

Storage Management

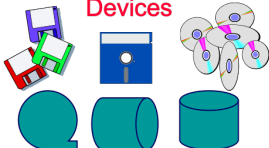
File systems...

A file system provides a mapping between the logical and physical views of a file, through a set of services and an interface. Simply put, the file system hides all the device-specific aspects of file manipulation from users.

The basic services of a file system include:

- keeping track of files (knowing location),
- I/O support, especially the transmission mechanism to and from main memory,
- management of secondary storage,
- sharing of I/O devices,
- providing protection mechanisms for information held on the system.

File system abstraction...

	<i>Objects</i>	<i>Typical operations</i>
Interactive (Shells)	files	copy, delete, rename
Applications and system programs	logical elements (records)	open/close, buffering seek (logical)
File System		file system check soft repair partitioning
Devices 	physical elements (head, cylinder, ...)	raw read/write, seek (physical) low-level format

Addressing levels...

There are three basic mapping levels (abstractions) from a logical to physical view of a file (contents):

- **File relative**— $\langle \text{filename}, \text{offset} \rangle$ form is used at the higher levels, where the file system is viewed as a collection of files.
- **Volume (partition) relative**—device-independent part of a file system use $\langle \text{sector}, \text{offset} \rangle$ (e.g., a partition is viewed as an array of sectors.)
- **Drive relative**—at the lowest level, $\langle \text{cylinder}, \text{head}, \text{sector} \rangle$ (also known as $\langle \text{track}, \text{platter}, \text{sector} \rangle$) is used.

File organization...

One of the key elements of a file system is the way the files are organized. File organization is the “logical structuring” as well as the access method(s) of files.

Common file organization schemes are:

- Sequential
- Indexed-sequential
- Indexed
- Direct (or hashed)

File space allocation...

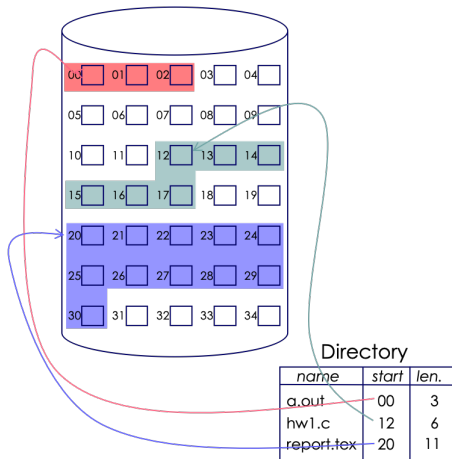
The file system allocates disk space, when a file is created. With many files residing on the same disk, the main problem is how to allocate space for them. File allocation scheme has impact on the efficient use of disk space and file access time.

Common file allocation schemes are:

- Contiguous
- Chained (linked)
- Indexed

All these techniques allocate disk space on a per block (smallest addressable disk unit) basis.

Contiguous allocation...



Allocate disk space like paged, segmented memory. Keep a free list of unused disk space.

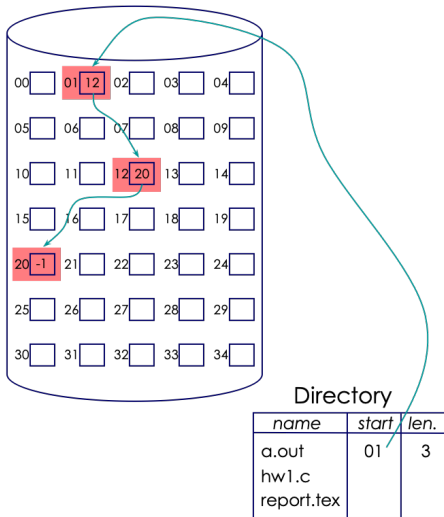
Advantages

- easy access, both random and sequential
- simple
- a few seeks

Disadvantages

- external fragmentation
- may not know the file size

Chained (linked) allocation...



Space allocation is similar to page frame allocation. Mark allocated blocks as in-use.

Advantages

- no external fragmentation
- files can grow easily

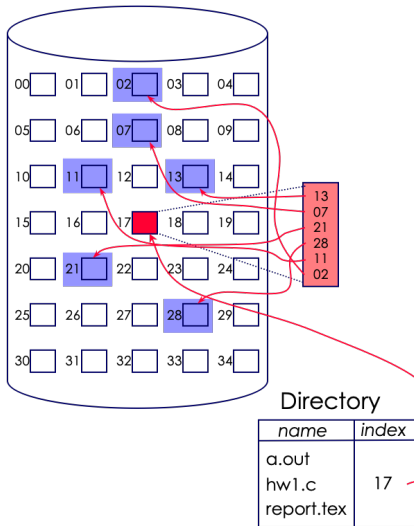
Disadvantages

- lots of seeking
- random access difficult

Example

- MSDOS FAT file system

Indexed allocation...



Allocate an array of pointers during file creation. Fill the array as new disk blocks are assigned.

Advantages

- small internal fragmentation
- easy sequential and random access

Disadvantages

- lots of seeking if the file is big
- maximum file size is limited to the size of a block

Example

- UNIX file system

Free space management...

Since the amount of disk space is limited (posing a management problem similar to that of physical memory), it is necessary to reuse the space released by deleted files. In general, file systems keep a list of *free* disk blocks (initially, all the blocks are free) and manage this list by one of the following techniques:

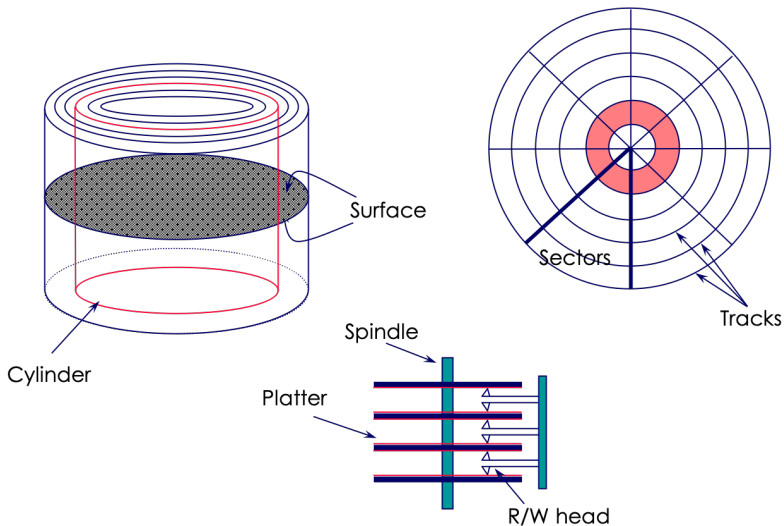
- **Bit vectors**
- **Linked lists** or **chains**—single list of a set of free block lists
- **Indexing**—single level, multiple levels

As you notice (again), we're dealing with “free space” management (like free memory)...

Other file system issues...

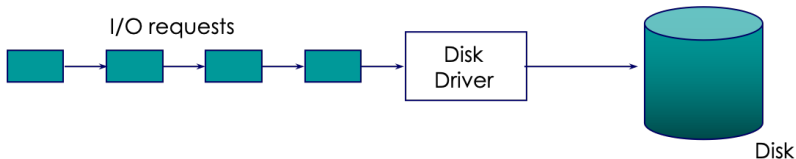
- *Disk blocking*
 - Multiple sectors per block for efficiency
- *Disk quotas*
 - How much space per user?
- *Reliability*
 - Backup/restore (disaster scenarios)
 - File system (consistency) check (e.g., UNIX `fsck`)
- *Performance*
 - Block or buffer caches (a collection of blocks kept in memory)

Disk structure...



Disk scheduling...

In multiprogramming systems, there may be several disk I/O requests at the same time. As a result, a disk driver is typically faced with a pool of I/O requests:



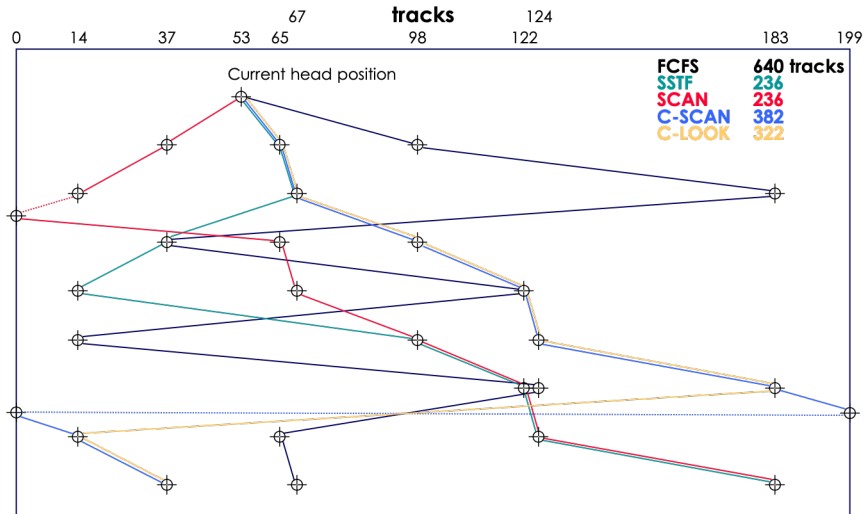
The most costly component of a disk I/O operation is the **seek time**. By scheduling multiple disk requests, the total seek time can be reduced. For example, shortest seek time first.

Disk scheduling strategies...

Commonly used strategies include (in addition to some common CPU scheduling policies!):

- *First Come First Served (FCFS)* or *FIFO*
- *Shortest Service Time First (SSTF)*
- *SCAN*—back and forth over disk
- *C-SCAN*—circular SCAN or one way SCAN and fast return
- *LOOK*—look for a request before moving in that direction
- *C-LOOK*—circular LOOK

A comparative example...



Disk management issues...

- **Formatting**

- *Physical*: divide the blank slate into sectors identified by headers containing such information as sector number; sector interleaving
- *Logical*: marking bad blocks; partitioning (optional) and writing a blank directory on disk; installing file allocation tables, and other relevant information (file system initialization)

- **Reliability**

- disk interleaving or striping
- RAIDs (Redundant Array of Inexpensive Disks): various levels, e.g., level 0 is disk striping)

- **Controller caches**

- newer disks have on-disk caches (128KB—512KB)

Elements of a storage management...

