International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 9, pp: 275-279 www.ijaem.net ISSN: 2395-5252

# Hidden Strategies in Os File Allocation

AJHAY.V, Dr.M. Sujithra M.C.A, M.Phil., PhD, Dr.A.D. Chitra M.C.A, M.Phil., PhD,

*2nd Year, M.Sc. Software Systems (Integrated), Coimbatore Institute of Technology, Coimbatore.*
*Assistant Professor, Department of Data Science, Coimbatore Institute of Technology, Coimbatore.*
*Assistant Professor, Department of Software Systems, Coimbatore Institute of Technology, Coimbatore.*

Date of Submission: 10-11-2020 Date of Acceptance: 24-11-2020

**ABSTRACT: I**n this Modern world, technology plays a vital role in human life in various means . To be on point its 65% Technology we use and rest goes into the array containing all other elements. And the percentage we use is rapidly increasing every now and then, that's how technology has grown in this Earth. In this book of technology and its growth, there occurred a need for fast processing of stuff as manual processing is slow in many means. Thus humans went onto find a processing machine for rapid processing rate … thus came computers. Computers are designed in such a way, to process stuff billions and million times quickly than manual processing … that too had some evolutions overtime to increase the processing rate . It was then sound and sutle after the invention of Computers. There came a query how to store processed files and how to allocate it into the computers just like humans store their food in a vessel and in a refrigerator. Then there arose the idea of storing files in the operating system. It's always been a **"Part of Parcel"** , when you comeup with something new , there will always be someone who comes as the criticizer , Similarly the file allocation methods has also cameup with some Pros and Cons and so finding some strategies and algorithms to eradicate the cons became the next mission for the humans.

## I. INTRODUCTION

File allocation method is really important and mandatory process required for storing a file successfully. The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods. The main idea behind these methods is to provide: Efficient disk space utilization.

Before leaning on to the Topic lets know some basic terms which we need to know:

- **A file** is a collection of data stored in one unit, identified by a filename. It can be a document, picture, audio or video stream, data library, application, or other collection of data.
- **An operating system (OS)** is system software that manages computer hardware, software resources, and provides common services for computer programs.
- **Allocation** is defined as the act of being portioned out for a certain reason.
- A **Disk** is a hard or floppy round, flat, and magnetic platter capable of having information read from and written to it. The most commonly found disks with a computer are the hard disks and floppy disks .
- The allocation methods define how the files are stored in the disk blocks. To make efficient use of the disk space **File allocation** techniques are stored.
- **File Structures** is the Organization of Data in Secondary Storage Device in such a way that minimize the access time and the storage space. A FileStructure is a combination of representations for data in files and of operations for accessing the data. A File Structure allows applications to read, write and modify data.
- **OS is responsible** for allocation of files and making best use of disk space.

## II. PROBLEM SPECIFICATION

We face difficulties in storing and retrieving the files we use if not allocated space for storing in the disk , Files can get corrupted if not stored in disk . No effective disk space is managed, Computer processing becomes meaningless if files are not stored for longer time as computers are involved in computations of files . Files can't be accessed fast , as we don't know the exact location of where the file is stored.

## III. IDEA BEHIND FILE ALLOCATION

- Efficient disk space utilization.
- Fast access to the file blocks.

## IV. TYPES

The files are allocated space through one of the three methods listed below:

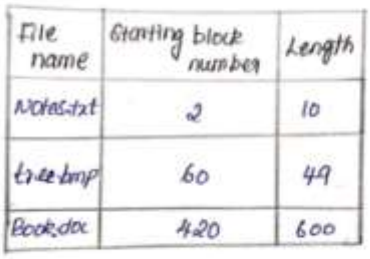
- Contiguous allocation
- Linked allocation
- Indexed allocation

### 1. Contiguous Allocation

In this scheme, each file occupies a contiguous set of blocks on the disk.

The entire available space is divided into blocks .In contiguous allocation,a file occupies contiguous blocks of memory. A file, to which the memory is allocated contiguously, can be very easily accessed either by sequential access method or random access method.

The advantage of this method is that a little information is required to keep track of a file's details. Only the starting block number and the length are required to work with a file. Below image gives a rough idea about contiguous allocation method.



| File name | Starting block number | Length |
|---|---|---|
| Notes.txt | 2 | 10 |
| tree.bmp | 60 | 49 |
| Book.doc | 420 | 600 |

Fig1. Rough idea

The major problem with contiguous allocation is external fragmentation: Although compaction can eliminate this problem, the time taken for compacting the available free blocks of memory is very high.

The other problem with this algorithm is determining how much space a file needs. It is very difficult to estimate before the file is created, the memory requirement of it.

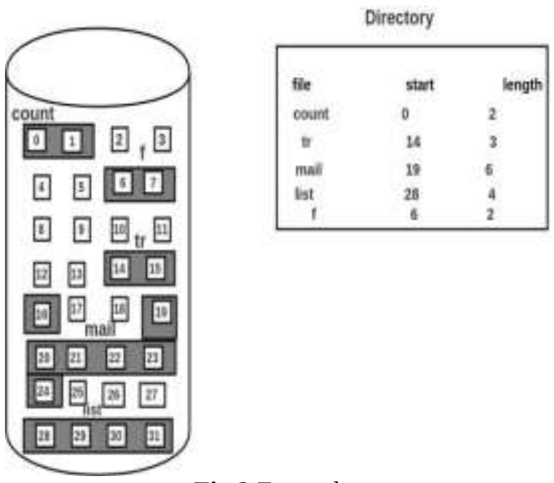
Whatever may be the algorithm we use it is almost impossible to allocate exactly the same amount of memory required by a file. The space assigned will be either more or less than what is actually required by the file. If the memory allocated is not sufficient for a file the user program is terminated mentioning the inadequacy of memory. To avoid this problem a file is always allocated more memory than what it requires. And, this trade-off sometimes leads to internal fragmentation, of course.

**For example,** if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: b, b+1, b+2,……b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.

The directory entry for a file with contiguous allocation contains

- Address of starting block
- Length of the allocated portion.

The file 'mail' in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.



| file | start | length |
|---|---|---|
| count | 0 | 2 |
| tr | 14 | 3 |
| mail | 19 | 6 |
| list | 28 | 4 |
| f | 6 | 2 |

**Fig 2.**Example

**Advantages:**

- Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the Kth block of the file which starts at block b can easily be obtained as (b+k).
- This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

**Disadvantages:**

- This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

## 2. Linked List Allocation

This algorithm solves the problems associated with contiguous allocation. Linked allocation maintains the file as a linked list of memory blocks, which need not be contiguous. The file starts at a particular location in the memory and ends somewhere else. The directory contains a pointer to the first and last blocks of the file. Since any free block can be assigned to a file, the problem of external fragmentation does not arise with linked allocation.

However, linked allocation also offers a problem. This method can be used only with sequential access. It does not work effectively with random access.

The pointers also pose some problems. A considerable percentage of memory will be utilized only by these pointers. Also, if a particular pointer is lost because of some problems in the system and if the operating system picks up a wrong pointer many errors are likely to occur.

However, these problems are not very serious and can be solved by taking some precautionary measures like collecting the memory blocks into clusters. Below image illustrates linked allocation method.

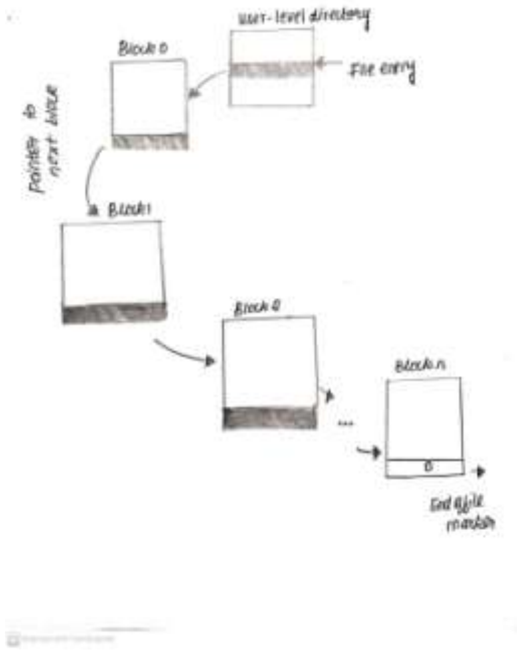


**Fig 3**. Rough idea

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 a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

The file 'jeep' in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.

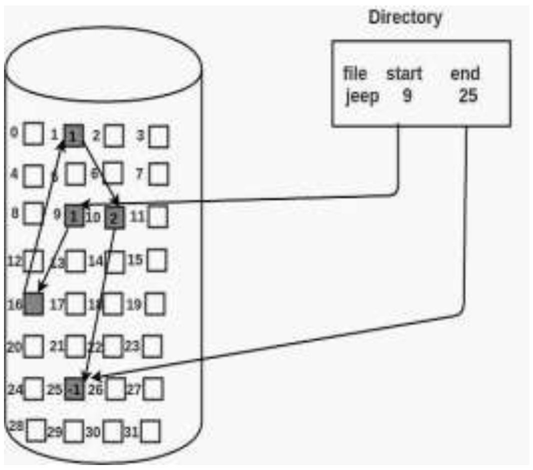


**Fig4**. Example

**Advantages:**

- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

**Disadvantages:**

- Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
- It does not support random or direct access. We cannot directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.
- Pointers required in the linked allocation incur some extra overhead

## 3. Indexed Allocation

In indexed allocation, all the pointers are brought together into one block known as the index block.

Each file contains an index block of its own, which maintains all the addresses of its disk blocks. The main block of the directory, in which a file is present,Contains the address of the file's index block.

Indexed allocation supports direct or random allocation. However, an indexed allocation is more complex than the previous two allocation algorithms. Below image illustrates indexed allocation.

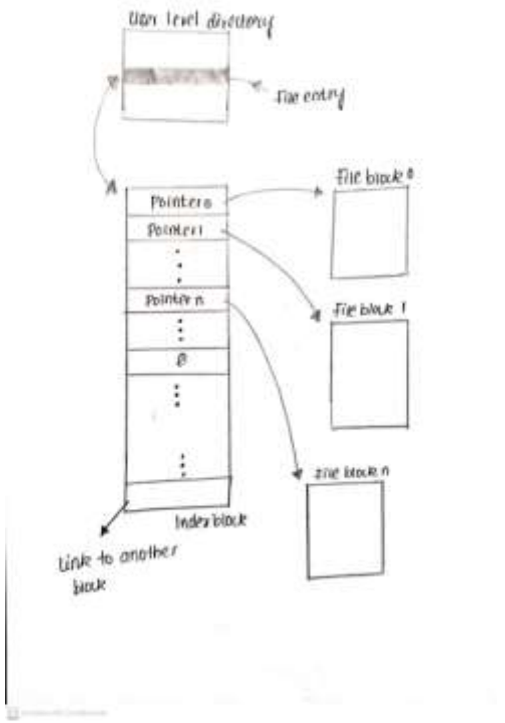


**Fig 5.**Rough idea

In this scheme, a special block known as the **Index block** contains the pointers to all the blocks occupied by a file. Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block as shown in the image:

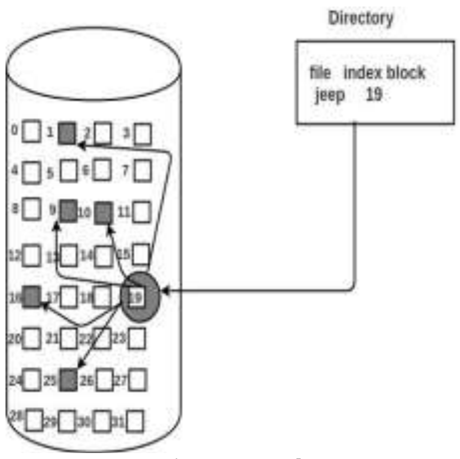


**Fig 6.**Example

**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 linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

For files that are very large, single index block may not be able to hold all the pointers.

## V. MECHANISMS USED TO RESOLVE ALLOCATION OF LARGE FILE ISSUE

1. **Linked scheme:** This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block.
2. **Multilevel index:** In this policy, a first level index block is used to point to the second level index blocks which inturn points to the disk blocks occupied by the file. This can be extended to 3 or more levels depending on the maximum file size.
3. **Combined Scheme:** In this scheme, a special block called the **Inode (information Node)** contains all the information about the file such as the name, size, authority, etc… and the remaining space of Inode is used to store the Disk Block addresses which contain the actual file as shown in the image below. The first few of these pointers in Inode point to the **direct blocks** i.e the pointers contain the addresses of the disk blocks that contain data of the file. The next few pointers point to indirect blocks. Indirect blocks may be single indirect, double indirect or triple indirect. **Single Indirect block** is the disk block that does not contain the file data but the disk address of the blocks that contain the file data. Similarly, **double indirect blocks** do not contain the file data but the disk address of the blocks that contain the address of the blocks containing the file data.

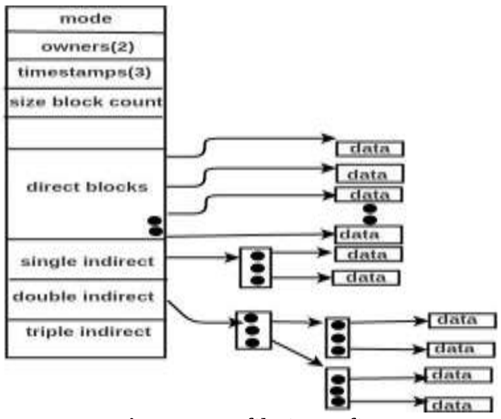
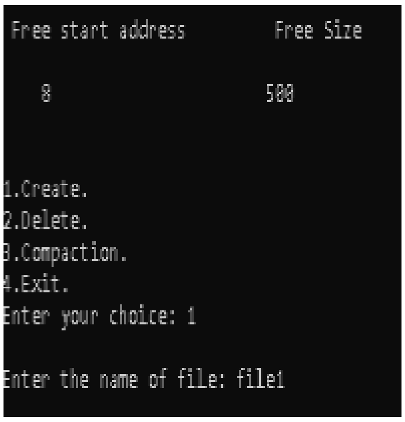


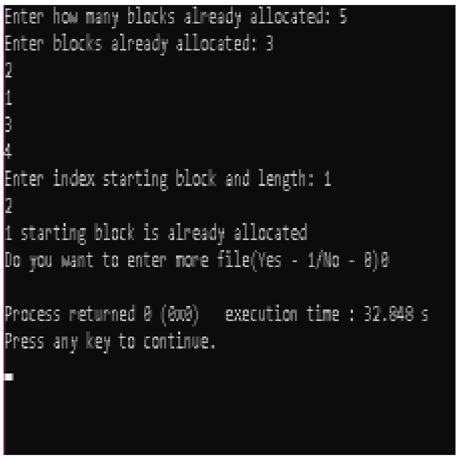
**Fig 7**.Large file issue fix

## VI.RESULTS FROM REAL TIME SIMULATIONS



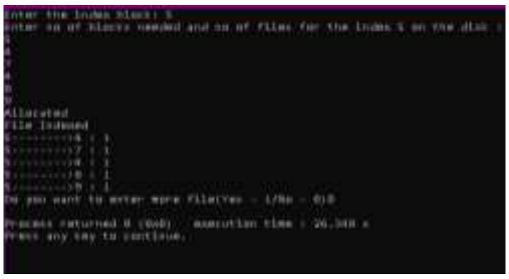
```
Free start address          Free Size

   0                          500



1.Create.
2.Delete.
3.Compaction.
4.Exit.
Enter your choice: 1

Enter the name of file: file1
```

**i)Contagious file allocation**



```
Enter how many blocks already allocated: 5
Enter blocks already allocated: 3
2
1
3
4
Enter index starting block and length: 1
2
1 starting block is already allocated
Do you want to enter more file(Yes - 1/No - 0)0

Process returned 0 (0x0)   execution time : 32.848 s
Press any key to continue.
```

**ii) Linkedlist file allocation**



```
Enter the index block: 5
Enter no of blocks needed and no of files for the index 5 on the disk :
5
6
7
[illegible]
8
9
Allocated
File Indexed
5-------->6 : 1
5-------->7 : 1
5-------->10 : 1
5-------->8 : 1
5-------->9 : 1
Do you want to enter more file(Yes - 1/No - 0)0

Process returned 0 (0x0)   execution time : 26.348 s
Press any key to continue.
```

**iii) Indexed file allocation**

SUMMARY

| | **Contiguous** | **Chained** | **Indexed** | |
|---|---|---|---|---|
| **Preallocation?** | Necessary | Possible | Possible | |
| **Fixed or variable size portions?** | Variable | Fixed blocks | Fixed blocks | Variable |
| **Portion size** | Large | Small | Small | Medium |
| **Allocation frequency** | Once | Low to high | High | Low |
| **Time to allocate** | Medium | Long | Short | Medium |
| **File allocation table size** | One entry | One entry | Large | Medium |

## VII. CONCLUSION

"By observed real time Simulations of all three file allocation methods , all three file allocation strategies has its own importance" nothing has an edge over the other.

From the file allocation methods desired solutions regarding allocating disk space for file were met , also solutions for undeisred results that result as a bi-product were also cleared off . Thus we came to know the importance of file allocation in the operating system.

## REFERENCES

i) www.faceprep.in
ii) www.geeksforgeeks.org
iii) https://en.wikipedia.org
iv) www.techterms.com
v) www.ijcsit.com
vi)Williamstallings – operating systems – internals and design principals 9th edition**.**
vii)Principles of Operating Systems: Design & Applications By Brian L. Stuart