File Management —:

/boot	super-block	inode list	data blocks

FS Layout

Boot Block : Occupies beginning of filesystem, typically first sector. Contains bootstrap code read into memory to boot (init, pid 0)

Super Block : Information about filesystem, free blocks, pointers to inode list.

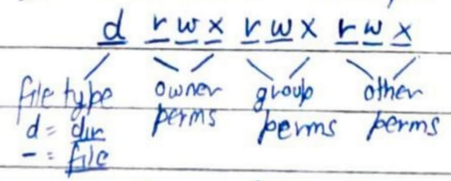
Inode List : List of inodes, where inodes provide file specific information.

Data Blocks : Data of files.

× Inode : Contains necessary info for processes accessing the file such as file ownership, access rights, location of file data & file size in the filesystem

- Exist in static copy form on disk (Secondary), from where kernel reads them into an in-core form for manipulation.
- Disk Inodes contain :

1) File owner identifier :



- Ownership divided between individual owner & group (set of users given access)
- SU has access to all files

2) File Type : regular, directory, FIFO, socket, etc.

3) File Access Perms -:
   - Owner, Group owner, Other users
   - Each class given access rights for read, write & execut

4) File Access Times -:
   - Time of last modification (mtime), last access (atime) & last inode modification (ctime)

5) # Links to file -:
   # of symbolic (soft & hard) links to the file (aliases)

6) Table of Contents for disk adresses of file data
   - File stored in discontiguous disk blocks, accessed via kernel as logical byte stream. Blocks identified by inode data.

7) File Size -:
   - File data addressable by # of bytes from beginning of file, starting from byte offset 0.

- In-Core Inodes contain, in addition to disk inodes file fields :

8) Status of inode -:
   - Indication of properties :
     × Inode is locked or not.
     × A process is waiting for inode to be unlocked
     × In-core rep. of inode is different as a result of change of inode data.
     × In-core file rep. differs from disk as a result of change to file data.
     × File is a mount point.

9) Logical device # of filesystem containing file

10) Inode # : Stored as linear array on disk, so kernel identifies disk inode by position.

11) Pointers to other incore inodes :

12) Reference Count : # instances of file active,
    # of processes accessing instance.

× Kernel identifies inodes based on file system & inode # & allocates in-core inodes.

× Maps device & inode # into a hash queue & searches queue.

× On unsuccessful search, a new inode is allocated from a maintained free list.

○ $\text{block \#} = \left[\frac{(\text{inode \#} - 1)}{\text{\# inodes per blk.}} \right]$ + start ~~address~~ block of inode list.

(inode # = 1-based indexing)

○ byte offset = $\left[(\text{inode \#} - 1) \ \% \ \text{\# inodes per blk} \right]$ * size of disk inode.

(indicates byte offset of inode in block).

× In case of no inodes of free list, a process is terminated with an error instead of slepp to prevent starvation.

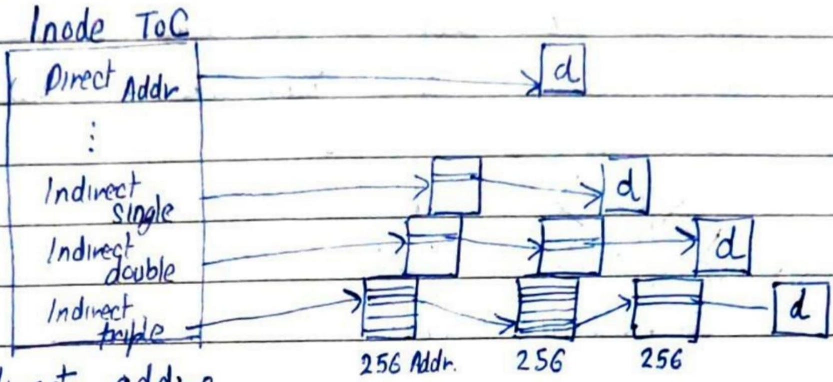
Releasing Inodes :

- On release, kernel decrements in-core reference count.
- When ref. count hits 0, if dirty write (in-core copy different from disk copy) then kernel initializes async write.
- Inode is then added to free list.
- If link count is 0, disk blocks for inode is freed, followed by freeing of inode.

Structure of Regular File -:

× Inode's table of contents contain block numbers containing file data in discontigious blocks.

- Prefer discontiguous block-based allocation for efficiency & reduced fragmentation. One block at a time is allocated.
- Table of Contents structure :



10 direct addrs.

1 single indirect → 256 direct blks.

1 double indirect → 256 single indirect

1 triple indirect → 256 double indirect (16M)

For 1Ki blocks : (1Ki = 1024)

10 direct blocks = 10KiB

1 single indirect = 256 KiB

1 double indirect = 64 MiB

1 triple indirect = 16 GiB (16,777,216 B)

Addr = 0 : Indicates absence of data block.

Eg Let size of block = 1024 B.

To access byte offset 9000, process needs to access block address = $\lceil 9000/1024 \rceil \approx 9$~~0~~ from ToC

leading to data block = ToC[9] = 367 (say).

To access byte offset 350000, process needs to access block address = $\lceil 350000/1024 \rceil = 342$.

which lies in double indirect block 0 at address 76

so required data block = $ABlock_{D1}[ToC[12]] = ABlock_{D1}[76$

⊗ Block address = 1 + Block offset in list.

Directories —:

x Provides hierarchical structure to filesystem.
x Aids in conversion of file name to inode #.
x Initial access : open, chdir, link syscalls.
x Inode of '/' = 0.

/<path> = absolute path
<|..|.> <path> = relative path

directory perms —: r : Search dir.
w : Write dirs & files.
x : Open, access dir,

Eg : Inodes in '/home' :

Byte Offset	Inode #	File
0	12	~~etc~~ .
16	16	..
32	162	etc

Eg : accessing /a/b/c
Fetch inode for /, given permission is available.
Search for inode # of a, given it exists.
Fetch inode for /a, given " " ".
Search for inode # of b, " " ",
Fetch inode of /a/b, " " " ".
Search for inode of c, " " ",
Fetch inode of /a/b/c.

x Inode of CWD stored in U-Area of a process.

E.g. Fetch /etc/ passwd

× Fetch inode with # 0 (Inode for /), (working dir = /).

× Search inode # for 'etc' in inode of / , inode# = 83 post reading inode of / & its disk blocks.

× 'etc' present, set working dir = /etc (inode # = 83)

× Fetch inode with # 83 (inode for /etc) (working dir = /etc)

× Search inode # for 'passwd' in inode of /etc, inode # = 2114

× 'passwd' present, set working path = /etc/passwd (inode # = 2114)

× No more path components, return inode # = 2114.

Super Block -:

× Holds information about :
× Size of filesystem
× Free Blocks (Count)
× List of Free Blocks (available)
× Index of next free blocks list's free block
× Size of inode list
× # of free inodes
× List of usable free ~~blocks~~ inodes
× Index of next free inode
× Flag for information about modification
× Lock fields.

× Fetching free inode for assignment to file (new file)
Inefficient way : Search disk inode list for free disk inode (type = 0) & allocate (linear search, expensive)
Improvisation : Cache free inode #s in super block.

× Remembered inodes :
- When superblock is free (empty free inode list), kernel searches disk & places as many free inodes as possible on free inode list.

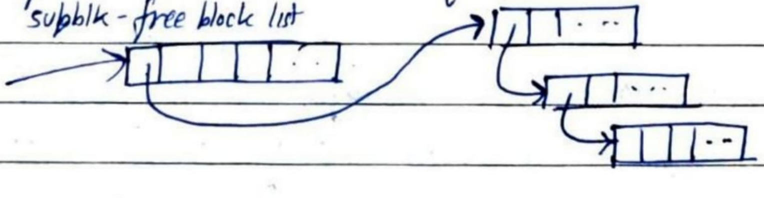
- When kernel needs more free inodes, search is resumed from last inode index that had been searched instead of the beginning. (remembered inode).
- Remembered Inode : Last inode from free inode list of super block. (Marked to trigger search).

Freeing Disk Inodes —: (ifree)

× On freeing of inode, add to superblock based on inode # : inode # lower than remembered inode implies update superblock free inode list & update remembered inode #. Inode # higher than
× remembered inode implies free inode not added to free inode list.
× Free inode list is unsorted

Allocating Disk Blocks —:

× Free disk blocks cached in superblock for allocation to files by kernel for processes.
× Data blocks stored in linked list, where each node contains array of free block #s & one entry points to index of next disk block node on list.



Linked list structure shifted (elements copied to parent) when last block of superblock list is to be allocated.

× Different structure for efficiency (discontiguous memory & large # of disk blocks & requirement of multiple blocks).