#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

int memory;

int block\_size;

int no\_file;

int no\_blocks;

typedef struct mem

{

int block\_id;

char filename[50];

int flag;

}mem;

typedef struct inp

{

char file[50];

int size;

int req\_block;

}input;

typedef struct linked

{

char filename[50];

int block;

struct linked \*next;

}list;

typedef struct indexed

{

int block\_id;

int blocks[100];

}indexs;

input value[100];

mem contiguous[100];

int checks()

{

int max = 0;

for(int i=0;i<no\_blocks-1;i++)

{

int count = 0;

if(contiguous[i].flag == 0)

{

for(int j=i+1;j<no\_blocks;j++)

{

if(contiguous[j].flag == 0)

count++;

else

break;

}

}

if(count > max)

max = count;

}

return max;

}

int generate()

{

int r;

r = rand()%no\_blocks;

return r;

}

void alloc\_contiguous()

{

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

input en[no\_file];

for(int j=0;j<no\_file;j++)

{

en[j] = value[j];

}

printf("%s%10s%10s\n","Filename\0","Start\0","Length\0");

for(int s=0;s<no\_file;s++)

{

while(1)

{

int r;

r = generate();

if(contiguous[r].flag == 1)

continue;

if(en[s].req\_block <= checks())

{

int start;

int end = 0;

int arr[en[s].req\_block];

int si;

for(int l=r;l<no\_blocks;l++)

{

start = l;

si = 0;

for(int m=l+1;m<no\_blocks;m++)

{

if(contiguous[m].flag == 1 || si == en[s].req\_block-1)

break;

else

{

arr[si] = m;

si++;

}

}

if(si == en[s].req\_block-1)

{

for(int ss=0;ss<si;ss++)

{

contiguous[arr[ss]].flag = 1;

strcpy(contiguous[arr[ss]].filename,en[s].file);

}

printf("%s%10d%10d\n",en[s].file,start,en[s].size);

end = 1;

break;

}

}

if(end == 1)

break;

}

else

{

printf("Not Allocated for %s!!! Continuous blocks not available\n",en[s].file);

break;

}

}

}

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

}

void insert(list \*f,char file[50],int b)

{

list \*s = f;

list \*p;

p = (struct linked \*)malloc(sizeof(struct linked));

strcpy(p->filename,file);

p->block = b;

while(s->next != NULL)

s = s->next;

p->next = s->next;

s->next = p;

}

int checking()

{

int count=0;

for(int j=0;j<no\_blocks;j++)

{

if(contiguous[j].flag == 0)

count++;

}

return count;

}

void alloc\_linked()

{

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

list \*files[100];

for(int m=0;m<no\_file;m++)

files[m] = (struct linked \*)malloc(sizeof(struct linked));

printf("%s%15s%10s\n","Filename\0","Startblock\0","Endblock\0");

for(int s=0;s<no\_file;s++)

{

int i=0,end=0;

while(i<1)

{

int r;

r = generate();

if(contiguous[r].flag == 1)

continue;

if(value[s].req\_block <= checking())

{

insert(files[s],value[s].file,contiguous[r].block\_id);

contiguous[r].flag = 1;

int count = 0;

while(1)

{

int n;

n = generate();

if(contiguous[n].flag == 1)

continue;

else

{

insert(files[s],value[s].file,contiguous[n].block\_id);

contiguous[n].flag = 1;

count++;

}

if(count == value[s].req\_block-1)

break;

}

}

else

{

printf("Not Allocated for %s!!! free blocks not sufficient\n",value[s].file);

end = 1;

break;

}

i++;

}

if(end != 1)

{

list \*ss = files[s]->next;

printf("%s%14d",ss->filename,ss->block);

while(ss->next !=NULL)

ss = ss->next;

printf("%12d\n",ss->block);

}

}

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

}

void alloc\_indexed()

{

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

indexs in[no\_file];

for(int s=0;s<no\_file;s++)

{

int i=0,end=0;

while(i<1)

{

int r;

r = generate();

if(contiguous[r].flag == 1)

continue;

if(value[s].req\_block+1 <= checking())

{

in[s].block\_id=contiguous[r].block\_id;

contiguous[r].flag = 1;

int count = 0;

while(1)

{

int n;

n = generate();

if(contiguous[n].flag == 1)

continue;

else

{

in[s].blocks[count] = contiguous[n].block\_id;

contiguous[n].flag = 1;

count++;

}

if(count == value[s].req\_block)

break;

}

}

else

{

printf("Not Allocated for %s!!! free blocks not sufficient\n",value[s].file);

end = 1;

break;

}

i++;

}

if(end != 1)

{

printf("Filename : %s\nIndexed block Number : %d\nBlocks : ",value[s].file,in[s].block\_id);

for(int k=0;k<value[s].req\_block;k++)

printf("%d ",in[s].blocks[k]);

printf("\n");

}

}

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

}

void main()

{

printf("Enter the totalmemory : ");

scanf("%d",&memory);

printf("Enter the block size : ");

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

no\_blocks = memory/block\_size;

printf("Enter no of files : ");

scanf("%d",&no\_file);

for(int i=0;i<no\_file;i++)

{

printf("Enter the file name : ");

scanf("%s",value[i].file);

printf("Enter the size : ");

scanf("%d",&value[i].size);

if(value[i].size%block\_size == 0)

value[i].req\_block = value[i].size/block\_size;

else

value[i].req\_block = value[i].size/block\_size + 1;

printf("%s %d %d\n",value[i].file,value[i].req\_block,value[i].size);

}

int choi;

while(1)

{

for(int k=0;k<no\_blocks;k++)

{

contiguous[k].block\_id = k+1;

contiguous[k].flag = 0;

}

printf("1:Contiguous\n2:Linked\n3:Indexed\n4:Exit\nYour Option : ");

scanf("%d",&choi);

if(choi == 4)

{

printf("Exited.....\n");

exit(0);

}

if(choi == 1)

alloc\_contiguous();

if(choi == 2)

alloc\_linked();

if(choi == 3)

alloc\_indexed();

}

}

/\*

Output :

amr@ubuntu:~/Desktop/OS$ gcc ex12.c -o ex12out

amr@ubuntu:~/Desktop/OS$ ./ex12out

Enter the totalmemory : 200

Enter the block size : 10

Enter no of files : 4

Enter the file name : file1

Enter the size : 35

file1 4 35

Enter the file name : file2

Enter the size : 20

file2 2 20

Enter the file name : file3

Enter the size : 50

file3 5 50

Enter the file name : file4

Enter the size : 63

file4 7 63

1:Contiguous

2:Linked

3:Indexed

4:Exit

Your Option : 1

-------------------Contiguous Allocation------------------

Filename Start Length

file1 3 35

file2 17 20

file3 13 50

Not Allocated for file4!!! Continuous blocks not available

------------------------------------------------------

1:Contiguous

2:Linked

3:Indexed

4:Exit

Your Option : 2

---------------------Linked Allocation--------------------

Filename block

file1 10 2 3 8

file2 11 20

file3 4 7 1 13 17

file4 12 9 16 19 14 5 18

------------------------------------------------------

1:Contiguous

2:Linked

3:Indexed

4:Exit

Your Option : 3

---------------------Indexed Allocation-------------------

Filename : file1

Indexed block Number : 19

Blocks : 5 16 11 14

Filename : file2

Indexed block Number : 7

Blocks : 12 1

Filename : file3

Indexed block Number : 17

Blocks : 3 2 6 8 10

Not Allocated for file4!!! free blocks not sufficient

------------------------------------------------------

1:Contiguous

2:Linked

3:Indexed

4:Exit

Your Option : 4

Exited.....

\*/