(i=0;i<50;i++) f[i]=0;

printf("Enter how many blocks that are already allocated"); scanf("%d",&p);

printf("\nEnter the blocks no.s that are already allocated"); for(i=0;i<p;i++)

{

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

}

X:

printf("Enter the starting index block & length"); scanf("%d%d",&st,&len); k=len; for(j=st;j<(k+st);j++)

{

if(f[j]==0)

{ f[j]=1;

printf("\n%d->%d",j,f[j]);

}

else

{

printf("\n %d->file is already allocated",j);

k++;

}

}

printf("\n If u want to enter one more file? (yes-1/no-0)");

scanf("%d",&c); if(c==1)

goto X;

else exit();

getch( );}

# EXPERIMENT.NO 9

**DEAD LOCK AVOIDANCE**

Simulate bankers algorithm for Dead Lock Avoidance (Banker‘s Algorithm)

# SOURCE CODE :

#include<stdio.h>

#include<conio.h>

#include<string.h> void main()

{

int alloc[10][10],max[10][10]; int avail[10],work[10],total[10]; int i,j,k,n,need[10][10];

int m;

int count=0,c=0;

char finish[10]; clrscr();

printf("Enter the no. of processes and resources:"); scanf("%d%d",&n,&m);

for(i=0;i<=n;i++) finish[i]='n';

printf("Enter the claim matrix:\n"); for(i=0;i<n;i++)

for(j=0;j<m;j++) scanf("%d",&max[i][j]);

printf("Enter the allocation matrix:\n"); for(i=0;i<n;i++)

for(j=0;j<m;j++) scanf("%d",&alloc[i][j]); printf("Resource vector:"); for(i=0;i<m;i++) scanf("%d",&total[i]); for(i=0;i<m;i++)

avail[i]=0; for(i=0;i<n;i++)

for(j=0;j<m;j++) avail[j]+=alloc[i][j]; for(i=0;i<m;i++) work[i]=avail[i]; for(j=0;j<m;j++) work[j]=total[j]-work[j]; for(i=0;i<n;i++) for(j=0;j<m;j++) need[i][j]=max[i][j]-alloc[i][j]; A:

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

{

c=0;

for(j=0;j<m;j++) if((need[i][j]<=work[j])&&(finish[i]=='n')) c++;

if(c==m)

{

printf("All the resources can be allocated to Process %d", i+1); printf("\n\nAvailable resources are:");

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

{

work[k]+=alloc[i][k];

printf("%4d",work[k]);

}

printf("\n");

finish[i]='y';

printf("\nProcess %d executed?:%c \n",i+1,finish[i]); count++;

}

}

if(count!=n) goto A;

else

printf("\n System is in safe mode"); printf("\n The given state is safe state"); getch();

}

# EXPERIMENT.NO 10 DEAD LOCKPREVENTION

Implement deadlock prevention technique

# Banker‘s Algorithm:

# SOURCE CODE :

#include<stdio.h>

#include<conio.h> void main()

{

char job[10][10];

int time[10],avail,tem[10],temp[10]; int safe[10]; int ind=1,i,j,q,n,t;

clrscr();

printf("Enter no of jobs: "); scanf("%d",&n); for(i=0;i<n;i++)

{

printf("Enter name and time: "); scanf("%s%d",&job[i],&time[i]);

}

printf("Enter the available resources:"); scanf("%d",&avail);

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

{

temp[i]=time[i]; tem[i]=i;

}

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

{

if(temp[i]>temp[j])

{

t=temp[i];

temp[i]=temp[j];

temp[j]=t; t=tem[i];

tem[i]=tem[j]; tem[j]=t;

}

}

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

{

q=tem[i]; if(time[q]<=avail)

{

safe[ind]=tem[i]; avail=avail-tem[q]; printf("%s",job[safe[ind]]); ind++;

}

else

{

printf("No safe sequence\n");

}

}

printf("Safe sequence is:"); for(i=1;i<ind; i++)

printf("%s %d\n",job[safe[i]],time[safe[i]]); getch();

}

# 

# EXPERIMENT.NO 11

Write a C program to simulate disk scheduling algorithms

* 1. FCFS b) SCAN c) C-SCAN

# PROGRAM

1. **FCFS DISK SCHEDULING ALGORITHM**

#include<stdio.h> main()

{

int t[20], n, I, j, tohm[20], tot=0; float avhm; clrscr();

printf(“enter the no.of tracks”); scanf(“%d”,&n);

printf(“enter the tracks to be traversed”);

for(i=2;i<n+2;i++)

scanf(“%d”,&t\*i+); for(i=1;i<n+1;i++)

{

tohm[i]=t[i+1]-t[i]; if(tohm[i]<0) tohm[i]=tohm[i]\*(-1);

}

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

tot+=tohm[i]; avhm=(float)tot/n;

printf(“Tracks traversed\tDifference between tracks\n”); for(i=1;i<n+1;i++)

printf(“%d\t\t\t%d\n”,t\*i+,tohm\*i+);

printf("\nAverage header movements:%f",avhm); getch();

}

# SCAN DISK SCHEDULING ALGORITHM

#include<stdio.h> main()

{

int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0; clrscr();

printf("enter the no of tracks to be traveresed"); scanf("%d'",&n);

printf("enter the position of head"); scanf("%d",&h);

t[0]=0;t[1]=h;

printf("enter the tracks"); for(i=2;i<n+2;i++)

scanf("%d",&t[i]); for(i=0;i<n+2;i++)

{

for(j=0;j<(n+2)-i-1;j++)

{

if(t[j]>t[j+1])

{

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

} } }

for(i=0;i<n+2;i++) if(t[i]==h)

j=i;k=i;

p=0;

while(t[j]!=0)

{

atr[p]=t[j]; j--; p++;

}

atr[p]=t[j]; for(p=k+1;p<n+2;p++,k++)

atr[p]=t[k+1]; for(j=0;j<n+1;j++)

{

if(atr[j]>atr[j+1])

d[j]=atr[j]-atr[j+1];

else

d[j]=atr[j+1]-atr[j];

sum+=d[j];

}

printf("\nAverage header movements:%f",(float)sum/n); getch();}

# C-SCAN DISK SCHEDULING ALGORITHM

#include<stdio.h> main()

{

int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0; clrscr();

printf("enter the no of tracks to be traveresed"); scanf("%d'",&n);

printf("enter the position of head"); scanf("%d",&h);

t[0]=0;t[1]=h;

printf("enter total tracks"); scanf("%d",&tot);

t[2]=tot-1;

printf("enter the tracks"); for(i=3;i<=n+2;i++)

scanf("%d",&t[i]); for(i=0;i<=n+2;i++)

for(j=0;j<=(n+2)-i-1;j++)

if(t[j]>t[j+1])

{

for(i=0;i<=n+2;i++) if(t[i]==h);

j=i;break;

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

}

p=0;

while(t[j]!=tot-1)

{

atr[p]=t[j]; j++;

p++;

}

atr[p]=t[j]; p++;

i=0;

while(p!=(n+3) && t[i]!=t[h])

{

atr[p]=t[i]; i++; p++;

}

for(j=0;j<n+2;j++)

{

if(atr[j]>atr[j+1])

d[j]=atr[j]-atr[j+1];

else

d[j]=atr[j+1]-atr[j];

sum+=d[j];

}

printf("total header movements%d",sum); printf("avg is %f",(float)sum/n);

getch();

}