UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA

FACULTY OF TELECOMMUNICATION AND INFORMATION ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

Operating systems

File Allocation Strategies

LAB MANUAL 7

Date:	
Name:	
Reg#:	**Group:**
Marks:	**Signature:**

EXPERIMENT:

Simulate file Allocation strategies:

Sequential AIM:

Simulate the file allocation strategies using file allocation methods

HARDWARE REQUIREMENTS:

Intel based Desktop Pc RAM of 512 MB

SOFTWARE REQUIREMENTS:

Turbo C/ Borland C/Dev C

THEORY:

File Allocation Strategies:

The main problem is how to allocate disk space to the files so that disk space is utilized effectively band files can be accessed quickly. We have 3 space allocation method.

Contiguous allocation (Sequential)

It requires each file to occupy a set of contiguous blocks on the hard disk where disk address define a linear ordering on the disk.

Disadvantages:

Difficult for finding space for a new file.

Internal and external fragmentation will be occurred.

OBJECTIVE

Write a C program to simulate the following file allocation strategies.

a) Sequential b) Linked c) Indexed

DESCRIPTION:

A file is a collection of data, usually stored on disk. As a logical entity, a file enables to divide data into meaningful groups. As a physical entity, a file should be considered in terms of its organization. The term "file organization" refers to the way in which data is stored in a file and, consequently, the method(s) by which it can be accessed.

SEQUENTIAL FILE ALLOCATION

In this file organization, the records of the file are stored one after another both physically and logically. That is, record with sequence number 16 is located just after the 15th record. A record of a sequential file can only be accessed by reading all the previous records.

LINKED FILE ALLOCATION

With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block.

INDEXED FILE ALLOCATION

Indexed file allocation strategy brings all the pointers together into one location: an index block. Each file has its own index block, which is an array of disk-block addresses. The ith entry in the index block points to the ith block of the file. The directory contains the address of the index block. To find and read the ith block, the pointer in the i th index-block entry is used.

ALGORITHM:

```
1. Start
2. Read the number of files
 3. For each file, read the number of blocks required and the starting block of the file.
 4. Allocate the blocks sequentially to the file from the starting block.
5. Display the file name, starting block, and the blocks occupied by the file.
6. stop
```

PROGRAM

SEQUENTIAL FILE ALLOCATION

```
#include <stdio.h>
#include <conio.h>
struct fileTable
{
char name[20];
int sb, nob; }
ft[30];
 void main()
{
int i, j, n;
char s[20];
clrscr();
printf("Enter no of files :");
scanf("%d",&n);
   for(i=0;i<n;i++)
   {
   printf("\nEnter file name %d :",i+1);
   scanf("%s",ft[i].name);
   printf("Enter starting block of file %d :",i+1);
   scanf("%d",&ft[i].sb);
   printf("Enter no of blocks in file %d :",i+1);
   scanf("%d",&ft[i].nob);
   }
   printf("\nEnter the file name to be searched-- ");
   scanf("%s",s);
   for(i=0;i<n;i++)
   if(strcmp(s, ft[i].name)==0)
   break;
   if(i==n)
   printf("\nFile Not Found");
   else
   {
   printf("\nFILE NAME START BLOCK NO OF BLOCKS BLOCKS OCCUPIED\n");
   printf("\n%s\t\t%d\t\t%d\t",ft[i].name,ft[i].sb,ft[i].nob);
   for(j=0;j<ft[i].nob;j++)
   printf("%d, ",ft[i].sb+j);
   }
   getch();
   }
```

INPUT:

```
Enter no of files :3
Enter file name 1 :A
Enter starting block of file 1 :85
Enter no of blocks in file 1 :6
Enter file name 2 :B
Enter starting block of file 2 :102
Enter no of blocks in file 2 :4
Enter file name 3 :C
Enter starting block of file 3 :60
Enter no of blocks in file 3 :4
Enter the file name to be searched -- B
```

OUTPUT:

```
FILE NAME     START BLOCK       NO OF BLOCKS       BLOCKS OCCUPIED
B                    102                    4                  102, 103, 104, 105
```

Or

Description:

Files are normally stored on the disks. So the main problem is how to allocate space to those files. So that disk space is utilized effectively and files can be accessed quickly. Three major strategies of allocating disc space are in wide use. Sequential, indexed and linked.

Sequential allocation:

In this allocation strategy, each file occupies a set of contiguously blocks on the disk. This strategy is best suited. For sequential files, the file allocation table consists of a single entry for each file. It shows the filenames, staring block of the file and size of the file. The main problem of this strategy is, it is difficult to find the contiguous free blocks in the disk and some free blocks could happen between two files.

Algorithm for Sequential File Allocation:

Step 1: Start the program.

Step 2: Get the number of memory partition and their sizes.

Step 3: Get the number of processes and values of block size for each process.

Step 4: First fit algorithm searches all the entire memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting process.

Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can be allocated to requesting process and allocates if.

Step 6: Worst fit algorithm searches the memory blocks for the largest hole and allocates it to the process.

Step 7: Analyses all the three memory management techniques and display the best algorithm which utilizes the memory resources effectively and efficiently.

Step 8: Stop the program.

```
/* Program to simulate sequential file allocation strategy */
#include < stdio.h>
#include<conio.h>
void main()
{
int f[50], i, st, len, j, c, k, count = 0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
printf("Files Allocated are : \n");
x: count=0;
printf("Enter starting block and length of files: ");
scanf("%d%d", &st,&len);
for(k=st;k<(st+len);k++)
 if(f[k]==0)
 count++;
if(len==count)
{
 for(j=st;j<(st+len);j++)
if(f[j]==0)
 {
 f[j]=1;
 printf("%d\t%d\n",j,f[j]);
 }
 if(j!=(st+len-1))
 printf(" The file is allocated to disk\n");
}
else
printf(” The file is not allocated \n");
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
 goto x;
else
 exit();
getch();
}
```

OUTPUT 1:

```
Files Allocated are :
Enter starting block and length of files: 14 3
14 1
15 1
16 1
The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files: 14 1
The file is not allocated
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files: 14 4
The file is not allocated
Do you want to enter more file(Yes - 1/No - 0)0
```

OUTPUT 2:

```
Files Allocated are :
Enter starting block and length of files: 17 4
17 1
18 1
19 1
20 1
The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files: 21 3
21 1
```

```
22 1
23 1
The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files: 19 4
The file is not allocated
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files: 25 5
25 1
26 1
27 1
28 1
29 1
The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)0
```

INDEXED FILE ALLOCATION

AIM: Write a C Program to implement Indexed File Allocation method.

Description:

Indexed allocation:

Indexed allocation supports both sequential and direct access files. The file indexes are not physically stored as a part of the file allocation table. Whenever the file size increases, we can easily add some more blocks to the index. In this strategy, the file allocation table contains a single entry for each file. The entry consisting of one index block, the index blocks having the pointers to the other blocks. No external fragmentation.

Algorithm for Indexed File Allocation:

Step 1: Start.

Step 2: Let n be the size of the buffer

Step 3: check if there are any producer

Step 4: if yes check whether the buffer is full

Step 5: If no the producer item is stored in the buffer

Step 6: If the buffer is full the producer has to wait

Step 7: Check there is any consumer.If yes check whether the buffer is empty

Step 8: If no the consumer consumes them from the buffer

Step 9: If the buffer is empty, the consumer has to wait.

Step 10: Repeat checking for the producer and consumer till required

Step 11: Terminate the process.

```
/* Program to simulate indexed file allocation strategy */
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
x:printf("Enter the index block: ");
scanf("%d",&ind);
if(f[ind]!=1)
{
 printf("Enter no of blocks needed and no of files for the index %d on the disk : \n",
ind);
 scanf("%d",&n);
}
```

```
else
printf("%d index is already allocated \n",ind);
 goto x;
}
y: count=0;
for(i=0;i<n;i++)
{
 scanf("%d", &index[i]);
 if(f[index[i]]==0)
 count++;
}
if(count==n)
{
 for(j=0;j<n;j++)
 f[index[j]]=1;
 printf("Allocated\n");
 printf("File Indexed\n");
 for(k=0;k<n;k++)
 printf("%d-------->%d : %d\n",ind,index[k],f[index[k]]);
}
else
{
printf("File in the index is already allocated \n");
printf("Enter another file indexed");
goto y;
```

```
}
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
 goto x;
else
 exit(0);
getch();
}
```

OUTPUT 1:

```
Enter the index block: 5
Enter no of blocks needed and no of files for the index 5 on the disk :
4
1 2 3 4
Allocated
File Indexed
5-------->1 : 1
5-------->2 : 1
5-------->3 : 1
5-------->4 : 1
Do you want to enter more file(Yes - 1/No - 0)1
Enter the index block: 4
4 index is already allocated
Enter the index block: 6
Enter no of blocks needed and no of files for the index 6 on the disk :

2

7 8

A5llocated

File Indexed

6-------->7 : 1

6-------->8 : 1

Do you want to enter more file(Yes - 1/No - 0)0
```

OUTPUT 2:

```
Enter the index block: 4

Enter no of blocks needed and no of files for the index 4 on the disk :

3

1 2 3

Allocated

File Indexed

4-------->1 : 1

4-------->2 : 1

4-------->3 : 1

Do you want to enter more file(Yes - 1/No - 0)0
```

Or

ALGORITHM:

1. Start

2. Read the number of files

3. Read the index block for each file.

4. For each file, read the number of blocks occupied and number of blocks of the file.

5. Link all the blocks of the file to the index block.

6. Display the file name, index block , and the blocks occupied by the file.

7. stop

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
main()
{
int n,m[20],i,j,ib[20],b[20][20];
clrscr();
printf("Enter no. of files:");
scanf("%d",&n);
for(i=0;i<n;i++)
{ printf("Enter index block :",i+1);
scanf("%d",&ib[i]);
printf("Enter blocks occupied by file%d:",i+1);
scanf("%d",&m[i]);
printf("enter blocks of file%d:",i+1);
for(j=0;j<m[i];j++)
scanf("%d",&b[i][j]);
} printf("\nFile\t index\tlength\n");
for(i=0;i<n;i++)
printf("%d\t%d\t%d\n",i+1,ib[i],m[i]);
printf("blocks occupiedare:");
 for(i=0;i<n;i++)
 { printf("fileno%d",i+1);
 for(j=0;j<m[i];j++)
 printf("\t%d--->%d\n",ib[i],b[i][j]);
 printf("\n");
 }
getch();
}
```

OUTPUT:

```
Enter no. of files:2
Enter index block 3
Enter blocks occupied by file1: 4
enter blocks of file1:9
 4 6 7
Enter index block 5
Enter blocks occupied by file2:2
enter blocks of file2: 10 8
 File index length
1 3 4
2 5 2
blocksoccupiedare:
file1
3--->9
```

3--->4

3--->6

3--->7

file2

5--->1

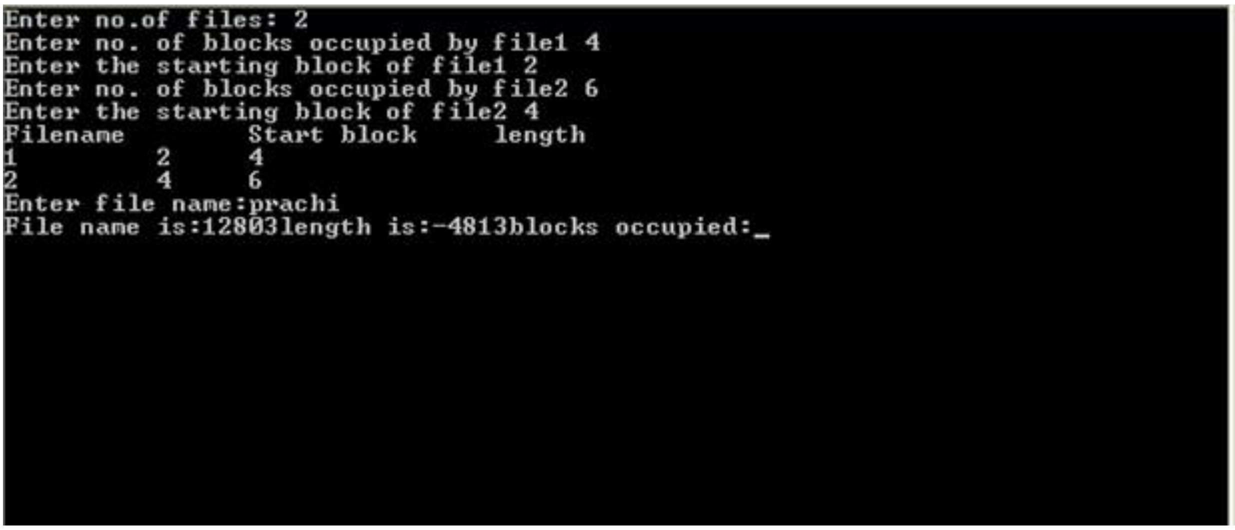
5--->8

Lab Task

P1: simulate sequential file allocation by creating sequential file and find out no. of sequential block occupied.

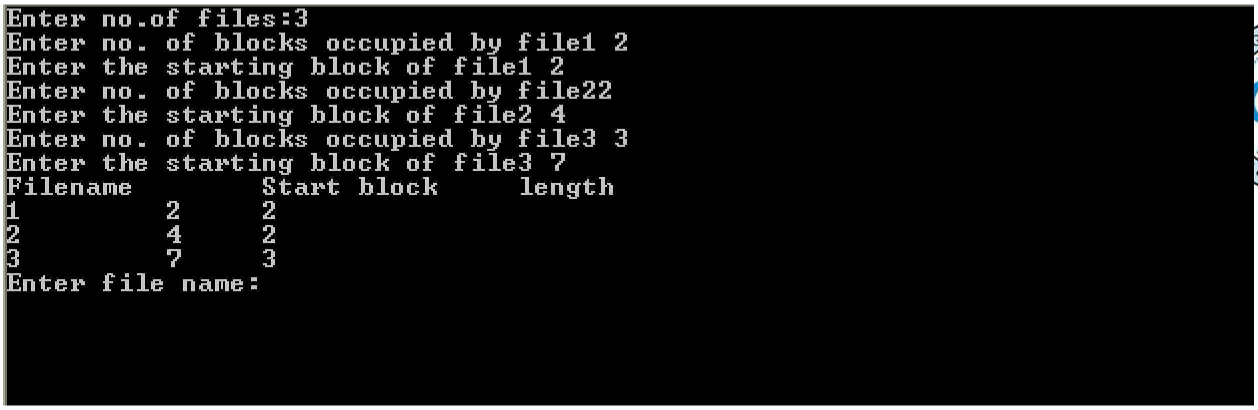
Input and Output:

Test case 1 :



```
Enter no.of files: 2
Enter no. of blocks occupied by file1 4
Enter the starting block of file1 2
Enter no. of blocks occupied by file2 6
Enter the starting block of file2 4
Filename         Start block     length
1          2     4
2          4     6
Enter file name:prachi
File name is:12803length is:-4813blocks occupied:_
```

Test case 2 :



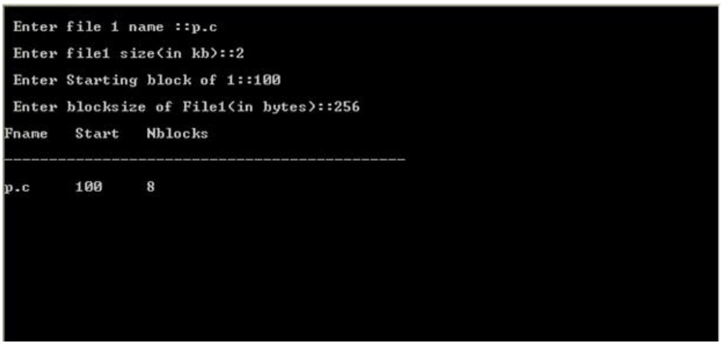
```
Enter no.of files:3
Enter no. of blocks occupied by file1 2
Enter the starting block of file1 2
Enter no. of blocks occupied by file22
Enter the starting block of file2 4
Enter no. of blocks occupied by file3 3
Enter the starting block of file3 7
Filename        Start block     length
1         2     2
2         4     2
3         7     3
Enter file name:
```

P2: Simulate sequential file allocation by creating single sequential file and find out no. of block required by taking input block size for a file.

Input and Output:

Test case :

:

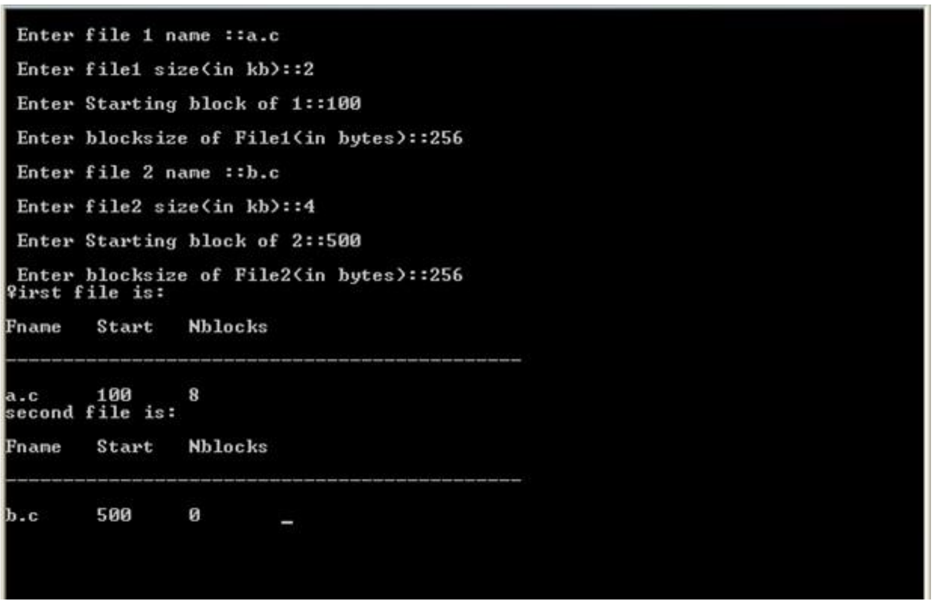


```
 Enter file 1 name ::p.c
 Enter file1 size(in kb)::2
 Enter Starting block of 1::100
 Enter blocksize of File1(in bytes)::256
Fname   Start   Nblocks
---------------------------------------------
p.c     100     8
```

P3: Simulate sequential file allocation by creating two files and find out no. of block required by taking input size of file and block size for both files.

Input and Output:

Test case :

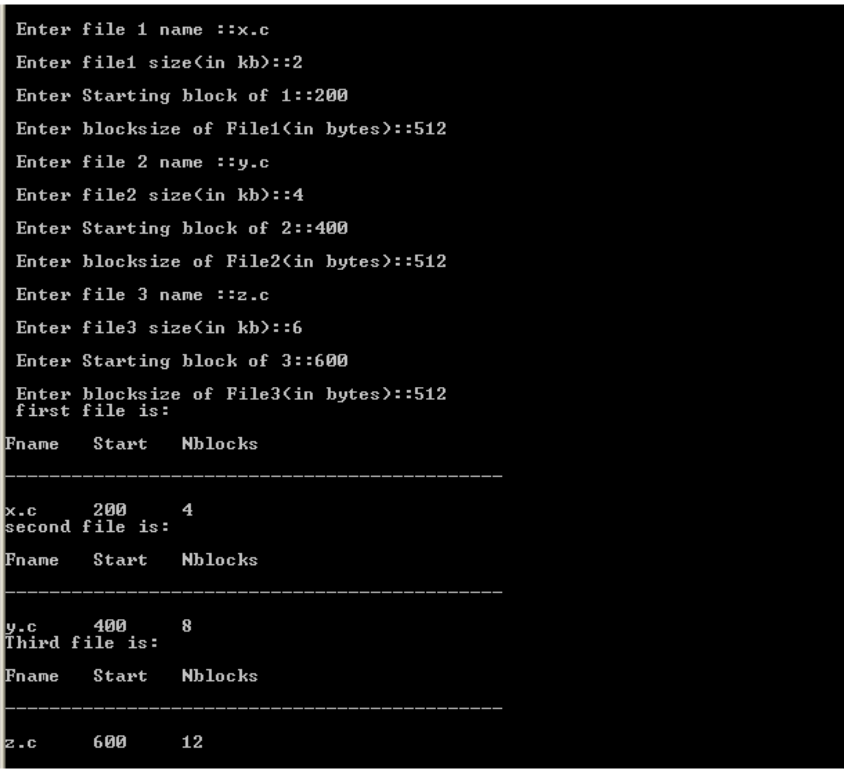


```
 Enter file 1 name ::a.c
 Enter file1 size(in kb)::2
 Enter Starting block of 1::100
 Enter blocksize of File1(in bytes)::256
 Enter file 2 name ::b.c
 Enter file2 size(in kb)::4
 Enter Starting block of 2::500
 Enter blocksize of File2(in bytes)::256
first file is:
Fname    Start    Nblocks
---------------------------------------------
a.c      100      8
second file is:
Fname    Start    Nblocks
---------------------------------------------
b.c      500      0       _
```

P4: Simulate sequential file allocation by creating three files and find out no. of block required by taking input size of file and block size for these files.

Input and Output:

Test case :

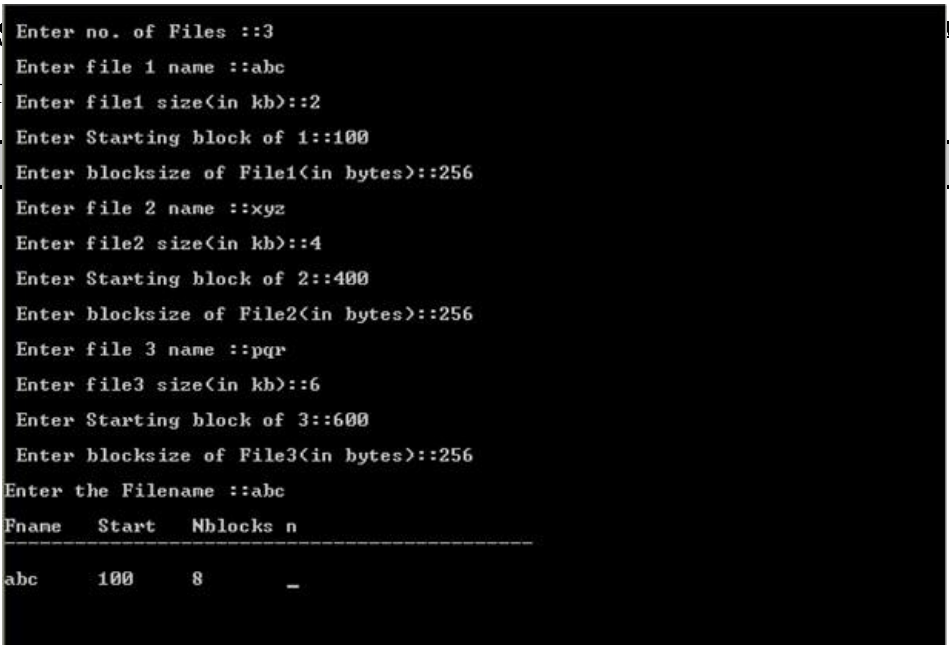


```
 Enter file 1 name ::x.c
 Enter file1 size(in kb)::2
 Enter Starting block of 1::200
 Enter blocksize of File1(in bytes)::512
 Enter file 2 name ::y.c
 Enter file2 size(in kb)::4
 Enter Starting block of 2::400
 Enter blocksize of File2(in bytes)::512
 Enter file 3 name ::z.c
 Enter file3 size(in kb)::6
 Enter Starting block of 3::600
 Enter blocksize of File3(in bytes)::512
 first file is:
Fname   Start   Nblocks
---------------------------------------------
x.c     200     4
second file is:
Fname   Start   Nblocks
---------------------------------------------
y.c     400     8
Third file is:
Fname   Start   Nblocks
---------------------------------------------
z.c     600     12
```

P5: Simulate sequential file allocation by creating n number of sequential file and find out no. of block required by taking input block size for n number of file.

Input and Output:

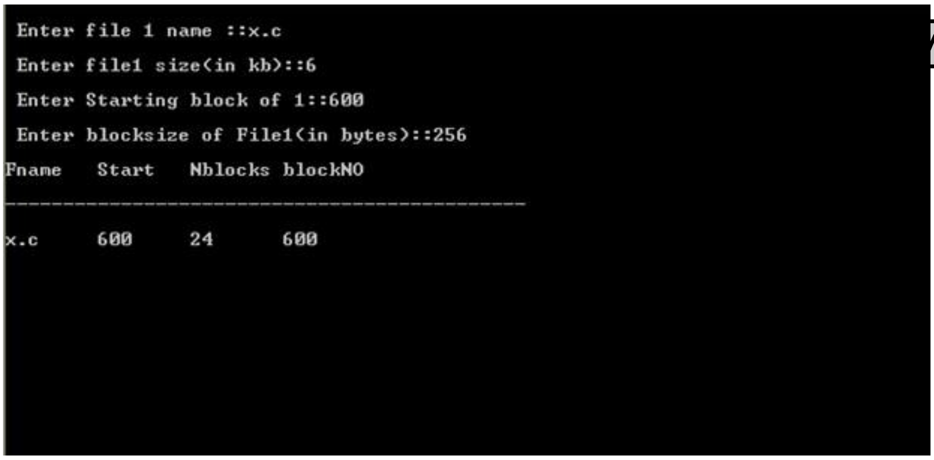
Test case :



```
 Enter no. of Files ::3
 Enter file 1 name ::abc
 Enter file1 size(in kb)::2
 Enter Starting block of 1::100
 Enter blocksize of File1(in bytes)::256
 Enter file 2 name ::xyz
 Enter file2 size(in kb)::4
 Enter Starting block of 2::400
 Enter blocksize of File2(in bytes)::256
 Enter file 3 name ::pqr
 Enter file3 size(in kb)::6
 Enter Starting block of 3::600
 Enter blocksize of File3(in bytes)::256
Enter the Filename ::abc
Fname   Start   Nblocks n
---------------------------------------------

abc     100     8       _
```

P6: Simulate sequential file allocation by creating single sequential file and find out no. of block required along with block address number.

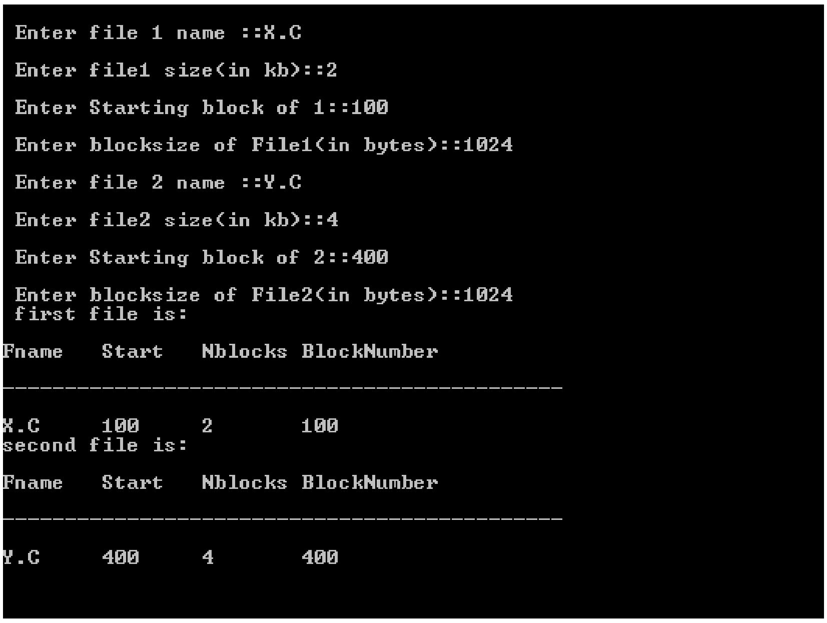
Test case :



```
 Enter file 1 name ::x.c

 Enter file1 size(in kb)::6

 Enter Starting block of 1::600

 Enter blocksize of File1(in bytes)::256

Fname   Start   Nblocks blockNO

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

x.c     600     24      600
```

P7: Simulate sequential file allocation by creating two files and find out no. of block required and block address number by taking input size of file and block size for both files.

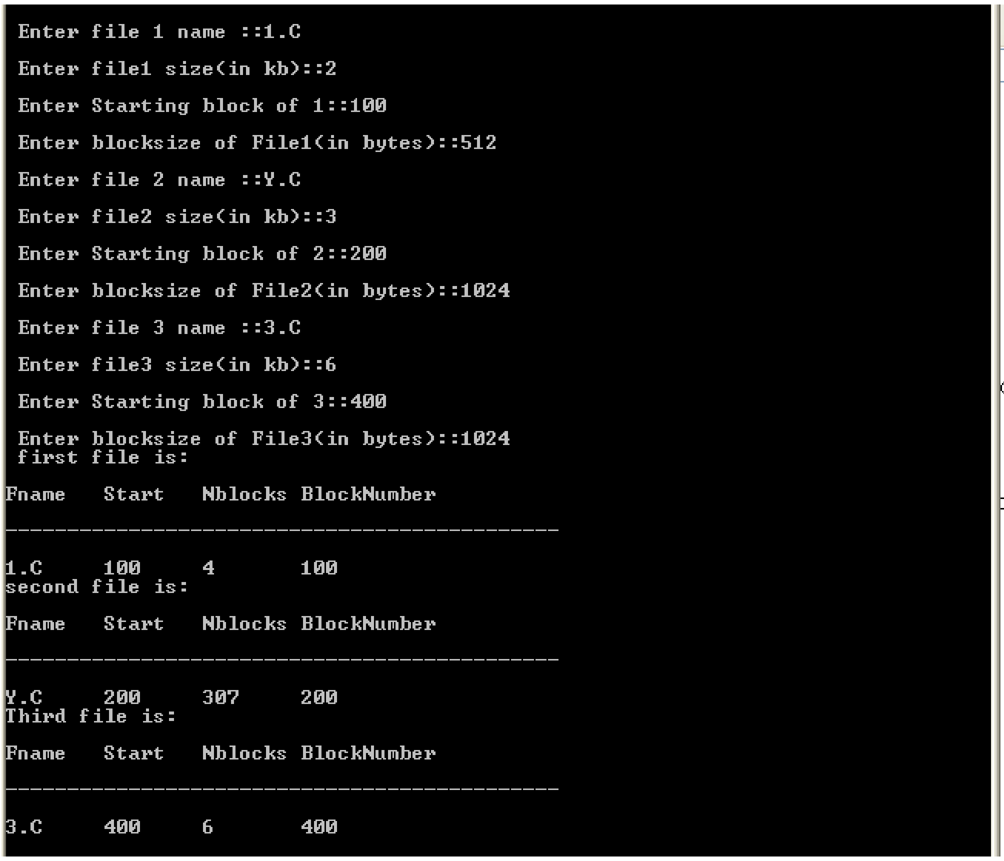
Test case :



```
 Enter file 1 name ::X.C

 Enter file1 size(in kb)::2

 Enter Starting block of 1::100

 Enter blocksize of File1(in bytes)::1024

 Enter file 2 name ::Y.C

 Enter file2 size(in kb)::4

 Enter Starting block of 2::400

 Enter blocksize of File2(in bytes)::1024
 first file is:

Fname   Start   Nblocks BlockNumber

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

X.C     100     2       100
second file is:

Fname   Start   Nblocks BlockNumber

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

Y.C     400     4       400
```

P8: Simulate sequential file allocation by creating three files and find out no. of block required and block address numbers by taking input size of file and block size for all files.

Test case :

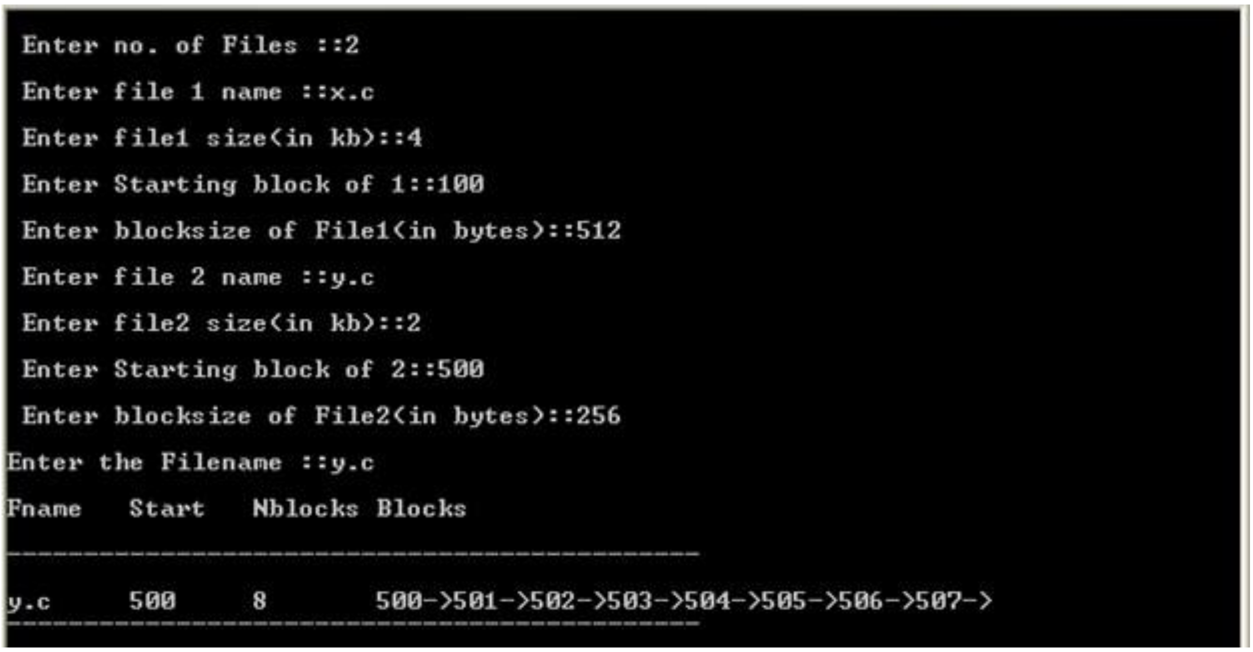


```
 Enter file 1 name ::1.C

 Enter file1 size(in kb)::2

 Enter Starting block of 1::100

 Enter blocksize of File1(in bytes)::512

 Enter file 2 name ::Y.C

 Enter file2 size(in kb)::3

 Enter Starting block of 2::200

 Enter blocksize of File2(in bytes)::1024

 Enter file 3 name ::3.C

 Enter file3 size(in kb)::6

 Enter Starting block of 3::400

 Enter blocksize of File3(in bytes)::1024
 first file is:

Fname   Start   Nblocks BlockNumber

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

1.C     100     4       100
second file is:

Fname   Start   Nblocks BlockNumber

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

Y.C     200     307     200
Third file is:

Fname   Start   Nblocks BlockNumber

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

3.C     400     6       400
```

P9: Simulate sequential file allocation by creating n number of sequential file and find out no. of block required and block address numbers by taking input block size for n number of file.

Input and output

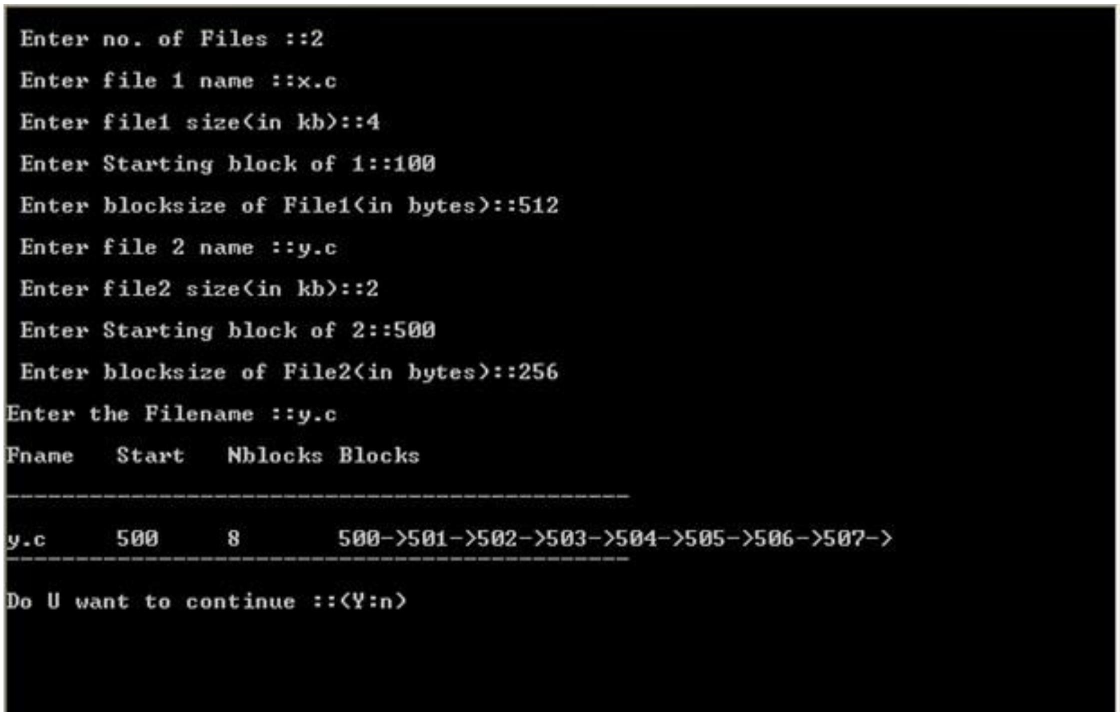
Test case :



```
 Enter no. of Files ::2

 Enter file 1 name ::x.c

 Enter file1 size(in kb)::4

 Enter Starting block of 1::100

 Enter blocksize of File1(in bytes)::512

 Enter file 2 name ::y.c

 Enter file2 size(in kb)::2

 Enter Starting block of 2::500

 Enter blocksize of File2(in bytes)::256

Enter the Filename ::y.c

Fname   Start   Nblocks Blocks

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

y.c     500     8       500->501->502->503->504->505->506->507->
---------------------------------------------
```

P10: Simulate sequential file allocation by creating n number of sequential file and find out no. of block required and block address numbers. Repeat the procedure.

Test case :



```
 Enter no. of Files ::2
 Enter file 1 name ::x.c
 Enter file1 size(in kb)::4
 Enter Starting block of 1::100
 Enter blocksize of File1(in bytes)::512
 Enter file 2 name ::y.c
 Enter file2 size(in kb)::2
 Enter Starting block of 2::500
 Enter blocksize of File2(in bytes)::256
Enter the Filename ::y.c
Fname   Start   Nblocks Blocks
---------------------------------------------
y.c     500     8       500->501->502->503->504->505->506->507->
---------------------------------------------
Do U want to continue ::(Y:n)
```

QUESTIONS:

1. Define File?

2.Define Directory?

3. Why we use file allocation strategies?

4.What are the advantages and dis-advantages Indexed Allocation?

5.Define Sequential File allocation

6.Why we use file allocation strategies?

7.What are the advantages and dis-advantages of Sequential File allocation?