Lecture 10 (File systems)

Requirements for long-term information storage:

- possible to store a large amount of information
- information must survive termination of a process using it
- multiple processes can concurrently access information

Disk = linear sequence of fixed-size blocks supporting r/w operation

---

File - logical unit of information created by process. File = name+extension. Information in files **persistent** - does not affected by process creation & termination.

Types of files:

- Byte sequence - unrestricted sequence of bytes, any meaning by user-level program
- Record sequence - sequence of fixed-length records with internal structure
- Tree - tree of records, each contains key field in fixed position

Regular files - contains user information. (ASCII or binary) Binary files have internal structure known to programs that use them.

Directories - system file for maintaining structure of file system.

Character special files - used to model serial I/O devices.

Block special files - model disks

File access:

- Sequential (process can read bytes or records in order, starting from the beginning to end without jumps)
- Random access (read bytes or records out of order)

Attributes == metadata: protection, password, creator ID, owner, read-only flag, hidden flag, system flag, lock flag, record length, key position, key length, creation time, time of last access, time of last change, current size, maximum size

File operations: create, delete, open, close, read, write, append, seek(file pointer to specific position inside file), get attributes, set attributes, rename.

Path names:

- Absolute (path from the root directory to the file)
- Relative (path in working directory)

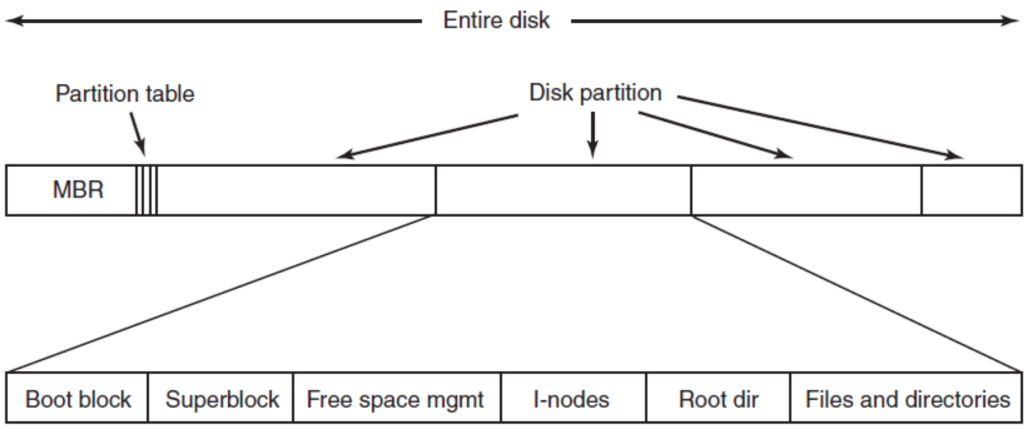
Directory operations: create, delete, opendir, closedir, readdir, rename, link(link to file), unlink

"." - current directory, ".." - parent directory or root directory

---

File system layout

File systems stored on disks & disk divided into partitions, with independent file systems



Sector 0 of disk = Master Boot Record (MBR) boot computer. **Boot sequence:** BIOS read & execute MBR → MBR locate active partition → reads in boot block → program in boot block loads OS.

Superblock contains **magic number** (identify file-system type); number of blocks, key administrative information

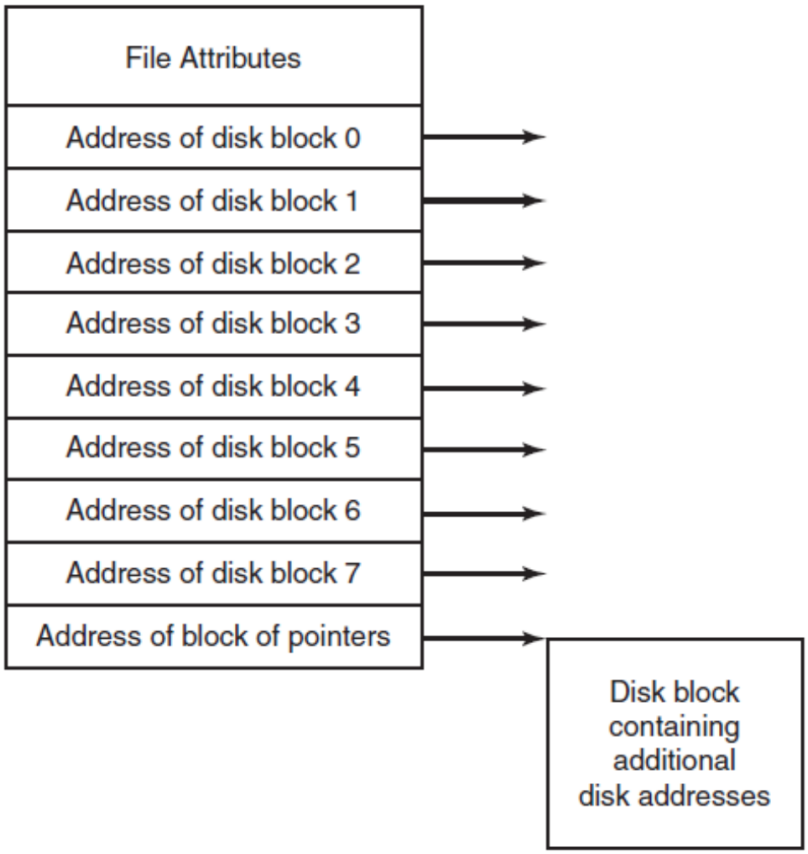
---

Implementing files

- **Contiguous allocation:** each file - contiguous run of disk blocks. Advantages: simplicity, read performance; Disadvantages: disk fragmentation. Physical

pieces of the single logical file (the movie) are called **extents**

- **Linked-list allocation:** first word of each block - pointer to the next, rest - data. Advantages: no fragmentation; Disadvantages: random access is slow, part of the block used for storage. **With File allocation table (FAT):** pointer word form each block in table. Advantage: lookup is fast, only data in block; Disadvantage: whole table must be in memory.
- **I-node:** data structure contains attributes & disk address of file's blocks. Advantages: only i-node for open file in memory, array proportional to max number number of open files at once.



Implementing directories:

File opened → OS locate directory entry on disk →directory entry provides info needed to ind disk blocks →

Shared files

Link - connection between directory and the shared file from another directory. File system now Directed acyclic graph (DAG)

Solutions for not visible changes for other user:

- Disk blocks are not listed in directories, but in a little data structure associated with the file itself
- **Symbolic link:** file contains reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution

---

File systems

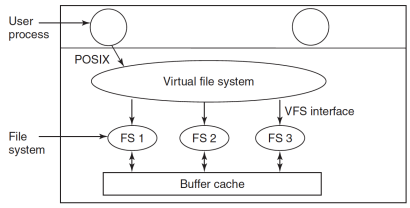
Log-structured: disk structured as log - circular buffer. I-nodes are scattered among disk instead of being at fix position. Pending writes are buffered into one segment & written to disk as single contiguous segment with description of content. **Cleaner** - thread that scans log circularly to compact it.

Journaling: keep log of next job for file system before it actually does it. If crush ⇒ reboot system & see from log crushed job. Logged operations must be **idempotent** - they can be repeated as often as necessary without harm. **Atomic transaction** - indivisible and irreducible series of database operations such that either all occur, or nothing occurs.

Removing a file:

- remove file from directory
- Its i-node to pool of free i-nodes
- Its all disk blocks to pool of free disk blocks

Virtual: common part to all file systems put in separate layer that calls underlying file systems to manage data.



Superblock - describes file system, v-node - describes file,

System booted → root file system registered by VFS → file system provide list of of addresses of required VFS functions as call vector → VFS calls any function supported by file system → VFS create v-node with necessary info → VFS makes entry to file descriptors' table & point it to new v-node → VFS return file descriptor to caller → file can be read, write & close

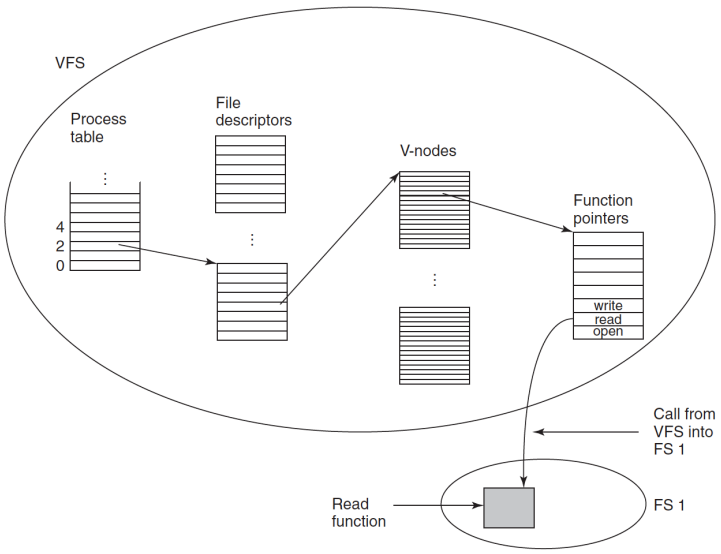


Figure 4-19. A simplified view of the data structures and code used by the VFS and concrete file system to do a read.

File system management & optimization

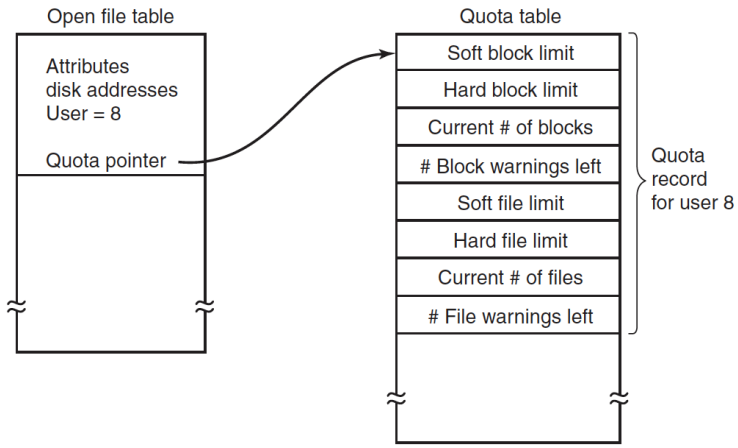
Disk space management:

- n consecutive bytes of disk space are allocated
- file split into blocks

Keeping track of free blocks:

- Linked list (linked list of disk blocks, with each block holding as many free disk block numbers as will fit)
- Bitmap (Free blocks are represented by 1s in the map, allocated blocks by 0s (or vice versa). Can be put in virtual memory)

Disk quota - maximum allotment of files and blocks for user



File system backups' problems:

1. Recover from disaster - disk crush, fire, flood...
2. Recover from stupidity - removed to recycle bin

Dumping:

- Physical (starts from block 0 to the end copy all blocks from input disk to output disk)
- Logical (starts at specified directories & dump all changed files)

Incremental dump - dump files which has not been modified.

If system crashes before all modified blocks have been written out ⇒ file system can be left in inconsistent state

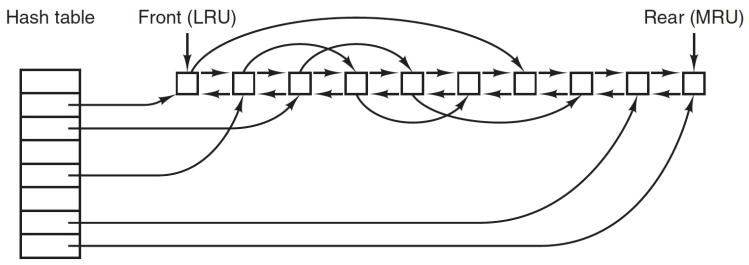
Consistency check:

- Blocks (each block have 1 either in block in use table or free block's table)
- Files (find duplicate file)

Missing block - block that doesn't occur in the table during consistency checking

Techniques to reduce disk access:

- Block cache or buffer cache (collection of block keep in cache to increase performance)



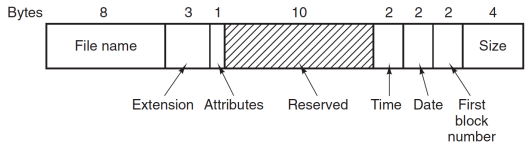
- Block Read Ahead - try to get blocks into the cache before they are needed to increase the hit rate
- Reducing disk-arm motion (put blocks accessing in sequence nearby)

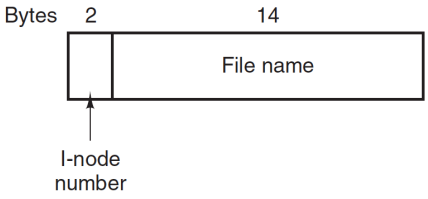
Write-through caches - caches in which all modified blocks are written to the disk immediately.

SSD - solid state disk - disk without any moving parts

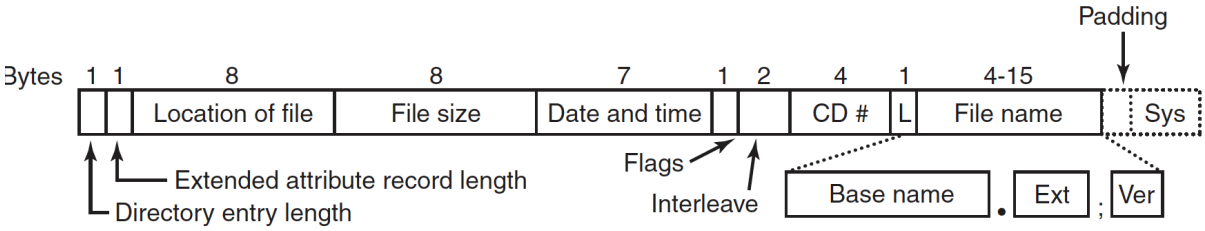
Defragmentation - process of moving blocks to get one contiguous space of free blocks

MS DOS directory entry: **UNIX V7 directory entry:**





ISO 9960 directory entry:



Rock ridge extensions - represent UNIX on CD-ROM:

1. PX - POSIX attributes

2. PN - Major and minor device numbers
3. SL - Symbolic link
4. NM - Alternative name
5. CL - Child location
6. PL - Parent location
7. RE - Relocation
8. TF - Time stamps

Joliet extensions - Windows on CD-ROM:

1. Long file names
2. Unicode character set
3. Directory nesting deeper than eight levels
4. Directory names with extensions

RAID 1 -два диска один зеркальный(точная копия) - обработка запросов, время записи одинаковое, overhead по дискам 50%

RAID 2 - коррекционные коды - исправление ошибок

RAID 3 - хранят только биты четности (parity bit for every word = 1 if number odd) each word on separate disk

RAID 4 - strips, parity bits

RAID 5 - parity bits store all over disks

RAID 6 - double parity bits store all over disks