Module5_Part1

OPERATING SYSTEMS

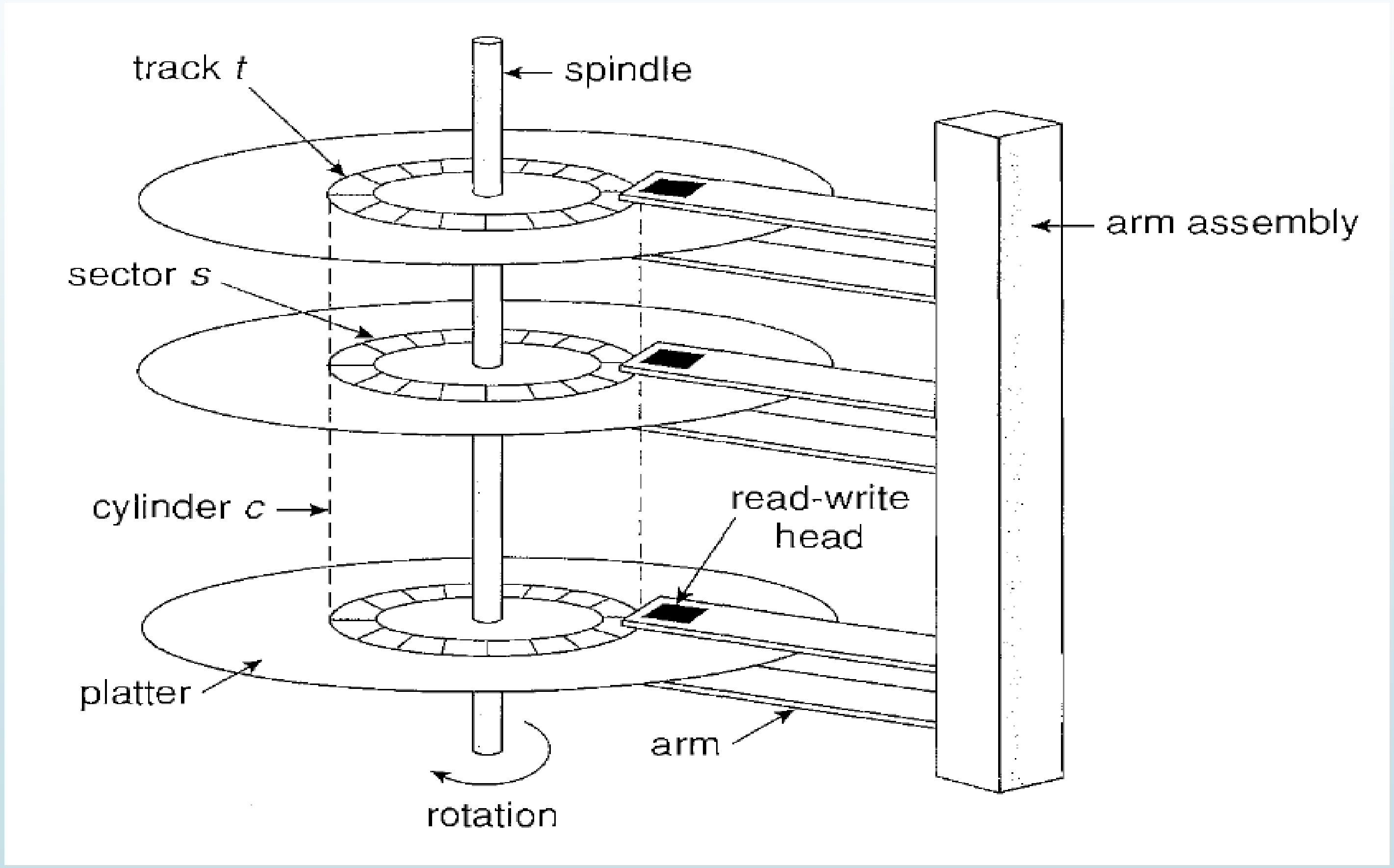
Textbook : Operating Systems Concepts by Silberschatz

Magnetic disks

- A magnetic disk is a storage device that uses a magnetization process to write, rewrite and access data.
- Each disk platter has a flat circular shape, like a CD. Common platter diameters range from 1.8 to 5.25 inches. The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters.
- A read -write head "flies" just above each surface of every platter.
- The surface of a platter is logically divided into circular tracks which are subdivided into sectors
- The heads are attached to a disk arm that moves all the heads as a unit.
- The set of tracks that are at one arm position makes up a cylinder. There may be thousands of concentric cylinders in a disk drive,
- Each track may contain hundreds of sectors. The storage capacity of common disk drives

  is measured in gigabytes.

Moving head disk mechanism



Magnetic disks

- When the disk is in use, a drive motor spins it at high speed. Most drives

  rotate 60 to 200 times per second.
- Disk speed has two parts.

  The transfer rate is the rate at which data flow between the drive and the computer.

  The positioning time sometimes called the **random access time** consists of the time

  necessary to move the disk arm to the desired cylinder, called the **seek time**.

  The time necessary for the desired sector to rotate to the disk head, called **the rotational latency.**
- Typical disks can transfer several megabytes of data per second, and they have seek times and rotational latencies of several milliseconds.
- A disk can be removable, allowing different disks to be mounted as needed.

Solid state Disk(SSD)

- Like a hard drive, an SSD is used **to store large volumes of data whether the system is on or off**, for extended periods of time. But unlike hard drives, an SSD has no moving parts,
- SSD incorporates the storage technique implemented in microchip-based flash memory, where data is electronically stored on flash memory chips. An SSD is an entirely electronic storage device, and its physical assembly contains no mechanical objects.
- SSDs in general are more reliable than HDDs, which again is a function of having no moving parts.
- Significantly faster data transfer rates
- SSDs are more durable than HDDs

Disk structure

- Disk is usually divided into sectors, tracks and cylinders.
- Modern disk drives are addressed as large one-dimensional arrays of logical blocks

  where the logical block is the smallest unit of transfer.
- The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially.
- Sector 0 is the first sector of the first track on the outermost cylinder. The mapping proceeds in order through that track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.
- convert a logical block number into a disk address that consists of a cylinder number, a track number within that cylinder, and a sector number within that track.

Disk formatting

- A new magnetic disk is a blank slate: it is just a platter of a magnetic recording material. Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low level formatting or physical formatting.

  low level formatting is used to initiate a hard drive and prepare it for data by creating the actual sectors and tracks on the drive.
- Low-level formatting fills the disk with a special data structure for each sector.
- The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size), and a trailer.
- The header and trailer contain information used by the disk controller, such as a sector number and an Error Correcting Code (ECC)

Disk formatting

- When the controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area.
- When the sector is read, the ECC is recalculated and compared with the stored value. If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad.
- Most hard disks are low-level-formatted at the factory as a part of the manufacturing process.
- Before it can use a disk to hold files, the operating system still needs to record its own data structures on the disk. It does so in two steps.

  1. patition the disk into one or more groups of cylinders. The operating system can treat each partition as though it were a separate disk. For instance, one partition can hold a copy of the operating system's executable code, while another holds user files.

Disk formatting

2. The second step is logical formatting or creation of a file system. In this step, the operating system stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space (a FAT or inodes) and an initial empty directory.

Module4_Part11

OPERATING SYSTEMS

Textbook : Operating Systems Concepts by Silberschatz

Directory implementation

- directory-allocation and directory-management
- directory-allocation and directory-management algorithms

Linear List

- The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks.
- This method is simple to program but time-consuming to execute.
- To create a new file, we must first search the directory to be sure that no existing file has the same name. Then, we add a new entry at the end of the directory.
- To delete a file, we search the directory for the named file and then release the space allocated to it.

Directory implementation

- The real disadvantage of a linear list of directory entries is that finding a file requires a linear search
- many operating systems implement a software cache to store the most recently used directory information. A cache hit avoids the need to constantly reread the information from disk.
- Hash Table
- Another data structure used for a file directory is a With this method, a linear list stores the directory entries, but a hash data structure is also used
- The hash table takes a value computed from the file name and returns
- a pointer to the file name in the linear list. Therefore, it can greatly decrease the
- directory search time

File allocation methods

- In almost every case, many files are stored on the same disk.
- The main problem is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly.
- Three major methods of allocating disk space are in wide use:

  contiguous,

  linked,

  indexed.

Contiguous Allocation

- Contiguous allocation requires that each file occupy a set of contiguous blocks

  on the disk.
- With this ordering, assuming that only one job is accessing the disk, accessing block *b* +1 after block *b* normally requires no head movement.
- When head movement is needed (from the last sector of one cylinder to the first sector of the next cylinder), the head need only move from one track to the next. Thus, the number

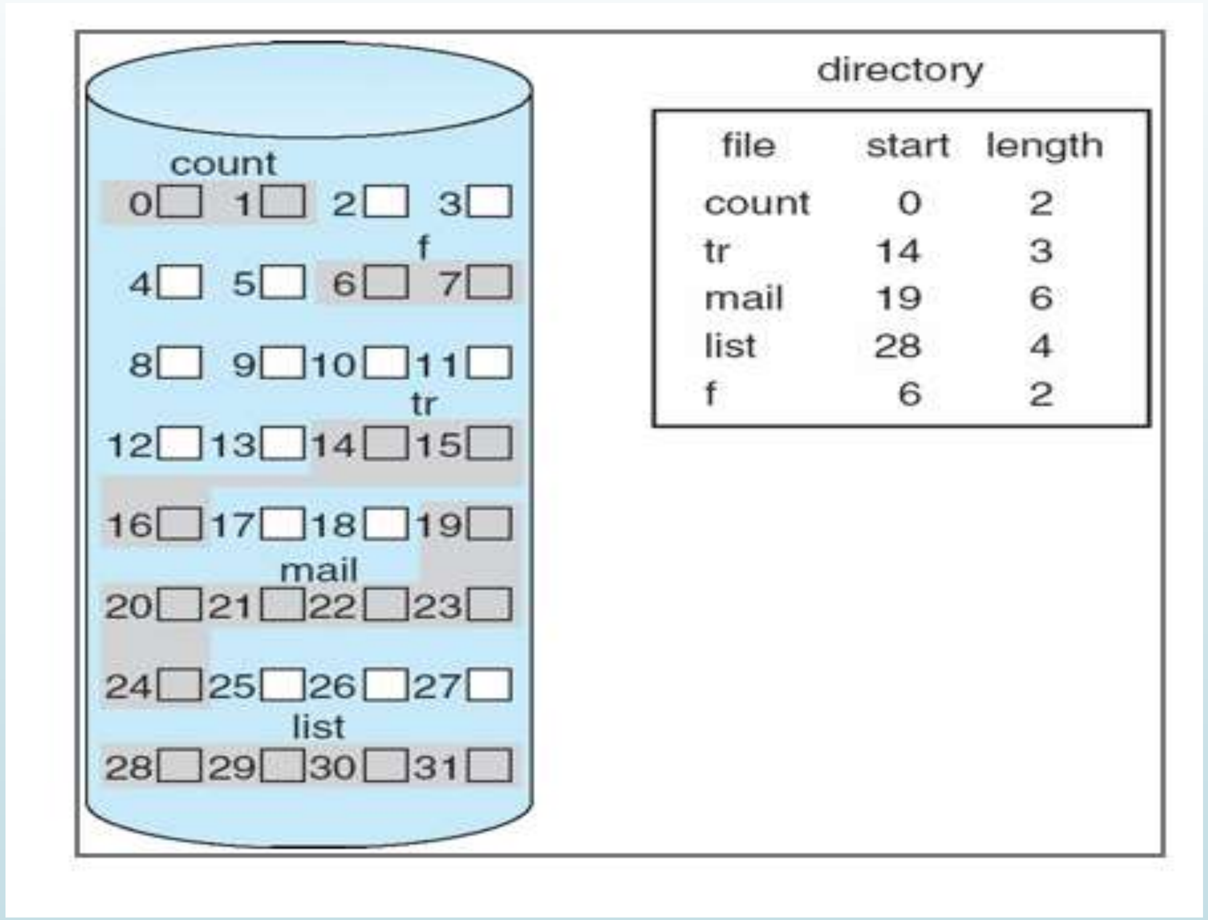
  of disk seeks required for accessing contiguously allocated files is minimal
- For **sequential access**, the file system remembers the disk address of the last block

  referenced and, when necessary, reads the next block. For **direct access** to block *i* of a

  file that starts at block *b*, we can immediately access block *b* + *i*. Thus, both sequential

  and direct access can be supported by contiguous allocation

Contiguous Allocation



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

- Contiguous allocation has some problems

  finding space for a new file.?
- The contiguous-allocation problem can be seen as a particular application

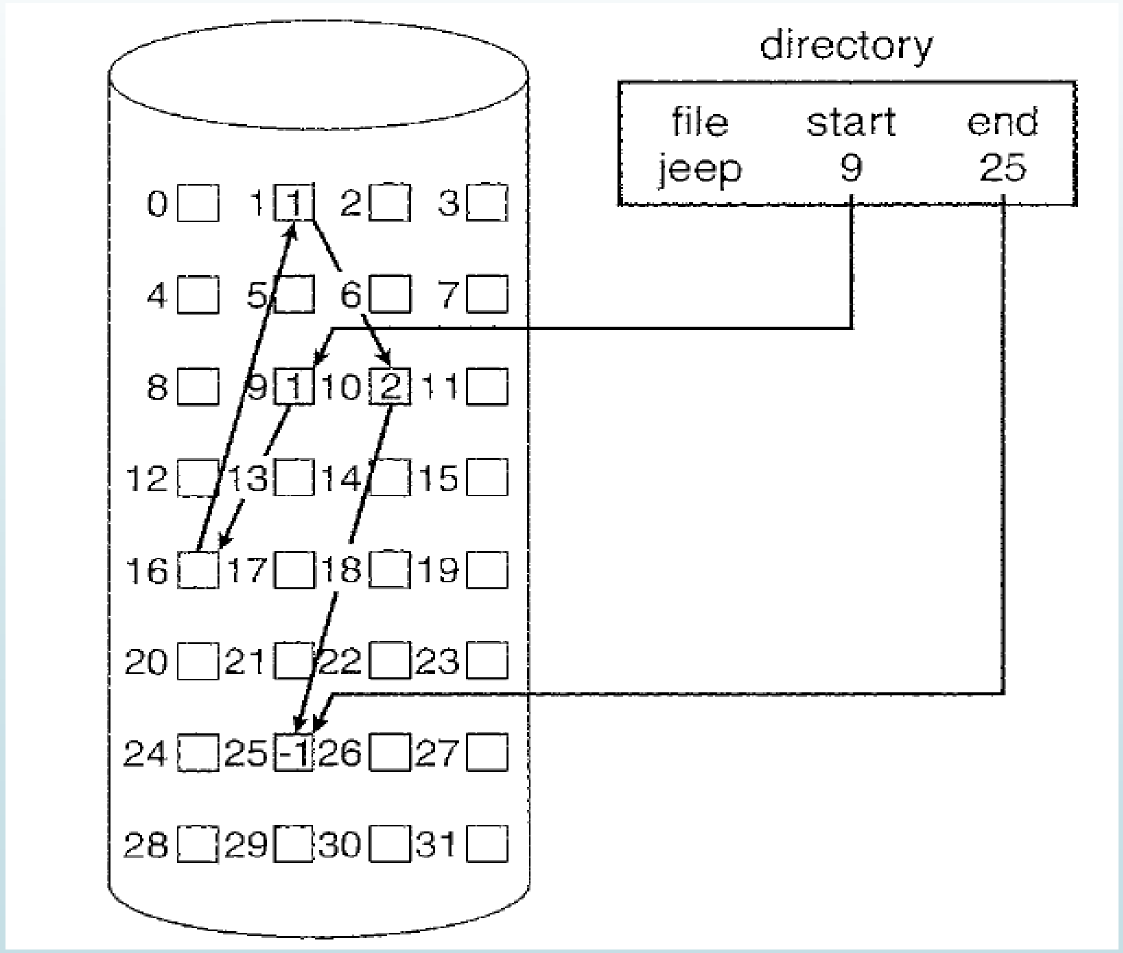
  of the general dynamic storage allocation problem
- First fit and best fit are the most common strategies used to select a free hole from

  the set of available holes.
- External fragmentation is another problem

Linked 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. For example, a file of five blocks might start at block 9 and continue at block 16, then block 1, then block 10, and finally block 25.
- Each block contains a pointer to the next block.


directory
file
start
end
jeep
9
25
0
1
1
2
3
4
5
6
7
8
9
1
10
2
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
-1
26
27
28
29
30
31

- To create a new file, we simply create a new entry in the directory.
- With linked allocation, each directory entry has a pointer to the first disk block of the file.
- This pointer is initialized to *nil* (the end-of-list pointer value) to signify an empty file.
- The size field is also set to 0.
- A write to the file causes the free-space management system to find a free block, and this new block is written to and is linked to the end of the file.
- To read a file, we simply read blocks by following the pointers from block to block.
- There is no external fragmentation with linked allocation, and any free block on the free-space list can be used to satisfy a request.
- The size of a file need not be declared when that file is created.
- A file can continue to grow as long as free blocks are available.
- Consequently, it is never necessary to compact disk space.

- Linked allocation disadvantages:
- Sequential access: To find the ith block of a file, we must start at the begining of that file and follow the pointers until we get to the ith block.
- The space required for the pointers: space required consists of space for file data as well as pointers for each block

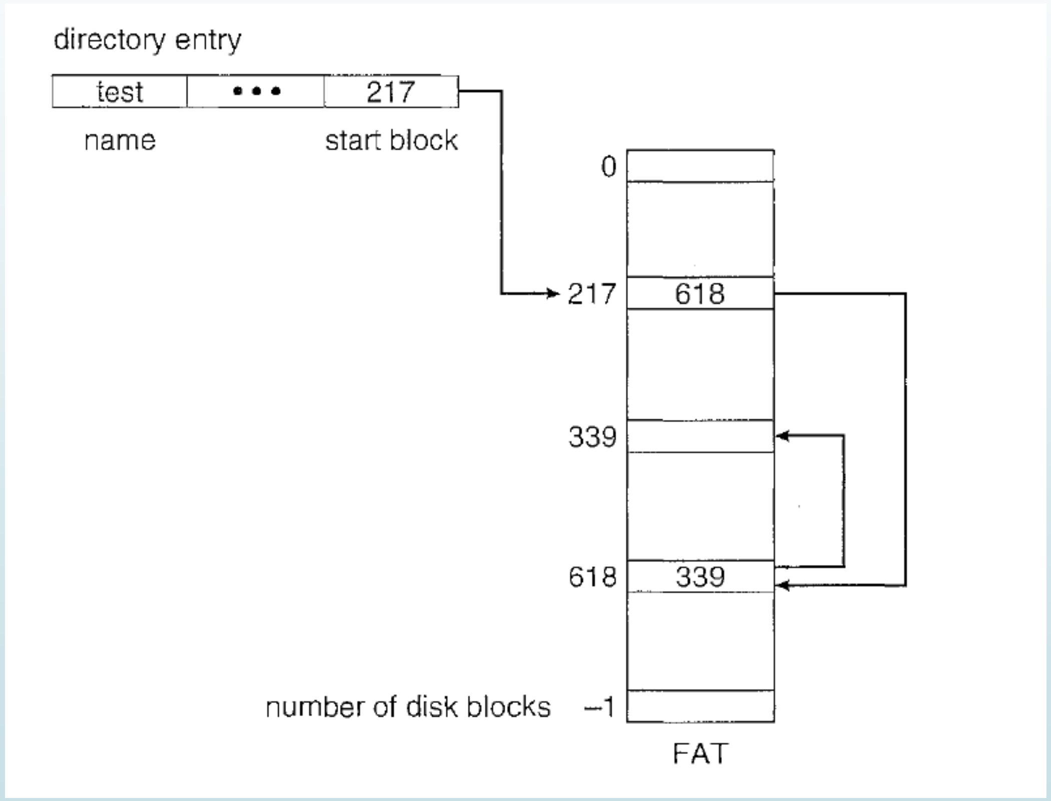
  solution to this problem is to collect blocks into multiples, called clusters and to allocate clusters rather than blocks. For instance, the file system may define a cluster as four blocks and operate on the disk only in cluster units
- Reliability: Recall that the files are linked together by pointers scattered all over the disk, and consider what would happen if a pointer were lost or damaged.

Variation on linked allocation :use of File Allocation Table (FAT)

- This simple but efficient method
- A section of disk at the beginning of each volume is set aside to contain the table.
- The table has one entry for each disk block and is indexed by block number
- The FAT is used in much the same way as a linked list.
- The directory entry contains the block number of the first block of the file.
- The table entry indexed by that block number contains the block number of the next block in the file.
- This chain continues until it reaches the last block, which has a special end-of-file value as the table entry.

- An unused block is indicated by a table value of 0. Allocating a new block to
  a file is a simple matter of finding the first 0-valued table entry and replacing
  the previous end-of-file value with the address of the new block. The 0 is then
  replaced with the end-of-file value.

Variation on linked allocation :use of File Allocation Table (FAT)

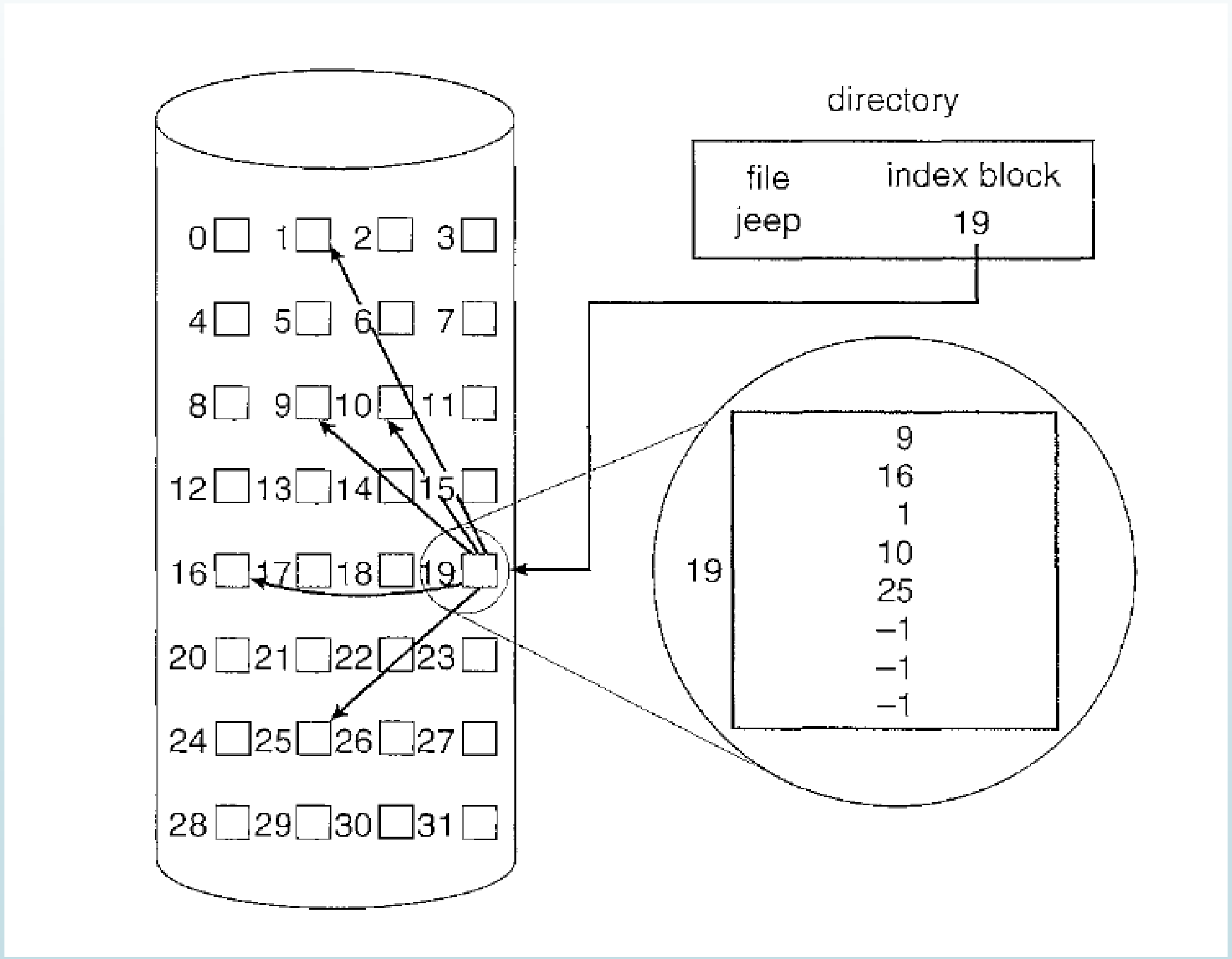


FAT

Indexed Allocation

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

- When the file is created, all pointers in the index block are set to *nil.*
- When the ith block is first written, a block is obtained from the free-space manager and its address is put in the ith index-block entry.



directory
file
index block
jeep
19
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
19
9
16
1
10
25
–1
–1
–1

Module4_Part11

OPERATING SYSTEMS

Textbook : Operating Systems Concepts by Silberschatz

Directory implementation

- directory-allocation and directory-management
- directory-allocation and directory-management algorithms

1.Linear List

- The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks.
- This method is simple to program but time-consuming to execute.
- To create a new file, we must first search the directory to be sure that no existing file has the same name. Then, we add a new entry at the end of the directory.
- To delete a file, we search the directory for the named file and then release the space allocated to it.

Directory implementation

- The real disadvantage of a linear list of directory entries is that finding a file requires a linear search
- many operating systems implement a software cache to store the most recently used directory information. A cache hit avoids the need to constantly reread the information from disk.
- 2.Hash Table
- Another data structure used for a file directory is a With this method, a linear list stores the directory entries, but a hash data structure is also used
- The hash table takes a value computed from the file name and returns
- a pointer to the file name in the linear list. Therefore, it can greatly decrease the
- directory search time