File Systems

CS 447
Monday 3:30-5:00
Tuesday 2:00-3:30

- Prof. RK Joshi, CSE, IIT Bombay

What is a file?

- Abstraction:
  - An ordered collection of data
  - May be realized by a physical mapping to disk blocks
- Attributes
  - (other than data itself) Name, type, location, size, protection, time identification, ownerships
- Operations
  - Create, write, read, reposition (seek), delete, truncate

Open-close (Session) Model

- Obtains an iterator (file pointer) on a file
- Identifies an active session with a file
  - Used for reference counting

File Access Methods

- Sequential Access
  - Read only in sequential order
  - Early OSs
  - May be acceptable for slow devices – backup tapes
- Random (Direct) Access
  - Without having to go through all previous records/data, access a specific location directly
- Indexed (key-based) access: can be built over direct access

Allocation of Disk Blocks to Files

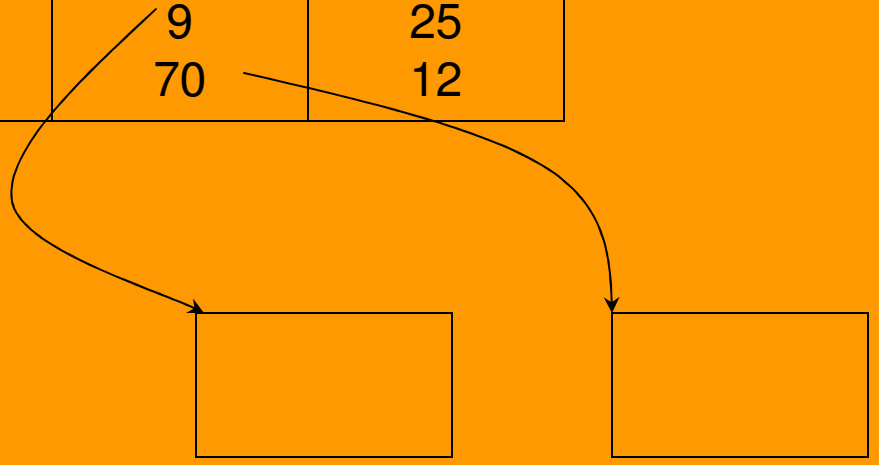
- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

Contiguous Allocation

- Access faster
- Simple to implement
- Max file size needs to be known
- Fragmentation possible
  - Compaction from time to time
- Best fit/worst fit/first fit
- Good for one time write media

Contiguous Allocation

File	Start	Size
A	9	25
B	70	12

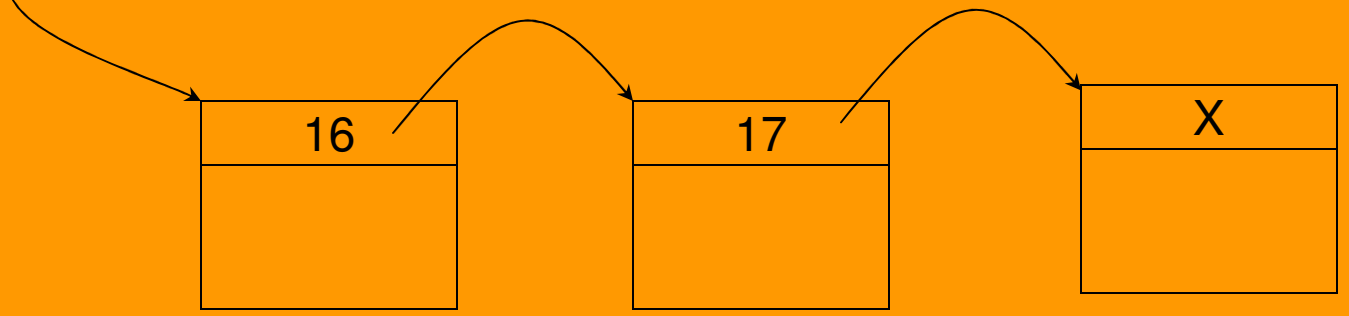


Linked Allocation

- Each block knows the next
- No fragmentation
- But performance may be a problem

Linked Allocation

File	Start	End
A	9	25
B	10	12



Microsoft FAT

- Linked block list is kept in the FAT rather than keeping the indices in actual blocks
- Once you read FAT in main memory, you have all the allocation information available – needed for fast access

FAT

Directory

Filename	Start Block	Other MD
afile	10	…

Linkage Table

0	.
.	.
.	.
.	.
10	12
11	25
12	11
.	.
.	.
25	30
.	.
30	-1
.	.
.	.
.	.
10000	

Indexed Allocation

- In linked allocation, direct accesses are not supported (if FAT is not used)
- In FAT, if one block gets corrupted, the subsequent blocks become unreachable
- → Index block can be used to solve these problems

Index Block

- For a given file, store all its block numbers in order
- Each file has an index block available
- Unix uses this scheme
  - Inode

Index block

Directory

Filename	Index Block	Other MD
afile	200	…

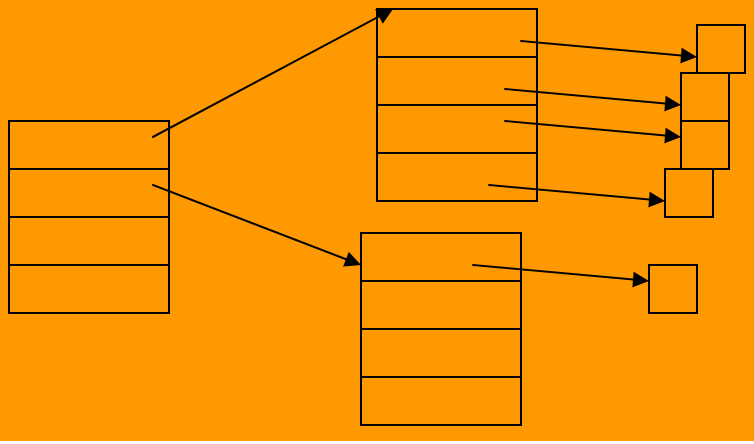
Index block of afile

.10
12
11
25
30

What if index block is not sufficient?

- Link index blocks as in linked allocation

- Have multilevel links

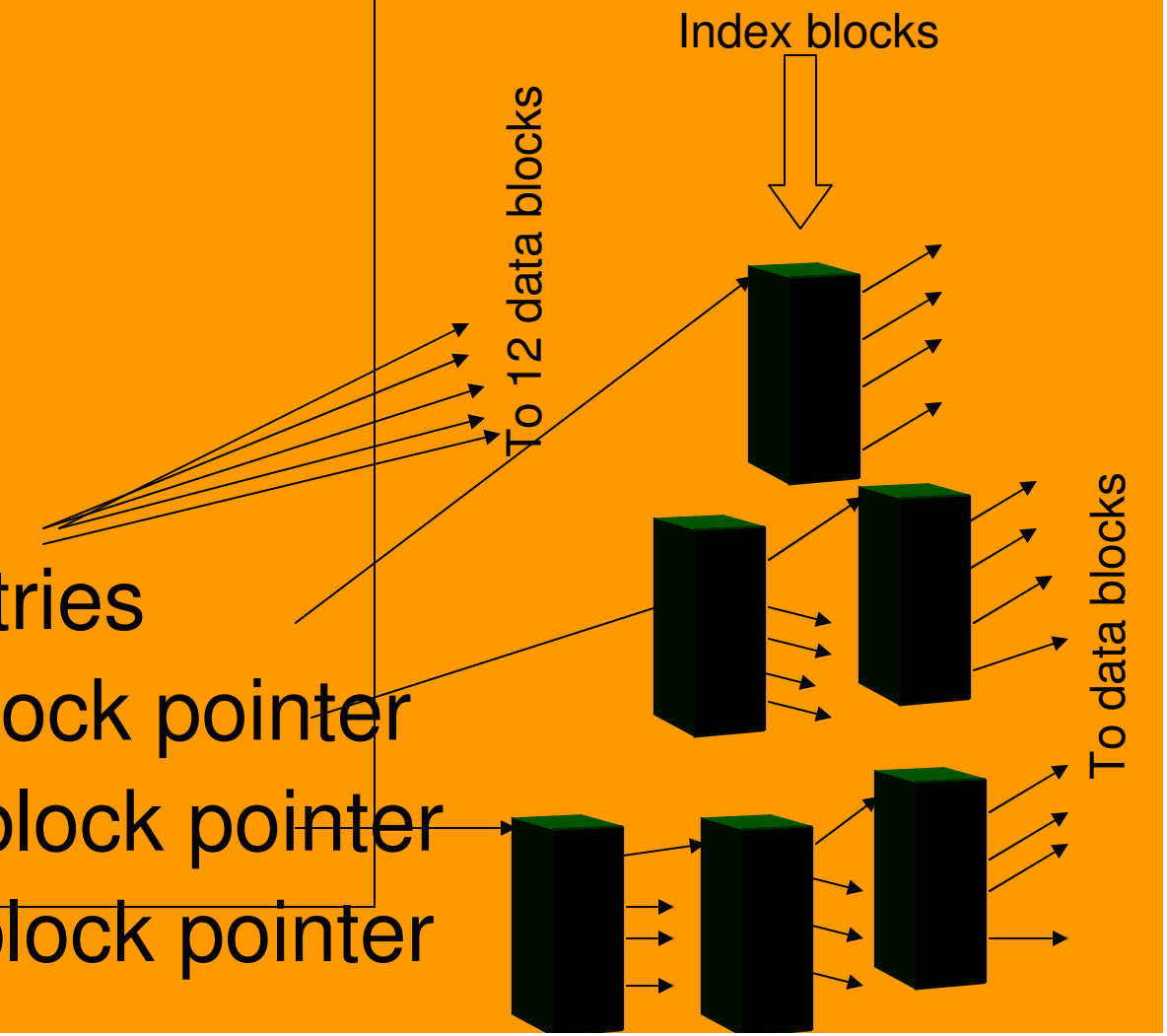


The Unix solution for large sized files

- A few entries in the index block
- The remaining entries in indirect index blocks

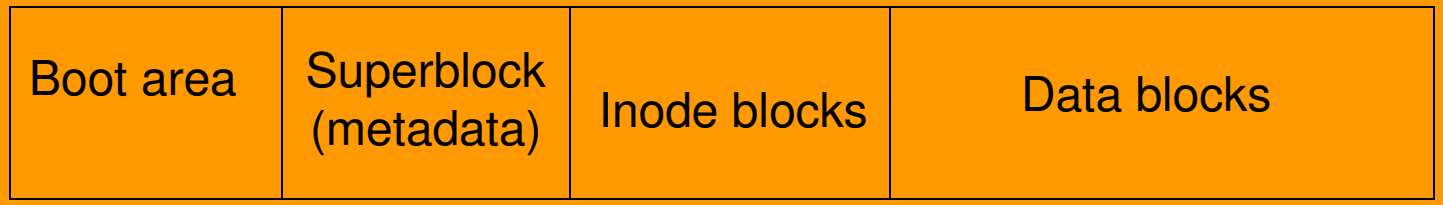
The Unix Inode

Mode
Owner
Time
Size
Count
DIB: K Direct entries
Single indirect block pointer
Double indirect block pointer
Tripple indirect block pointer



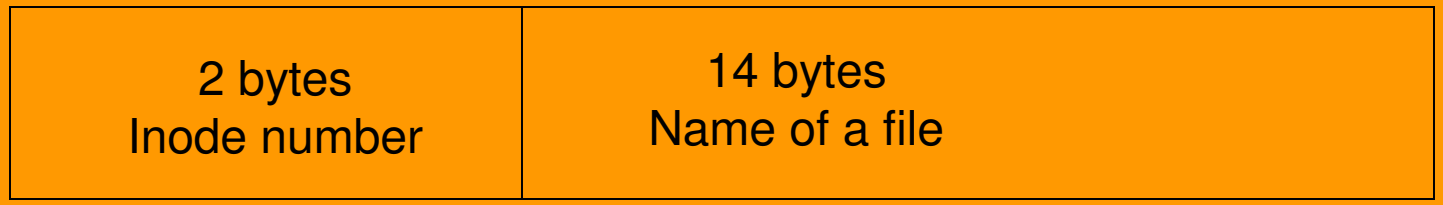
System V file system S5FS

- Disk map



System V file system S5FS

- An entry in directory

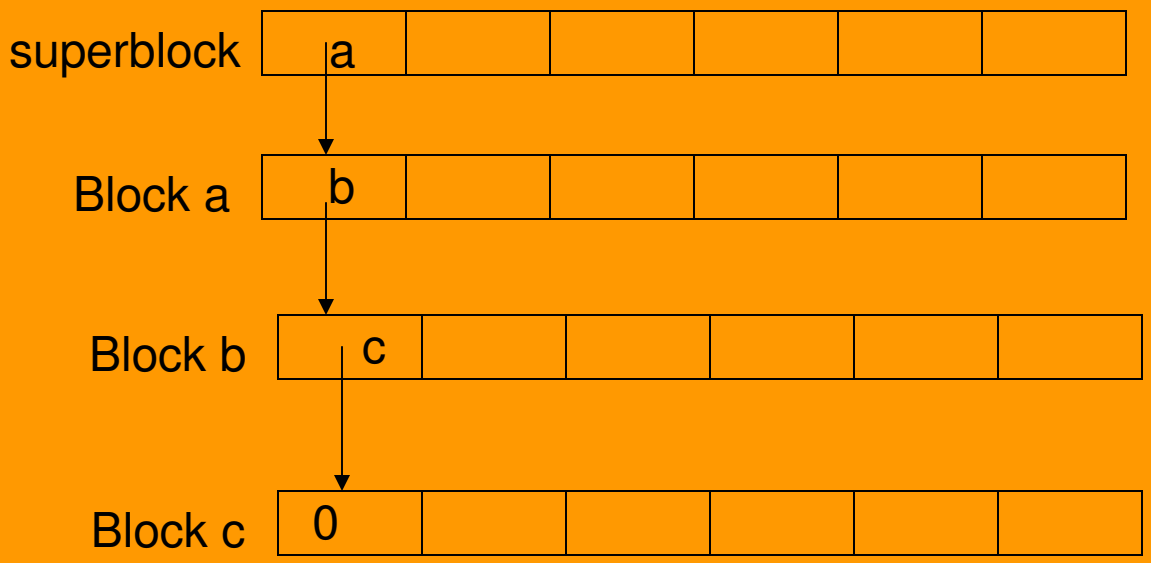


System V file system S5FS

- Superblock
  - Size of the FS in blocks
  - Size of the inode list in blocks
  - No of free blocks
  - No of free inodes
  - Free block list – has to be complete
  - Free inode list – can be partially complete

S5FS

- Free data block list



S5FS (System V file system) Inode

- Inode size = 64 Bytes
- Mode: 2
- Hardlinks: 2
- Uid: 2
- Gid: 2
- Size: 4
- Address block: 3*13 entries: 39
- Genno: incremented each time the inode is used for a new file: 1
- Atime: last access time: 4
- Mtime: last modification time: 4
- Ctime: last change time: 4

Address block on s5fs

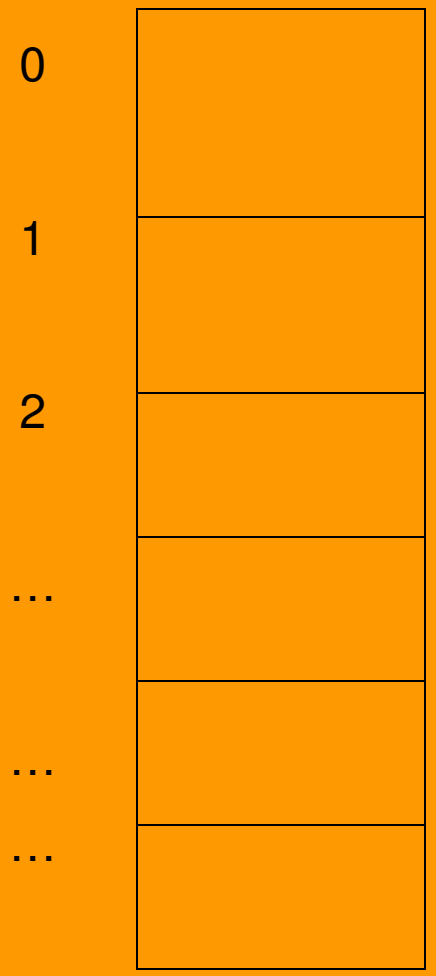
- 13*3 bytes
- 10 direct blocks
- 1 single indirect block
- 1 double indirect block
- 1 tripple indirect block

- Block size 1KB, 4 bytes per entry

2 bytes of Mode

- Type: if regular file/directory/block device/character device: 4
- suid, sgid: 2
- Perms: owner, group, other: 3*3 (rwx): 9
- Sticky bit: 1
  - directories can have sticky bit turned on so that
    - files created by other users cannot be deleted from the directory;
    - but any one with write permissions can write

Pathname translation



Inode list

Start from / i.e. node no. 2 (root directory)

If directory: search directory entries and locate next inode

If file: return inode

Pathname translation: example

- /home/fac/rkj/.bashrc
- Lookup (path p)

  currentinode = incore(car (p));

  case (currentinode.type == FILE) and (cdr (p)!=NULL): error, exit

  case (currentinode.type == FILE) and (cdr (p)==NULL): return currentinode;

  case (currentinode.type == DIR) and (cdr (p)!=NULL): return lookup (CDR (p));

  case (currentinode.type == DIR) and (cdr (p)==NULL): return currentinode;

incomplete

Pathname translation: example

- /home/fac/rkj/.bashrc
- Lookup (path p, inode ref)

  currentinode =getinode (car (p); ref);

  case (currentinode.type == FILE) and (cdr (p)!=NULL): error, exit

  case (currentinode.type == FILE) and (cdr (p)==NULL): return currentinode;

  case (currentinode.type == DIR) and (cdr (p)!=NULL): return lookup (CDR (p), currentinode);

  case (currentinode.type == DIR) and (cdr (p)==NULL): return currentinode;

/home/rkj/.bashrc /:2, home:10, rkj:15, .bashrc:20

- / → inode 2
- Lookup (home/.., 2)
- Getinode (home, 2) → 10
- Lookup (rkj/.bashrc, 10)
- Getinode (rkj, 10) → 15
- Lookup (.bashrc, 15)
- Getinode (.bashrc, 15) -> 20
- Return 20