**17. simulate the FILE ALLOCATION techniques:**

**a. Sequential File Allocation:**

**AIM:**

To implement sequential file allocation

**DESCRIPTION:**

In this technique, all the files are allocated in a sequential manner. The blocks will be allocated in an order for each file.

**ALGORITHM:**

**Step 1:**Start

**Step 2:**declare stdio.h,string.h

**Step 3:**declare st[20],b[20],b1[20],ch,i,j,n,blocks[20][20],sz[20] as integer

**Step 4:**read no of files(n)

**Step 5:**for i<-0 repeat upto n

begin

read f[i] (file name)

read sz[i] (file size in kb)

read st[i] (starting block)

read b[i] (block size of file in bytes)

end for

**Step 6:**for i<-0 repeat upto n

begin

b1[i]=(sz[i]\*1024)/b[i]

end for

**Step 7:**for i<-0 repeat upto n

begin

for j<-0 repeat upto b1[i]

begin

blocks[i][j]=st[i]+j;

end for

end for

**Step 8:**while(1)

begin

read s(file name)

for i<-0 repeat upto n

begin

if(strcmp(s,f[i]==0)

begin

write "FNAME START NBLOCKS BLOCKS"

write "---------------------------"

write f[i],st[i],b1[i],b[i]

write "->" blocks[i][j]

end if

end for

write "--------------------"

write "do u want to continue(y/n)(0/1)"

read ch

if(ch!=1)

begin

break

end if

end while

**Step 9:**stop

**SOURCE CODE:**

#include<stdio.h>

#include<string.h>

main()

{

int st[20],b[20],b1[20],ch,i,j,n,blocks[20][20],sz[20];

char f[20][20],s[20];

printf("enter no of files\n");

scanf("%d",&n);

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

{

printf("enter filename\n",i+1);

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

printf("enter the size of the file\n ");

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

printf("enter the starting block\n");

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

printf("enter the block size in bytes\n");

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

}

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

{

b1[i]=(sz[i]\*1024)/b[i];

}

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

{

for(j=0;j<b1[i];j++)

{

blocks[i][j]=st[i]+j;

}

}

while(1)

{

printf("enter the file name to which the memory has to be allocated\n");

scanf("%s",s);

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

{

if(strcmp(s,f[i])==0)

{

printf("\nFNAME\t\tSTART\t\tNBLOCKS\t\tBLOCKS\n\n");

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

printf("%s\t\t%d\t\t%d\t\t%d\n",f[i],st[i],b1[i],b[i]);

for(j=0;j<b1[i];j++)

{

printf("--->%d",blocks[i][j]);

}

}

}

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

printf("do u want to continue\n 1.yes\n 2.no\n");

printf("enter ur choice\n");

scanf("%d",&ch);

if(ch!=1)

{

break;

}

}

}

**OUTPUT:**

Enter no of files 3

Enter filename M

Enter the size of the file 1

Enter the starting block 1000

Enter the block size in bytes 256

Enter filename B

Enter the size of the file 2

Enter the starting block 2000

Enter the block size in bytes 512

Enter filename R

Enter the size of the file 3

Enter the starting block 3000

Enter the block size in bytes 1024

Enter the filename to which the memory has to be allocated M

FNAME START NBLOCKS BLOCKS

M 1000 4 256

---🡪1000---🡪1001-🡪1002--🡪1003

Do u want to continue

1.yes

2. no

Enter ur choice 1

Enter the filename to which the memory has to be allocated B

FNAME START NBLOCKS BLOCKS

B 2000 4 512

-🡪2000---🡪2001-🡪2002--🡪2003

Do u want to continue

1.yes

2. no

Enter ur choice 1

Enter the filename to which the memory has to be allocated R

FNAME START NBLOCKS BLOCKS

R 3000 3 1024

---🡪3000---🡪3001-🡪3002

Do u want to continue

1.yes

2. no

Enter ur choice 2

**b. Indexed File Allocation:**

**AIM:**

Implementation of indexed file allocation.

**DESCRIPTION:**

In this type of strategy, the files are allocated based on the indexes that are created for each fragment of the file such that each and every similar indexed file is maintained by the primary index thereby providing flow to the file fragments.

ADVANTAGES:

* This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
* It overcomes the problem of external fragmentation.

DISADVANTAGES:

* The pointer overhead for indexed allocation is greater than liked allocation.

**ALGORITHM:**

**Step 1:** start

**Step 2:** include the header file stdio and string

**Step 3:** declare the necessary variables

Global: n of type int

Local: b[20],b1[20],I,j,blocks[20][20],ch of int type

F[20][20],s[20] as character

**Step 4:** read no of files(n)

**Step 5:** for i<-0 repeat to n

begin

read filename(f[i])

read filesize in kb (sz[i])

read block size of files in bytes(b[i])

end for

**Step 6:** for i<-0 repeat to n

begin

b1[i]=(sz[i]\*1024)/b[i]

write “enter blocks for i+1 file”

for j<-0 repeat to b1[i]

begin

read blocks[i][j]

end for

end for

**Step 7:** while(1)

begin

read file name(s)

for i<-0 repeat to n

begin

if(string compare(f[i],s) equals to zero)

begin

write”file name file size block size no of blocks blocks”

write” -------------”

write f[i],sz[i],b[i],b1[i]

for j<-0 repeat to b1[i]

begin

write”->”blocks[i][j]

end for

end if

end for

write”----------”

write”do u want to continue(0/1)”

read choice(ch)

end while

**Step 8:** stop

**SOURCE CODE:**

#include<stdio.h>

#include<string.h>

int n;

main()

{

int b[20],b1[20],i,j,blocks[20][20],ch,sz[20];

char f[20][20],s[20];

printf("enter no of files\n");

scanf("%d",&n);

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

{

printf("enter file name\n");

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

printf("enter file size in kb\n");

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

printf("enter block size of files in bytes\n");

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

}

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

{

b1[i]=(sz[i]\*1024)/b[i];

printf("enter blocks for %d file",i+1);

for(j=0;j<b1[i];j++)

{

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

}

}

while(1)

{

printf("enter filename:");

scanf("%s",s);

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

{

if(strcmp(f[i],s)==0)

{

printf("\nfilename\t filesize\t block size\t no of blocks\t blocks\t");

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

printf("\n%s\t%d\t%d\t%d\t",f[i],sz[i],b[i],b1[i]);

for(j=0;j<b1[i];j++)

{

printf("%d->",blocks[i][j]);

}

}

}

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

printf("\ndo u want to continue(0/1)\n");

printf("enter ch\n");

scanf("%d",&ch);

if(ch!=1)

break;

}

}

**Output:**

enter no of files 3

enter file name A

enter file size in kb 1

enter block size of files in bytes 256

enter file name B

enter file size in kb 2

enter block size of files in bytes 512

enter file name S

enter file size in kb 1

enter block size of files in bytes 1024

enter blocks for 1 file 200 209 212 215

enter blocks for 2 file 700 709 716 723

enter blocks for 3 file 912

enter filename:A

filename filesize block size no of blocks blocks

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

A 1 256 4 200->209->212->215->

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

do u want to continue(0/1)

enter ch 1

enter filename:B

filename filesize block size no of blocks blocks

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

B 2 512 4 700->709->716->723->

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

do u want to continue(0/1)

enter c 1

enter filename:S

filename filesize block size no of blocks blocks

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

S 1 1024 1 912->

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

do u want to continue(0/1)

enter ch 2

**c. Linked File Allocation:**

**AIM:**

To implement the linked file allocation technique.

**DESCRIPTION:**

In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk. The directory entry contains pointer to the starting and ending file block. Each block contains a pointer to the next block occupied by the file. This is very flexible in terms of file size. File size can be increased easily since the system doesnot have to look for a contiguous chunk of memory.

**ALGORITHM:**

**Step 1:** start

**Step 2:** include the header file stdio and string

**Step 3:** declare the necessary variables

Global:int n

Local :int b[20],b1[20],i,j,blocks[20][20],ch,x,sb[20],eb[20],sz[20];

char f[20][20],s[20];

**Step 4:**read no of files(n)

**Step 5:**for i<-0 repeat upto n

begin

read the filename (f[i])

read the filesize in kb (sz[i])

read blocksize of files in bytes(b[i])

end for

**Step 6:**for i<-0 repeat upto n

begin

b1[i]=(sz[i]\*1024)/b[i]

read the starting block of file(sb[i])

read the ending block of file (eb[i])

write “enter blocks for file”

for j<-0 repeat upto b1[i]-2

begin

write "enter the j+1 block:"

read x

if(x>sb[i]&&x<eb[i])

begin

blocks[i][j]=x

end

else

printf("invalid block\n");

end for

end for

**Step 7:** while(1)

begin

read filename(s)

for i<-0 repeat to n

begin

if(strcmp(f[i],s)==0)

begin

write “\nfname\t fsize\tbsize\tnblocks\tblock\n"

write"\n------------------------------------------------\n"

write”f[i],sz[i],b[i],b1[i]”

write “---->"sb[i]

for j<-0 repeat <b1[i]-2

begin

write “---->",blocks[i][j]

end for

write”---->" eb[i]);

end if

end for

write “\n----------------------------------------------------------\n"

write "do u want to continue(n- 0\ny-1\n)"

read ch

end while

**Step 8:** stop

**SOURCE CODE:**

#include<stdio.h>

#include<string.h>

int n;

main()

{

int b[20],b1[20],i,j,blocks[20][20],ch,x,sb[20],eb[20],sz[20];

char f[20][20],s[20];

printf("enter no of files\n");

scanf("%d",&n);

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

{

printf("enter the filename\n",i+1);

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

printf("enter the filesize in kb\n");

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

printf("enter blocksize of files in bytes\n");

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

}

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

{

b1[i]=(sz[i]\*1024)/b[i];

printf("enter the starting block of file %d:",i+1);

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

printf("enter the ending block of file %d:",i+1);

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

printf("enter blocks for %d file\n",i+1);

for(j=0;j<b1[i]-2;j++)

{

printf("enter the %d block:",j+1);

scanf("%d",&x);

if(x>sb[i]&&x<eb[i])

{

blocks[i][j]=x;

}

else

{

printf("invalid block\n");

}

}

}

while(1)

{

printf("enter filename\n");

scanf("%s",&s);

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

{

if(strcmp(f[i],s)==0)

{

printf("\nfname\t fsize\tbsize\tnblocks\tblock\n");

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

printf("%s\t%d\t%d\t%d\n",f[i],sz[i],b[i],b1[i]);

printf("%d---->",sb[i]);

for(j=0;j<b1[i]-2;j++)

{

printf("%d---->",blocks[i][j]);

}

printf("%d---->",eb[i]);

}

}

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

printf("do u want to continue\n 0\n 1\n");

printf("enter ur choice\n");

scanf("%d",&ch);

}

}

**OUTPUT:**

Enter no of files 3

Enter filename M

Enter the size of the file 2

Enter the block size in bytes 256

Enter filename B

Enter the size of the file 3

Enter the block size in bytes 512

Enter filename R

Enter the size of the file 1

enter the block size in bytes 1024

Enter the starting block of file 1 : 1000

Enter the ending block of file 1 : 2000

Enter blocks for 1 file

Enter 1 block : 1100

Enter 2 block : 1200

Enter 3 block : 1300

Enter 4 block : 1400

Enter 5 block : 1500

Enter 6 block : 1600

Enter the starting block of file 1 : 3000

Enter the ending block of file 1 : 4000

Enter 1 block : 3100

Enter 2 block : 3200

Enter 3 block : 3300

Enter 4 block : 3400

Enter blocks for 3 file

Enter the filename to which the memory has to be allocated M

FNAME Fsize bsize nblocks block

M 2 256 8

---🡪1000---🡪1001-🡪1002--🡪1003🡪1400🡪1500🡪1600-🡪2000-🡪

Do u want to continue

1.yes

2. no

Enter ur choice 1

Enter the filename to which the memory has to be allocated B

c

Do u want to continue

1.yes

2. no

Enter ur choice 1

Enter the filename to which the memory has to be allocated R

Fname Fsize bsize nblocks block

R 1 1024 1

Do u want to continue

1.yes

2. no

Enter ur choice 2