Exp. No. 15a Contiguous Allocation

Aim

To implement file allocation on free disk space in a contiguous manner.

File Allocation

The three methods of allocating disk space are:

1. Contiguous allocation
2. Linked allocation
3. Indexed allocation

Contiguous

- Each file occupies a set of contiguous block on the disk.
- The number of disk seeks required is minimal.
- The directory contains address of starting block and number of contiguous block (length) occupied.
- Supports both sequential and direct access.
- First / best fit is commonly used for selecting a hole.

Algorithm

1. Assume no. of blocks in the disk as 20 and all are free.
2. Display the status of disk blocks before allocation.
3. For each file to be allocated:
   a. Get the *filename*, *start* address and file *length*
   b. If *start* + *length* > 20, then goto step 2.
   c. Check to see whether any block in the range (start, start + length-1) is allocated. If so, then go to step 2.
   d. Allocate blocks to the file contiguously from start block to start + length – 1.
4. Display directory entries.
5. Display status of disk blocks after allocation
6. Stop

Program

```
/* Contiguous Allocation - cntalloc.c */

#include <stdio.h>
#include <string.h>

int num=0, length[10], start[10];
char fid[20][4], a[20][4];

void directory()
{
   int i;
   printf("\nFile Start Length\n");
```

```
   for(i=0; i<num; i++)
      printf("%-4s %3d %6d\n",fid[i],start[i],length[i]);
}

void display()
{
   int i;
   for(i=0; i<20; i++)
      printf("%4d",i);
   printf("\n");
   for(i=0; i<20; i++)
      printf("%4s", a[i]);
}

main()
{
   int i,n,k,temp,st,nb,ch,flag;
   char id[4];

   for(i=0; i<20; i++)
      strcpy(a[i], "");
   printf("Disk space before allocation:\n");
   display();
   do
   {
      printf("\nEnter File name (max 3 char) : ");
      scanf("%s", id);
      printf("Enter start block : ");
      scanf("%d", &st);
      printf("Enter no. of blocks : ");
      scanf("%d", &nb);
      strcpy(fid[num], id);
      length[num] = nb;
      flag = 0;

      if((st+nb) > 20)
      {
         printf("Requirement exceeds range\n");
         continue;
      }

      for(i=st; i<(st+nb); i++)
         if(strcmp(a[i], "") != 0)
            flag = 1;
      if(flag == 1)
      {
         printf("Contiguous allocation not possible.\n");
         continue;
      }
      start[num] = st;
      for(i=st; i<(st+nb); i++)
```

```
         strcpy(a[i], id);;
       printf("Allocation done\n");
       num++;

       printf("\nAny more allocation (1. yes / 2. no)? : ");
       scanf("%d", &ch);
    } while (ch == 1);
    printf("\n\t\t\tContiguous Allocation\n");
    printf("Directory:");
    directory();
    printf("\nDisk space after allocation:\n");
    display();
}
```

Output

```
Disk space before allocation:
   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19

Enter File name (max 3 char) : cp
Enter start block : 14
Enter no. of blocks : 3
Allocation done
Any more allocation (1. yes / 2. no)? : 1

Enter File name (max 3 char) : tr
Enter start block : 18
Enter no. of blocks : 3
Requirement exceeds range

Enter File name (max 3 char) : tr
Enter start block : 10
Enter no. of blocks : 3
Allocation done
Any more allocation (1. yes / 2. no)? : 1

Enter File name (max 3 char) : mv
Enter start block : 0
Enter no. of blocks : 2
Allocation done
Any more allocation (1. yes / 2. no)? : 1

Enter File name (max 3 char) : ps
Enter start block : 12
Enter no. of blocks : 3
Contiguous allocation not possible.
Any more allocation (1. yes / 2. no)? : 2

                  Contiguous Allocation
Directory:
File Start Length
cp    14      3
tr    10      3
mv     0      2

Disk space after allocation:
   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  mv  mv                               tr  tr  tr      cp  cp  cp
```

Result

Thus contiguous allocation is done for files with the available free blocks.

Ex. No. 15b LINKED FILE ALLOCATION

Date:

Aim

To st

Linked

- Each file is a linked list of disk blocks.
- The directory contains a pointer to first and last blocks of the file.
- The first block contains a pointer to the second one, second to third and so on.
- File size need not be known in advance, as in contiguous allocation.
- No external fragmentation.
- Supports sequential access only.

Indexed

- In indexed allocation, all pointers are put in a single block known as index block.
- The directory contains address of the index block.
- The i^{th} entry in the index block points to i^{th} block of the file.
- Indexed allocation supports direct access.
- It suffers from pointer overhead, i.e wastage of space in storing pointers.

Algorithm

1. Get no. of files
2. Accept filenames and no. of blocks fo each file
3. Obtrain start block for each file
4. Obtain other blocks for each file
5. Check block availability before allocation
6. If block is unavailable then report error
7. Accept file name
8. Display linked file allocation blocks for that file
9. Stop

Program

```
#include <stdio.h>
#include <conio.h>
#include <string.h>


main()
{
   static int b[20], i, j, blocks[20][20];
   char F[20][20], S[20], ch;
   int sb[20], eb[20], x, n;
   clrscr();
   printf("\n Enter no. of Files ::");
   scanf("%d",&n);
```

```
   for(i=0;i<n;i++)
   {
      printf("\n Enter file %d name ::", i+1);
      scanf("%s", &F[i]);
      printf("\n Enter No. of blocks::", i+1);
      scanf("%d",&b[i]);
   }

   for(i=0;i<n;i++)
   {
      printf("\n Enter Starting block of file%d::",i+1);
      scanf("%d", &sb[i]);
      printf("\nEnter blocks for file%d::\n", i+1);
      for(j=0; j<b[i]-1;)
      {
         printf("\n Enter the %dblock ::", j+2);
         scanf("%d", &x);
         if(b[i] != 0)
         {
            blocks[i][j] = x;
            j++;
         }
         else
            printf("\n Invalid block::");
      }
   }

   printf("\nEnter the Filename :");
   scanf("%s", &S);
   for(i=0; i<n; i++)
   {
      if(strcmp(F[i],S) == 0)
      {
         printf("\nFname\tBsize\tStart\tBlocks\n");
         printf("\n-------------------------------------\n");
         printf("\n%s\t%d\t%d\t", F[i], b[i], sb[i]);
         printf("%d->",sb[i]);
         for(j=0; j<b[i]; j++)
         {
            if(b[i] != 0)
               printf("%d->", blocks[i][j]);
         }
      }
   }
   printf("\n-----------------------------------------\n");
   getch();
}
```

Output

```
 Enter no. of Files ::2

 Enter file 1 name ::fcfs
 Enter No. of blocks::3

 Enter file 2 name ::sjf
 Enter No. of blocks::2

 Enter Starting block of file1::8
Enter blocks for file1::
 Enter the 2block ::3
 Enter the 3block ::5

 Enter Starting block of file2::2
Enter blocks for file2::
 Enter the 2block ::6

Enter the Filename ::fcfs

Fname   Bsize   Start   Blocks
-----------------------------------------
fcfs      3        8      8->3->5
-----------------------------------------
```

Result

Thus blocks for file were allocation using linked allocation method.