Operating Systems

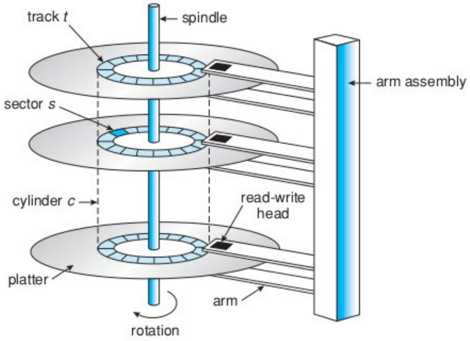
Storage

Carlos Alberto Llano R.

12 de noviembre de 2015

Contents

Overview - Mass-Storage Structure



Overview - Mass-Storage Structure

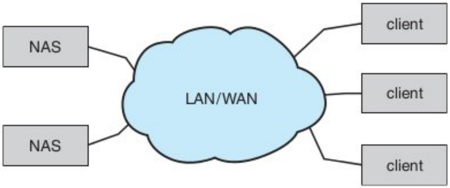
The transfer rate	is the rate at which data flow between the drive and the computer.
The seek time	is the time necessary to move the disk arm to the desired cylinder.
rotational latency	is the time necessary for the desired sector to rotate to the disk head.
bandwidth	is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Overview - Solid-State Disks (SSDs)

- They have no moving parts and faster because they have no seek time or latency.
- They consume less power. However, they are more expensive.
- They have less capacity than the larger hard disks.
- They may have shorter life spans than hard disks.
- They have high performance.
- They are used in some laptop computers to make them smaller, faster, and more energy-efficient.

Disk Attachment - Network-Attached Storage (NAS)

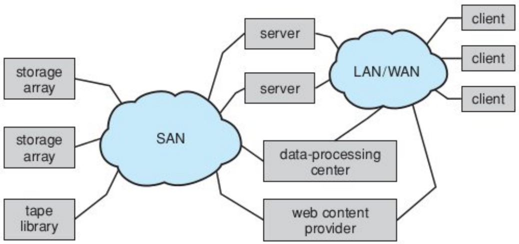
The NAS device is a special-purpose storage system that is accessed remotely over a data network.



Clients access network-attached storage via a remote-procedure-call.
The remote procedure calls (RPCs) are carried via TCP or UDP over an IP network—usually the same local- area network (LAN) that carries all data traffic to the clients.

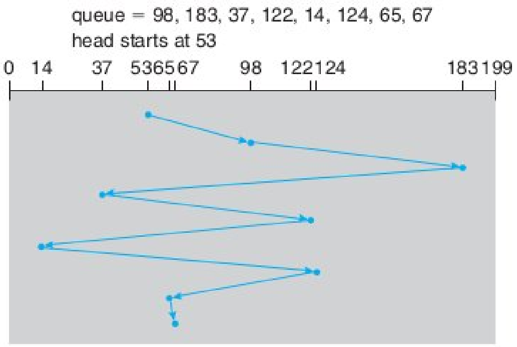
Disk Attachment - Storage-Area Network (SAN)

A storage-area network (SAN) is a private network (using storage protocols rather than networking protocols) connecting servers and storage units.



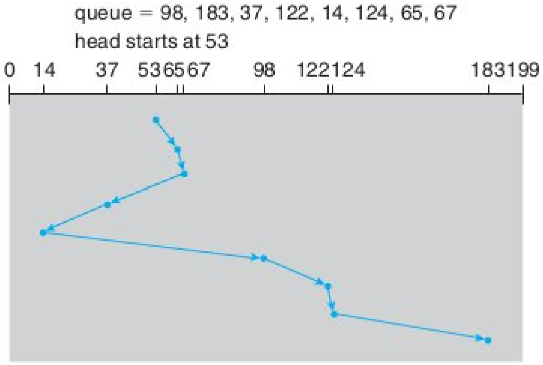
Disk Scheduling - FCFS Scheduling

The simplest form of disk scheduling is, of course, the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service.



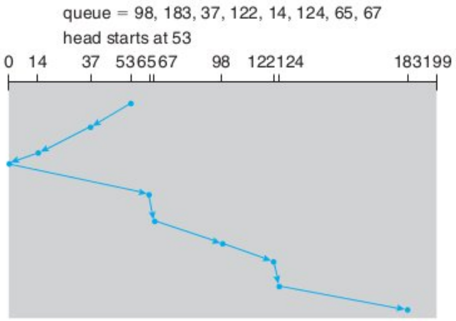
Disk Scheduling - SSTF Scheduling

The shortest-seek-time-first (SSTF) algorithm selects the request with the least seek time from the current head position. In other words, SSTF chooses the pending request closest to the current head position.



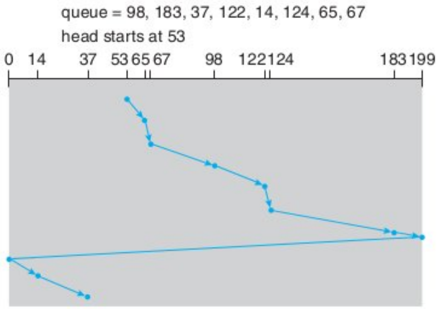
Disk Scheduling - Scan (or Elevator) Scheduling

In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues.



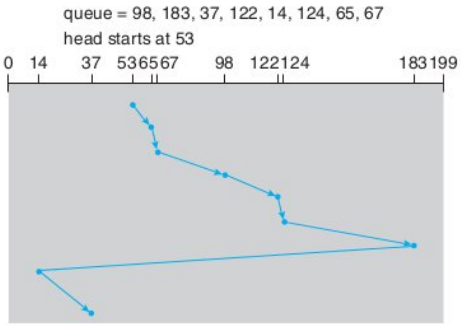
Disk Scheduling - Circular SCAN (C-SCAN) scheduling

is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip.



Disk Scheduling - LOOK Scheduling

The arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to the end of the disk. Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they look for a request before continuing to move in a given direction.



Overview

Several on-disk and in-memory structures are used to implement a file system.

These structures vary depending on the operating system and the file system, but some general principles apply.

Overview

On disk, the file system may contain information about how to boot an operating system stored there, the total number of blocks, the number and location of free blocks, the directory structure, and individual files.

Overview - Structures on disk

A boot control block (per volume) can contain information needed by the system to boot an operating system from that volume. If the disk does not contain an operating system, this block can be empty. It is typically the first block of a volume. it is called the boot block or the partition boot sector.

Overview - Structures on disk

A volume control block (per volume) contains volume (or partition) details, such as the number of blocks in the partition, the size of the blocks, a free-block count and free-block pointers. It is stored in the master file table.

Overview - Structures on disk

A directory structure (per file system) is used to organize the files.

A per-file FCB contains many details about the file. It has a unique identifier number to allow association with a directory entry.

In some filesystems, this information is stored within the master file table, which uses a relational database structure, with a row per file.

Overview - Structures on memory

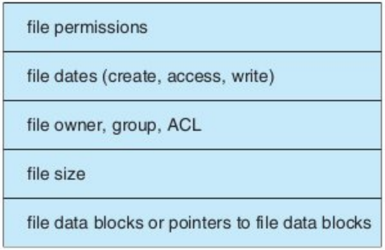
The in-memory information is used for both file-system management and performance improvement via caching.

The data are loaded at mount time, updated during file-system operations, and discarded at dismount.

Overview - Structures on memory

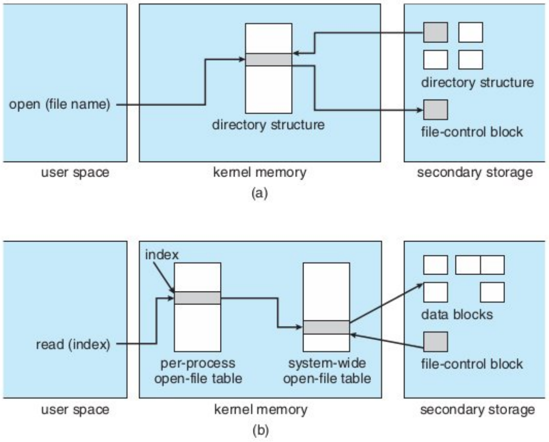
- **mount table** contains information about each mounted volume.
- **directory-structure cache** holds the directory information of recently accessed directories.
- **The system-wide open-file table** contains a copy of the FCB of each open file.
- **The per-process open-file table** contains a pointer to the appropriate entry in the system-wide open-file table.
- **Buffers** hold file-system blocks when they are being read from disk or written to disk.

Overview - FCB (File Control Block)



file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

Overview - file-system implementation are summary



Overview - file-system implementation are summary - open

1. To create a new file, an application program calls the logical file system.
2. The logical file system knows the format of the directory structures.
3. To create a new file, it allocates a new FCB.
4. The system then reads the appropriate directory into memory updates it with the new file name and FCB.
5. and writes it back to the disk.

Overview - file-system implementation are summary - read

1. It must be opened. The open() call passes a file name to the logical file system.
2. The open() system call first searches the system-wide open-file table to see if the file is already in use by another process.
3. If it is, a per-process open-file table entry is created pointing to the existing system-wide open-file table.
4. If the file is not already open, the directory structure is searched for the given file name.
5. Once the file is found, the FCB is copied into a system-wide open-file table in memory.
6. This table not only stores the FCB but also tracks the number of processes that have the file open.

Overview - file-system implementation are summary - read

7. Next, an entry is made in the per-process open-file table, with a pointer to the entry in the system-wide open-file table and some other fields.
8. When a process closes the file, the per-process table entry is removed, and the system-wide entry's open count is decremented.

Linear list

The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks.

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.

Linear list

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.

A sorted list allows a binary search and decreases the average search time.

A more sophisticated tree data structure, such as a balanced tree, might help here.

Hash table

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.

The major difficulties with a hash table are its generally fixed size and the dependence of the hash function on that size.

Contiguous Allocation

Three major methods of allocating disk space are in wide use: contiguous, linked, and indexed.

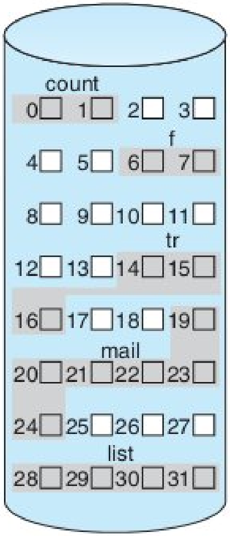
Each method has advantages and disadvantages. Although some systems support all three, it is more common for a system to use one method for all files within a file-system type.

Contiguous Allocation

Contiguous allocation requires that each file occupy a set of contiguous blocks on the disk.

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.

Contiguous Allocation



directory

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

Contiguous Allocation - problems

1. One difficulty is finding space for a new file.
2. Another problem with contiguous allocation is determining how much space is needed for a file.

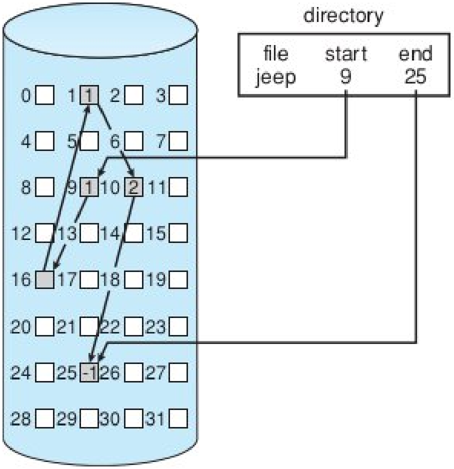
Linked Allocation

Linked allocation solves all problems of contiguous 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.

Each block contains a pointer to the next block.

Linked Allocation

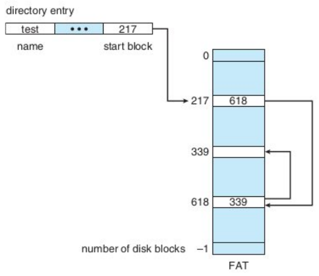


Linked Allocation - problems

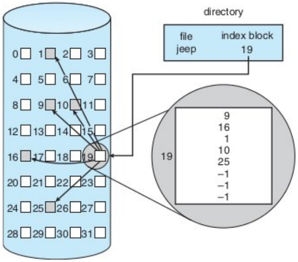
1. Each access to a pointer requires a disk read, and some require a disk seek.
2. Another disadvantage is the space required for the pointers.
   - If a pointer requires 4 bytes out of a 512-byte block, then 0.78 percent of the disk is being used for pointers.

Linked Allocation - File Allocation Table (FAT)

This simple but efficient method of disk-space allocation was used by the MS-DOS OS. 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. FAT would be cached.



Indexed Allocation



Free-Space Management

To keep track of free disk space, the system maintains a free-space list. The free-space list records all free disk blocks—those not allocated to some file or directory.

To create a file, we search the free-space list for the required amount of space and allocate that space to the new file.

This space is then removed from the free-space list.

When a file is deleted, its disk space is added to the free-space list.

Bit Vector

The free-space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

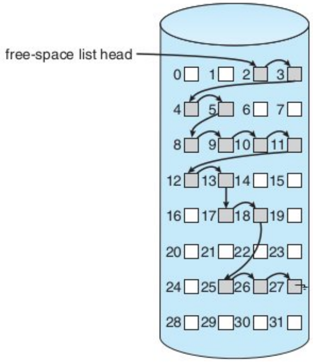
For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free and the rest of the blocks are allocated.

The free-space bit map would be
001111001111110001100000011100000 ...

The main advantage of this approach is its relative simplicity.

Linked List

Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.



FAT32 and NTFS

- FAT32 is one of the most commonly used file systems around. Most Operating systems, including Windows 98+, OS X, Linux, support the FAT32 file system.
- FAT32 does not support file compression, encryption and other functions supported by NTFS.
- FAT32 has a upper limit of 2TB on Hard drives and a limit of 4GB on single files.
- NTFS on the other hand, has a limit of 16Exabyte for both hard drives and single files.

FAT32 and NTFS

- FAT32 is one of the most commonly used file systems around. Most Operating systems, including Windows 98+, OS X, Linux, support the FAT32 file system.
- FAT32 does not support file compression, encryption and other functions supported by NTFS.
- FAT32 has a upper limit of 2TB on Hard drives and a limit of 4GB on single files.
- NTFS on the other hand, has a limit of 16Exabyte for both hard drives and single files.
- For small volumes, FAT16 or FAT32 usually provide faster access to files than NTFS.

Ext2

- Ext2 stands for second extended file system.
- It was introduced in 1993. Developed by Remy Card.
- Ext2 does not have journaling feature.
- On flash drives, usb drives, ext2 is recommended, as it doesn't need to do the over head of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext2 file system size can be from 2 TB to 32 TB

Ext3

- Ext3 stands for third extended file system.
- It was introduced in 2001. Developed by Stephen Tweedie.
- Starting from Linux Kernel 2.4.15 ext3 was available.
- The main benefit of ext3 is that it allows journaling.
- Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext3 file system size can be from 2 TB to 32 TB
- You can convert a ext2 file system to ext3 file system directly (without backup/restore).

Ext4

- Ext4 stands for fourth extended file system.
- It was introduced in 2008.
- Starting from Linux Kernel 2.6.19 ext4 was available.
- Maximum individual file size can be from 16 GB to 16 TB
- Overall maximum ext4 file system size is 1 EB (exabyte)
- Directory can contain a maximum of 64,000 subdirectories (as opposed to 32,000 in ext3)
- You can also mount an existing ext3 fs as ext4 fs (without having to upgrade it).
- In ext4, you also have the option of turning the journaling feature "off".