#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<string.h>

#include<math.h>

int noOfFiles,noOfBlocks,block\_size;

typedef struct dir

{

char fname[20];

int start;

int length;

int end;

struct dir \*next;

}dir;

typedef struct node

{

char fname[20];

int blockid;

int fileblocktable[100];

struct node \*next;

struct node \*link;

}node;

typedef struct

{

char fname[20];

int fsize;

int fileblocks;

}fileDetails;

void contiguous\_alloc(node\* mainmem,fileDetails filedata[]);

void insert(node \*head,node data);

void indexed\_alloc(fileDetails filedata[],node \*mainmem);

void linked\_alloc(fileDetails filedata[],node \*mainmem);

void main()

{

int mem\_size,choice,i;

node data;

node \*mainmem;

node \*temp;

fileDetails filedata[100];

char c;

mainmem=malloc(sizeof(node));

mainmem->next=NULL;

printf("Enter the main memory size:");

scanf("%d",&mem\_size);

printf("Enter the block size:");

scanf("%d",&block\_size);

noOfBlocks=(int)mem\_size/block\_size;

printf("Total no. of blocks available:%d\n",noOfBlocks);

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

{

data.blockid=i+1;

strcpy(data.fname,"free");

insert(mainmem,data);

}

printf("Number of files to be allocated:");

scanf("%d",&noOfFiles);

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

{

printf("\nName of file %d:",i+1);

scanf("%s",filedata[i].fname);

printf("Size of file %d(in KB):",i+1);

scanf("%d",&filedata[i].fsize);

filedata[i].fileblocks=ceil((float)filedata[i].fsize/(float)block\_size);

}

do

{

printf("\n\nFILE ALLOCATION TECHNIQUES\n");

printf("1.Contiguous\n");

printf("2.Linked\n");

printf("3.Indexed\n");

printf("Enter choice:");

scanf("%d",&choice);

temp=mainmem->next;

while(temp!=NULL)

{

strcpy(temp->fname,"free");

temp=temp->next;

}

srand((unsigned)time(NULL));

switch(choice)

{

case 1:

contiguous\_alloc(mainmem,filedata);

break;

case 2:

linked\_alloc(filedata,mainmem);

break;

case 3:

indexed\_alloc(filedata,mainmem);

break;

}

printf("\nDo you want to continue?:");

scanf("%s",&c);

} while (c=='y');

}

void insert(node \*head,node data)

{

node \*t;

node \*newnode;

newnode=(node\*)malloc(sizeof(node));

newnode->blockid=data.blockid;

strcpy(newnode->fname,data.fname);

newnode->next=NULL;

t=head;

while(t->next!=NULL)

{

t=t->next;

}

t->next=newnode;

}

void contiguous\_alloc(node\* mainmem,fileDetails filedata[])

{

dir d[noOfFiles];

int i,j;

int alloc=0;

int randno;

int occur[noOfBlocks+1];

node \*t;

node \*start;

int found;

int count\_rand;

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

{

found=0;

count\_rand=0;

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

occur[j]=0;

printf("outer\n");

while(count\_rand!=noOfBlocks )

{

randno = (rand() % noOfBlocks) + 1;

while(occur[randno]!=0)

{

randno = (rand() % noOfBlocks ) + 1;

}

printf("randno:%d",randno);

count\_rand++;

occur[randno]=1;

t=mainmem;

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

{

t=t->next;

}

start=t;

found=1;

for(j=0;j<filedata[i].fileblocks;j++)

{

if(t==NULL)

{

break;

}

if(strcmp(t->fname,"free")==0)

{

t=t->next;

continue;

}

else

{

found=0;

break;

}

}

if(found==1)

{

d[alloc].start=start->blockid;

d[alloc].length=filedata[i].fileblocks;

strcpy(d[alloc].fname,filedata[i].fname);

for(j=0;j<filedata[i].fileblocks;j++)

{

strcpy(start->fname,filedata[i].fname);

start=start->next;

}

break;

}

}

if(found==0)

{

printf("\nMEMORY UNAVAILABLE\n");

}

else

{

alloc++;

}

}

printf("No. of files allocated:%d\n",alloc);

printf("\nDirectory\n");

printf("\tFile Name\tStart\tLength\n");

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

{

printf("\t%s\t\t%d\t%d\n",d[i].fname,d[i].start,d[i].length);

}

}

void linked\_alloc(fileDetails filedata[],node \*mainmem)

{

node \*temp,\*start,\*new;

int i,j,k;

int randno;

int found;

dir d[noOfFiles];

int startpos;

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

{

for(j=0;j<filedata[i].fileblocks;j++)

{

found=0;

while(found==0)

{

randno = (rand() % noOfBlocks ) + 1;

temp=mainmem;

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

{

temp=temp->next;

}

if(strcmp(temp->fname,"free")==0)

{

strcpy(temp->fname,filedata[i].fname);

found=1;

if(j==0)

{

new=temp;

strcpy(d[i].fname,filedata[i].fname);

d[i].start=temp->blockid;

}

else if(j==filedata[i].fileblocks-1)

{

new->link=temp;

temp->link=NULL;

d[i].end=temp->blockid;

}

else

{

new->link=temp;

new=new->link;

}

}

}

}

}

printf("\nDirectory\n");

printf("\tFile Name\tStart\tEnd\n");

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

{

printf("\t%s\t\t%d\t%d\n",d[i].fname,d[i].start,d[i].end);

}

printf("\nIndividual File listing\n");

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

{

printf("File Name: %s\n",d[i].fname);

startpos=d[i].start;

temp=mainmem;

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

{

temp=temp->next;

}

printf("\tData-block %d\n",temp->blockid);

temp=temp->link;

while(temp!=NULL)

{

printf("\tData-block %d\n",temp->blockid);

temp=temp->link;

}

}

}

void indexed\_alloc(fileDetails filedata[],node \*mainmem)

{

node \*temp,\*start,\*indexblock;

int i,j,k;

int indexblockid;

int randno;

int found;

dir d[noOfFiles];

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

{

found=0;

while(found!=1)

{

randno = (rand() % noOfBlocks ) + 1;

temp=mainmem;

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

{

temp=temp->next;

}

if(strcmp(temp->fname,"free")==0)

{

found=1;

strcpy(temp->fname,filedata[i].fname);

}

}

indexblock=temp;

strcpy(d[i].fname,filedata[i].fname);

d[i].start=indexblock->blockid;

for(j=0;j<filedata[i].fileblocks;j++)

{

found=0;

while(found!=1)

{

randno = (rand() % noOfBlocks ) + 1;

temp=mainmem;

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

{

temp=temp->next;

}

if(strcmp(temp->fname,"free")==0)

{

found=1;

strcpy(temp->fname,filedata[i].fname);

indexblock->fileblocktable[j]=temp->blockid;

}

}

}

}

printf("\nDirectory\n");

printf("\tFile Name\tIndexed Block\n");

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

{

printf("\t%s\t\t%d\n",d[i].fname,d[i].start);

}

printf("\n\nIndex Table\n");

printf("File Name\t\tBlock Indexed\n");

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

{

indexblockid=d[i].start;

temp=mainmem;

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

{

temp=temp->next;

}

printf("\n%s",temp->fname);

for(j=0;j<filedata[i].fileblocks;j++)

{

printf("\t\t\tData-block %d\n",temp->fileblocktable[j]);

}

}

}

/\*

Enter the main memory size:500

Enter the block size:10

Total no. of blocks available:50

Number of files to be allocated:5

Name of file 1:a

Size of file 1(in KB):25

Name of file 2:b

Size of file 2(in KB):30

Name of file 3:c

Size of file 3(in KB):35

Name of file 4:d

Size of file 4(in KB):45

Name of file 5:e

Size of file 5(in KB):55

FILE ALLOCATION TECHNIQUES

1.Contiguous

2.Linked

3.Indexed

Enter choice:1

outer

randno:13outer

randno:27outer

randno:47outer

randno:50randno:43randno:14randno:24randno:39outer

randno:32No. of files allocated:5

Directory

File Name Start Length

a 13 3

b 27 3

c 47 4

d 39 5

e 32 6

Do you want to continue?:y

FILE ALLOCATION TECHNIQUES

1.Contiguous

2.Linked

3.Indexed

Enter choice:2

Directory

File Name Start End

a 49 45

b 2 22

c 43 36

d 28 31

e 37 18

Individual File listing

File Name: a

Data-block 49

Data-block 38

Data-block 45

File Name: b

Data-block 2

Data-block 32

Data-block 22

File Name: c

Data-block 43

Data-block 7

Data-block 50

Data-block 36

File Name: d

Data-block 28

Data-block 29

Data-block 14

Data-block 11

Data-block 31

File Name: e

Data-block 37

Data-block 41

Data-block 48

Data-block 26

Data-block 16

Data-block 18

Do you want to continue?:y

FILE ALLOCATION TECHNIQUES

1.Contiguous

2.Linked

3.Indexed

Enter choice:3

Directory

File Name Indexed Block

a 35

b 39

c 43

d 49

e 40

Index Table

File Name Block Indexed

a Data-block 23

Data-block 45

Data-block 5

b Data-block 38

Data-block 42

Data-block 8

c Data-block 33

Data-block 12

Data-block 17

Data-block 16

d Data-block 44

Data-block 14

Data-block 9

Data-block 13

Data-block 4

e Data-block 21

Data-block 48

Data-block 25

Data-block 47

Data-block 2

Data-block 32

Do you want to continue?:n

\*/