Part 4

File Systems

4.1 Files
4.2 Directories
4.3 File system implementation
4.4 File system management and optimization
4.5 Example file systems

4.1 Files

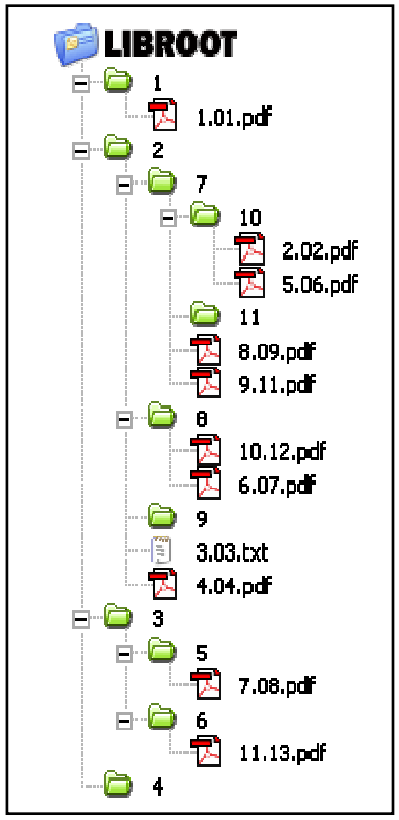
Files
File Concept

- Long-term Information Storage
  - Must store large amounts of data
  - Information stored must survive the termination of the process using it
  - Multiple processes must be able to access the information concurrently
- File
  - Used to store information on disks and other extenal media in units
  - Process can read them and write new ones if need be
- File system
  - Part of operating system dealing with files
  - Includes two independent parts: set of file and directory structure, organize and provide information about all files in system

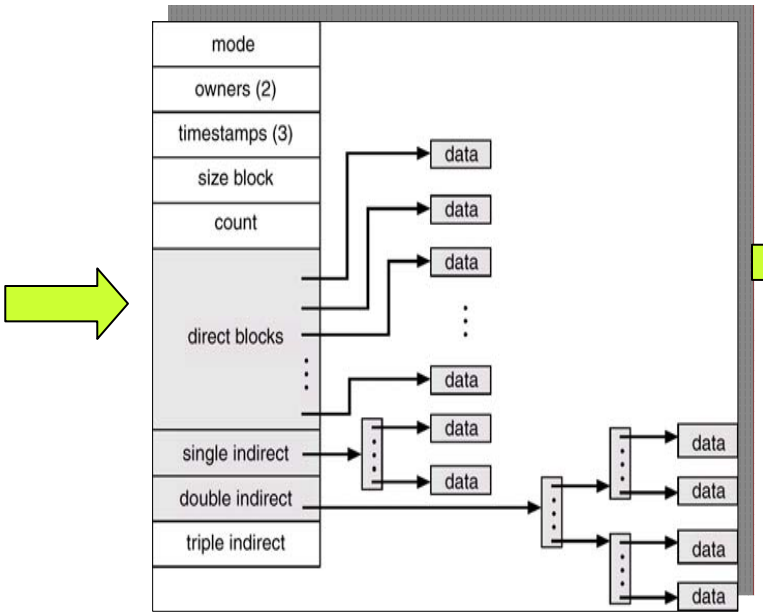
Files
File Concept

User Abstraction		Hardware Resource
Process/Thread		CPU
Address Space	⇐ **OS** ⇒	Memory
Files		Disk

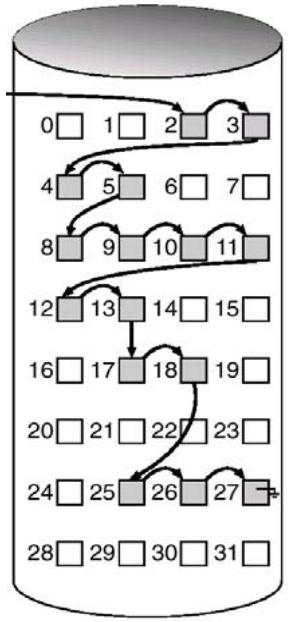
Files
File Concept



file system



operating system



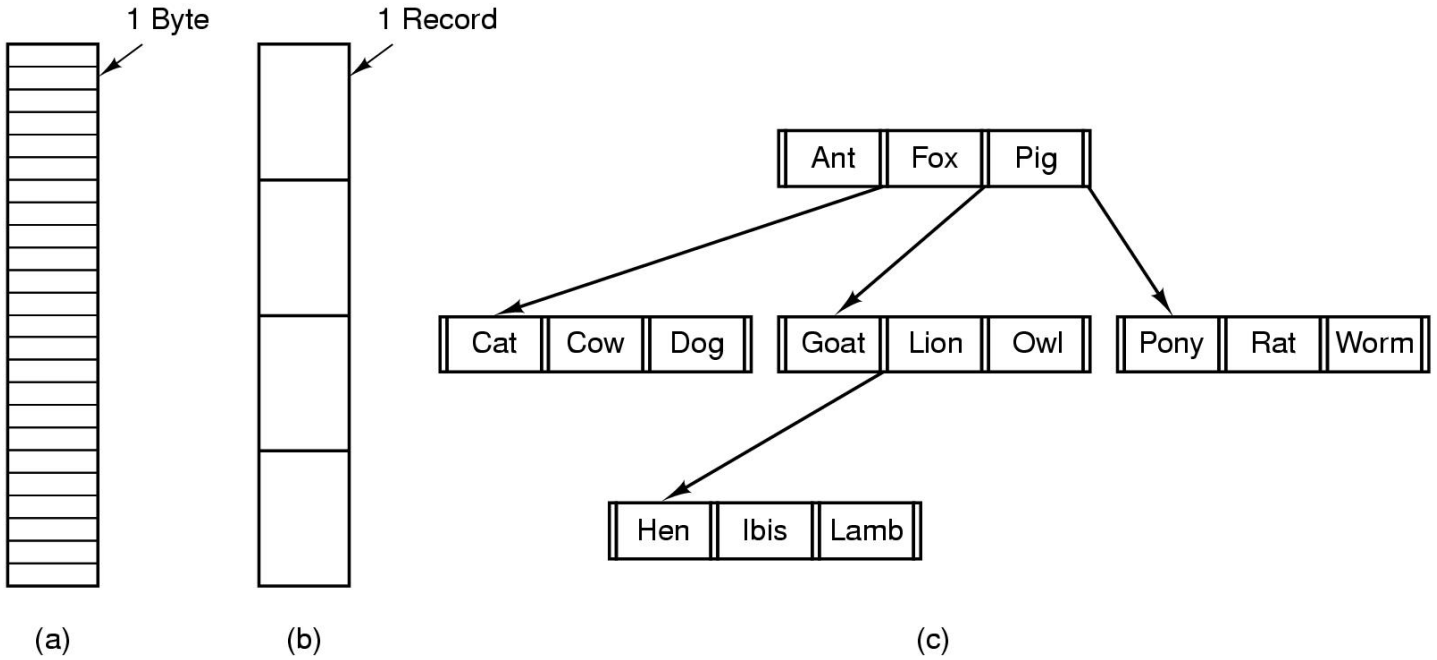
physical disk

Files
File Naming

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

Typical file extensions.

Files
File Structure

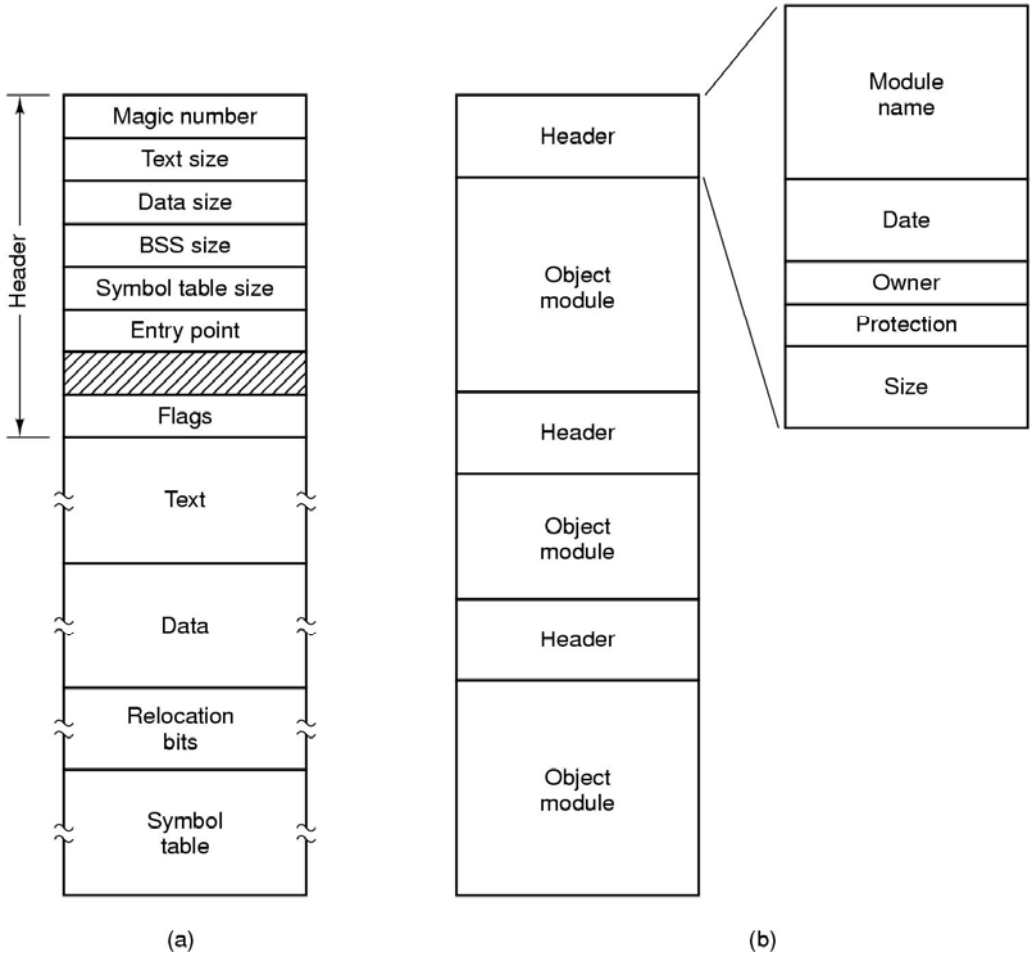


- Three kinds of files
  - (a) byte sequence
  - (b) record sequence
  - (c) tree

Files
File Types

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program
    - Source
    - Object
    - Executable
  - Regular, special (character, block)

Files
File Types



(a) An executable file (b) An archive

Files
File Access

- Sequential access
  - read all bytes/records from the beginning
  - cannot jump around, could rewind or back up
  - convenient when medium was mag tape
- Random access
  - bytes/records read in any order
  - essential for data base systems
  - read can be …
    - move file marker (seek), then read or …
    - read and then move file marker

Files
File Attributes

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Possible file attributes

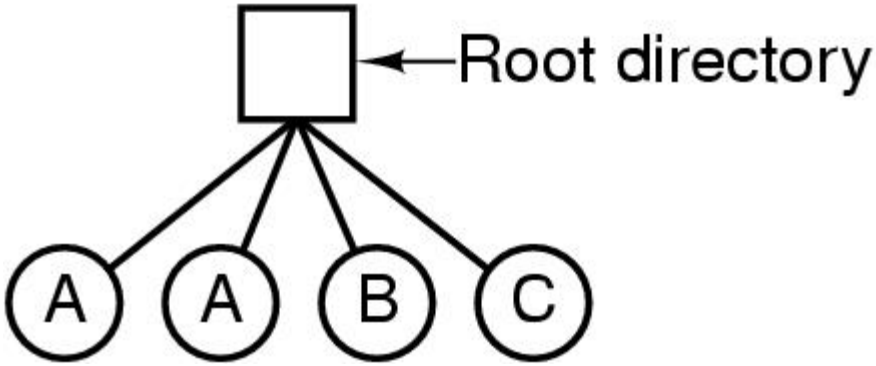
Files

File Operations

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set Attributes
11. Rename

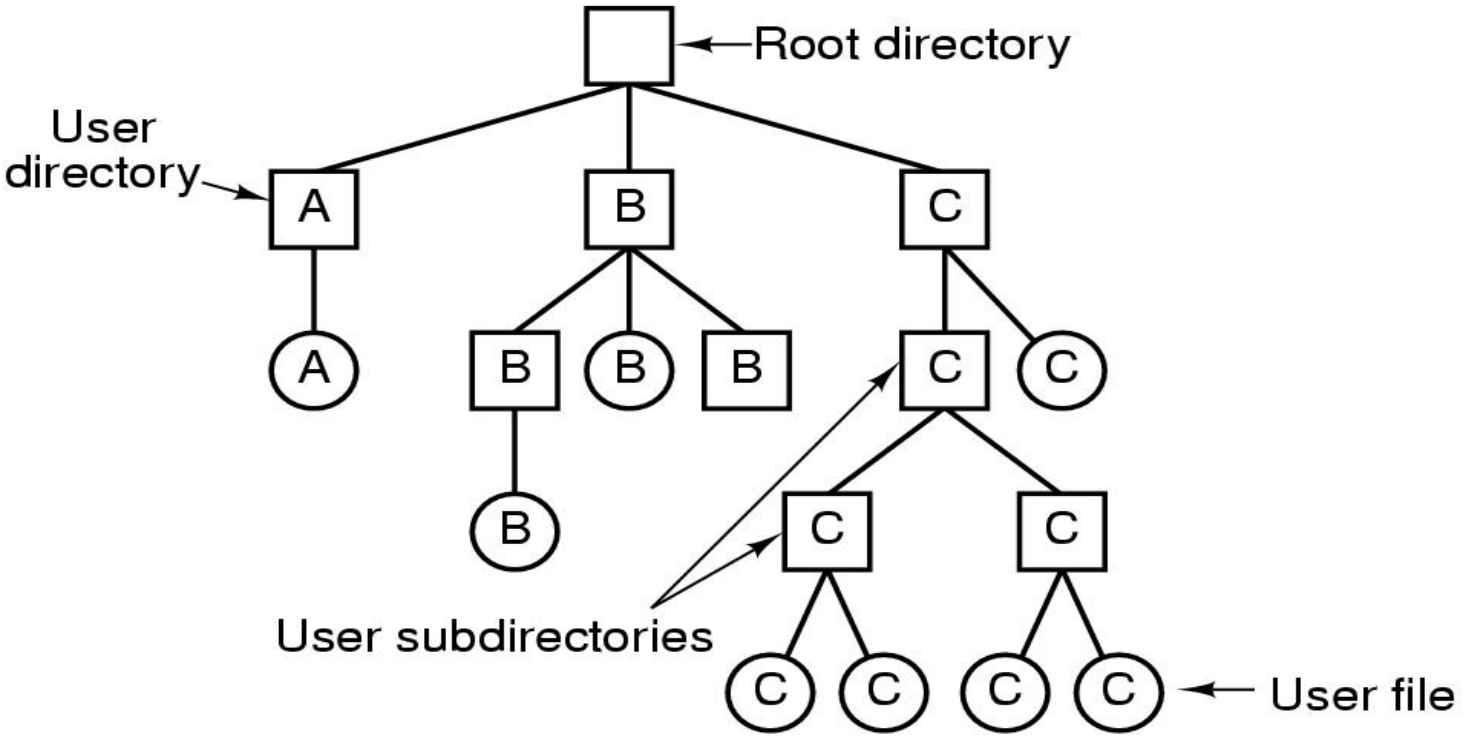
4.2 Directories

Directories
Single-Level Directory Systems



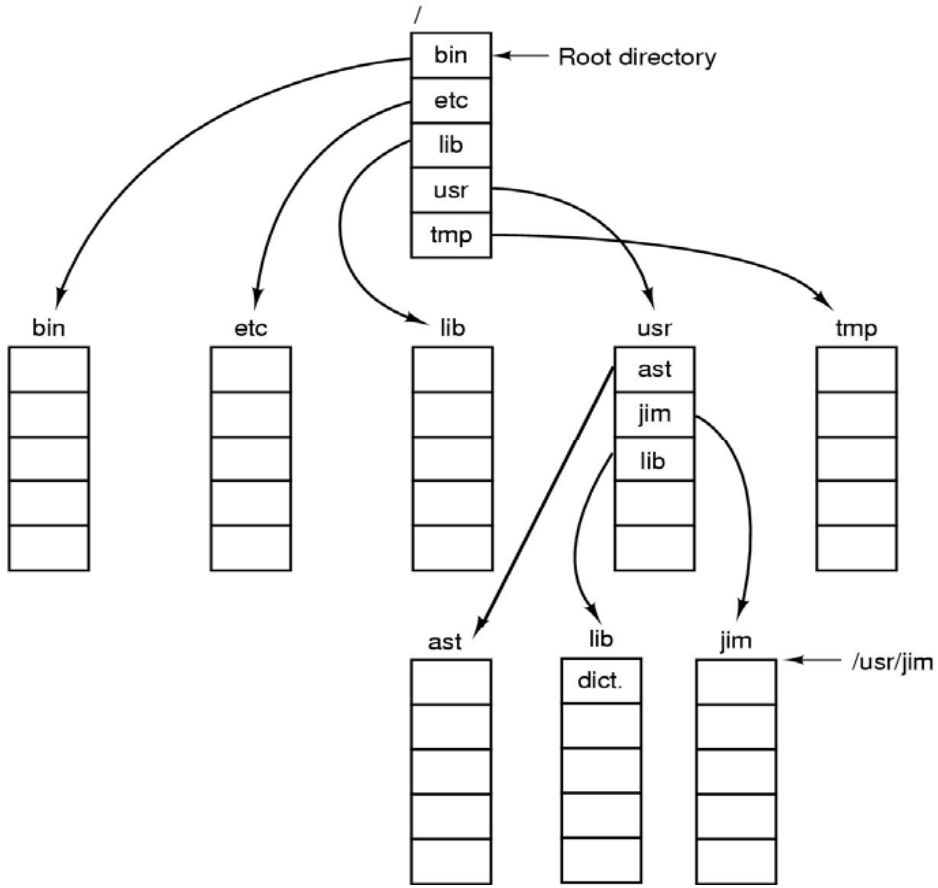
- A single level directory system
  - contains 4 files
  - owned by 3 different people, A, B, and C

Directories
Hierarchical Directory Systems



A hierarchical directory system

Directories
Path Names



A UNIX directory tree

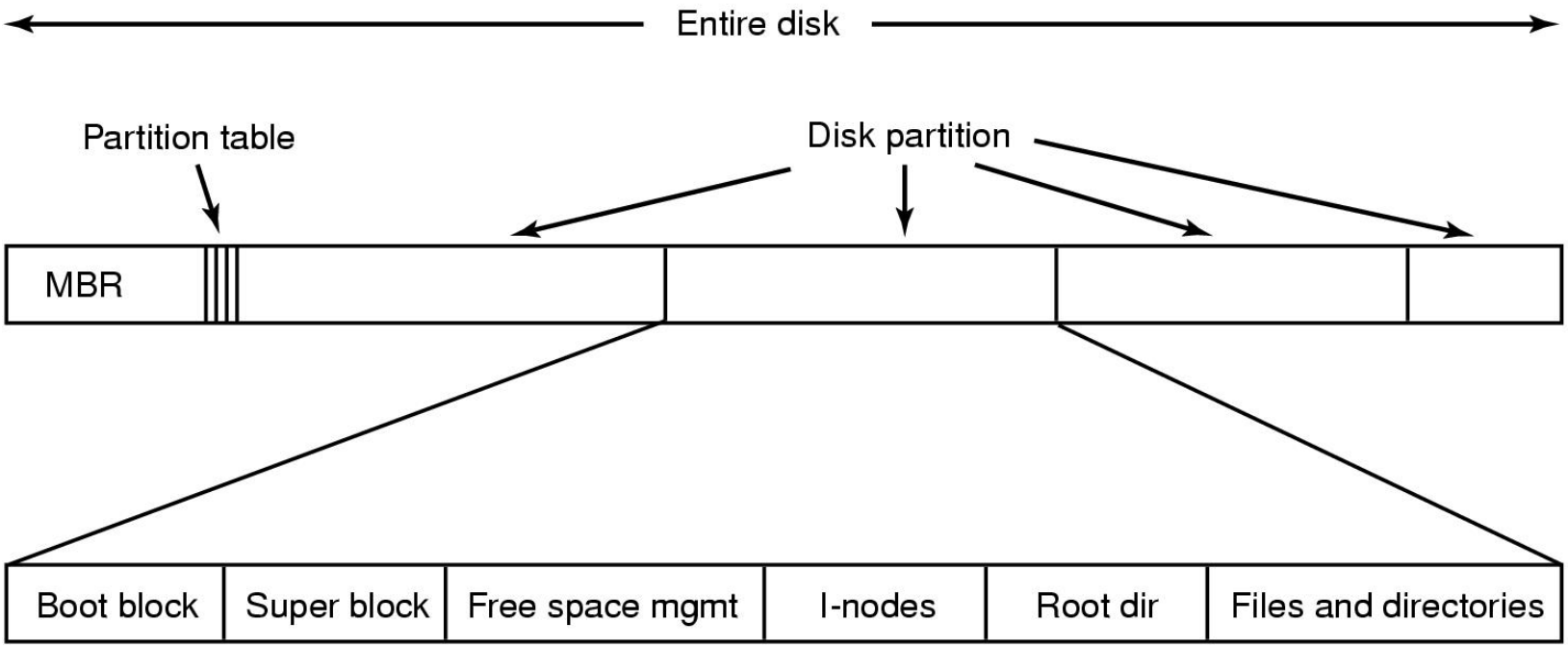
Directories

Directory Operations

1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link
8. Unlink

4.3 File System Implementation

File System Implementation



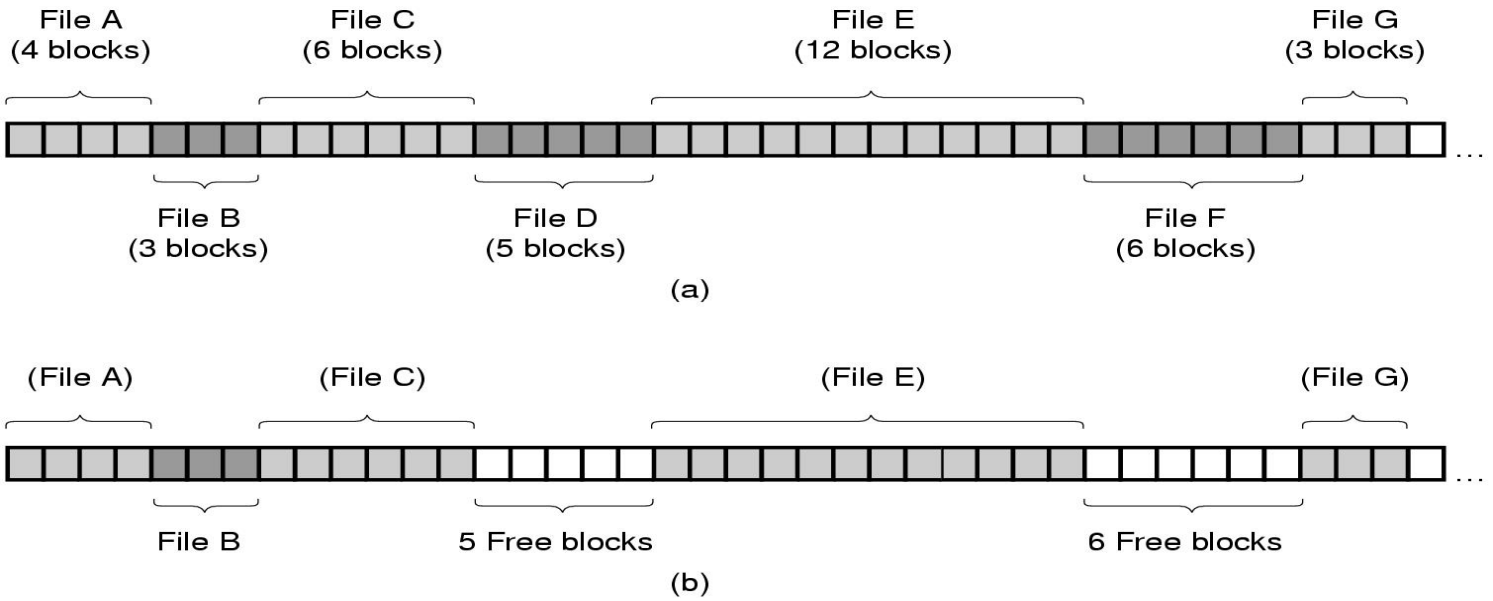
A possible file system layout

File System Implementation
Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
- Contiguous allocation
- Linked allocation
- Indexed allocation

File System Implementation

Allocation Methods: Contiguous allocation (1)



(a) Contiguous allocation of disk space for 7 files

(b) State of the disk after files *D* and *E* have been removed

File System Implementation

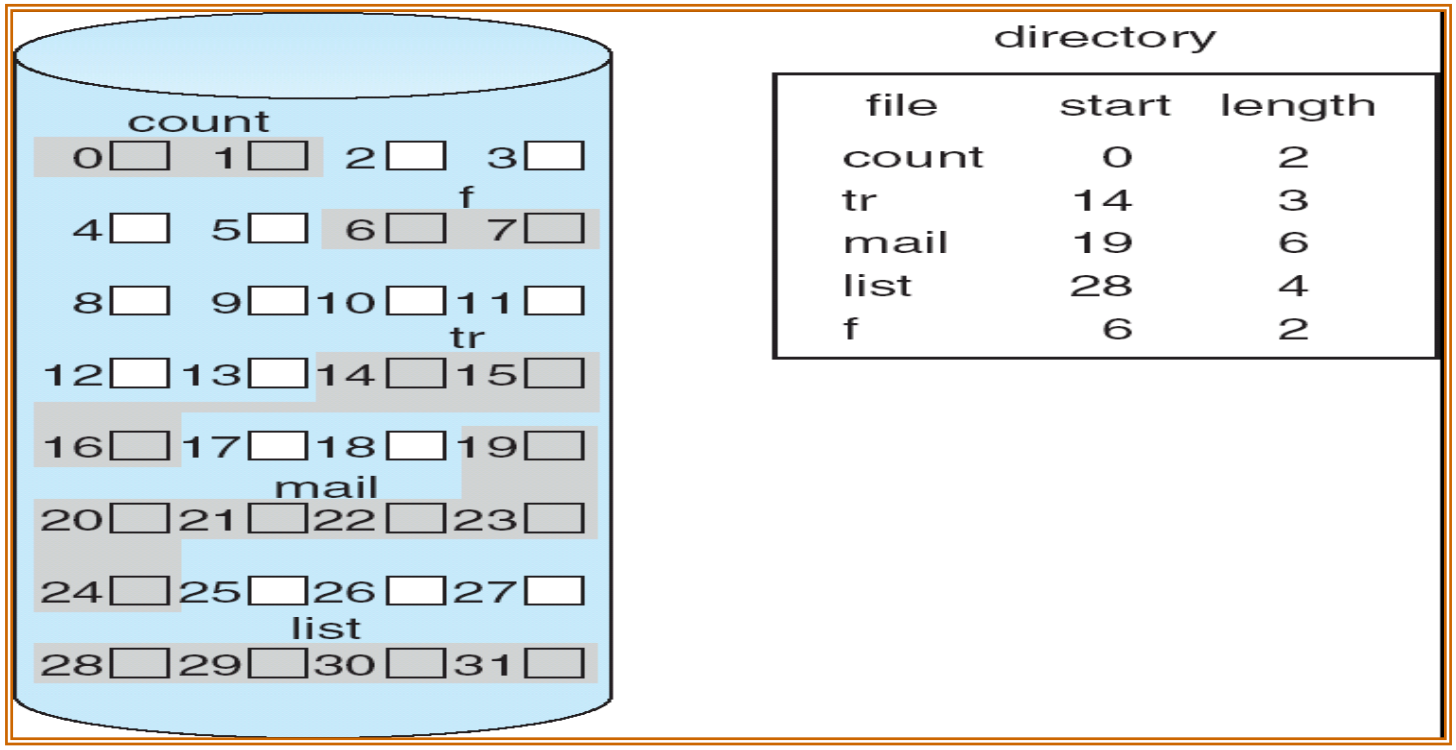
Allocation Methods: Contiguous allocation (2)

- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Random access
- Wasteful of space (dynamic storage-allocation problem)
- Files cannot grow
- Use for CD-ROM because the length of file are known in advance and no deletion

File System Implementation

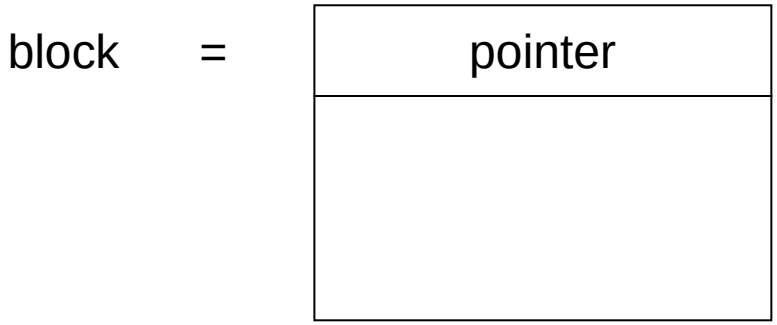
Allocation Methods: Contiguous allocation (3)

Contiguous Allocation of Disk Space



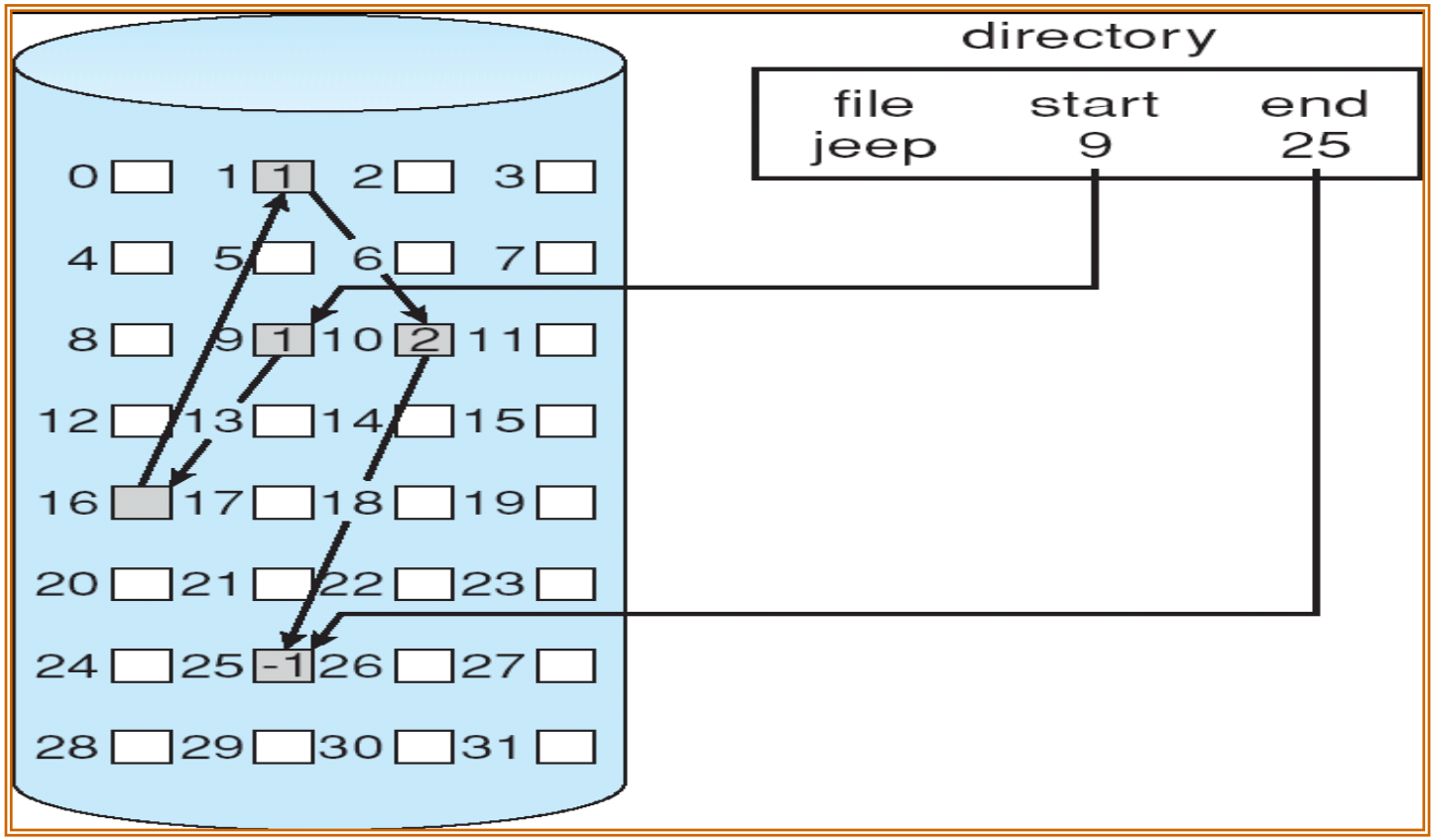
File System Implementation
Allocation Methods: Linked allocation (1)

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
- Simple – need only starting address
- Free-space management system – no waste of space
- No random access



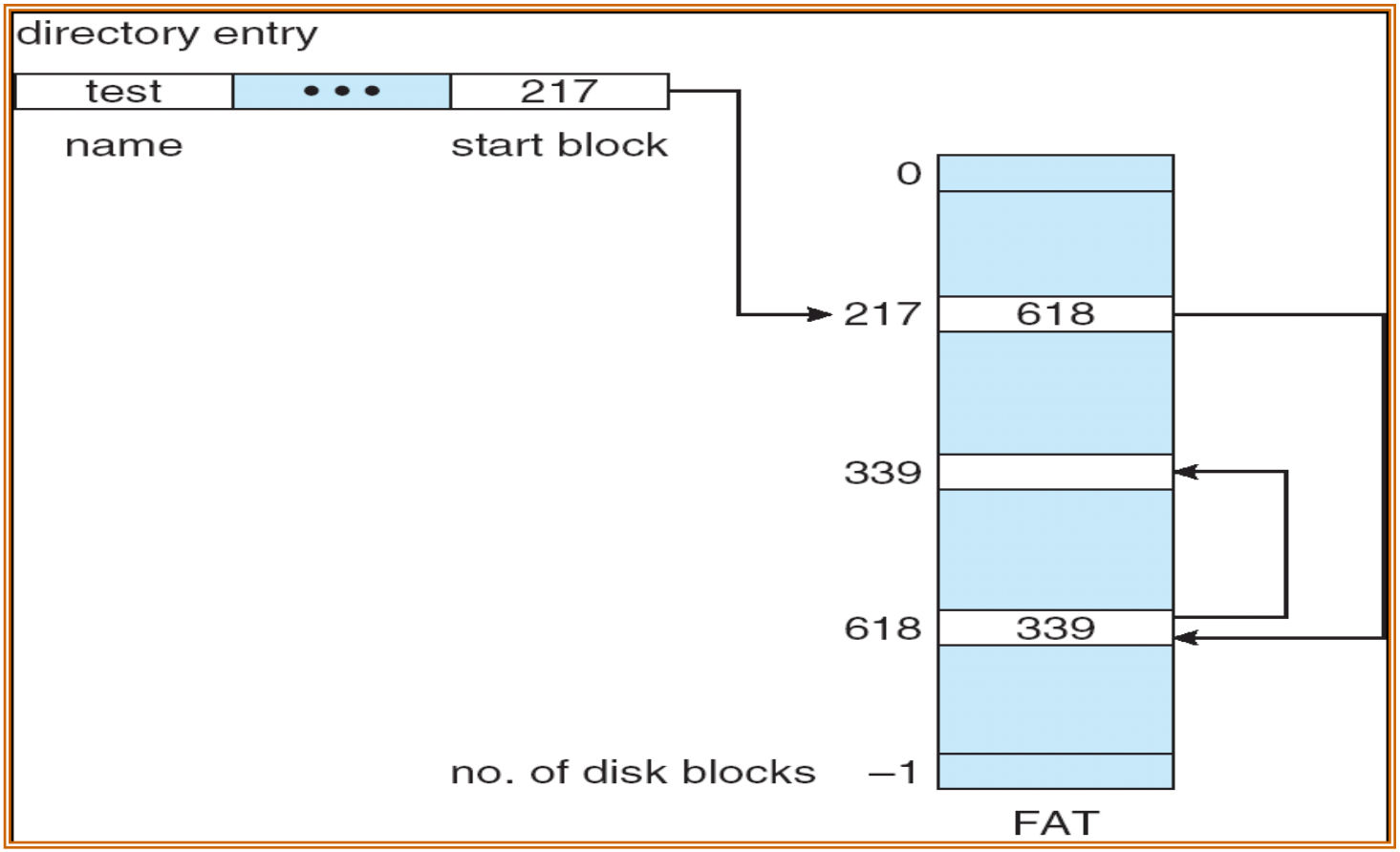
File System Implementation

Allocation Methods: Linked allocation (2)



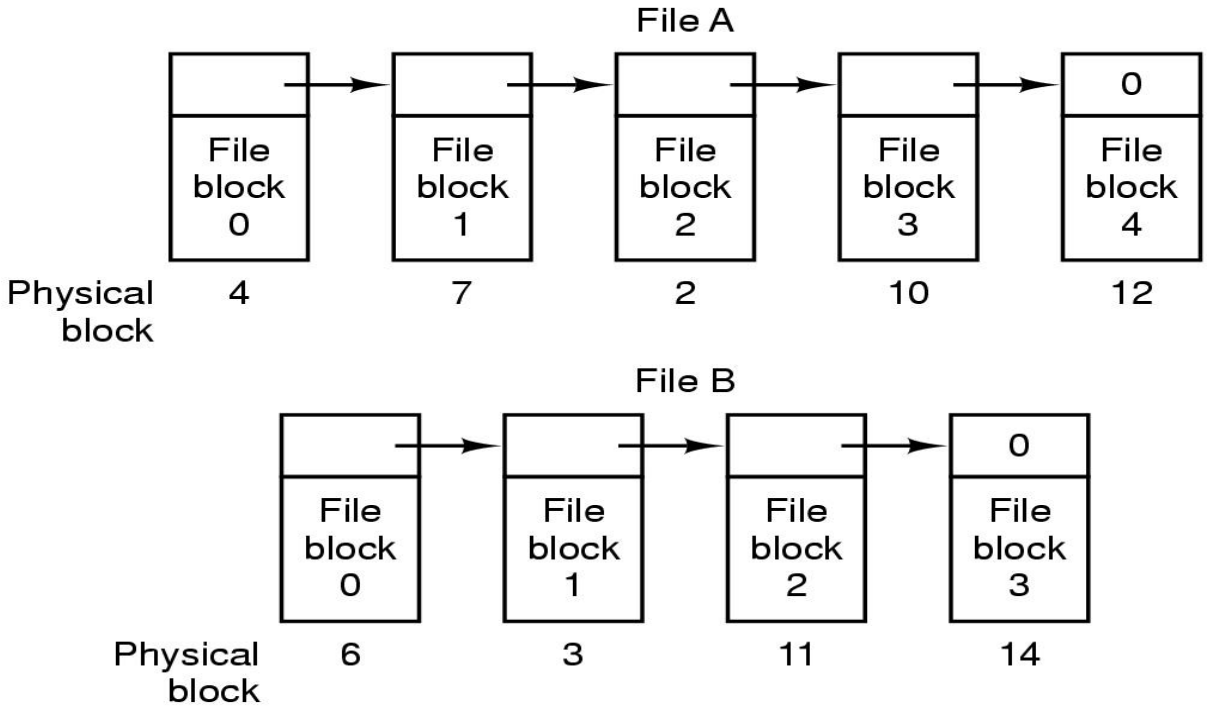
File System Implementation
Allocation Methods: Linked allocation (3)

- File-Allocation Table



File System Implementation

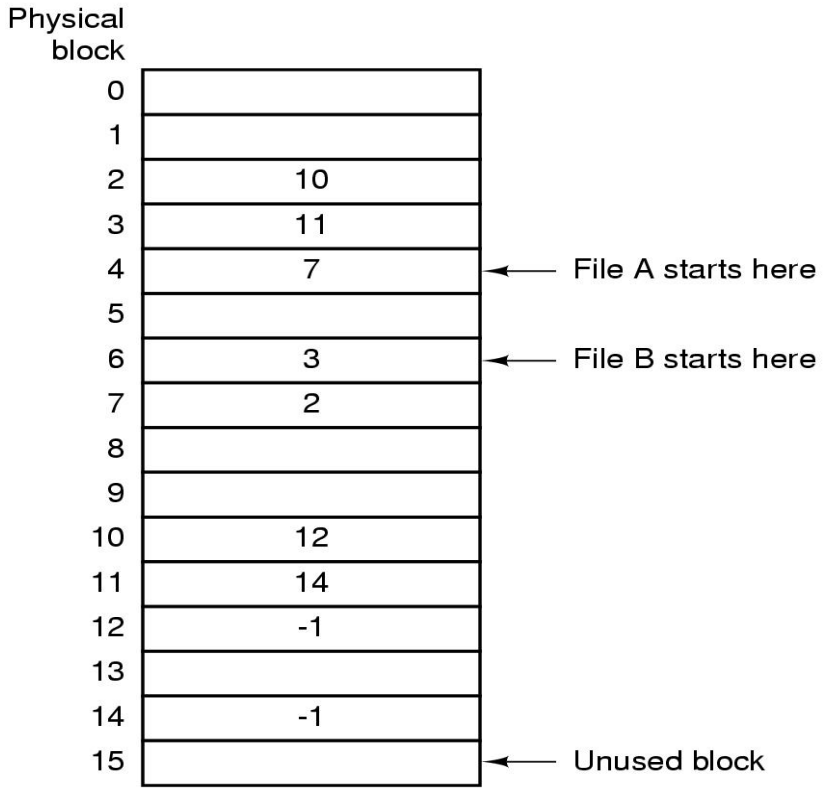
Allocation Methods: Linked allocation (4)



Storing a file as a linked list of disk blocks

File System Implementation

Allocation Methods: Linked allocation (5)

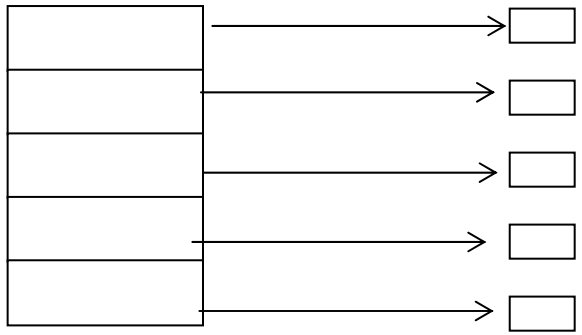


Linked list allocation using a file allocation table in RAM

FAT File Allocation Table

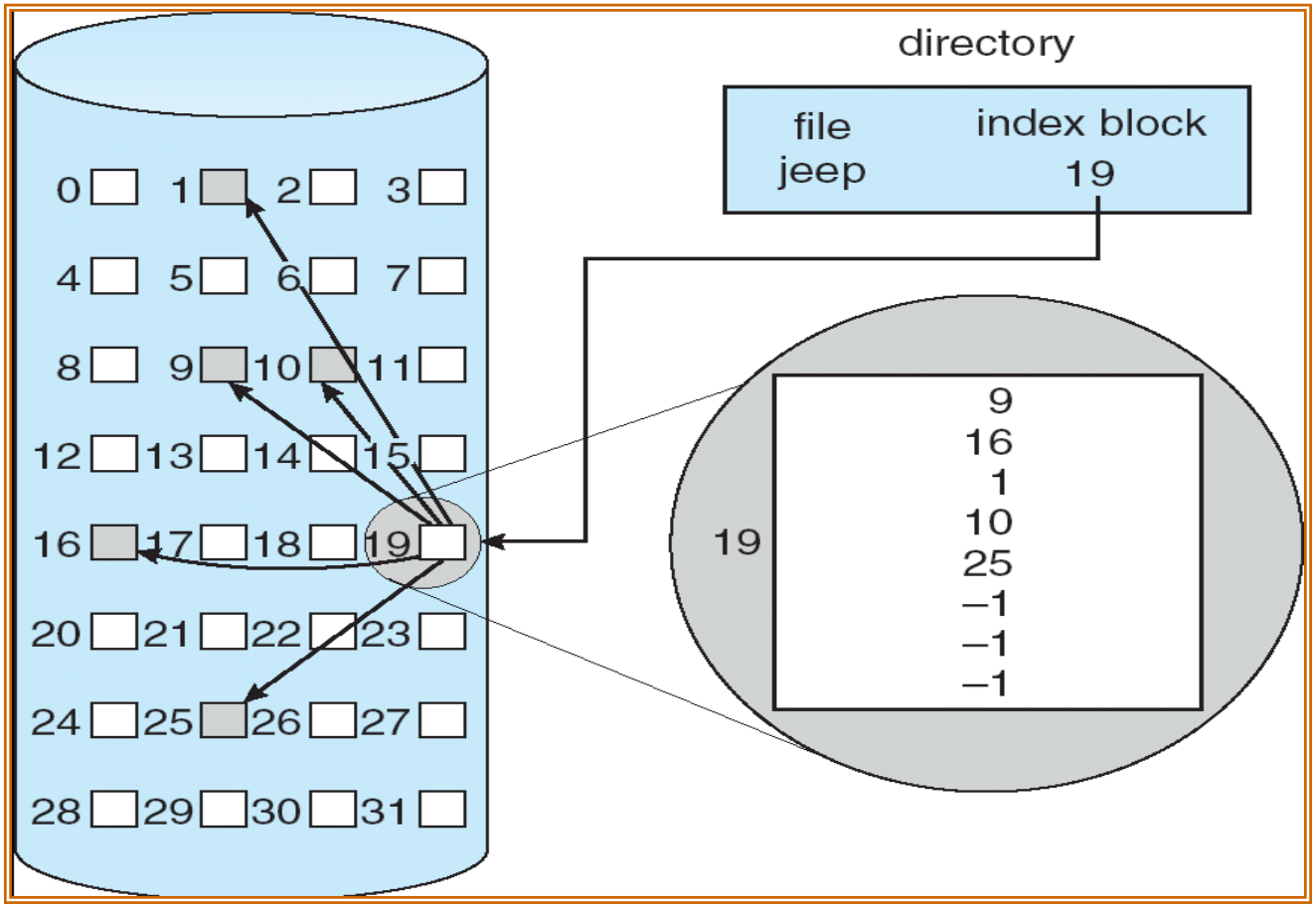
File System Implementation
Allocation Methods: Indexed allocation (1)

- Brings all pointers together into the *index block.*
- Logical view.
- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.

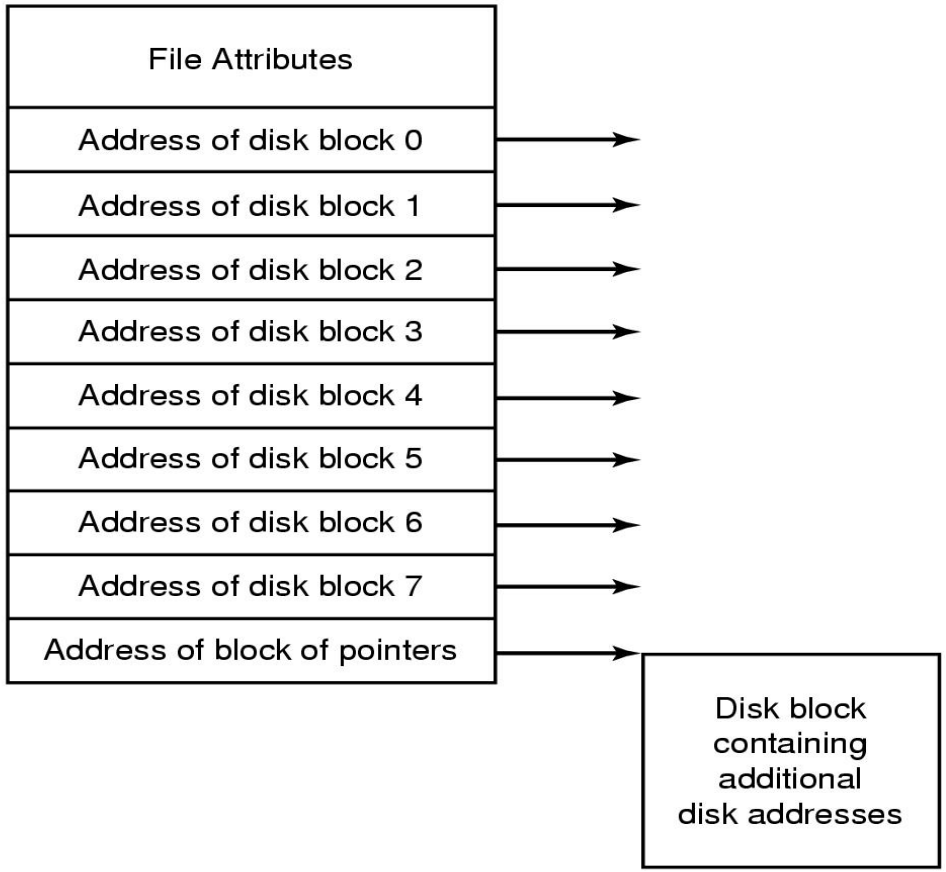


index table

File System Implementation
Allocation Methods: Indexed allocation (2)



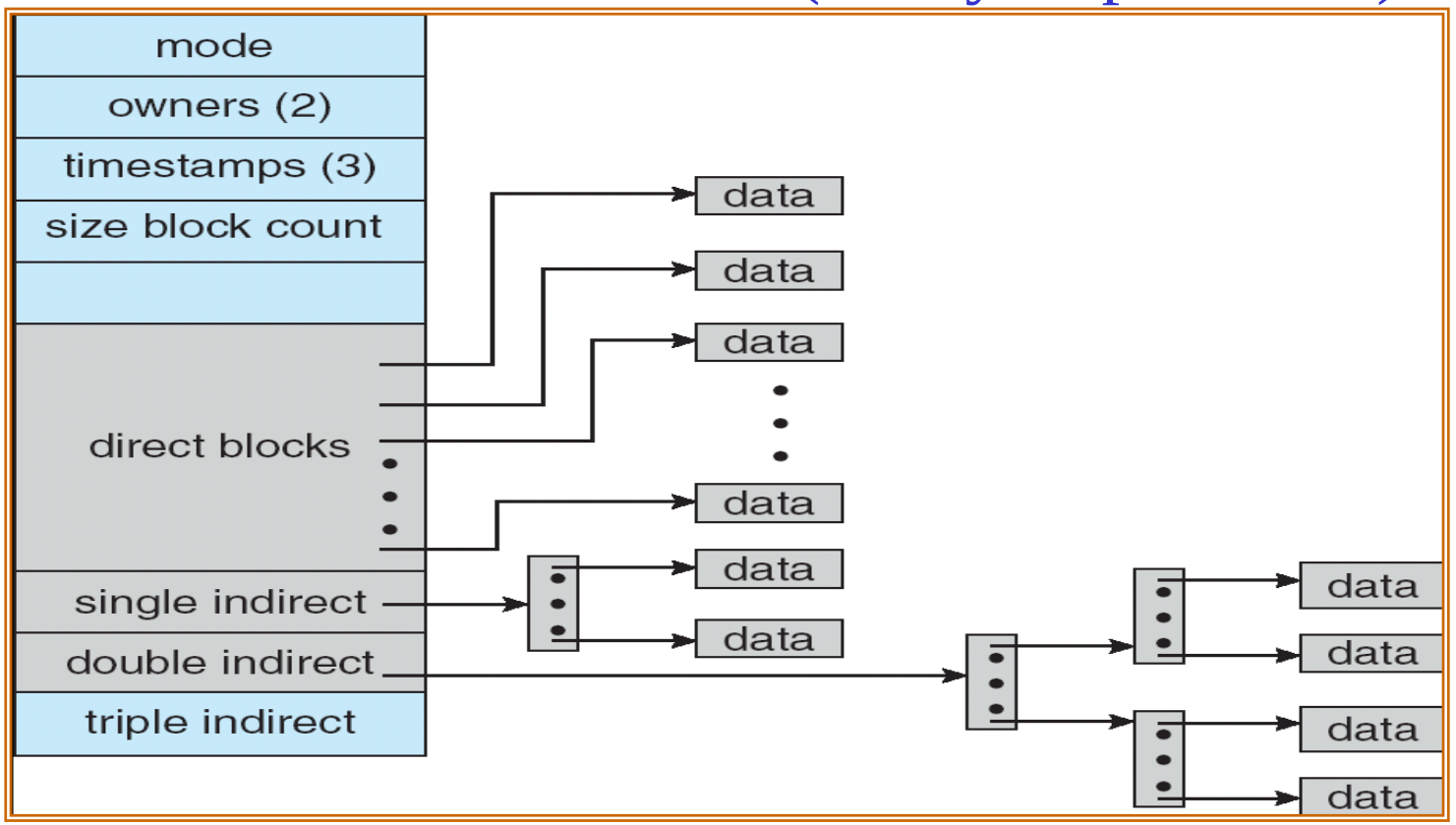
File System Implementation
Allocation Methods: Indexed allocation (3)



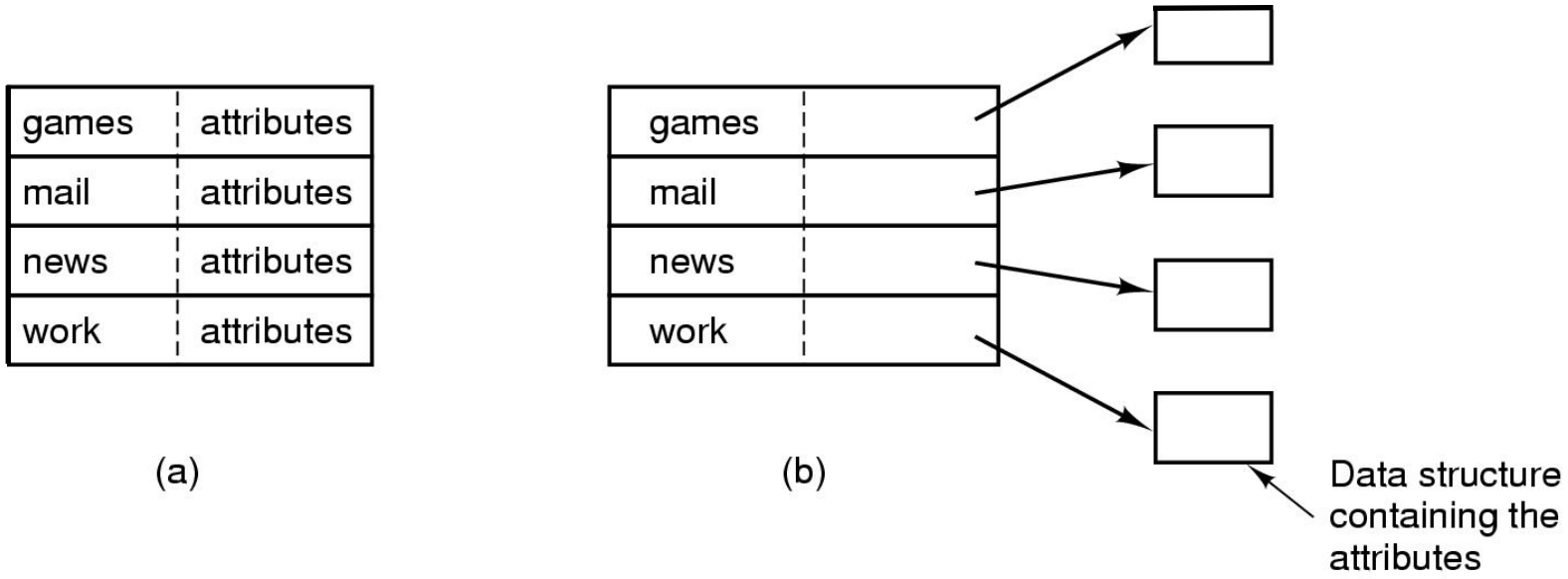
An example i-node

File System Implementation
Allocation Methods: Indexed allocation (4)

- Combined Scheme: UNIX (4K bytes per block)



File System Implementation
Implementing Directories (1)



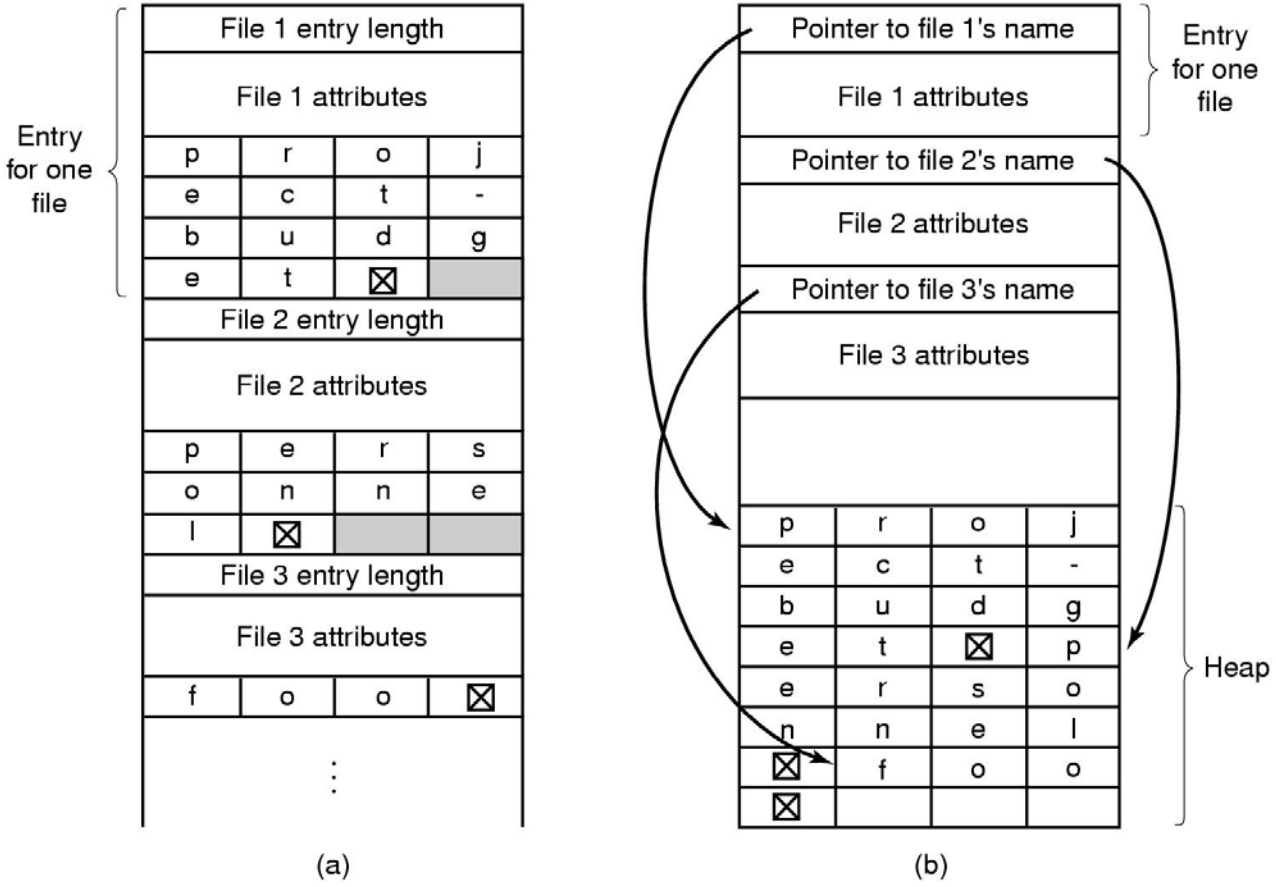
(a) A simple directory

fixed size entries

disk addresses and attributes in directory entry

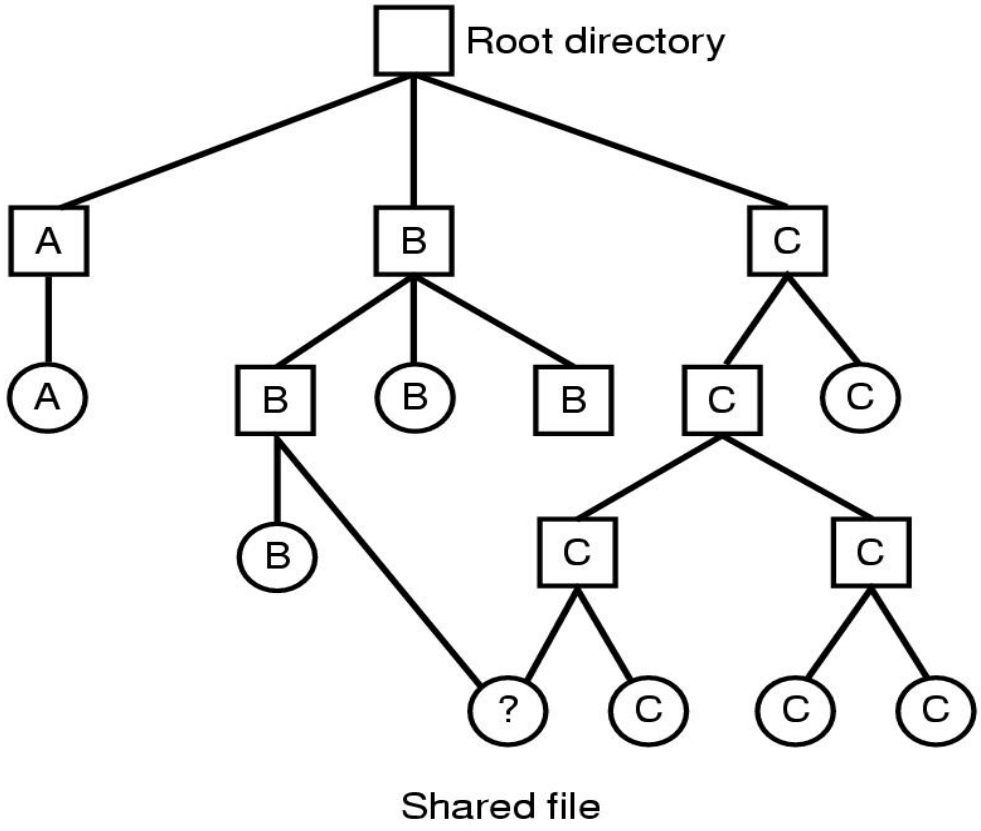
(b) Directory in which each entry just refers to an i-node

File System Implementation
Implementing Directories (2)



- Two ways of handling long file names in directory
  - (a) In-line
  - (b) In a heap

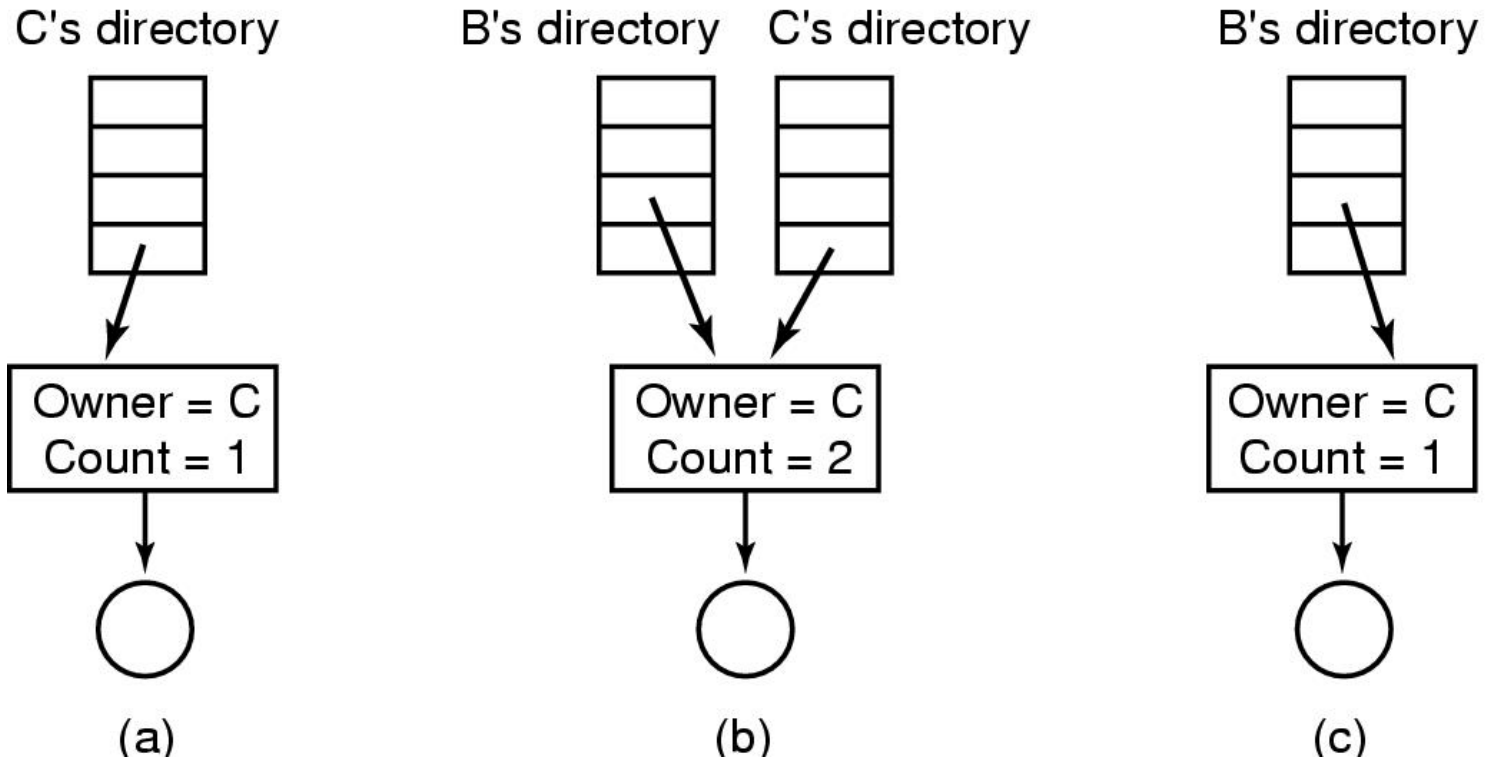
File System Implementation
Shared Files (1)



File system containing a shared file

File System Implementation

Shared Files (2)



(a) Situation prior to linking

(b) After the link is created

(c) After the original owner removes the file

File System Implementation
Log-Structured File Systems

- With CPUs faster, memory larger
  - disk caches can also be larger
  - increasing number of read requests can come from cache
  - thus, most disk accesses will be writes

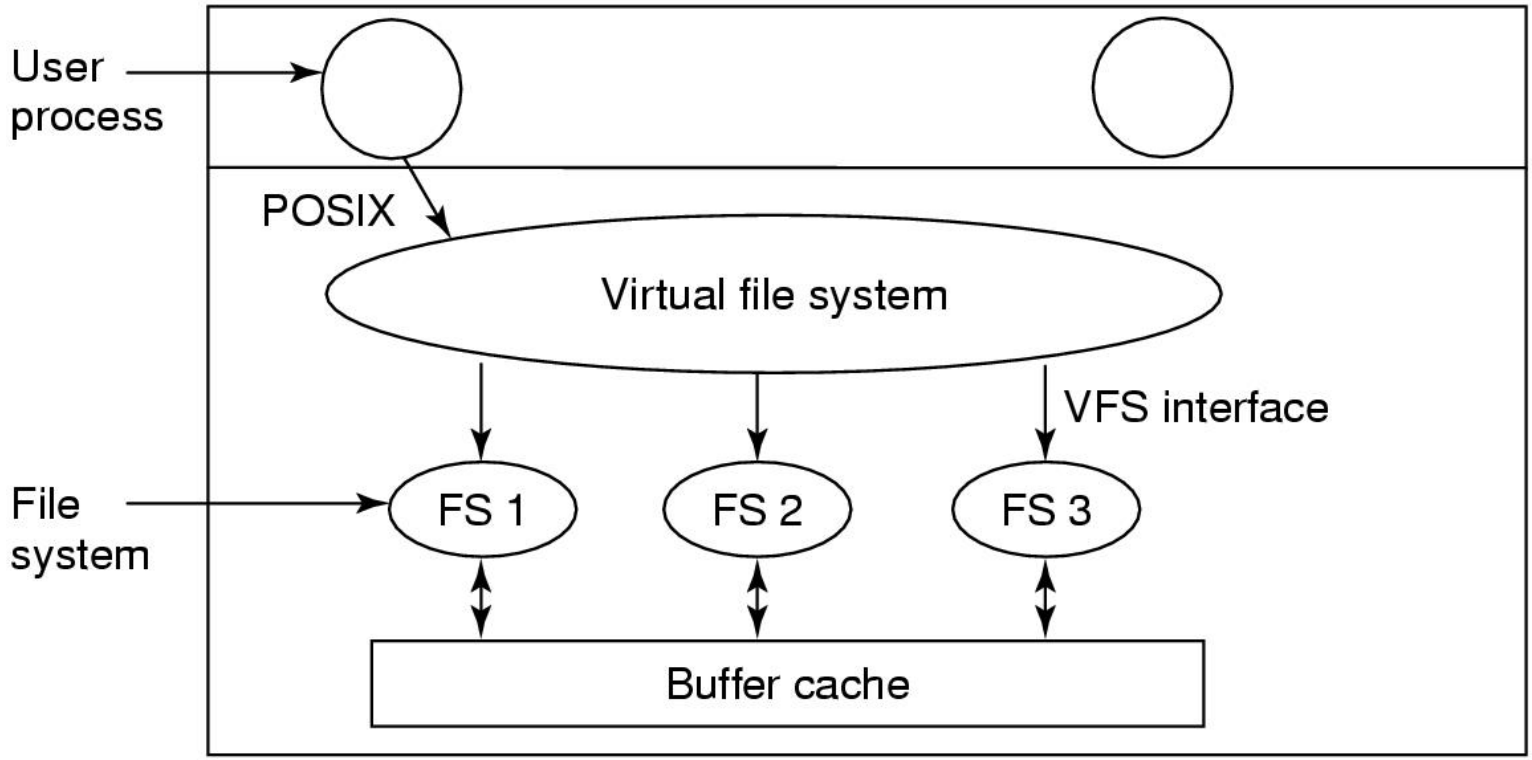
- LFS Strategy structures entire disk as a log
  - have all writes initially buffered in memory
  - periodically write these to the end of the disk log
  - when file opened, locate i-node, then find blocks

File System Implementation
Journaling File Systems

Operations required to remove a file in UNIX:

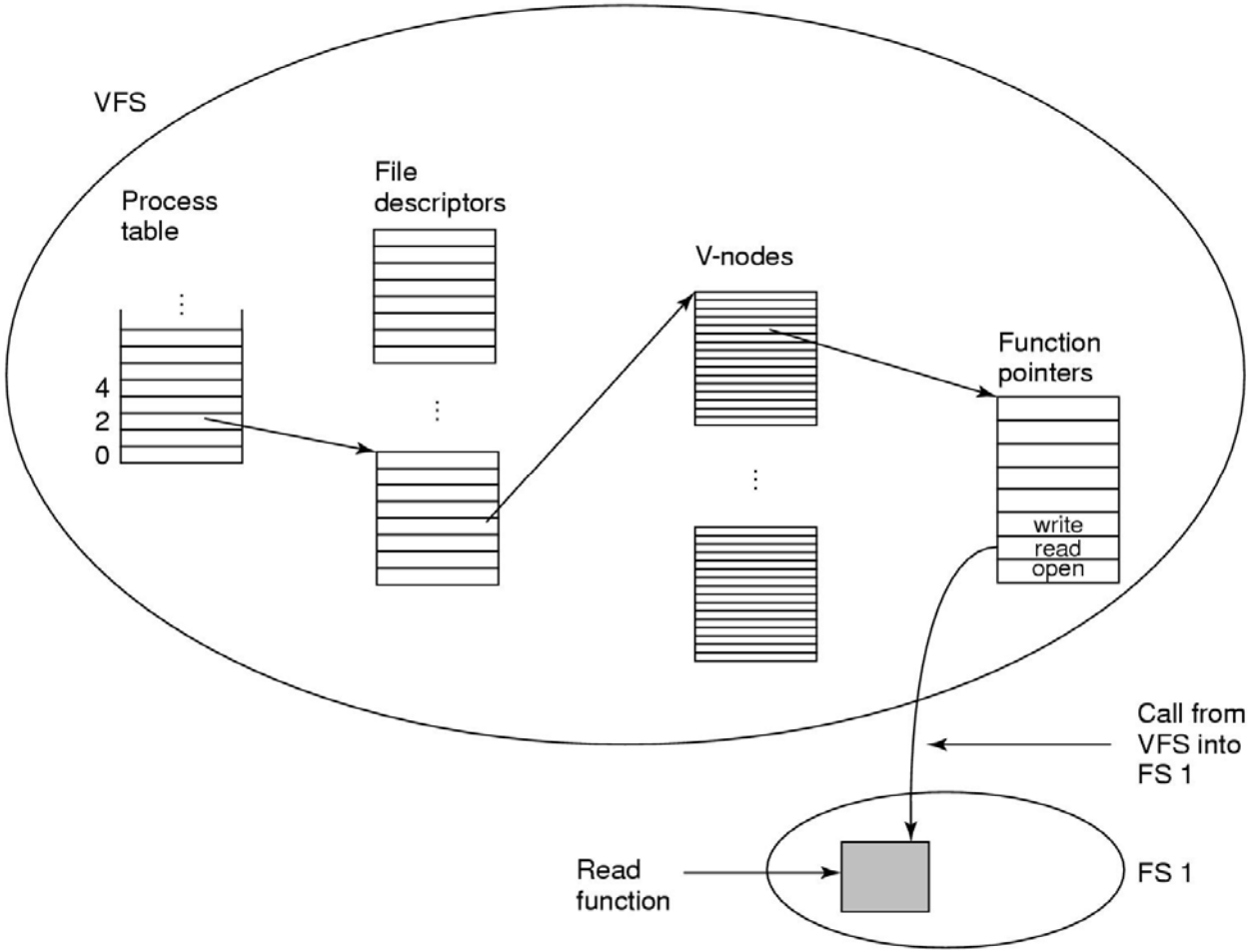
- Remove the file from its directory.
- Release the i-node to the pool of free i-nodes.
- Return all the disk blocks to the pool of free disk blocks.

File System Implementation
Virtual File Systems (1)



Position of the virtual file system.

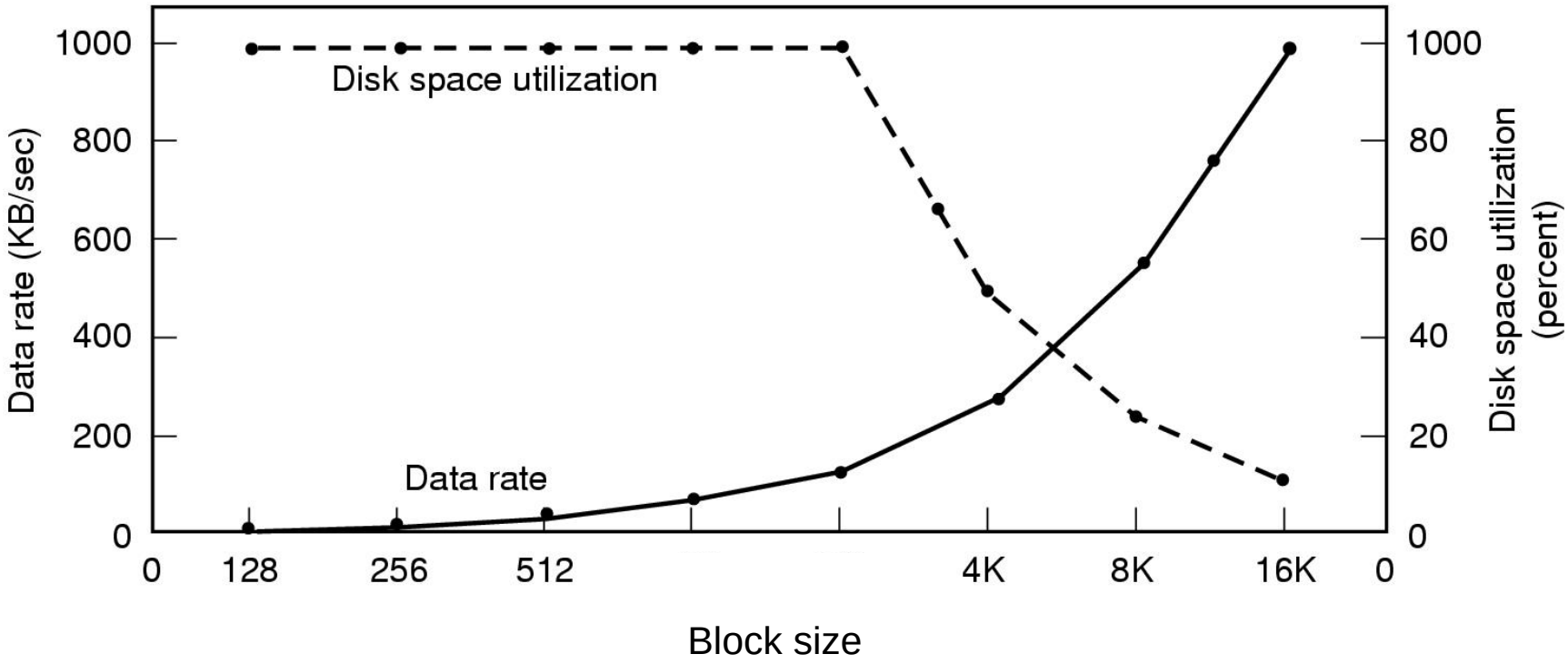
File System Implementation
Virtual File Systems (2)



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

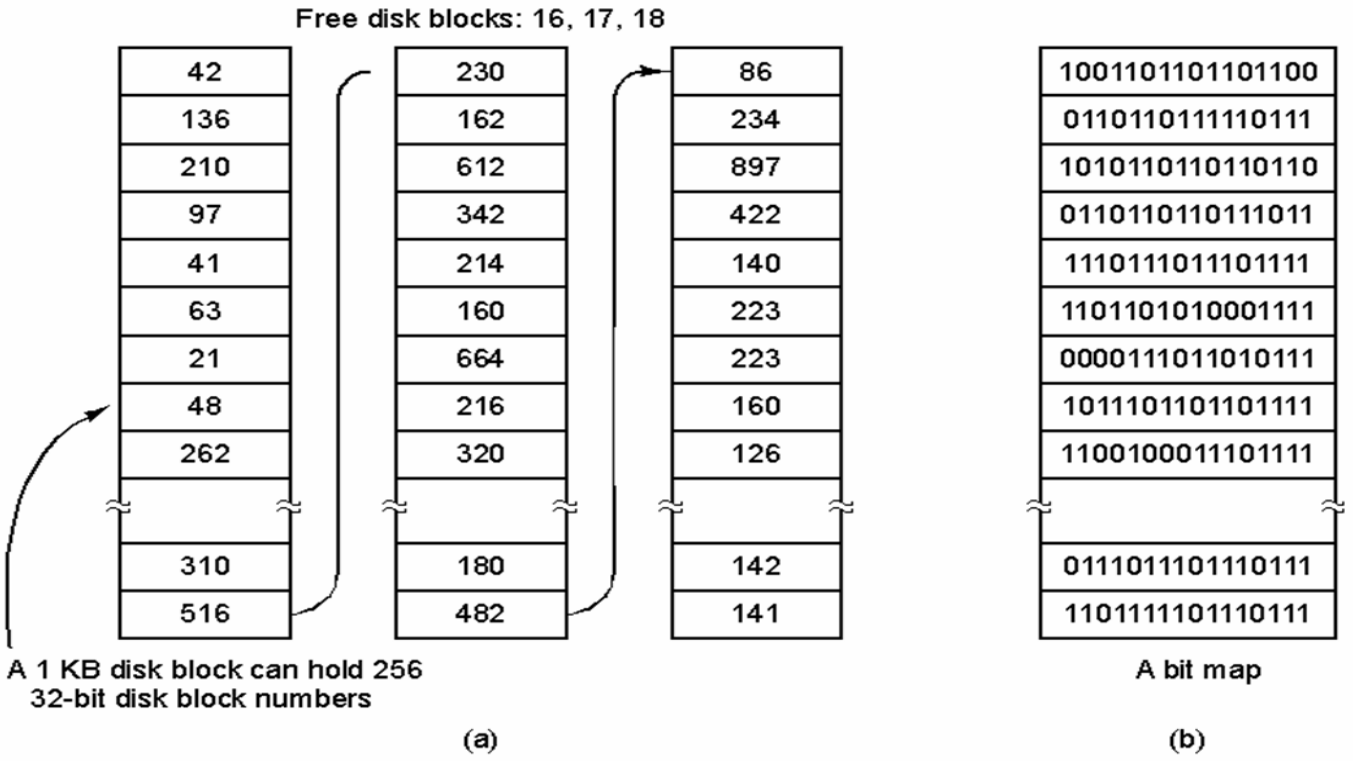
4.4 File system Management and Optimization

File system Management and Optimization
Disk Space Management (1)



- Dark line (left hand scale) gives data rate of a disk
- Dotted line (right hand scale) gives disk space efficiency
- All files 2KB

File system Management and Optimization
Disk Space Management (2)



(a) Storing the free list on a linked list

(b) A bit map

File system Management and Optimization

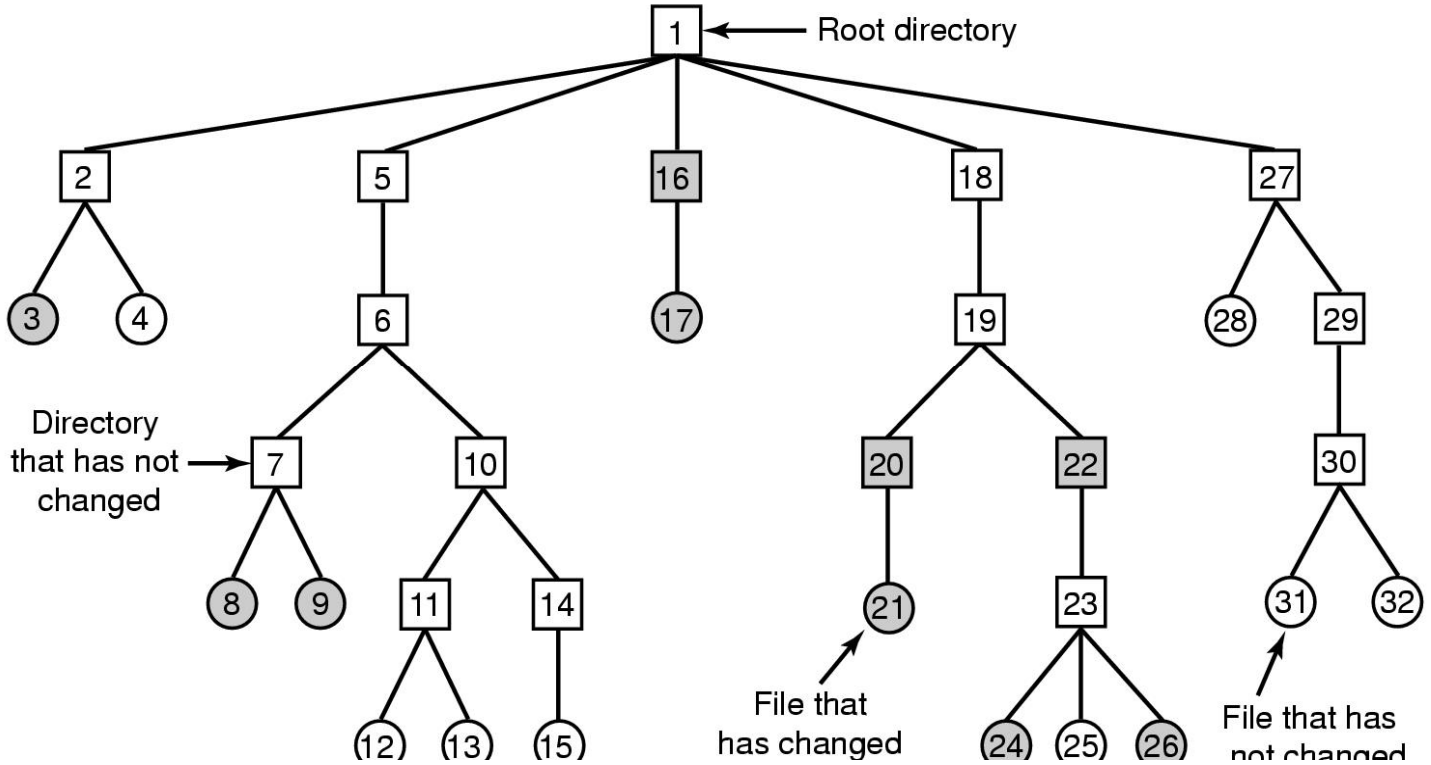
File System Backups (1)

Backups to tape are generally made to handle one of two potential problems:

- Recover from disaster.
- Recover from stupidity.

File system Management and Optimization

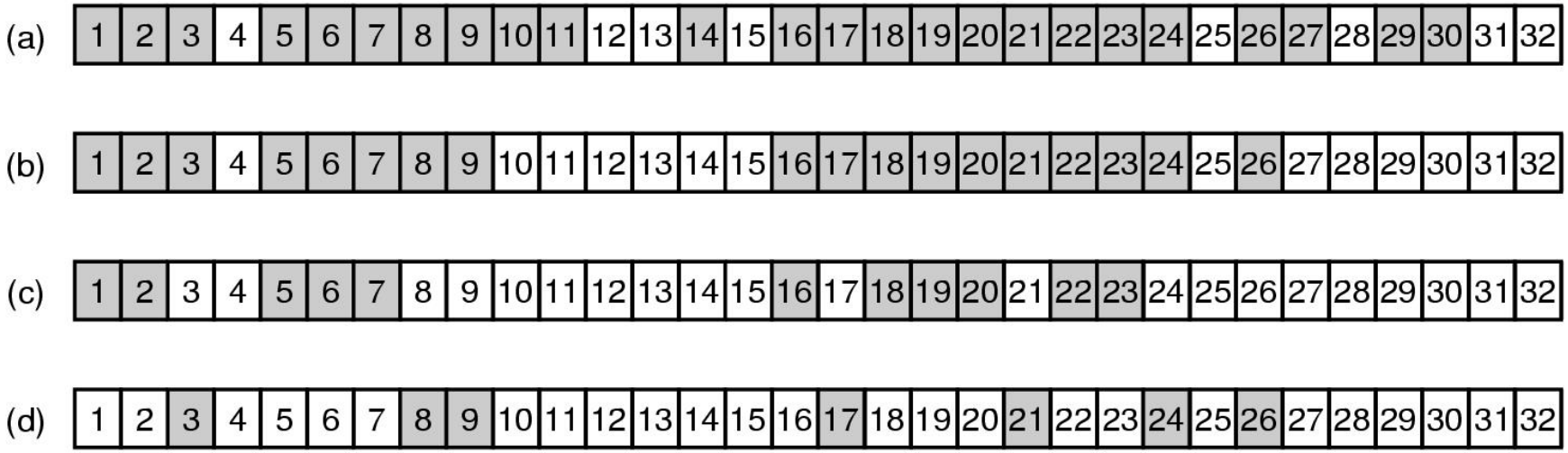
File System Backups (2)



A file system to be dumped. Squares are directories, circles are files. Shaded items have been modified since last dump. Each directory and file is labeled by its i-node number.

File system Management and Optimization

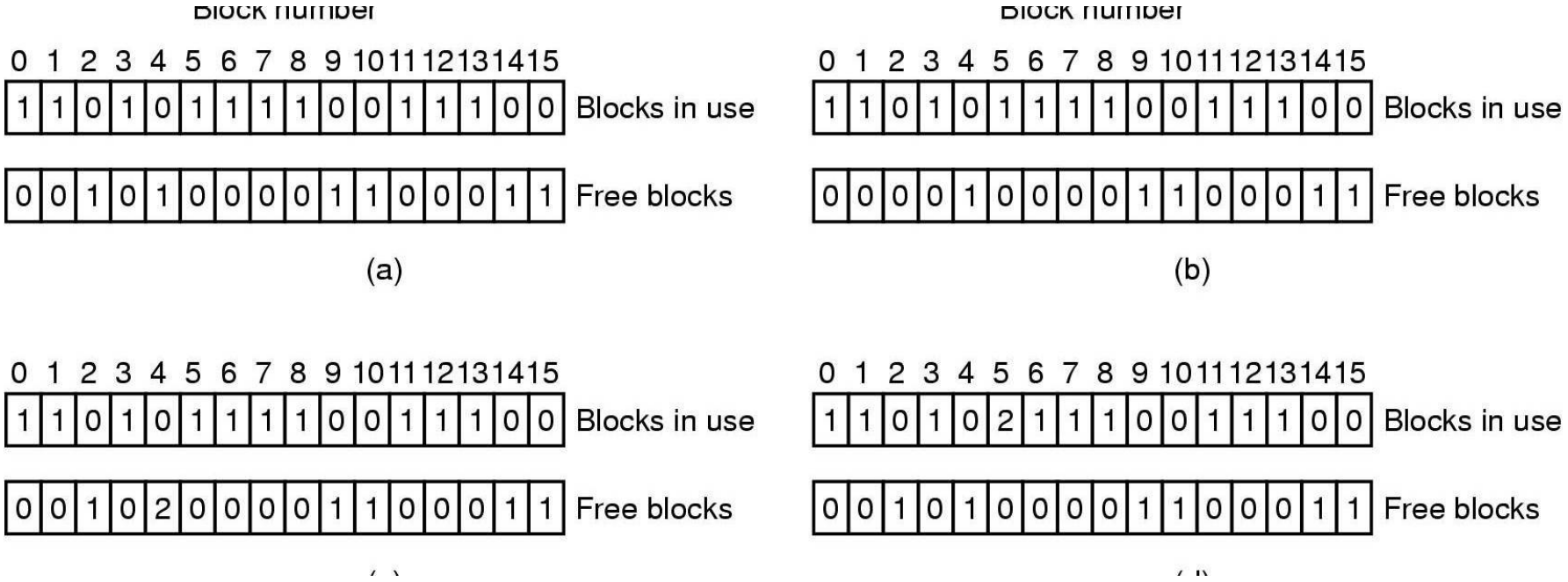
File System Backups (3)



Bitmaps used by the logical dumping algorithm.

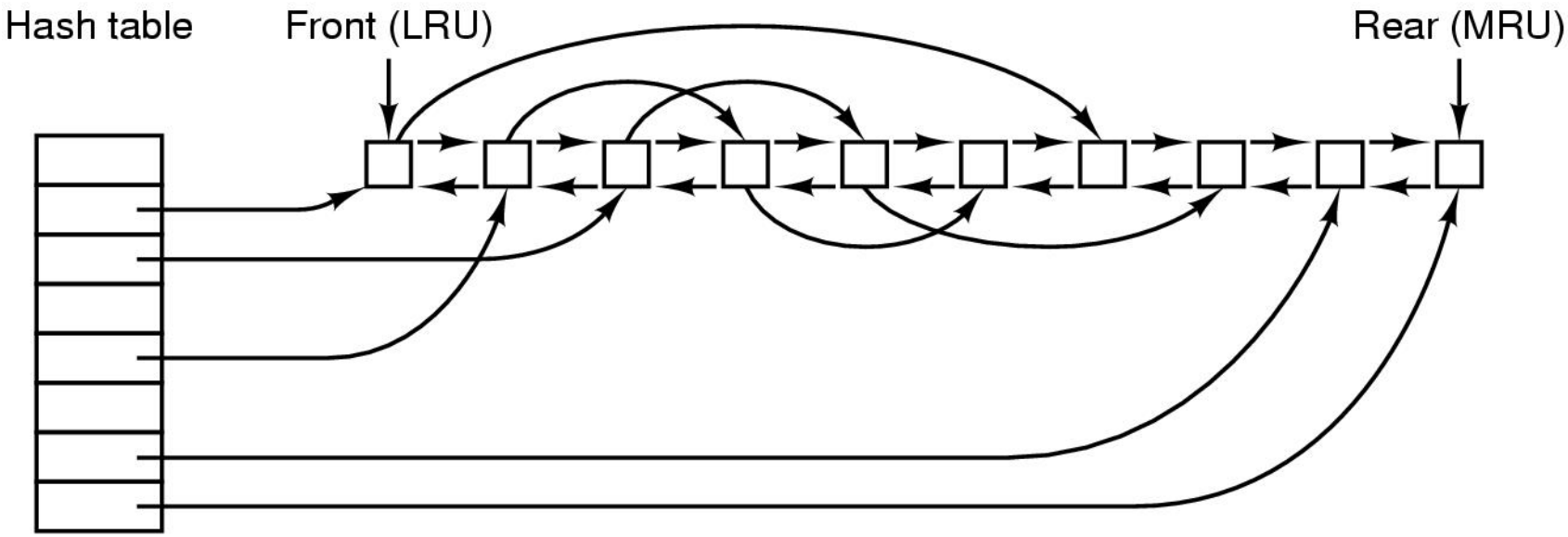
File system Management and Optimization

File System Consistency



File system states. (a) Consistent. (b) Missing block. (c) Duplicate block in free list. (d) Duplicate data block.

File system Management and Optimization
Caching (1)



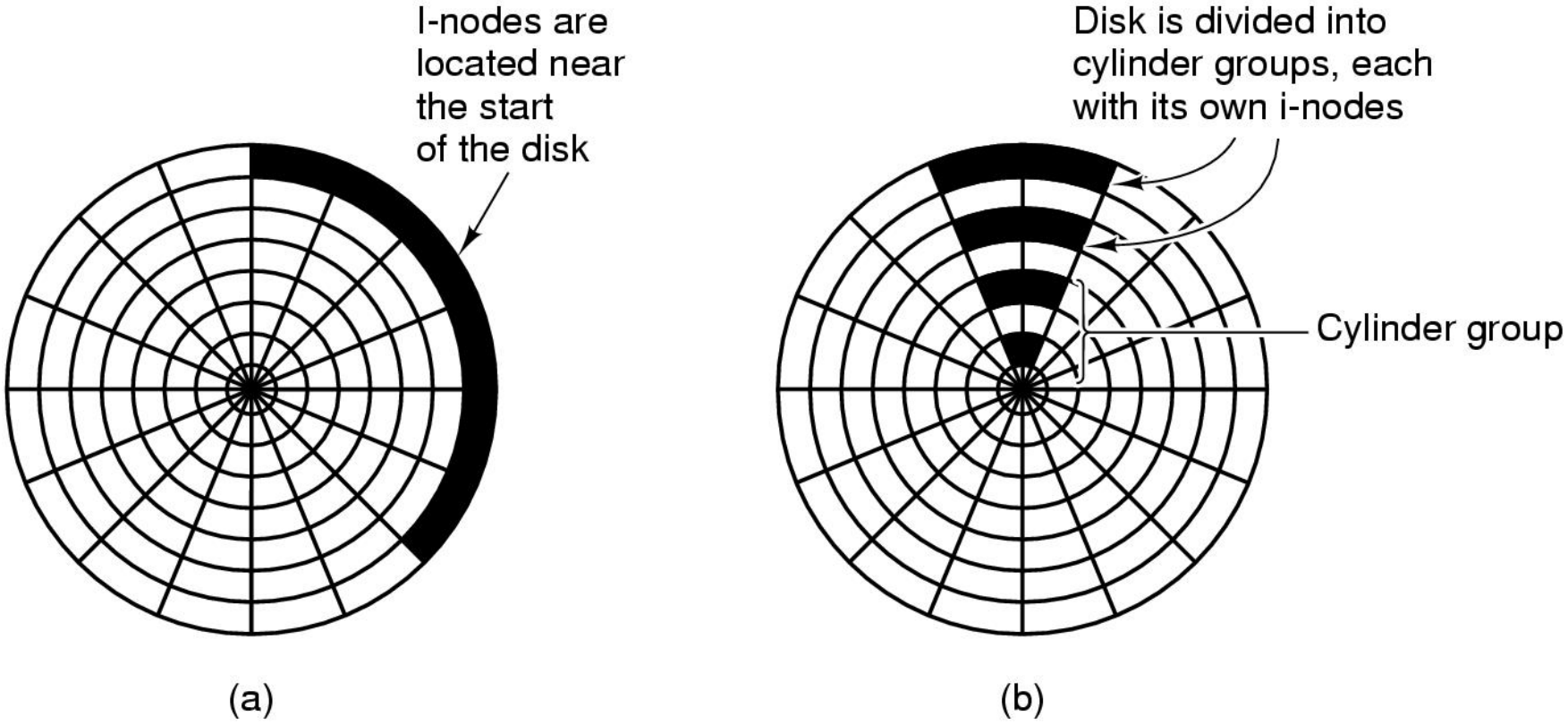
The buffer cache data structures.

File system Management and Optimization
Caching (2)

- Some blocks, such as i-node blocks, are rarely referenced two times within a short interval.

- Consider a modified LRU scheme, taking two factors into account:
  - Is the block likely to be needed again soon?
  - Is the block essential to the consistency of the file system?

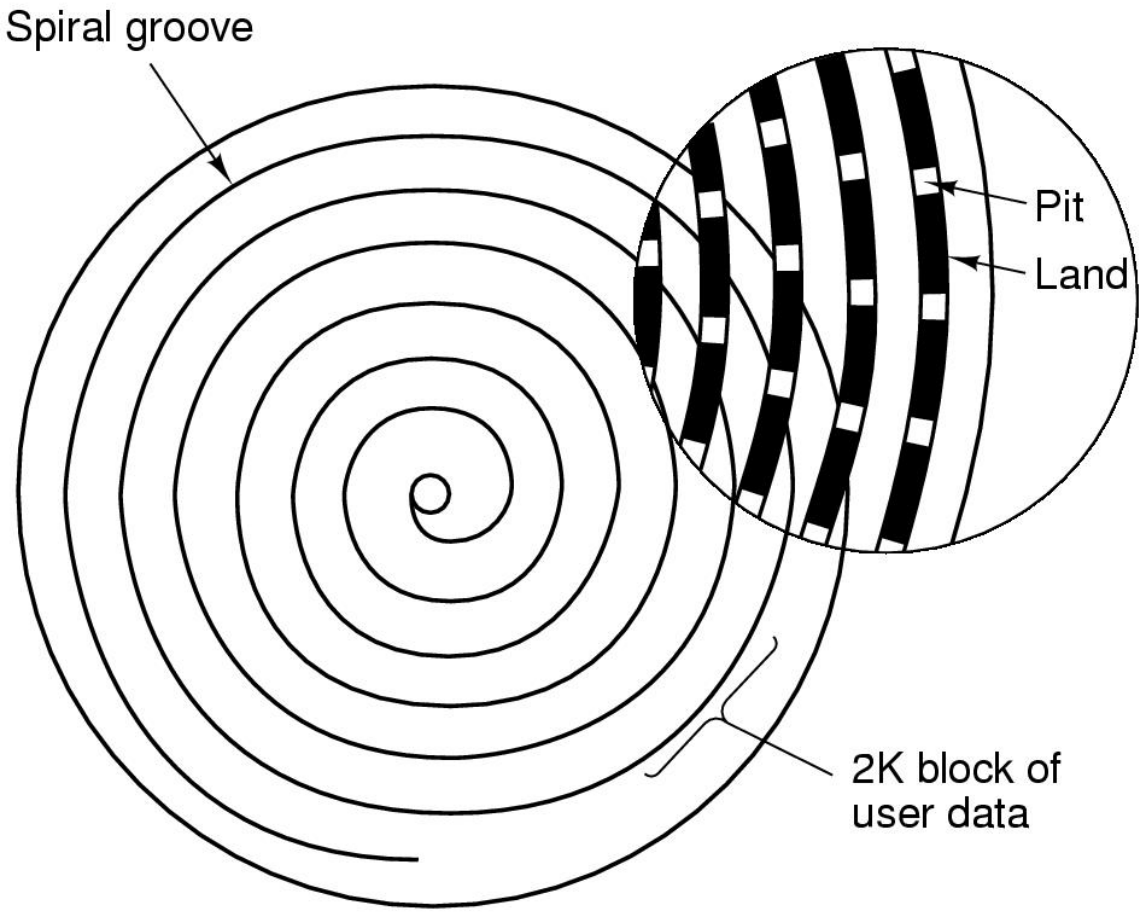
File system Management and Optimization
Reducing Disk Arm Motion



(a) I-nodes placed at the start of the disk.
(b) Disk divided into cylinder groups, each with its own blocks and i-nodes.

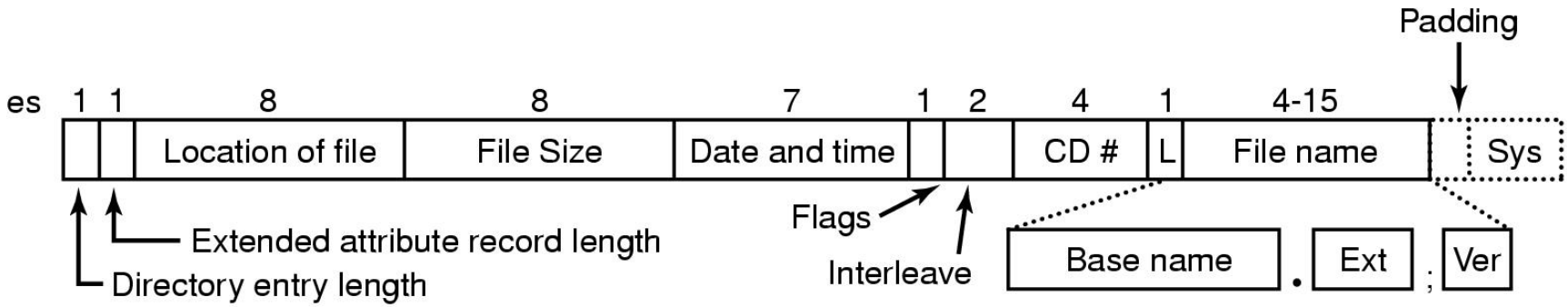
4.5 Example File Systems

Example File Systems
CD-ROM File Systems (1)



Recording structure of a CD or CD-ROM

Example File Systems
CD-ROM File Systems (2)



The ISO 9660 directory entry

Example File Systems
Rock Ridge Extensions (3)

Rock Ridge extension fields:

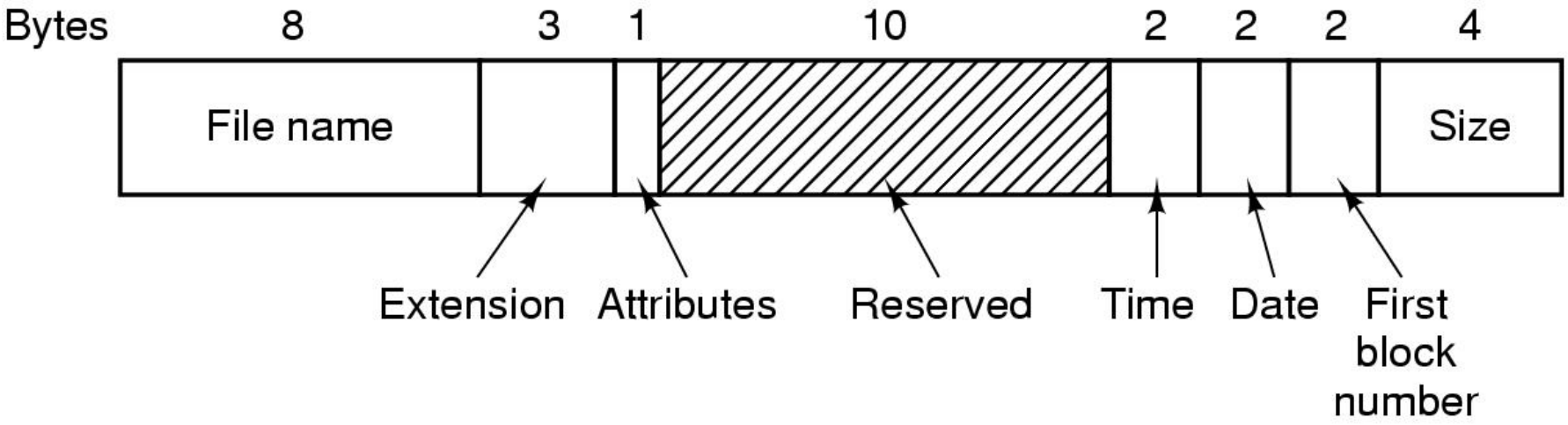
- PX - POSIX attributes.
- PN - Major and minor device numbers.
- SL - Symbolic link.
- NM - Alternative name.
- CL - Child location.
- PL - Parent location.
- RE - Relocation.
- TF - Time stamps.

Example File Systems
Joliet Extensions (4)

Joliet extension fields:

- Long file names.
- Unicode character set.
- Directory nesting deeper than eight levels.
- Directory names with extensions

Example File Systems
The MS-DOS File System (1)



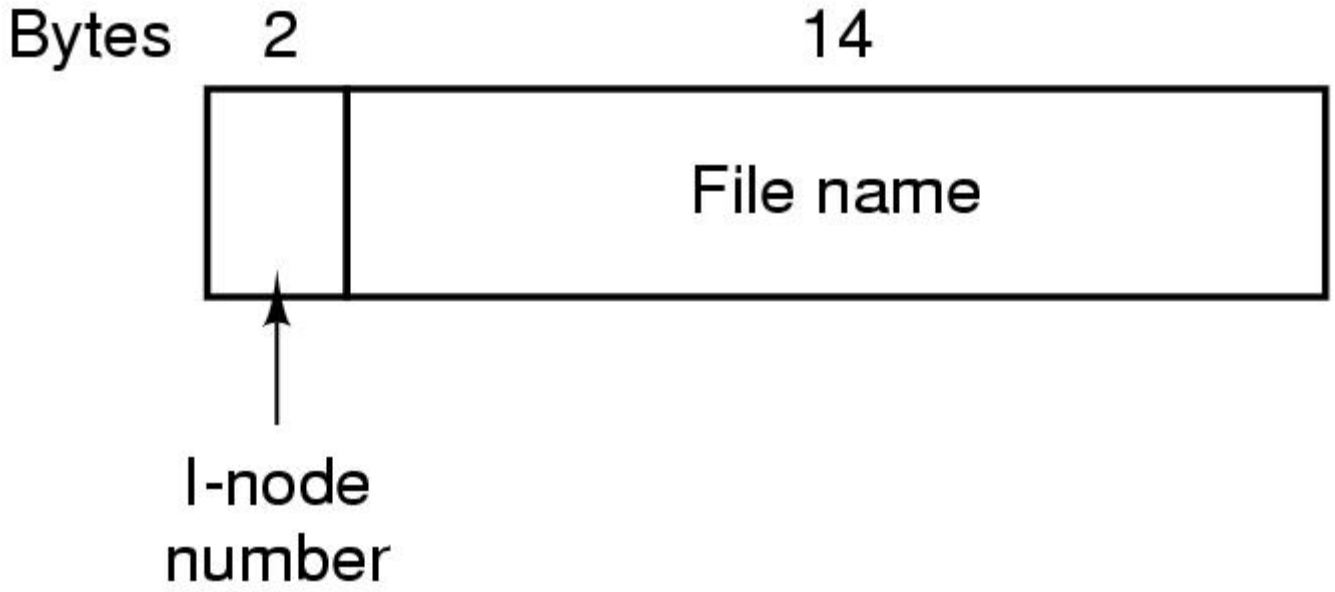
The MS-DOS directory entry

Example File Systems
The MS-DOS File System (2)

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

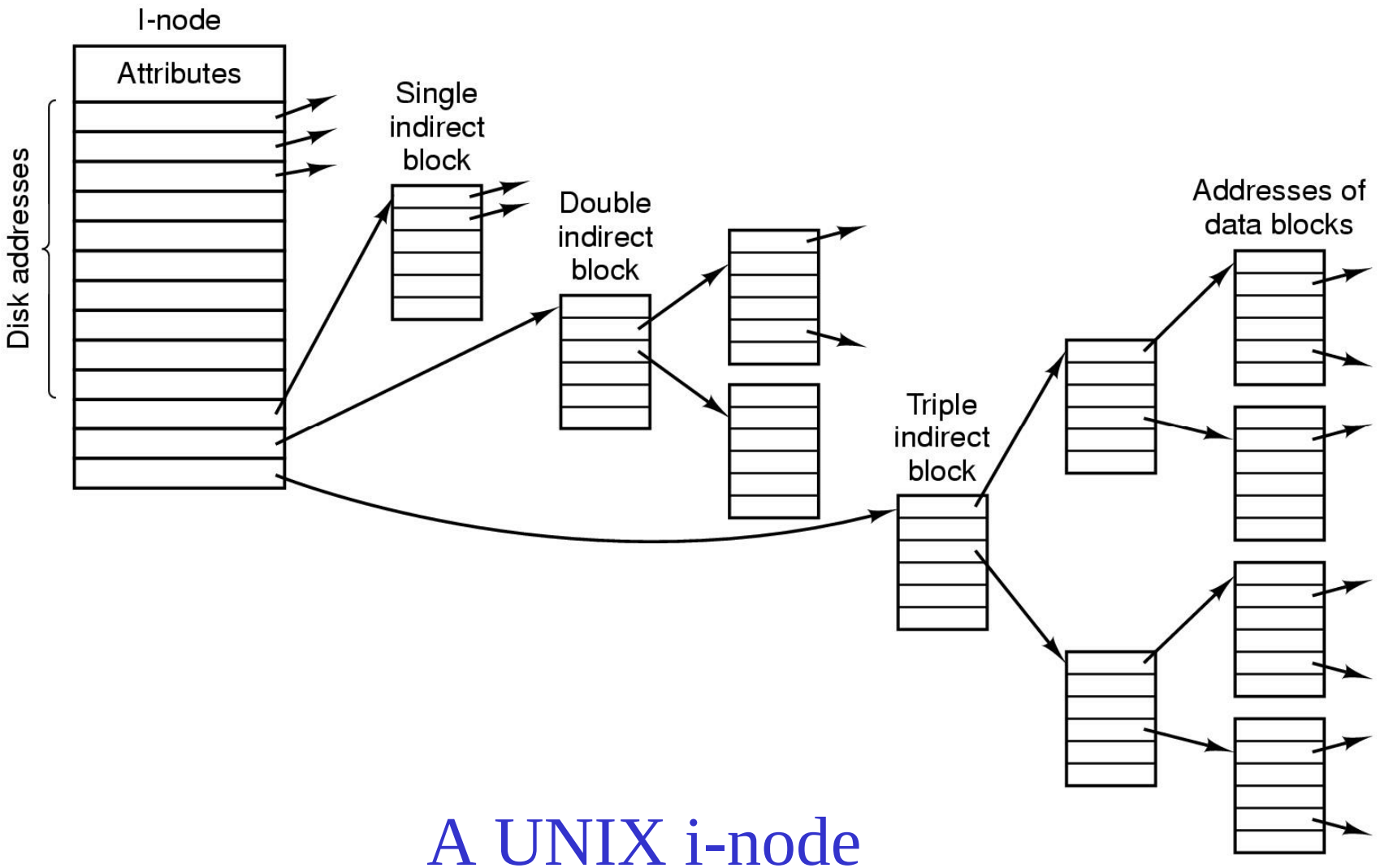
- Maximum partition for different block sizes
- The empty boxes represent forbidden combinations

Example File Systems
The UNIX V7 File System (1)



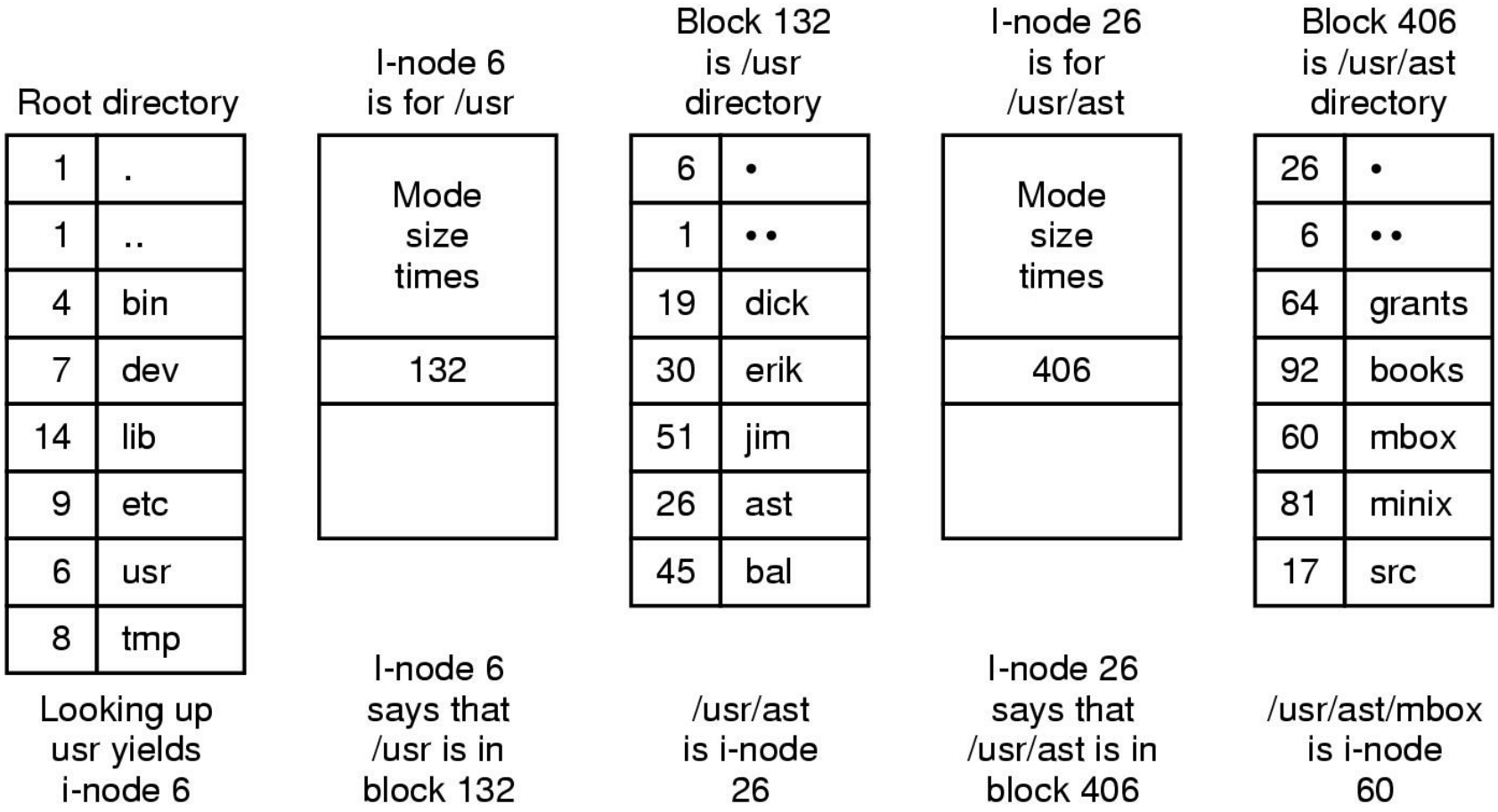
A UNIX V7 directory entry

Example File Systems
The UNIX V7 File System (2)



A UNIX i-node

Example File Systems
The UNIX V7 File System (3)



The steps in looking up */usr/ast/mbox*