ISSN: 2321-7782

Volume 1, Issue 1, June 2013

International Journal of Advance Research in Computer Science and Management Studies

Research Paper

Available online at: www.ijarcsms.com

Disk Space Management Methods

Ramakrishna Chowdary
M.C.A. Department
Rao & Naidu Engineering College [RNEC]
Ongole, Prakasam
Andhra Pradesh – India

Abstract: Operating system maintains a list of free disk space to keep track of all disk blocks which are not being used by any file. Whenever a file has to be created, the list of free disk space is searched for and then allocated to the new file. The amount of space allocated to this file is then removed from the free space list. When a file is deleted, its disk space is added to the free space list.

But the problem is how to allocate space to files for effective disk space utilization and quick access.

In this paper we have to consider major methods to manage free disk space/blocks.

Keywords: Disk allocation methods, Continuous, Non-continuous

I. Introduction

An important function of the file system is to manage space on the secondary storage, which includes keeping track of both disk blocks allocated to files and the free block available for allocation.

The main problems in allocating space to files are:

- Effective utilization of disk space.
- Fast accesses of files

Management of disk blocks is a familiar problem that we have encountered and discussed in relation to main memory management. But, secondary storage introduces two additional problems:

- Slow disk access time and
- Fast accesses of files

In spite of that, many considerations are similar to both environments, particularly, contiguous and non-contiguous allocation of the files. Each method has its advantages and disadvantages.

Two widely used allocation techniques are contiguous and non-contiguous (Indexing and chaining).

II. Disk Allocation Methods

The direct-access of disks and keeping files in adjacent areas of the disk is highly desirable. But the problem is how to allocate space to files for effective disk space utilization and quick access. Also, as files are allocated and freed, the space on a disk becomes fragmented. The major methods of allocating disk space are:

1. Continuous
2. Non-continuous (Indexing and Chaining)

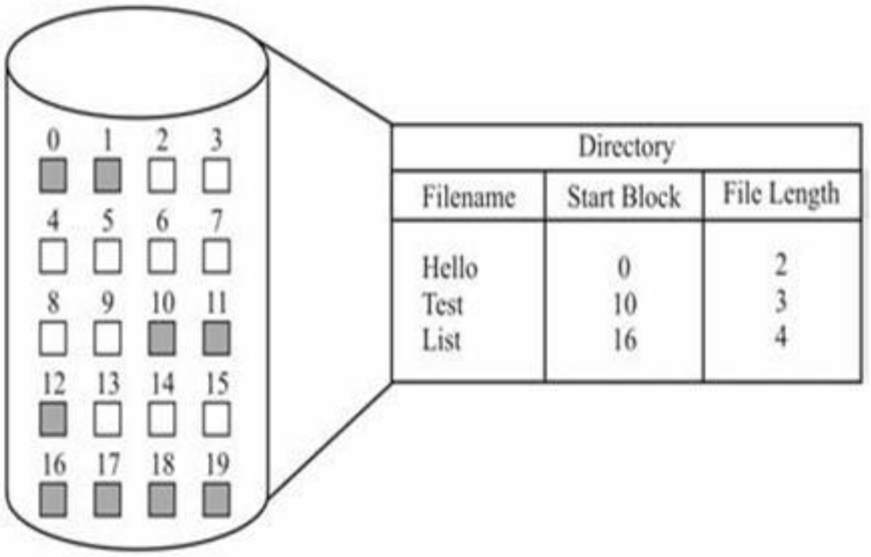
1. Continuous:

In contiguous allocation, files are assigned to contiguous area of secondary storage. A linear ordering of disk addresses is seen on the disk.

The advantage of this approach is that successive logical records are physically adjacent and require no head movement. So disk seek time is minimal and speeds up access of records. Also, this scheme is relatively simple to implement. The technique, in which the operating system provides units of file space on demand by user running processes, is known as dynamic allocation of disk space.

Contiguous allocation merely retains the disk address (start of file) and length (in block units) of the first block. If a file is n blocks long and it begins with location b (blocks), then it occupies b, b+1, b+2,…, b+n-1 blocks. First-fit and best-fit strategies can be used to select a free hole from the available ones. But the major problem here is searching for sufficient space for a new file.

The below diagram depicts a contiguous allocation method.



Directory		
Filename	Start Block	File Length
Hello	0	2
Test	10	3
List	16	4

As shown in the above example, 'Hello' file is stored on disk and the starting block from where the file begins is block number 0 (zero) and the length of the file is 2 blocks long. So, block number 0 and 1 are allocated for the file 'Hello'. Same way allocation for 'Test' and 'List' file will be done.

This diagram exhibits similar fragmentation problems as in variable memory partitioning. This is because the allocation and deal location could result in regions of free disk space broken into chunks (pieces) within active space, which is called external fragmentation.

A user specifies in advance the size of the area needed to hold a file to be created. If the desired amount of contiguous space is not available, the file cannot be created. But it supports both sequential and direct accessing. For sequential access, almost no seeks are required. Even direct access to seek and read is fast. Also, calculation of blocks holding data is quick and easy as we need just offset from the start of the file.

2. Non-Continuous:

This scheme has replaced the previous ones. The popular non-contiguous storages because the files do tend either to grow or shrink over time and users rarely know in advance how large their files will be contiguous.

Storage allocation systems are being replaced by more dynamic non-contiguous storage allocation systems. In this we will study two schemes:

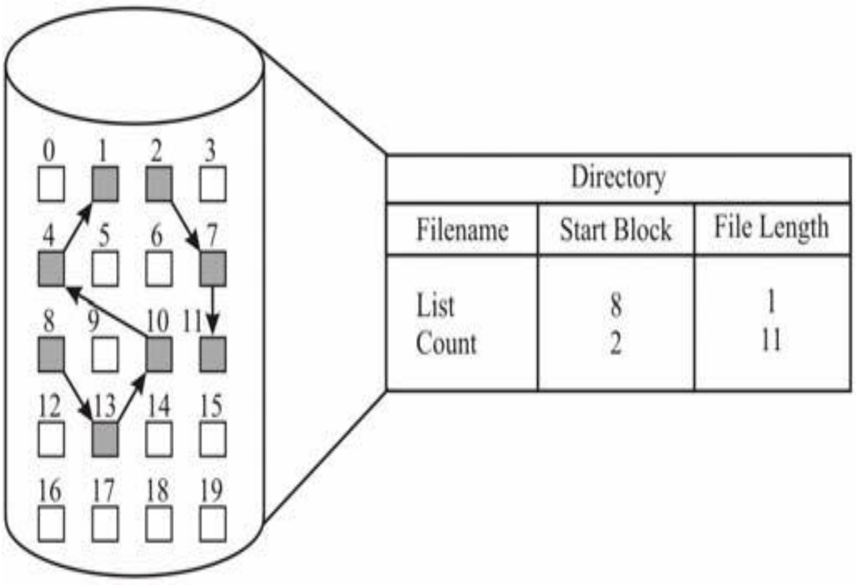
A. Linked/Chained Collection
B. Indexed Allocation

[A]. Linked/Chained Collection:

Linked allocation is essentially a disk-based version of the linked list. The disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last block of the file. Also each block contains pointers to the next block, which are not made available to the user.

It can be used effectively for sequential access only but there also it may generate long seeks between blocks. Another issue is the extra storage space required for pointers. Yet the reliability problem is also there due to loss/damage of any pointer.

The below diagram depicts linked /chained allocation where each block contains the information about the next block.



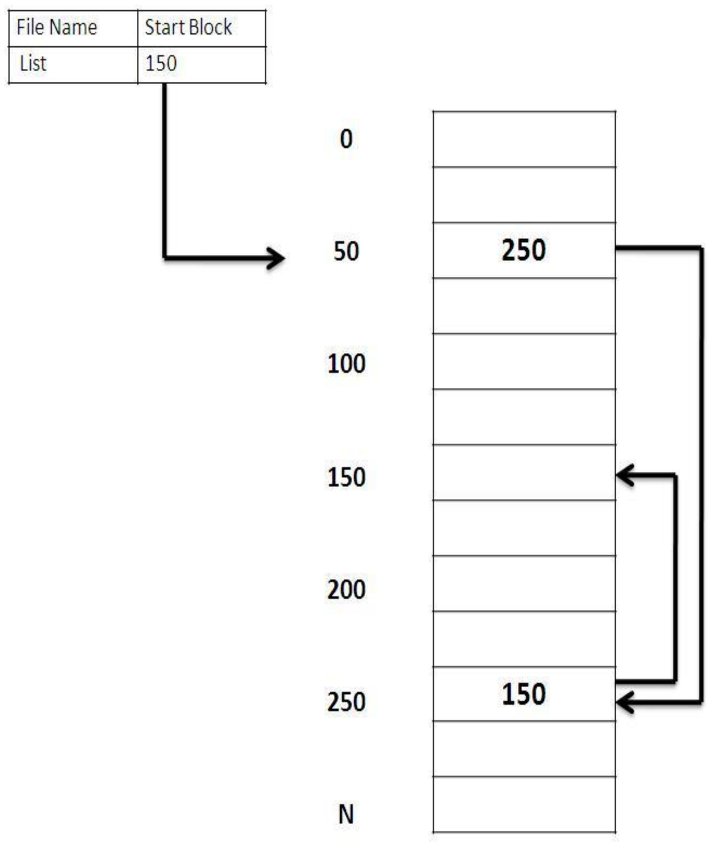
MS-DOS and OS/2 use another variation on linked list called FAT (File Allocation Table). The beginning of each partition contains a table having one entry for each disk block and is indexed by the block number.

The directory entry contains the block number of the first block of the file. The table entry indexed by block number contains the block number of the next block in the file.

The Table pointer of the last block in the file has an EOF pointer value. This chain continues until EOF (end of file) table entry is encountered.

We still have to linearly traverse next pointers, but at least we don't have to go to the disk for each of them. 0 (Zero) table value indicates an unused block. So, allocation of free blocks with FAT scheme is straightforward, just search for the first block with a 0 table pointer. MS-DOS and OS/2 use this scheme. The below figure shows file allocation table **(FAT).**

❖ **Directory:**

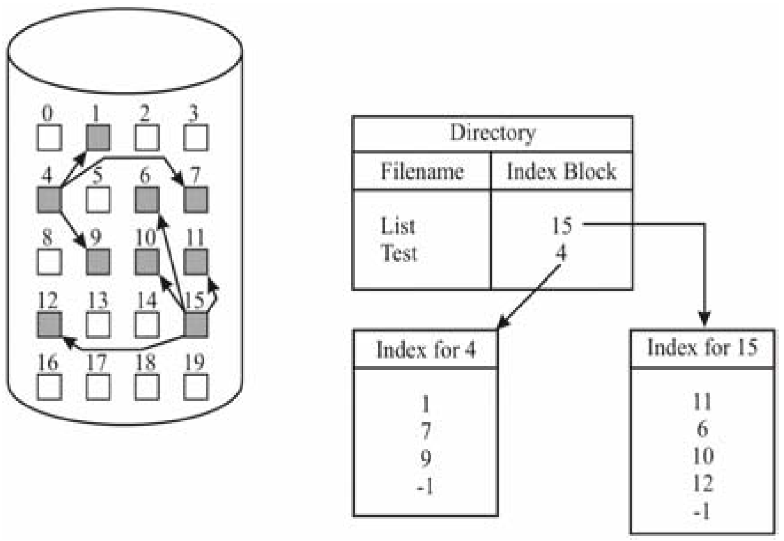


[B]. Indexed Allocation:

Index allocation addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file; the index has one entry for each portion allocated to file.

Typically, the file indexes are not physically stored as part of the file allocation table. Rather, the index for a file is kept in a separate block, and entry for the file in the allocation table points to that block. The allocation may be on the basis of either fixed size blocks or variable size portions.

The indexed allocation scheme is diagrammatically shown in below.



The advantage of this scheme is that it supports both sequential and random access. The searching may take place in index blocks themselves. The index blocks may be kept close together in secondary storage to minimize seek time. Also space is wasted only on the index which is not very large and there's no external fragmentation.

III. Conclusion

A file consists of a collection of records. The way in which these records may be accessed determines its logical organization, and to some extent it's physical Organization on disk. If a file is primary to be processed as a whole, then a sequential file organization is the simplest and most appropriate. If sequential access is needed but random access to individual file is also desired, then an indexed sequential file may give better performance. If access to the file is principally at random, then an indexed file may be the most appropriate.

We have to study the major methods and find that indexed allocation supports both sequential and direct access to the file, and thus is the most popular form of file allocation.

References

Books:

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne – Operating System Concept – 6th Edition JOHN WILEY & SONS, INC.
2. Dan Sydow - Programming the Be Operating System - O'Reilly Publisher.
3. Dr. Arvind Mohan Parashar, Chandresh Shah, Saurab Mishra, Sunilkumar Ojha - Computer Science & Application. Agra: Upkar Prakashan.
4. Pankaj Jalote - An Integrated Approach to Software Engineering – 3rd Edition Narosa Publishing House
5. An Operating Systems, Raphael A. Finkel 2nd Edition - Prentice Hall