File system is the most visible part of the operating system

Files are stored in disks

Allocation of space for files in disks

keep track of free disk blocks.

An allocation method refers to how disk blocks are allocated for file.

Disk space should be allocated so that disk space is utilized effectively and files can be accessed quickly.

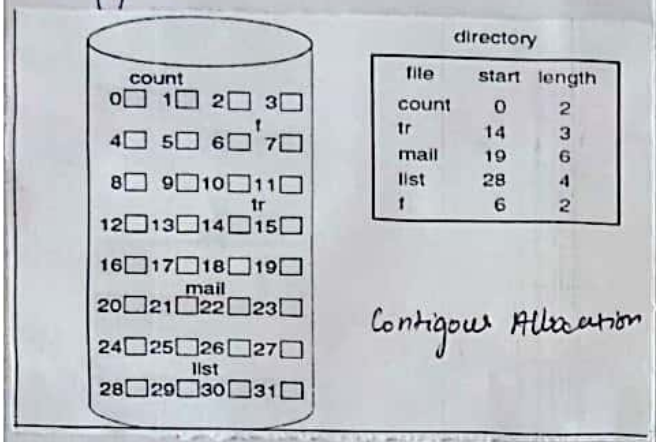
- contigous allocation
- linked allocation
- Indexed allocation.

Contigous Allocation:

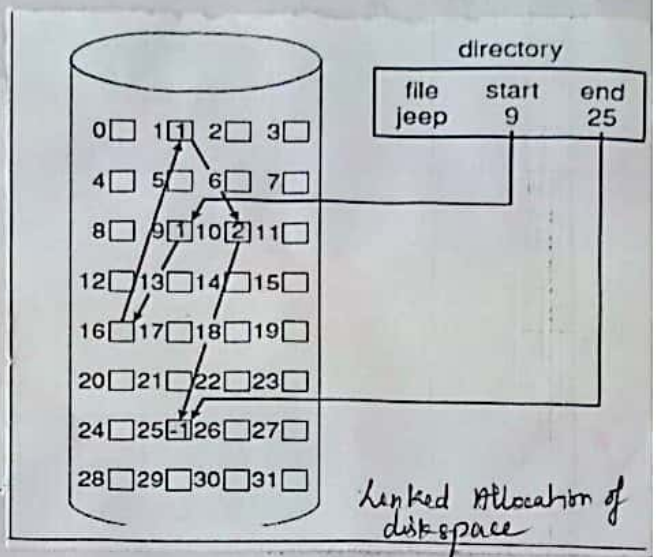
- Each file occupies a set of contigous blocks on the disk.
- If only one job is accessing the disk, accessing block b+1 after block b required no head movement.
  - Disk seems are minimal.
- Simple - only starting location (block #) and length (number of blocks) are required.
- Sequential and Direct access.
- wasteful of space (dynamic storage - allocation problem)
  - How to satisfy a request of size 'n' from a list of free holes.
  - First fit / Best fit.
- External Fragmentation - compaction.
- Difficult to determine how much space is needed for a file.
- Files cannot grow. can overestimate the space needed, results in wastage of space.
  - Find a larger hole, copy contents to new space, free old space.
- Even if file size is known in advance, file may grow over a long period of time - internal fragmentation.
- Modified contigous allocation.
  - Initially, contigous chunk of space is allocated.
  - If this is not large enough, another chunk of space (extent) is allocated
  - Example: Veritas file system

Linked Allocation :

- Solves all problems of contiguous allocation.
- Each file is a linked list of disk blocks : blocks may be scattered anywhere on the disk.
- Directory entry has a pointer to the first and the last blocks of the file.

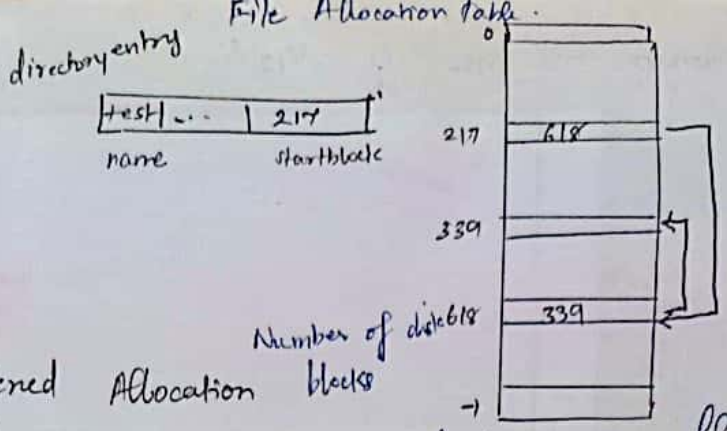


Contiguous Allocation



Linked Allocation of disk-space

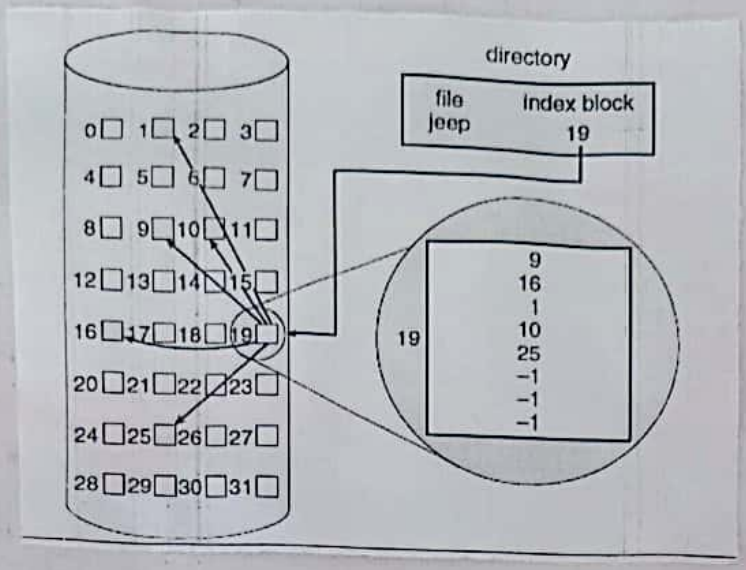
- No external fragmentation.
- Size of a file need not be declared when the file is created.
- File can continue to grow as long as free blocks are available
- Compaction not needed.
- No random access (only sequential-access)
- Space required for pointers
  - Collect blocks into multiples called clusters (say, 4 blocks)
  - Less number of pointers.
  - Fewer disk head seeks.
  - Increased internal fragmentation.
- Reliability :
  - If pointers are lost/damaged ?
  - Use doubly linked list.
  - Store the file name and relative block number in each block.
- Variation of linked allocation — File-allocation table (FAT) — disk-space allocation used by MS-DOS.
- A section of disk at the beginning of each volume contains the table.



Indexed Allocation blocks

- Solves the external fragmentation and size-declaration problem of contiguous allocation.
- Solves the direct access problem in linked allocation.
- Brings all pointers together into one block, the index block.
- i^{th} entry in the index block points to the i^{th} block of the file.
- Supports direct access.
- No external fragmentation.
- Suffers from wasted space.
  - More pointer overhead
  - When there is a file occupying only 2 blocks, in linked allocation, space for only 2 pointers is wasted, in indexed allocation, one whole block is wasted.
- How large the index block should be?
  - As small as possible
  - If index block is too small, block may not be able to hold enough pointers for a large file.
- Consider a file of maximum size of 256K (2^{18}) words and block size of 512 (29) words
  - File occupies 29 blocks.
  We need only 1 block for index table.
- Mechanisms for index block variants
  - Linked scheme – link together index blocks (no limit on size)
    - An index block will contain addresses of disk blocks as well as address of next index block.
  - Two-level index
    - First-level index block points to a second level index block
    - Second-level index block points to file blocks.

- Maximum file size is 512 3.



Indexed Allocation of disk space

Combined Scheme :

- Another alternative used in UNIX-based file systems, 15 pointers of the index block in the file's inode.
- The first 12 of these pointers point to direct blocks; (address of blocks that contain data of the file. Thus data for small file do not need a separate index block).
- If the block size is 4KB, then upto 48 KB of data can be accessed directly. The next 3 pointers point to indirect blocks.
- The first points to a single indirect block, which is an index block containing not data but the addresses of blocks that do contain data.
- The second points to a double indirect block, address of block that contains address of blocks that contains pointer to the actual data block.
- The last pointer contains the address of a triple indirect block.

- This method is used in CD-ROM and DVD-ROM drives.

Constant Angular Velocity:

The disk rotation speed can stay constant. The density of bits decreases from inner tracks to outer tracks to keep the data rate constant. This method is used in hard disks and is known as Constant angular velocity (CAV).

① Disk Scheduling

- The responsibilities of operating system is to use the hardware efficiently.
- For the disk drive,
  - fast access time
  - High disk bandwidth.
- The access time has 2 major components:
  - ① The seek time is the time for the disk arm to move the heads to the cylinder containing the desired sector.
  - ② The rotational latency is the additional time for the disk to rotate the desired sector to the disk head.
  - ③ The disk bandwidth is the total number of bytes transferred, divided by the total time b/w the first request for service and the completion of the last transfer.

- Whenever a process needs I/O to or from the disk, it issues a system call to the Operating System.
- If the desired disk drive and controller are available, the request can be serviced immediately.
- If the drive or controller is busy, any new requests for service will be placed in the queue of pending requests for that drive.
- For a multiprogramming system with many processes, the disk queue may often have several pending requests.
- When one request is completed, the operating system chooses which pending request to service next.

Several disk scheduling algorithm can be used.

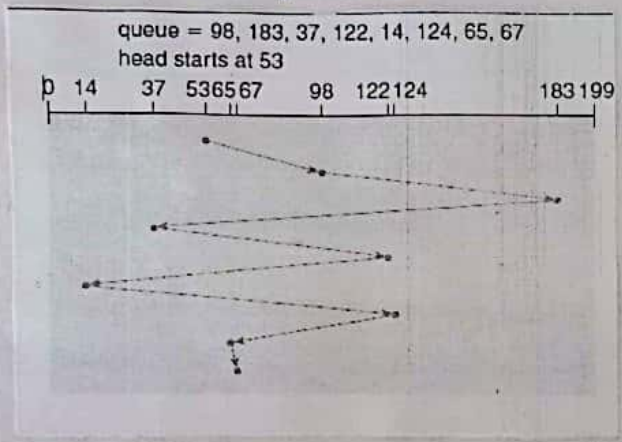
1. FCFS (First Come First Served)
2. SSTF (shortest - seek - time - first) scheduling.
3. SCAN Scheduling. [several algorithms exist to schedule the servicing of disk I/O requests.]
4. C-SCAN Scheduling.
5. LOOK Scheduling
6. C-LOOK scheduling.

Request Queue - 98, 183, 37, 122, 14, 124, 65, 67

Head pointer - 53.

FCFS (First Come first Served)

- The simplest form of disk scheduling
- This algorithm is intrinsically fair, but it generally does not provide the fastest service.

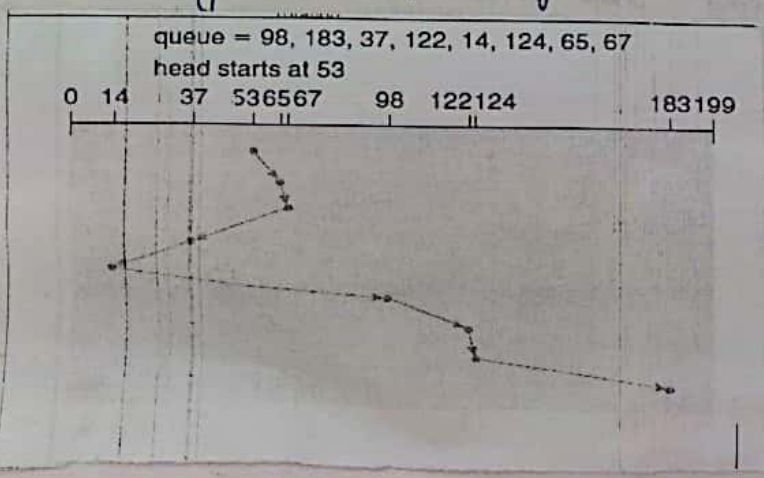


Total head Movement =
(98-53) + (183-98) + (183-37) + (122-37) + (122-14) + (124-14) + (124-65) + (67-65)

= 640 cylinders

SSTF (Shortest - seek - time - first) algorithm

- Select the request with the minimum seek time from the current head position.
- choose the pending request close to the current head position
- SSTF scheduling is a form of SJF scheduling.



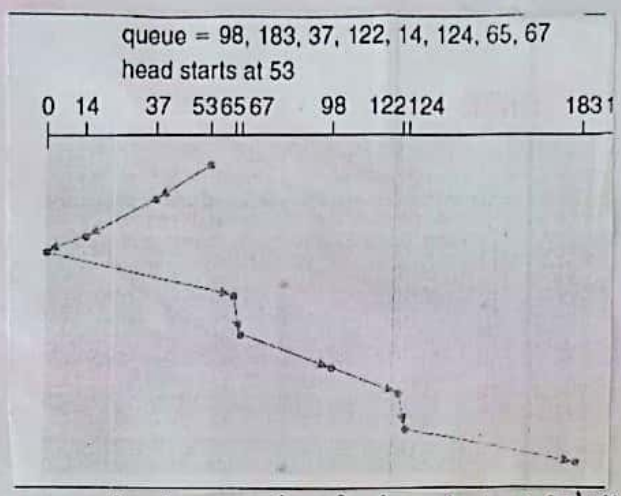
disadv:
starvation of requests.
- Not optimal

Total head Movement =
(65-53) + (67-65) + (67-37) + (37-14) + (98-14) + (122-98) + (124-122) + (183-124)

= 208 cylinders.

SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk.
- The head movement is then reversed and servicing continues.
- Sometimes called the elevator algorithm.



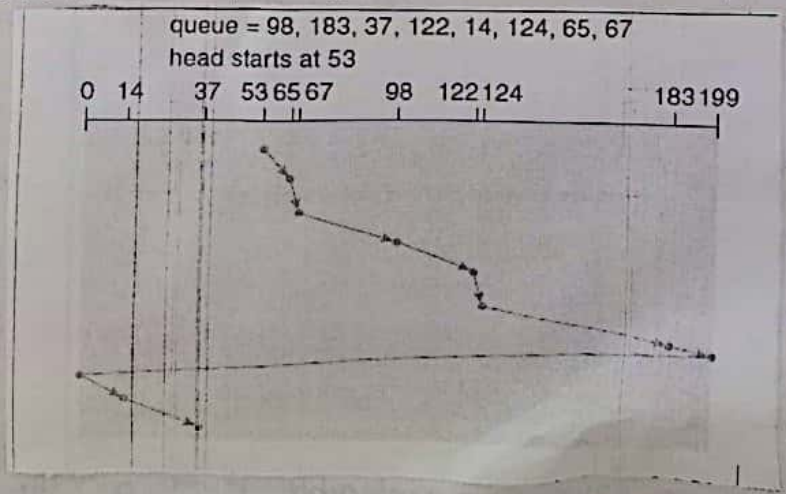
Total head Movements = 236

Total head movement = (53-37) + (37-14) + (14-0) + (65-0) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) = 236

C-SCAN : Circular SCAN scheduling is a variant of SCAN designed to

- provide a more uniform wait time.
- The head moves from one end of the disk to the other servicing requests as it goes.
- When it reaches the other end, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

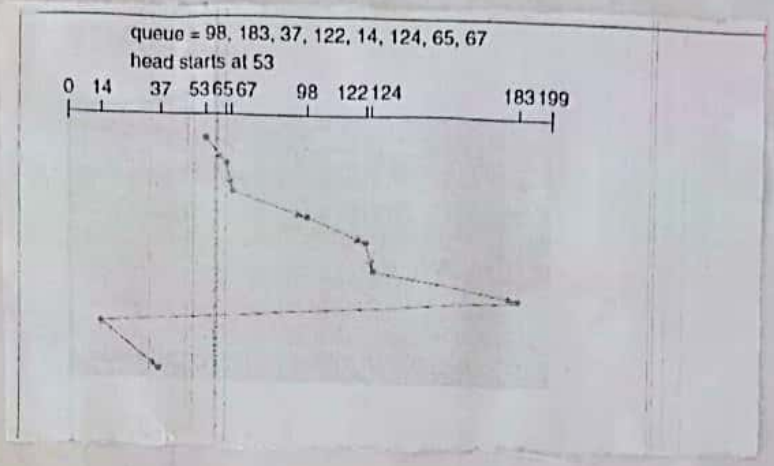
Total head movment = (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) + (199-0) + (14-0) + (37-14) = 382 cylinders



C-LOOK :

- Variant of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately.
- The arm does not go all the way to the end of the disk.

Total head Movements = (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (37-14) = 153



Selecting a Disk Scheduling Algorithm :

- SSTF is common and has a natural appeal because it increases performance over FCFS.
- SCAN and CSCAN perform better for systems that place a heavy load on the disk - because less starvation.
- performance depends on the number and types of requests.

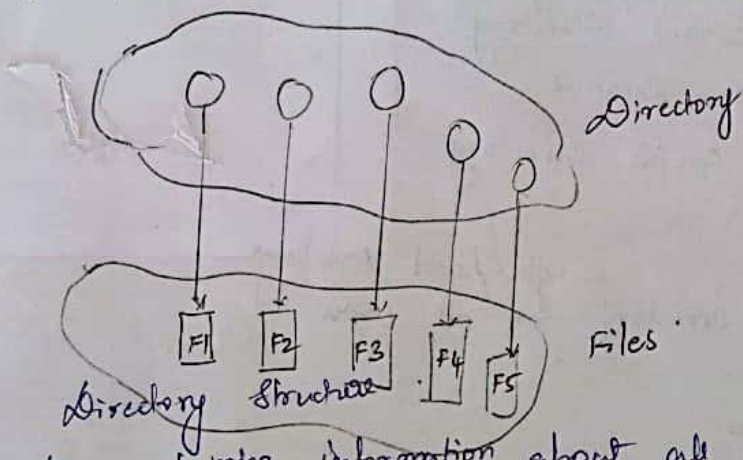
* Request for disk service can be influenced by the file-allocation method.
  - For a contigously allocated file - limited head movement.
  - For linked / indexed file - may include blocks that are widely scattered on the disk - greater head movement.

* Location of directories and index blocks is also important
  - opening a file requires searching the directory structure, then the files data.
  - If directory entry is on the first cylinder and the files' data are on the final cylinder, greater head movement.
  - Caching directories and index blocks in main memory help to reduce disk-arm movement.
  - Either SSTF or LOOK is a reasonable choice for the default algorith

Disk Structure

Systems stores millions of files on disk.
should organize to manage this large number of files
Disk can be subdivided into partitions
Disk or partition can be used raw - without a file system, or formatted with a file system

- partitions also known as minidisks, slices.
- Each disk has atleast one partition
- partitions can store multiple operating system.
- Entity containing file system known as a Volume.
- Each volume containing file system also tracks that file system's info in device directory or volume table of contents.
- Device directory records information such as name, location, size, type for all files in that volume



- A collection of nodes containing information about all files
- Both the directory structure and the files reside on disk.
- Backups of these two structures are kept on tapes.

Operations performed on Directory

Search for a file
Create a file
Delete a file
List a Directory

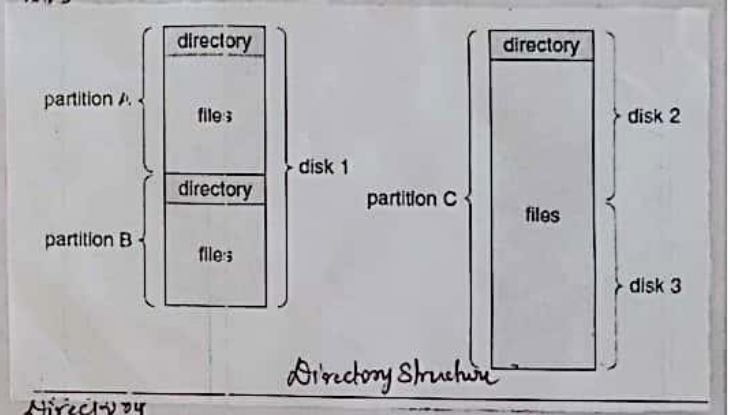
- Rename a file - position in the directory may be changed.
- Traverse the file system

Organize the Directory (logically) to obtain

- Efficiency - locating a file quickly
- Naming - convenient to users.
  - Two users can have same name for different files
  - The same file can have several different names.
- Grouping - Logical grouping of files by properties (eg all Java programs, all games, ...)

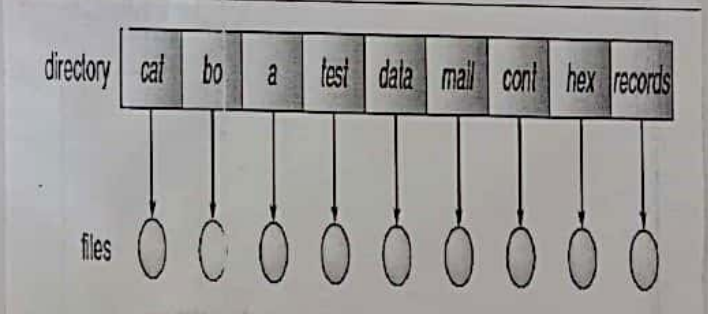
Different directory structures are:

1. Single-Level Directory
2. Two-Level Directory
3. Tree-Structured Directory
4. Absolute and Relative
5. General Graph Directory



Single-Level Directory

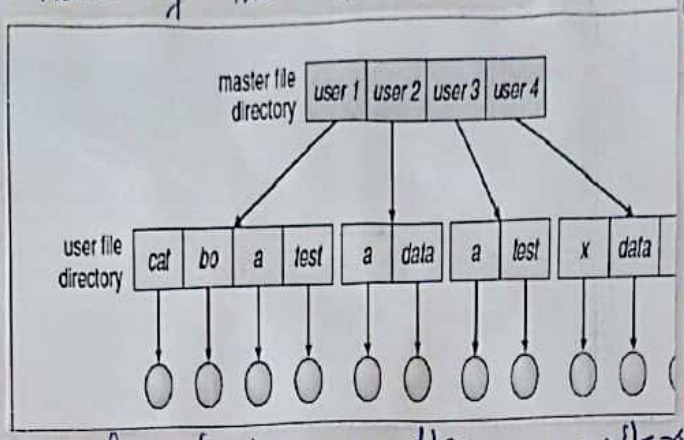
- A single directory for all users
- Simplest
- Limitations
  - Number of files increases
  - System has more than one user
  - Naming problem.
- Even a single-user finds it difficult to remember names.



Two-Level Directory

- Separate directory for each user.
- can have the same file name for different user
- creation and deletion of files confined to the user's UFD.
- Creation and deletion of UFDs

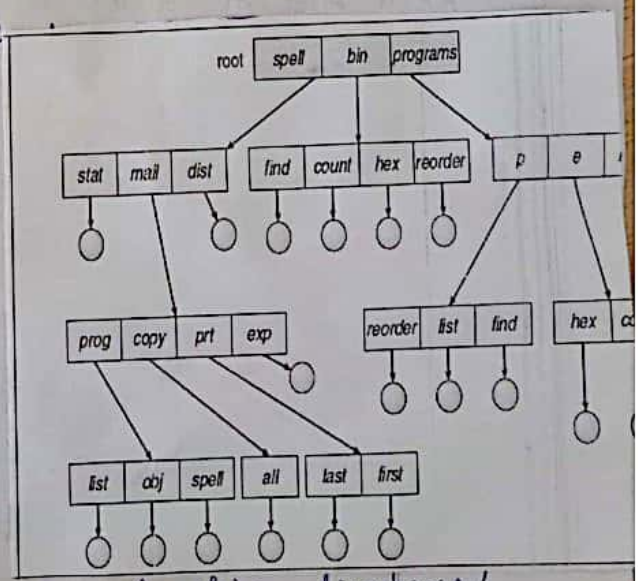
- User can't cooperate and access one another's files.
- To access another user's file should specify the path.
- path name - name of the MFD (root), name of UFD, name of file



- placing system files - loaders, assemblers, compilers
  - Copied to each UFD - waste of space.
  - Special user directory to contain the system files
  - Search path.

Tree-structured Directories:

- Tree is the most common directory structure.
- Directory - a special kind of file - bit to differentiate.
- System calls for creating and deleting directories
- Tree has a root directory.
- Every file has a unique path.



- Each user has a current directory (working directory)
  - first current directory is searched.
  - cd /spell/mail/prog
  - type list

- Absolute or Relative path Name :
  If root/spell/mail is the current directory, pst/first is the relative path name, absolute path name is root/spell/mail/pst/first.
- Creating a new file is done in current directory
- Delete a file rm <file-name>
- Creating a new subdirectory is done in current directory
  mkdir <dir-name>
  Example : if in current directory /mail
  mkdir count
- Deletion of a directory
  - If empty, can be deleted
  - If not empty
    - will not delete unless empty
    - All files and subdirectories will also be deleted.

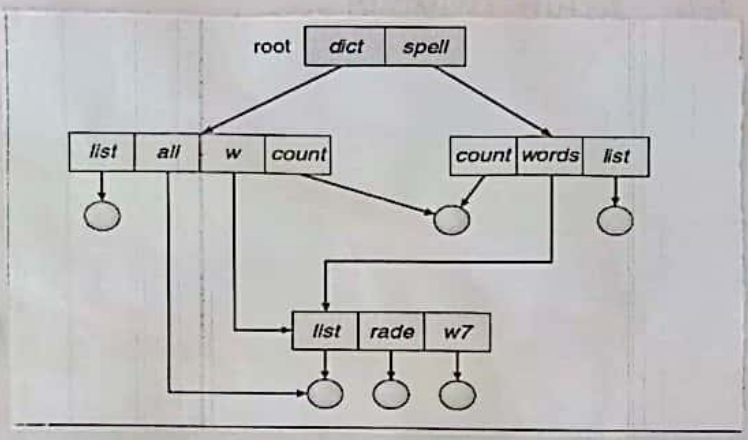
- Can access files of other user
  - By specifying path names - absolute or relative
  - By changing current directory to other user's directory
  - users are allowed to define their own search paths - local directory, system file directory, other users directory
- prohibits the sharing of files and directories among different users.

Acyclic-Graph Directories

- Have shared subdirectories and files.
- Sharing
  - Not the same as having 2 copies of the same file.
  - Link (Hard Link, Soft Link)
  - Duplicate all information in both sharing directories

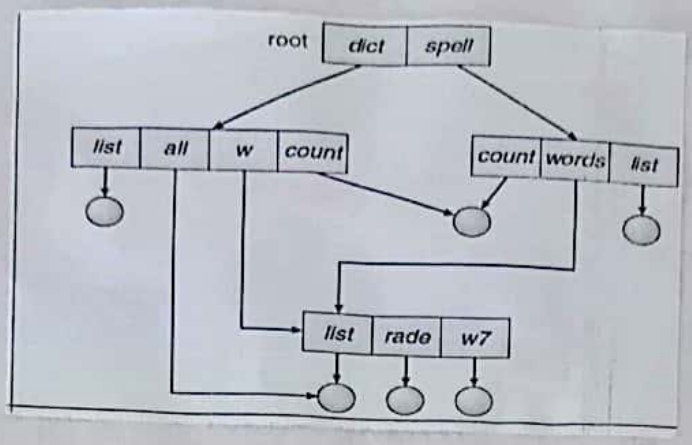
Two different absolute path names for the same file.

Deletion of shared files:
- Remove the file when anyone deletes it
- Leaves dangling pointers if original file is removed.
- In symbolic links, if the link file is removed no harm.
- If original file is removed, leaves dangling pointers.
- In hard links, a link count is maintained
- Link count gives the number of links to a file
- preserve the file until all references are deleted.



General Graph Directory
- If cycle are allowed to exist in directories
  - possibility of searching a subdirectory twice - infinite loop possible.
  - Limit the number of directories accessed during a search
  - Allow only links to file not subdirectories.

Deletion of a file:
- possibility to have a non-zero reference count - self-referencing directory.
- Garbage collection:
  - Traverse the file system, mark everything that can be accessed
  - In second pass, collect everything that is not marked onto list of free space.
  - Extremely time consuming for a disk based file system

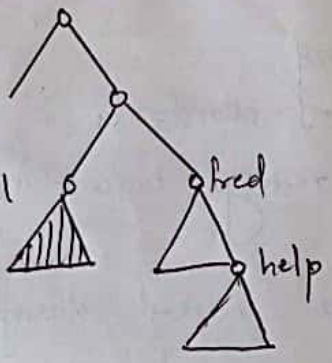


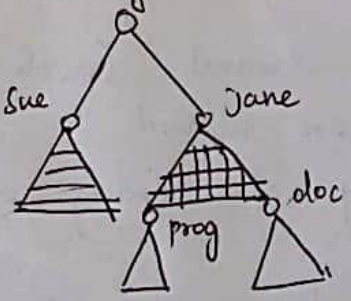
- Acyclic graphs are easier to work with
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK.

(4) a File System Mounting

- File System must be mounted before accessed
- Single file system can be built out of multiple partitions
- mount (name of device containing the file system to be mounted, location at file system to be mounted, location at file system to attach (mount point))
  - Mount point - directory at which the mounted file system will be attached.

File System mounting





- Mounted device should have a valid file system.
- A mount table is kept in the kernel which maintains information about the mount point, mounted file system
- Mount point shall be empty.

b File Sharing

- Sharing of files is desirable when users want to collaborate
- In a multiuser and access control needed for the sharing
- Systems use concepts of owner and group
- Owner is responsible for changing attributes and granting access

for a file

File Sharing - Multiple Users :

- Group is a subset of users who can share access to the file.
- User IDs and group IDs are stored with other attributes in a file
- UNIX - example
- Even with multiple local file systems, ID checking and permission matching are not straightforward, once the file systems are mounted
- With advent of networks, communication between remote computers possible.
- Sharing of resources across the world has become possible
  - Manually via programs like FTP.
  - Automatically, seamlessly using distributed file systems (remote directories are visible from a local machine).

File Sharing - Remote File Systems :

Client-Server model allows clients to mount remote file systems from servers.

- Server can serve multiple clients
- client can use multiple servers.
- NFS is standard UNIX client-server file sharing protocol.
- Common Internet File System (CIFS) is standard windows protocol.

File Sharing - Failure Modes

- Local file systems fail due to failure of disk, corruption of directory structure, disk controller failure etc.
- Errors caused by users or system administrators also can causes files to be lost or directories to be deleted.
- Remote file systems add new failure modes, due to network failure, server failure.
- Interruption of networks can be due to hardware failure, poor hardware configuration or network implementation issues.
- Say, a client is using a remote file system.
  - client may be opening, reading, writing, closing files.
  - If server crashes / shuts down, client has to terminate or wait.

- Termination can result in losing data.
- Most protocols allow delaying of operation so that the remote host becomes available again

o Recovery from failure can involve state information being maintained in the client and the server.

o NFS implements a stateless DFS.
- Includes all information in each request, allowing easy recovery but less security - forged read and write.

© File sharing - Consistency Semantics.

- Consistency Semantics specify how multiple users are to access a shared file simultaneously.
- Specifies when modifications of data by one user will be seen by other users.
- A file session is a series of file accesses b/w an open() and close() operations.
- Andrew File System (AFS) has session semantics:
  - Writes to an open file by a user are not visible immediately to other users that have the same file open.
  - Once a file is closed, the changes made to it are visible only in sessions starting later.
  - Already open instances do not reflect these changes.
  - Writes only visible to sessions starting after the file is closed.

Protection:

- To keep files safe from physical damage (reliability) and from improper access (protection)
- Reliability
  - Writes to an open file visible immediately to other users who have this file open.
  - File has a single image that interleaves all ops regardless of their origin.

UNIX SEMANTICS :

- Writes to an open file visible immediately to other users who have this file open.
- File has a single image that interleaves all operations, regardless of their origin.

- Immutable - shared - File Semantics :
  - Once a file is declared as shared by its creator, it cannot be modified.
  - The contents of an immutable file cannot be altered
  - The files are read only.

(d) Protection

- To keep files safe from physical damage (reliability) and from improper access (protection)
- Reliability
  - Duplicate copies of files
  - periodically copy disk files to tape at regular intervals
- File owner/creator should be able to control:
  - What can be done on a file
  - by whom.
- Types of access that can be controlled ..
  - Read, write, Execute, Append, Delete, List, Renaming, Copying etc.

Access Control Lists and Groups:

- Associate with each file and directory an access-control list (ACL)
  - ACL has user names and types of access allowed for each user
  - When a user requests access to a particular file, the access list is checked.
  - If the user is listed for that particular access, access is allowed - Else user is denied access.
  - problem with ACL is the length of the list.
  - Need to know in advance the list of users in the system
  - The directory entry must be of variable size.

- Condensed access-control list with user categories.
  - owner, group and Universe.
- Combination of both (Solaris)
  - By default, the three categories of access is allowed.
  - For specific files and directories more fine grained access control is allowed.

- In UNIX, directory and file protection are handled similarly
- Each file has three fields - owner, group and others.
- Each field has three bits

		r	w	x
a) owner access	7	1	1	1
b) group access	6	1	1	0
c) public access	1	0	0	1

Other Protection Approaches:-
- Associate a password with each file
- passwords to be chosen randomly and changed often
- User may have to remember a number of passwords
- If only one password is used for all the files, once it is discovered, all files become accessible.
- Some systems allow user to associate password with a subdirectory

Learning Objectives:

e- Disk management - Disk Initialization
Booting from disk
bd-block Recovery.

f- Swap-space Management.

Disk Management - Disk formatting.

- A new disk is just a platter of magnetic recording material.
- low level formatting, or physical formatting - Dividing a disk into sectors that the disk controller can read and write.
- A special data structure is assigned for each sector.
  - Header (sector no).
    data area,
    trailer (error-correcting code)

Disk controller
- Data part
- ECC calculation
- stored.

Rel: Data - ECC already stored
ECC compared - same
errors good

- The ECC can be used to correct if there are errors in a few bits of data.
- If error can be found, it reports a recoverable soft error.
- Many hard disks are low-level formatted as a part of the manufacturing process.
- If the controller low-level-formats the disk, the block size can be specified (512, 256, 1024, etc). usa/admin → size of sector.
- To use a disk to hold files, the O.S still needs to record its own data structures on the disk.
  - partition the disk into one or more groups of cylinders.
  - Each partition can then be treated as a separate disk.

- logical formatting or "Creation of a file system"
  - Initial file-system data structures are stored onto the disk
  - Data structures may include maps of free and allocated space (FAT) and an initial empty directory.

Boot block :

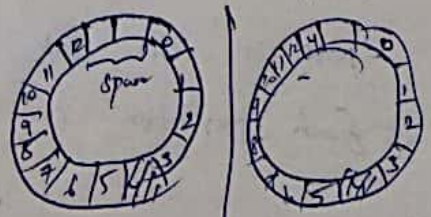
- For a computer to start running, needs an initial program to run bootstrap program.
- Bootstrap program initializes all aspects of the system, find the OS kernel on the disk, loads the kernel to memory, jumps to an initial address to start execution.
- The bootstrap loader can be stored in ROM.

- The bootstrap loader in the ROM brings into memory the full bootstrap program from the disk.
- Full bootstrap program is stored in boot blocks in the disk.
- Disk having a boot partition is called a boot disk (system disk)

Bad blocks.

- Disks are prone to failure becoz of moving part (Cylinder heads)
- If failure is complete, disk has to be replaced, data restored from backup.
- Sometimes some sectors alone become defective - called bad blocks
- Manually handle bad blocks - scan
- Methods such as sector sparing used to handle bad blocks.
  - Controller maintains a list of bad blocks on the disk..
  - This list is initialized during low-level format and is updated over the life of the disk
  - low-level formatting sets aside spare sectors
  - The controller can be told to replace each bad sector logically with one of the spare sectors.

Sparing



- Sector slipping
  - 17 - defective 203 - spare
  - 202 to 203, 201 to 202.

slipping

Swap Space Management.

- Swap space - Virtual memory uses disk space as an extension of main memory.
- Swap-space can be carried out of the normal file system or more commonly, it can be in a separate disk partition.
- Some OS allow the use of multiple swap spaces (both file and dedicated swap partitions) - Linux.

* Swap space - a large file within the file system / dedicated swap partition
  - Normal file-system routines are used to create it, name it and allocate space.
  - Navigating the directory structure and the disk-allocation data structures takes time.
  - Can improve performance by caching disk blocks in main memory and allocating physically contiguous blocks.

* Swap-space - As a separate raw partition.
  - Create a fixed amount of swap space during disk partitioning
  - No file system or directory structure.
  - A swap space manager allocates and deallocates blocks from the raw partition.
  - Manages uses algorithms optimized for speed rather than for storage efficiency.
  - Internal fragmentation - till next boot.

(5) a) FREE SPACE MANAGEMENT

- To keep track of free disk space.
- To reuse the space from deleted files for new files.
- Free-space list records all free disk blocks.
- To create a file, search the space in the free list and allocate that space to the new file.
- Remove the space from the free-space list when a file is deleted, add the space to the free-space list.

Implementation of free space list 1- Bit Vector
2- Linked list
3. Grouping
4 Counting

Bit Vector:

The free space list is implemented as a bit map / bit vector. Each block in represented by 1 bit. If the block is free, the bit is 1 If the block in allocated, the bit is 0.

$$bit[i] = \begin{cases} 0 \Rightarrow block[i] \text{ free} \\ 1 \Rightarrow block[i] \text{ occupied} \end{cases}$$

Say, blocks 2, 5, 8, 22 are free..

11011011011111111111111011

adv - Relative simplicity and its efficiency in finding the first free blocks or n consecutive free blocks on the disk.

- Sequentially check each word in the bit map to see whether the value is 0 or not.
- Calculation of block number (Number of bits per word) × (number of 0-value words) + offset of first 1 bit.

- Bit map requires extra space and is kept in main memory

- Example: block size = 2^{12} bytes
disk size = 2^{30} bytes (1 gigabyte)
Number of blocks = $2^{30}/2^{12} = 2^{18}$
Number of bits in bit map 2^{18} (or 32k bytes).

A 1 TB disk 4 KB block size needs 256 MB to store it bit map.

Linked List:

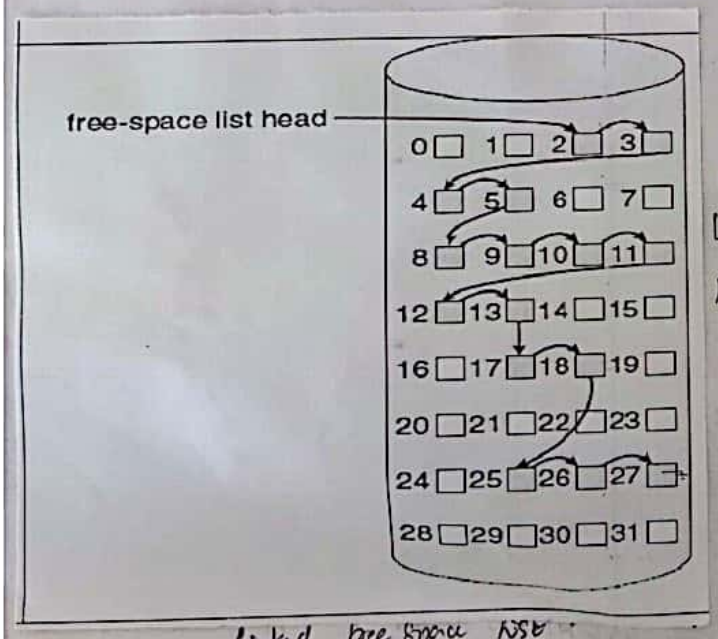
- Link together all the free blocks
- Keep a pointer to the first free block on the disk and cache this in memory.
- The first block contains a pointer to the next free block and so
- Cannot get contigous space easily.
- No waste of space.
- Traversing the list takes time.
- Usually, the first free block is allocated
- FAT incorporates free-block accounting into the allocation data struct

Grouping:

- Store addresses of n free blocks in the first free block;
- nth block contains the addresses of another n free blocks.
- Addresses of large number of free blocks can be found quickly

Counting:

- Keep the address of the first free block and the number n of free contigeous blocks that follow the first block.
- Each Entry - a disk address and a count.
- Mostly the overall list is shorter.
- These entries can be stored in a balanced tree, rather than a linked list, for efficient look up, insertion and deletion.



Linked free space list

Bit Vector

11000011000000111001111111000111

Grouping

Block 2 → 3,4,15

Block 5 → 6,9,10

Block 10 → 11,12,13

Block 13 → 14,28,25

Block 25 → 26,27

Counting

2	4
8	5
17	2
25	3

File Types - Name, Extension

(5)

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

(12)

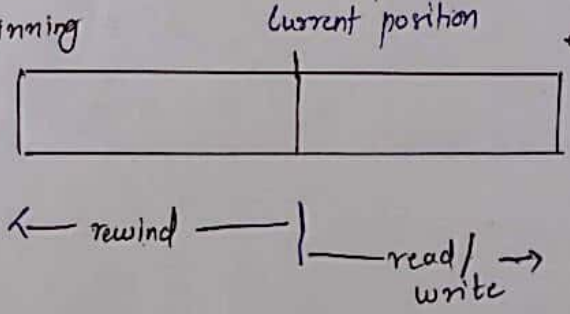
Access Methods:

Information stored in files must be accessed.
Different methods of access is possible
- Some system support only one method.
- Some support multiple methods, right one chosen based on application.
- Sequential access, direct access, indexed sequential access.

Sequential Access:

- Simplest, most common access
- Information processed one record after another

read next
write next
reset



Direct Access

- It is also called as Relative access
- File is made up of fixed-length logical records that allow programs to read and write records in no particular order
- The direct-access method is based on a disk model of a file, since disks allow random access to any file block
- For direct access, the file is viewed as a numbered sequence of blocks or records.
- Read block 14, read block 53 and then write block 7.
- There are no restrictions on the order of reading/writing for a direct-access file.
- The files is viewed as a numbered sequence of blocks or records.
- Useful for immediate access to large volume of data.
- Easy to read, write and delete a record.

```
read n
write n
position to n
read next
write next
rewrite n.
```

n - relative block number.

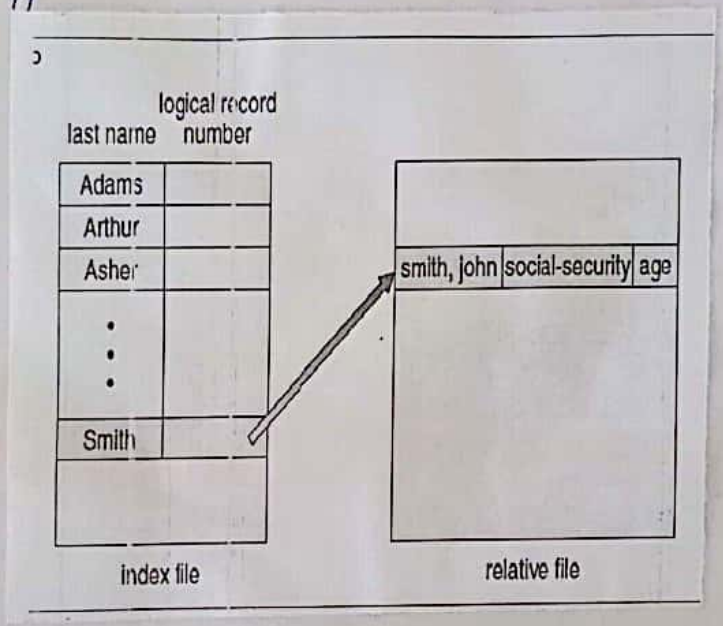
- read (n) rather than read next()
- write (n) rather than write next()

Simulation of Sequential Access on a Direct-access file.

Sequential Access	Implementation for Direct Access
reset	cp=0.
read next	read cp; cp=cp+1
write next	write cp. cp=cp+1.;

Other Access Methods :

- An index can be built which contains pointers to various blocks
- To find a record, search the index, use the pointer to access the file and the desired record.
- When file size becomes large, the index file also become large
- An index can be maintained for the index file
  - primary index file will have pointers to secondary index files.
  - Secondary index file point to blocks.



Explain about Kernel I/O subsystem and transforming I/O hardware operations? (14)

- Kernel is an essential part of operating system which resides in main memory. It provides various services like creation, termination, communication and execution of process.
- Kernel I/O subsystem is responsible to provide many services related to I/O.

Scheduling:
- Kernel schedules a set of I/O requests to determine a good order in which to execute them.
- When an application issues a blocking I/O system call, the request is placed on the queue for that device.
- The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.

Buffering
- Kernel I/O subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a device with an application operation.
- Buffering is done to cope with a special mismatch b/w the producer and consumer of a data stream or to adjust b/w devices that have different data transfer sizes.

Caching:
- Kernel maintains cache memory which is a region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.

Spooling and Device Reservation:
- A spool is a buffer that holds output for a device, such as printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, if spooling is managed by a system daemon process.

Error Handling:
- An operating system that uses protected memory can guard against many kinds of hardware and application errors.

I/O protection:

- The I/O system must protect against accidental or deliberate erroneous I/O.
- User applications are not allowed to perform I/O in usermode.
- All I/O requests are handled through system calls that must be performed in Kernel mode.

Kernel Data Structures:

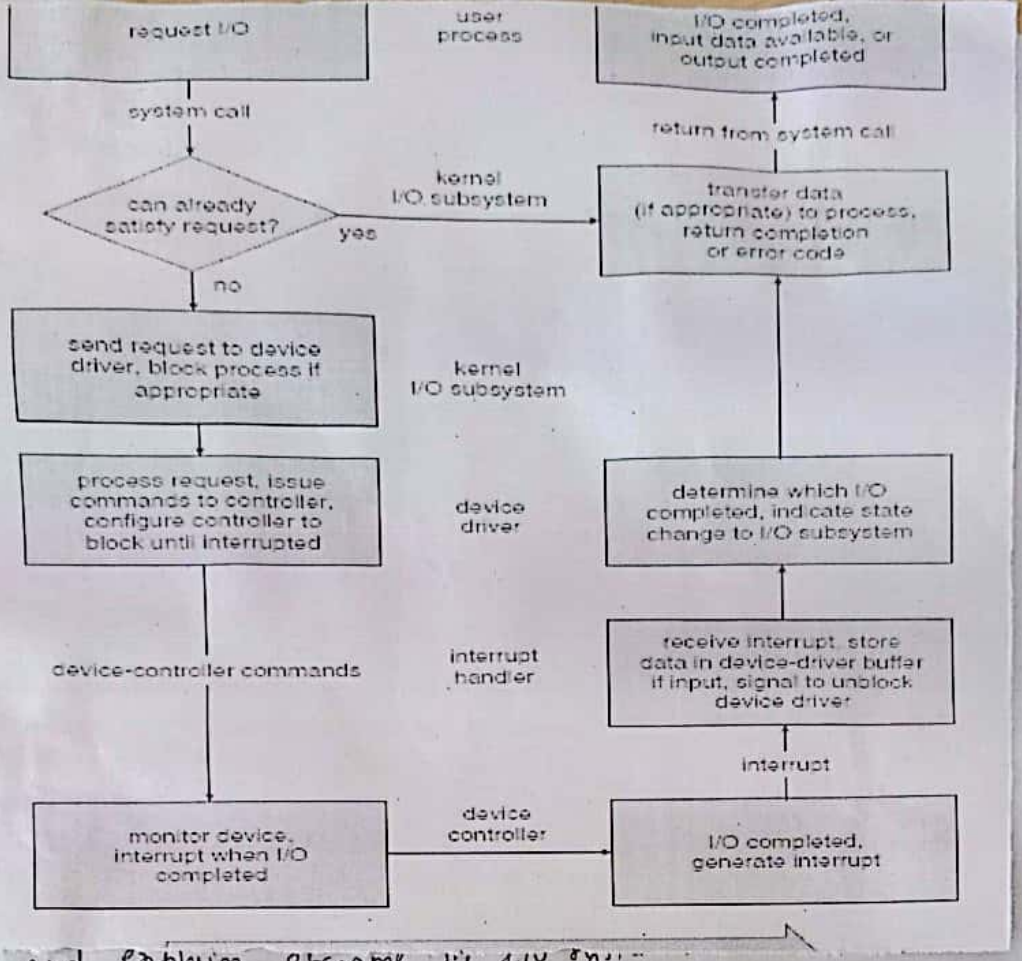
- The Kernel maintains a number of important data structures pertaining to the I/O system, such as the open file table.
- These structures are object oriented, and flexible to allow access to a wide variety of I/O devices through a common Interface

Transforming I/O requests to Hardware operations:

The OS uses some hardware mechanisms for processing I/O.

Consider a processing of read request:

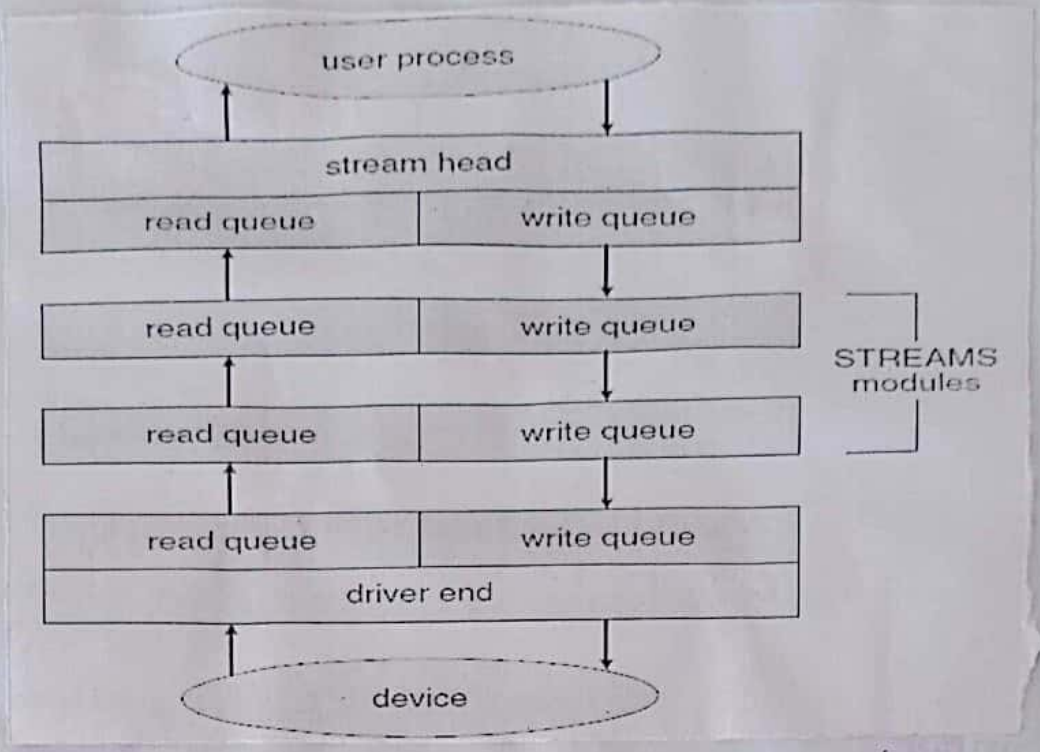
1. A process issues a blocking read() system call to file descriptor.
2. If the data is already available in buffer cache, the data is returned and I/O request is completed.
3. If the data is not available, the process is removed from the queue and placed on the wait queue and then the I/O request is scheduled.
4. The device driver allocates Kernel buffer space to receive data and schedules I/O.
5. The device controller operates the device hardware to perform data transfer.
6. The transfer is managed by a DMA controller, which generates an interrupt when the transfer completes
7. The interrupt handler performs the interrupt processing by using interrupt-vector table.
8. The device-driver receives the signal, determines the requests status and informs to the Kernel I/O subsystem.
9. The Kernel performs the data transfer and the process resumes execution at the completion of system call.



State and explain streams in I/O sys.

STREAMS:

- Stream is used as full duplex communication between a device driver and user level process.
  User process is interfaces with STREAM head, The device driver interacts with the device end.
- A stream consists of
  1. STREAM head interfaces with the user process.
  2. Driver end interfaces with the device
  3. Zero or more STREAM modules between them.
- Each module contains a read queue and write queue. Message passing is used to communicate b/w queues.
- zero or more modules can be pushed on to the stream, using ioctl(). These modules may filter and/or modify the data as it passes through the stream.
- Each module has a ~~...~~
  Flowcontrol can be optionally supported, in which each case module will buffer data until the adjacent module is ready to receive it.
- Without flow control, data is passed along as soon as it is ready.



- User proces communicate with the stream head using other read() and write()
- Streams I/O is asynchronous (non-blocking) except for the interface b/w the user process and the stream head.
- adv - flexibility of writing device drivers.
- The device driver must respond to interrupts from its device. If the adjacent module is not prepared to accept data and the device drivers buffers are all full, then data is typically dropped.
- Streams are widely used in UNIX, and are the preferred approach for device drivers.
  For example: UNIX implements sockets using streams.

- Learning Objectives: (6) a) (16)
- Layered file system
- On-disk structures
- In-memory structures
- Directory Implementation.

File-System Structure:

- A file is a collection of related information.
- File system resides on secondary storage (disks)
- File systems provide efficient and convenient access to the disk by allowing data to be stored, located and retrieved easily.

Application Programs
⇓
Logical File System
⇓
File Organization Module
⇓
Basic file System
⇓
I/O control
⇓
Devices

I/O Control Level.

- I/O Control Level:
  - device drivers and interrupt handlers to transfer information b/w main memory and disk system.
  - Example of input to a device driver - retrieve block 123
  - output from a device driver - low-level hardware specific instructions to the disk controller.
- Basic File System -
  - Issues generic commands to the device driver to read and write physical blocks on the disk.
  - * Manages memory buffers and caches.
  - * A block in the buffer can hold the contents of a disk block
  - * Cache holds frequently used file-system metadata.

File Organization Module :
- Knows about files and their logical and physical blocks.
- Knows the location of the file in the disk.
- Logical blocks are numbered 0 to N.
- physical blocks do not match the logical numbers.
- Has a free-space manager, which tracks unallocated blocks.

Logical File System :
- Manages metadata Information
- Includes all details about a file except the actual content of the file.
- Manages the directory structure.
- Maintains file-structure via file-control blocks.
- File control block (FCB) (inode in UNIX) has information about a file - owner, size, permissions, location of file contents.

Advantages :
- Duplication of code is minimized
- I/O control and basic file system code can be used by multiple file systems.
- Each file system can then have its own logical file system and file-organization modules.

Disadvantages :
- Can introduce more operating-system overhead, resulting in decreased performance
- Decision about how many layers to use, what each layer should do is a challenge.

- Many file-systems are in use today
  - UNIX file system, FAT, FAT 32, NTFS, ext3, ext4, Google

File System Implementation

- Several on-disk and in-memory structures are used
- on-disk structures contain information about
  - How to boot an OS stored there, total number of disk blocks, number and location of free disk blocks, directory structure, individual files.
- In-memory structures:
  - File-system Management, caching

On-Disk Structures

- Boot control Block
  - Contains information needed to boot an operating system.
  - If the disk has no OS, this block is empty.
  - Usually, the first block of the volume
  - In UNIX, called boot block.
  - In NTFS, called partition boot sector.
- Volume Control Block:
  - Contains volume (partition) details
  - Number of blocks in partition
  - Size of blocks
  - Free-block count and free-block pointer
  - In UNIX, called superblock
  - In NTFS, it is stored in the master file table.
- Directory structure (for each file system)
  - To organize files.
  - In UNIX, includes file names and associated inode numbers
  - In NTFS, it is stored in the master file table.
- per-file Control block (FCB):
  - Contains details about the files.
  - has a unique identifier number to allow association with a directory entry
  - In NTFS, it is stored in the master file table.

In-Memory Structures

- Mount table:
  - Information about each mounted volume
- Directory-structure cache
  - Holds directory information of recently accessed directories
- System-wide open-file table
  - contains a copy of the FCB of each open file
- per process open-file table:
  - Points to the appropriate entry in the system-wide open-file table.
- Buffers
  - To hold file-system blocks when they are read from disk or written to disk.

In memory Structures - Create a New file

- Application program calls logical file system.
- Logical file system knows the name of the directory structures
- Allocates a new FCB.
- System reads appropriate directory into memory.
- Updates the directory with the new file name and FCB and writes back to disk.

A Typical File Control Block

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

In-Memory Structures - Open an Existing File

- open() call passed a file name to the logical file system
- The call first searches the system-wide open file table to see if the file is already in use by another process
- If it is, a per-process open-file table entry is created pointing to the existing system-wide open-file table entry.

- If file is not already open, the directory structure is searched for the given file name.

- parts of directory structure are cached in memory
- Once the file is found, the FCB is copied into an entry in the system-wide open-file table in memory.
- The FCB entry also keeps track of the number of processes that have opened the file
- An entry is made in the per-process open-file table
- This entry points to the system-wide open-file table.
- This entry also has information about where the next read/write should be done on the file, access mode in which the file is open.
- open() returns a pointer to the entry in the per-process file-system table
- All file operations after the open() using this pointer
- In UNIX this pointer is called file descriptor (file handle in windows)

In-Memory Structures - close a file

- When a process closes a file, the per-process open-file table entry is removed.
- The count in the system-wide open-file table entry is decremented
- When the count becomes zero, the updated metadata is copied to the directory structure in the disk.

(6)b Directory Implementation

- Linear list of file names with pointers to the data blocks.
  - Simple to program
  - Time-consuming to execute.
  - To create a new file, search the directory if another file with a similar name exists.
  - If no such file, a new entry is added to the end of the directory.
  - To delete a file, search the directory for the file and release the space allocated to it.

To reuse this deleted entry
- Mark the entry as unused by assigning it a special name.
- Attach it to a list of free directory entries.
- Copy the last entry in the directory to the freed location and decrease the length of the directory
- Maintain the entries as a linked list to reduce the time required for deletion.

Disadvantage of a linear list:
- Finding a file required linear search - will make access slow.
- Operating systems implement a software cache to store the most recently used directory information.
- Maintaining a sorted list allows a binary search and reduces the search time.

Page No.:

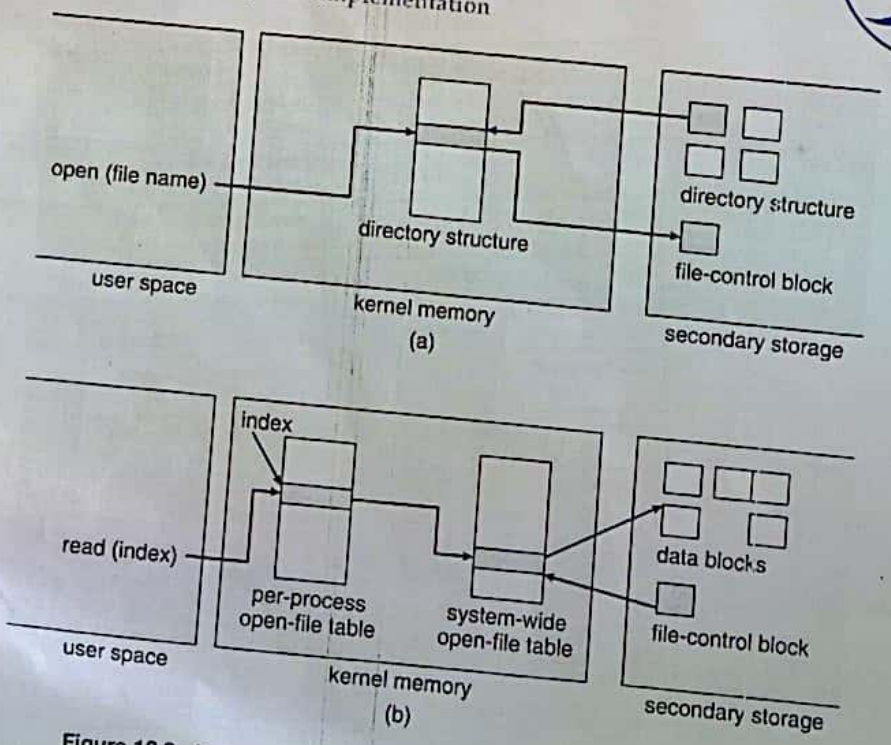


Figure 12.3 In-memory file-system structures. (a) File open. (b) File read.

file-system tables

Use a hash data structure:

- A linear list stores the directory entries, a hash data structure is also used.
- Hash table takes a value computed from the file name and returns a pointer to the file name in the linear list.
- Reduces the search time
- Provision must be made for collisions - two file names hash to the same location.

(4) a Efficiency & performance of I/O systems

Efficiency dependent on:

- disk allocation and directory algorithms
- types of data kept in file directory entry.

performance:

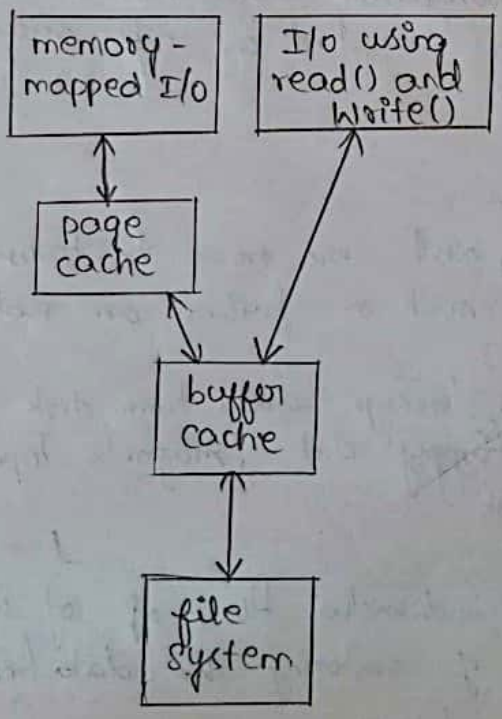
- disk cache - separate section of main memory for frequently used blocks.
- free-behind and read-ahead - techniques to optimize sequential acc
- improve PC performance by dedicating section of memory as virtual disk or RAM disk.
- page cache:

  A page cache caches pages rather than disk blocks using virtual memory techniques.

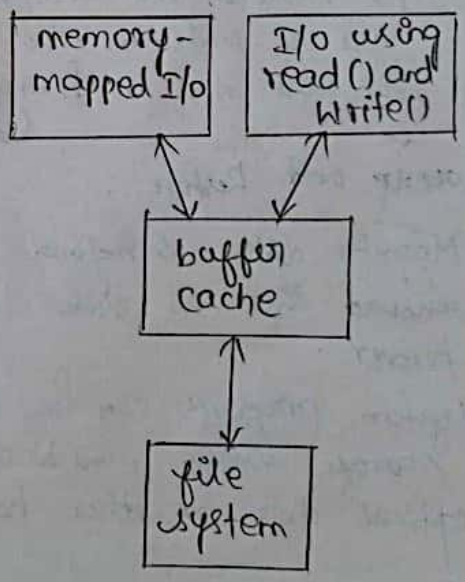
  Memory-mapped I/O uses a page cache.
  Routine I/O through the file system uses the buffer (disk) cache

- Unified Buffer cache:

  A unified buffer cache uses the same page cache to cache both memory-mapped pages and ordinary file system I/O.



I/O without a unified buffer cache



I/O with a unified buffer cache

Recovery:

Files and directories are kept both in main memory and on disk, and care must be taken to ensure that system does not result in loss of data or in data inconsistency.

The methods which are used for recovery of files and directories

1. Consistency checking
2. Backup and restore
3. Log structured file systems.

Consistency checking:

- The directory information in main memory is generally more up to date than is the corresponding information on the disk because cached directory information is not necessarily written to disk as soon as the update takes place.
- Frequently, a special program is run at reboot time to check for and correct disk inconsistencies.
- The consistency checker - a systems program such as
  - chkdsk in MS-DOS - Compares the data in the directory structure with the data blocks on disk and tries to fix any inconsistencies it finds.
  - The allocation and free-space-management algorithms dictate what types of problems the checker can find and how successful it will be in fixing them.

Backup and Restore:

- Magnetic disks sometimes fail, and care must be taken to ensure that the data lost in such a failure are not lost forever.
- System programs can be used to backup data from disk to another storage device, such as a floppy disk, magnetic tape, optical disk or other hard disk.
- Recovery from the loss of an individual file, or of an entire disk, may then be a matter of restoring the data from backup.
- A typical backup schedule as follows:

Day1 : Copy to a backup medium all files from the disk. This is called a full backup.

Day2: Copy to another medium all files changed since day 1. This is an incremental backup.

Day3: Copy to another medium all files changed since day 2

:

Day N: Copy to another medium all file changed since day N-1. Then go back to day 1.

Log-Structured File Systems:

Log-based transaction-Oriented (a Journaling) filesystems borrow techniques for database, guaranteeing that any given transaction either completes successfully or can be rolled back to a safe state before the transaction commence.

- All metadata changes are written sequentially to a log.
- A set of changes for performing a specific task (moving a file) is a transaction.
- As changes are written to the log they are said to be committed, allowing the system to return to its work.
- In the meantime, the changes from the log are carried out on the actual file system and a pointer keeps track of which changes in the log have been completed and which have not yet been completed.
- When all changes corresponding to a particular transaction have been completed, that transaction can be safely removed from the log.
- At any given time, the log will contain information pertaining to uncompleted transaction that not yet been completed.
- From the log, the remaining transactions can be completed or if the transaction was aborted, then the partially completed changes can be undone