File Systems

Tanenbaum Chapter 4

Silberschatz Chapters 10, 11, 12

File Systems

Essential requirements for long-term information storage:

- It must be possible to store a very large amount of information.
- The information must survive the termination of the process using it.
- Multiple processes must be able to access the information concurrently.

File Structure

- None:
  - File can be a sequence of words or bytes
- Simple record structure:
  - Lines
  - Fixed Length
  - Variable Length
- Complex Structure:
  - Formatted documents
  - Relocatable load files
- Who decides?

File Systems

Think of a disk as a linear sequence of fixed-size blocks and supporting reading and writing of blocks. Questions that quickly arise:

- How do you find information?
- How do you keep one user from reading another’s data?
- How do you know which blocks are free?

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

Figure 4-1. Some typical file extensions.

File Access Methods

- Sequential Access
  - Based on a magnetic tape model
  - read next, write next
  - reset
- Direct Access
  - Based on fixed length logical records
  - read n, write n
  - position to n
  - relative or absolute block numbers

File Structure

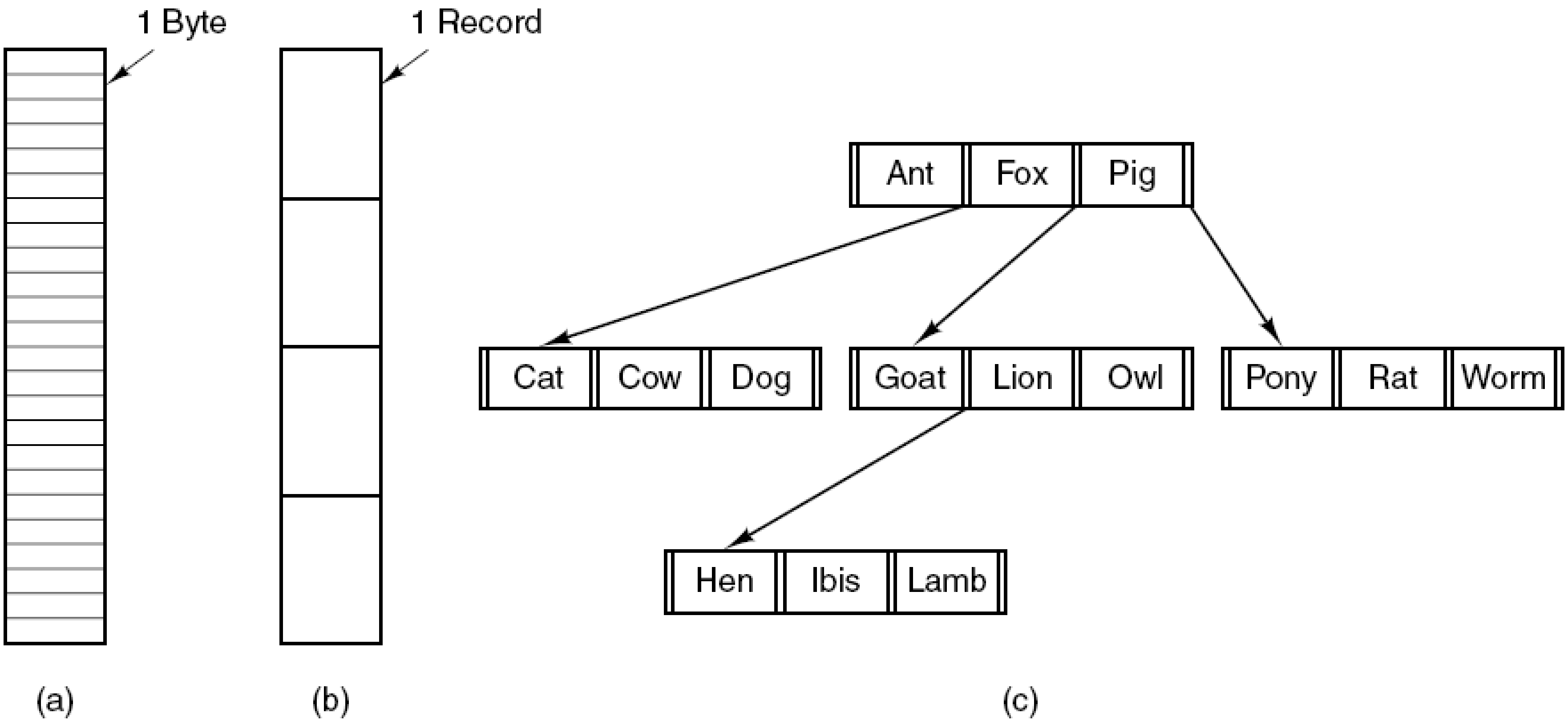


Figure 4-2. Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

File Types

- Regular Files:
  - ASCII files or binary files
  - ASCII consists of lines of text; can be displayed and printed
  - Binary, have some internal structure known to programs that use them
- Directory
  - Files to keep track of files
- Character special files (a character device file)
  - Related to I/O and model serial I/O devices
- Block special files (a block device file)
  - Mainly to model disks

File Types

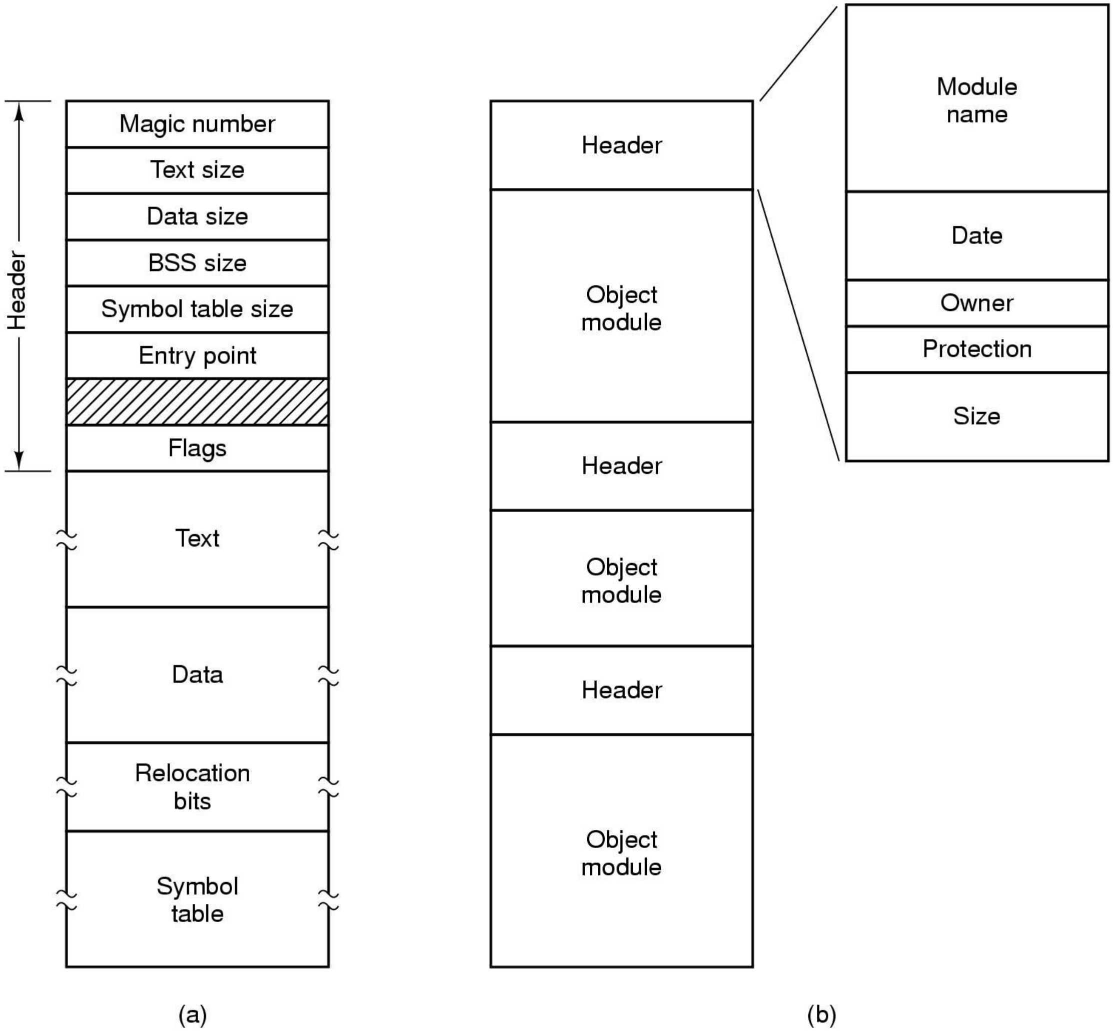
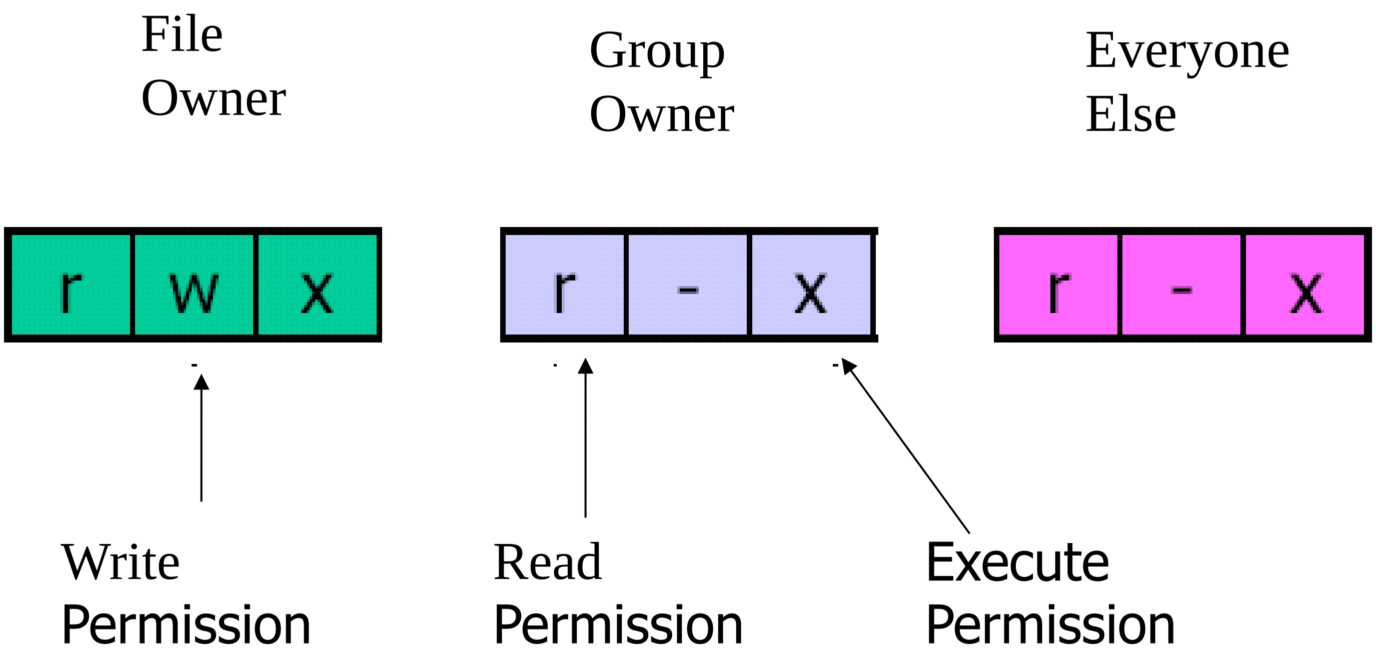


Figure 4-3. (a) An executable file. (b) An archive.

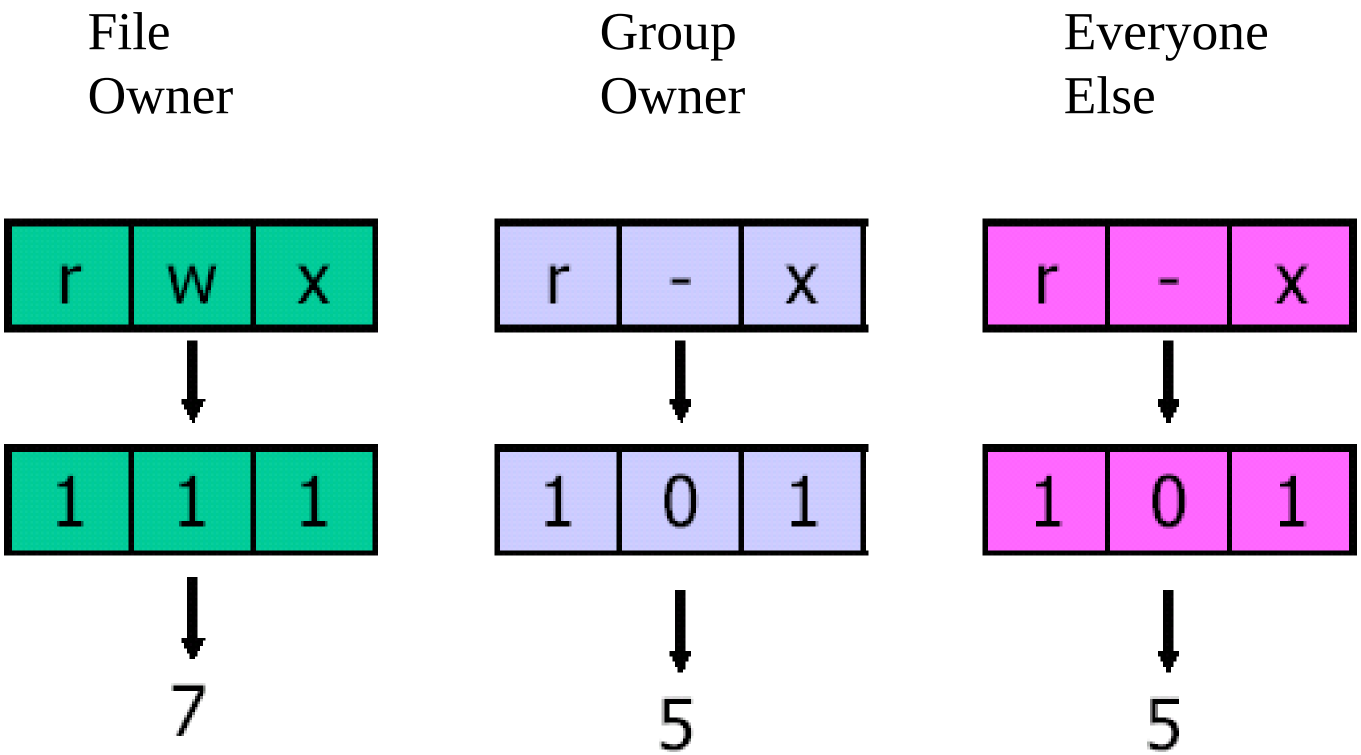
File Attributes

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 was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Figure 4-4a. Some possible file attributes.


File Owner
Group Owner
Everyone Else
r w x
r - x
r - x
Write Permission
Read Permission
Execute Permission

UNIX


File
Owner
Group
Owner
Everyone
Else
r
w
x
r
-
x
r
-
x
1
1
1
1
0
1
1
0
1
7
5
5

File Operations

The most common system calls relating to files:

- Create
- Delete
- Open
- Close
- Read
- Write
- Append
- Seek
- Get Attributes
- Set Attributes
- Rename

Information in a Device Directory

- File name:
- File Type:
- Address:
- Current Length
- Maximum Length
- Date Last accessed (for archiving)
- Date Last updated (for dumping)
- Owner ID
- Protection information

Hierarchical Directory Systems (1)

Single-level directory system:
The simpliest

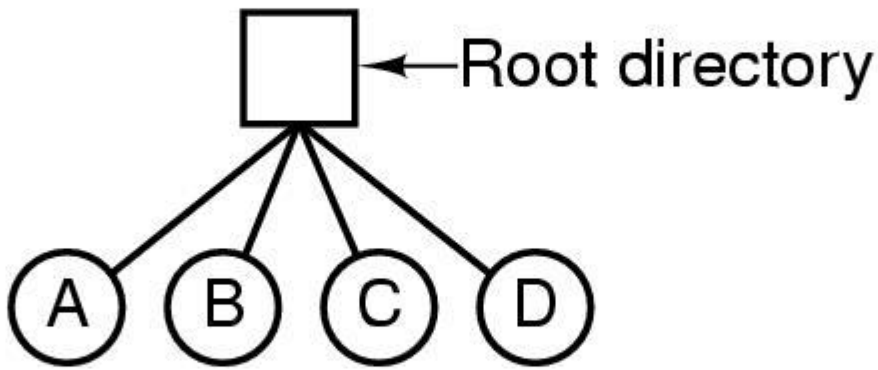


Figure 4-6. A single-level directory system containing four files.

Hierarchical Directory Systems (2)

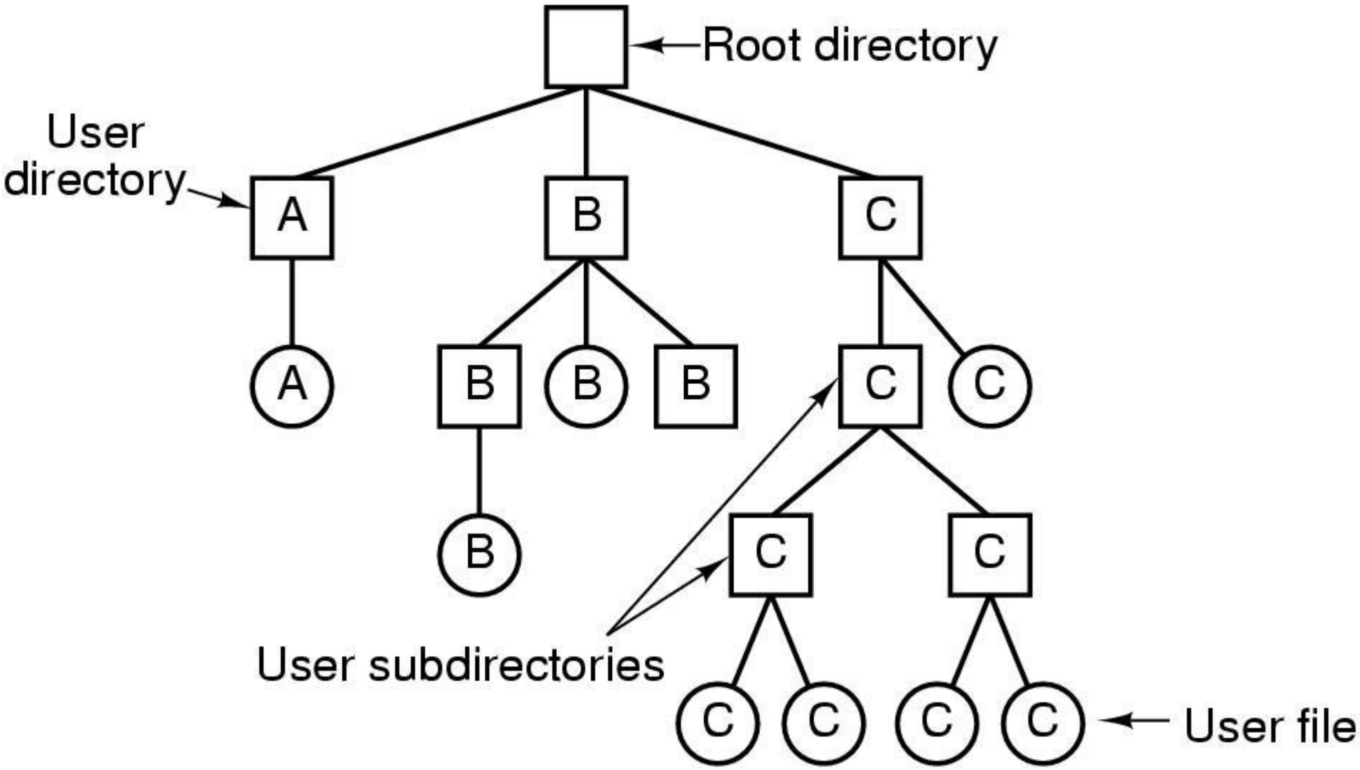


Figure 4-7. A hierarchical directory system.

Directory Operations

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Objectives for a Directory System

- Make it efficient
  - It should be easy to locate a file quickly
- Make file (and directory) naming convenient
  - Allow 2 users to have the same name for different files
  - Allow the same file to have more than 1 name
- Allow logical grouping of files
  - All word processing files together
  - All c++ files together
  - etc.

Path Names

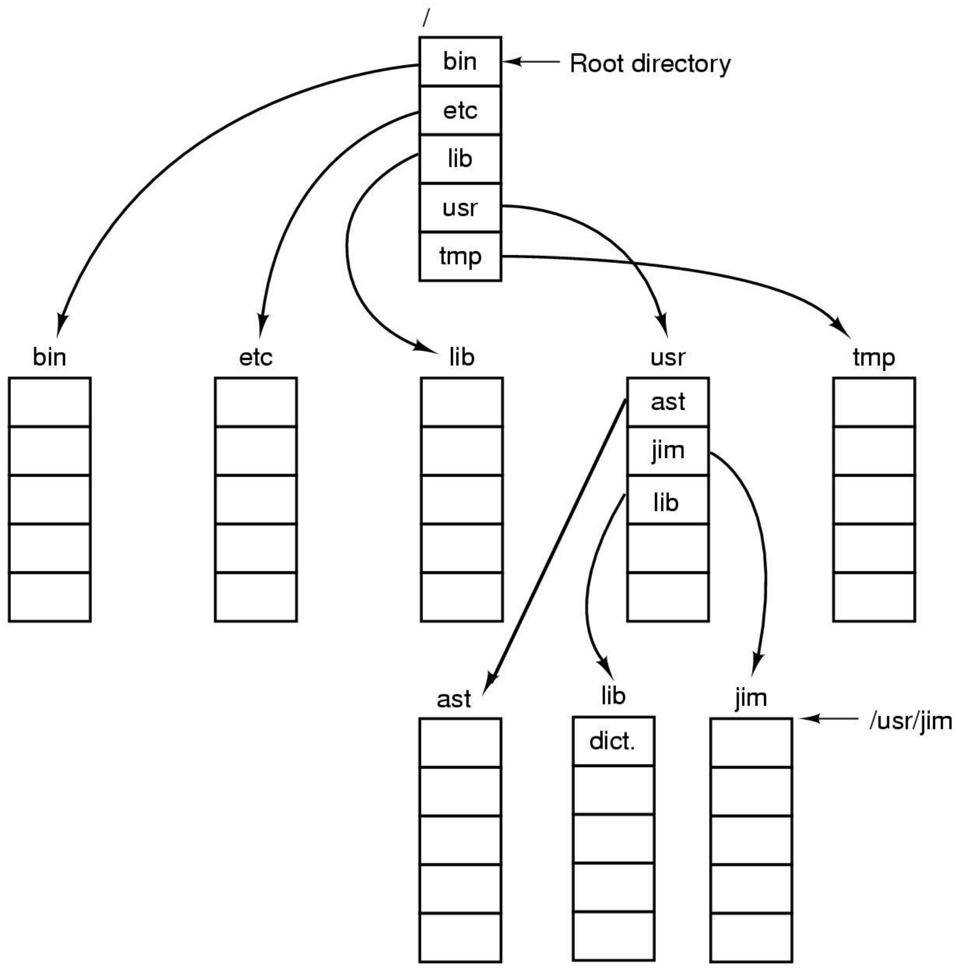


Figure 4-8. A UNIX directory tree.

Directory operation

- Hard link
  - Linking allows a file to appear in more than one directory; increments the counter in the file's i-node
- Symbolic link
  - A name is created pointing to a tiny file naming another file

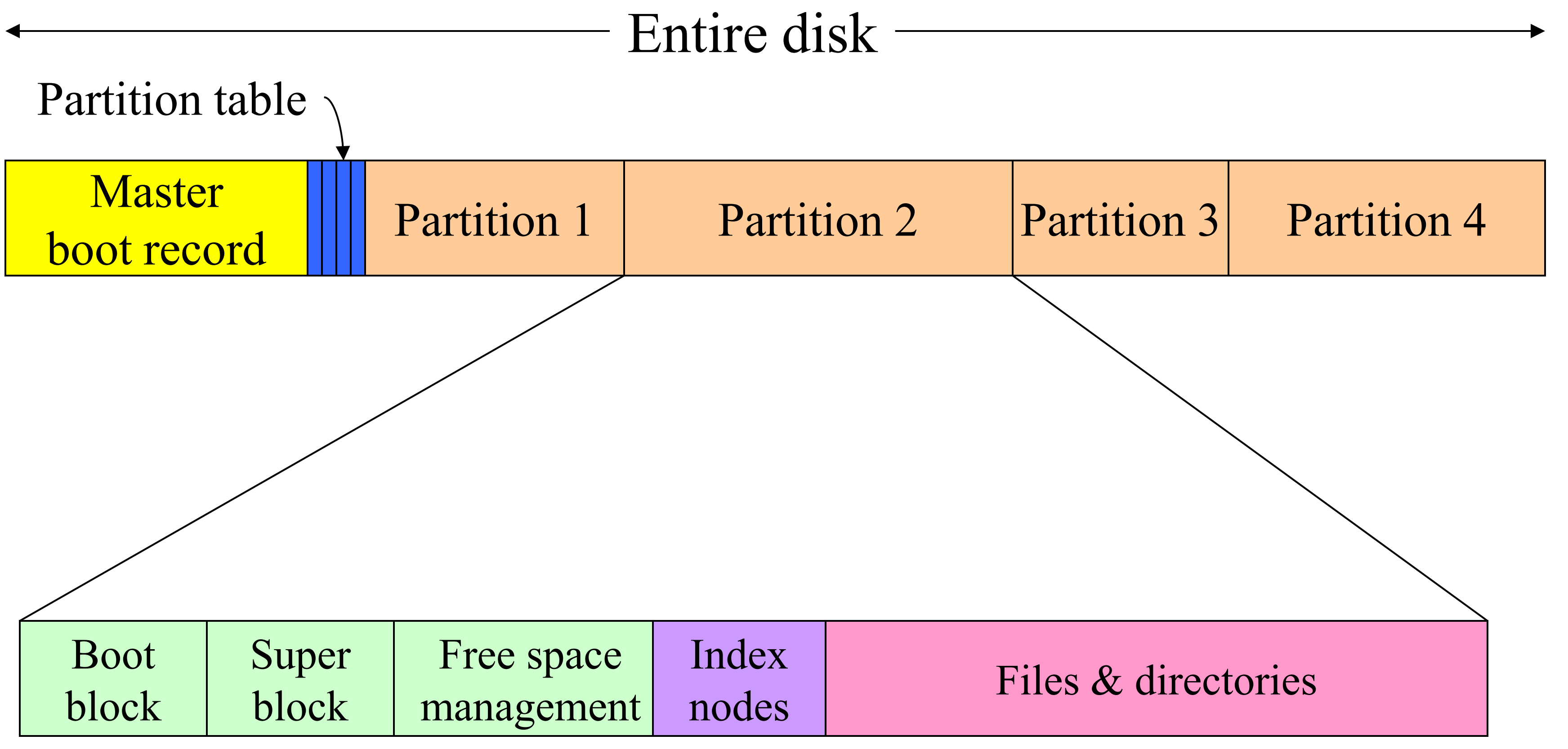
File System Implementation

- Users:
  - How files are names, what operations are allowed on them, what the directory tree looks like
- Implementors
  - How files and directories are stored, how disk space is managed and how to make every thing work efficiently and reliably

File System Layout

- File system are stored on disks.
- Most disks are divided up into several partitions
- Sector 0 is called MBR (master boot record), to boot the computer
- BIOS reads in and executes MBR, MBR locates the active partition, reads in the boot block, and execute
- The boot block reads in the OS contained in the partition
- Superblock: contains all the key parameters about a file system; read into memory the booted or the FS is used

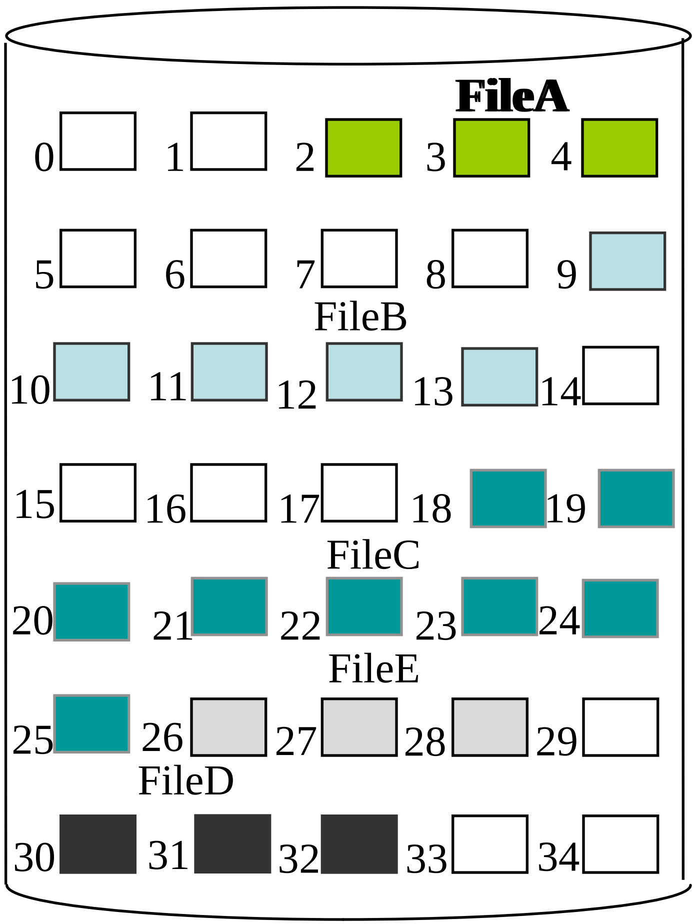
Carving up the disk



Allocation Methods
Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Number of blocks needed identified at file creation
  - May be increased using file extensions
- Advantages:
  - Simple to implement
  - Good for random access of data
- Disadvantages
  - Files cannot grow
  - Wastes space

Contiguous Allocation

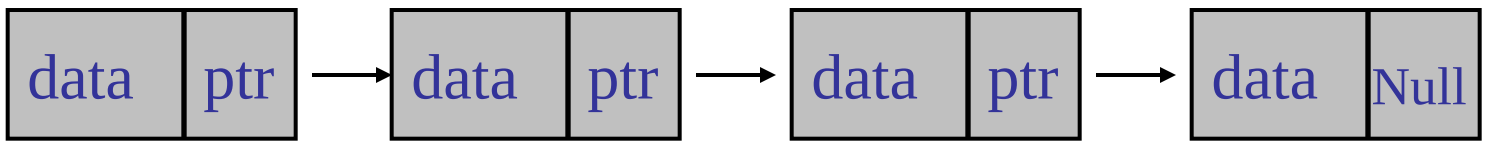


File Allocation Table

File Name	Start Block	Length
FileA	**2**	**3**
FileB	9	5
FileC	18	8
FileD	30	2
FileE	26	3

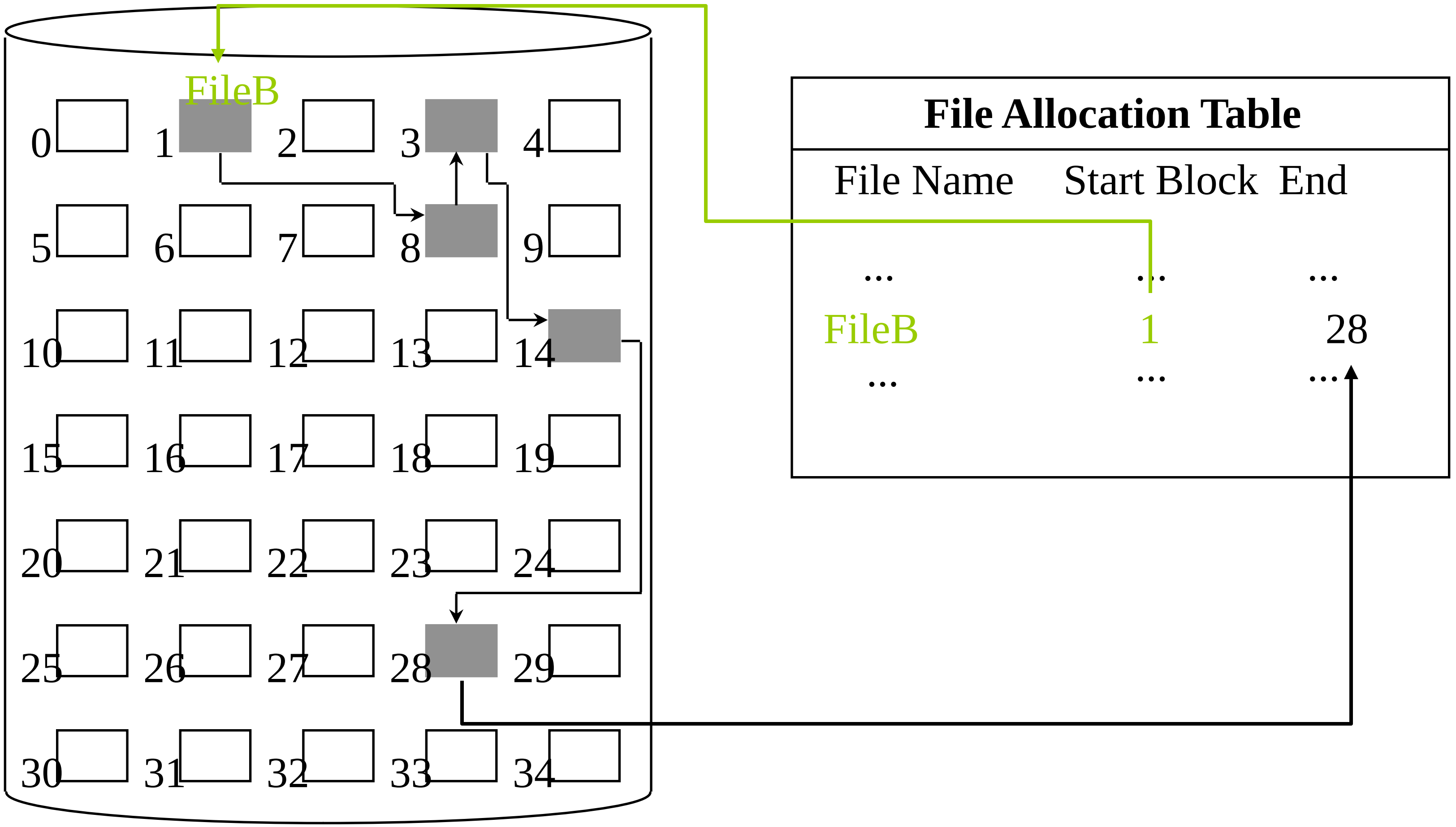
Allocation Methods
Linked Allocation

- Each file consists of a linked list of disk blocks.

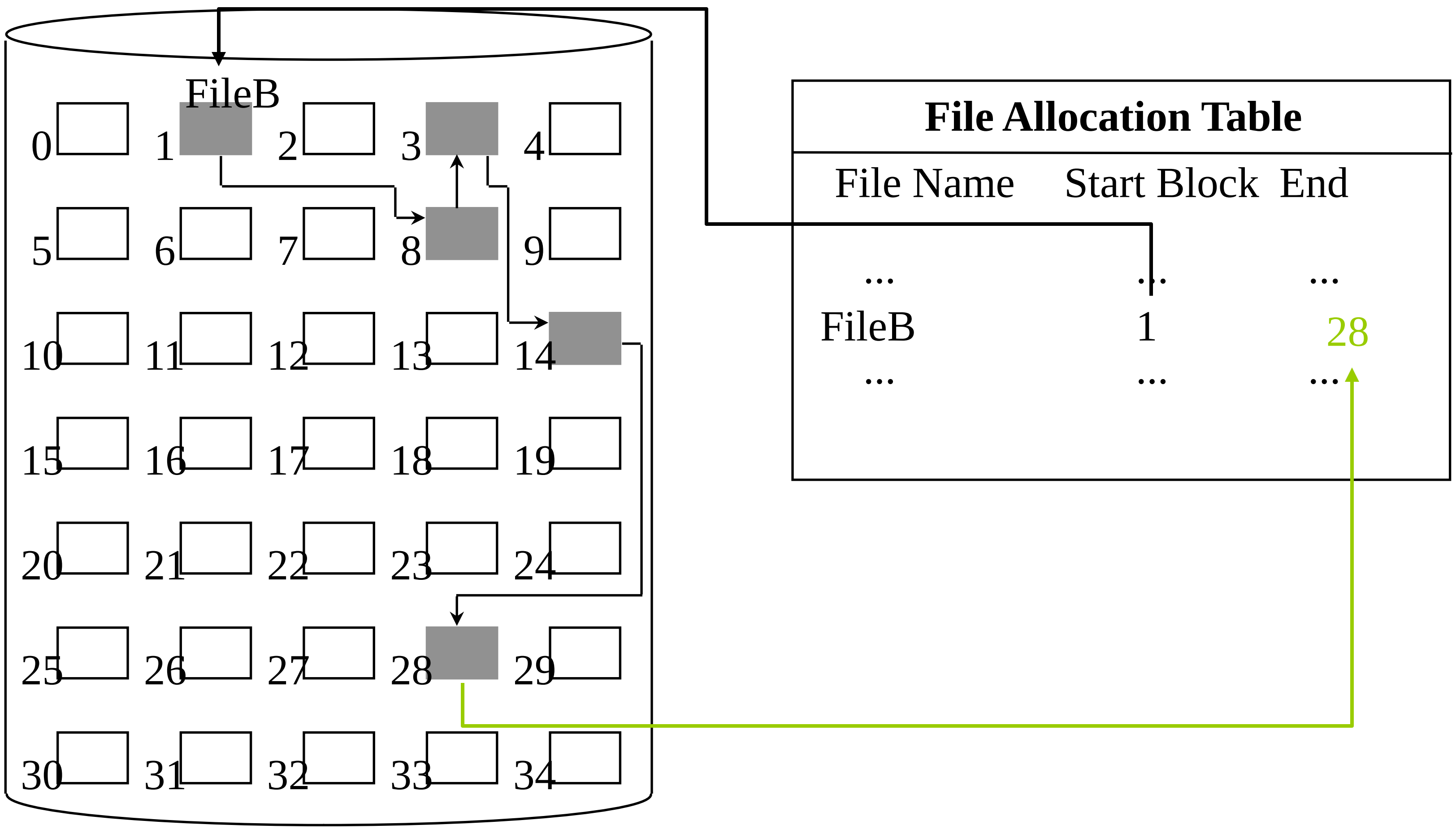


- Advantages:
  - Simple to use (only need a starting address)
  - Good use of free space
- Disadvantages:
  - Random Access is difficult

Linked Allocation

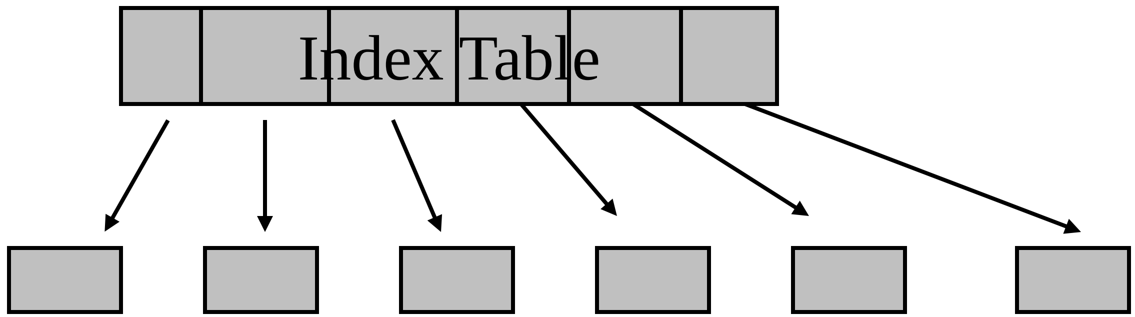


Linked Allocation



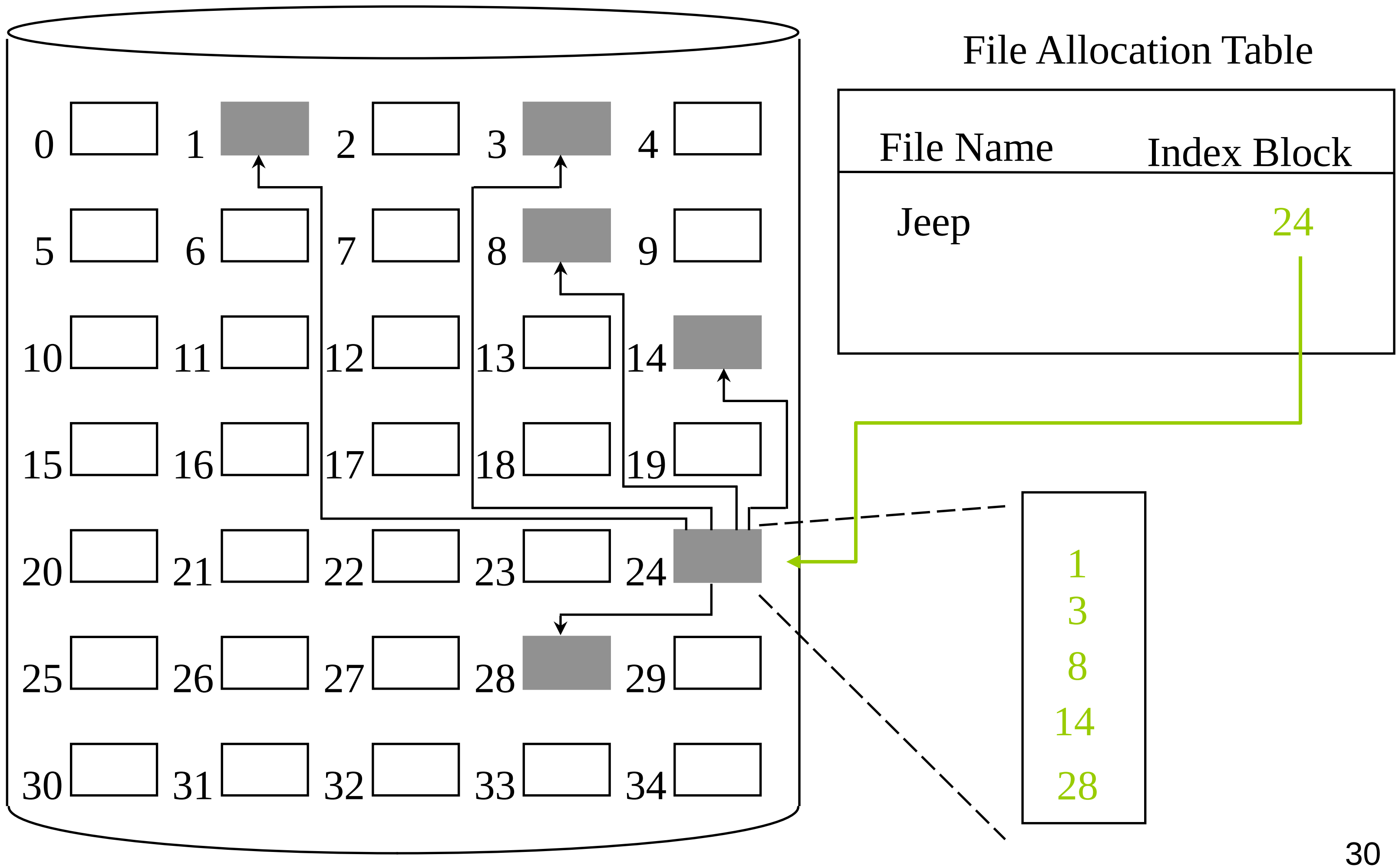
Allocation Methods
Indexed Allocation

- Collect all block pointers into an index block.

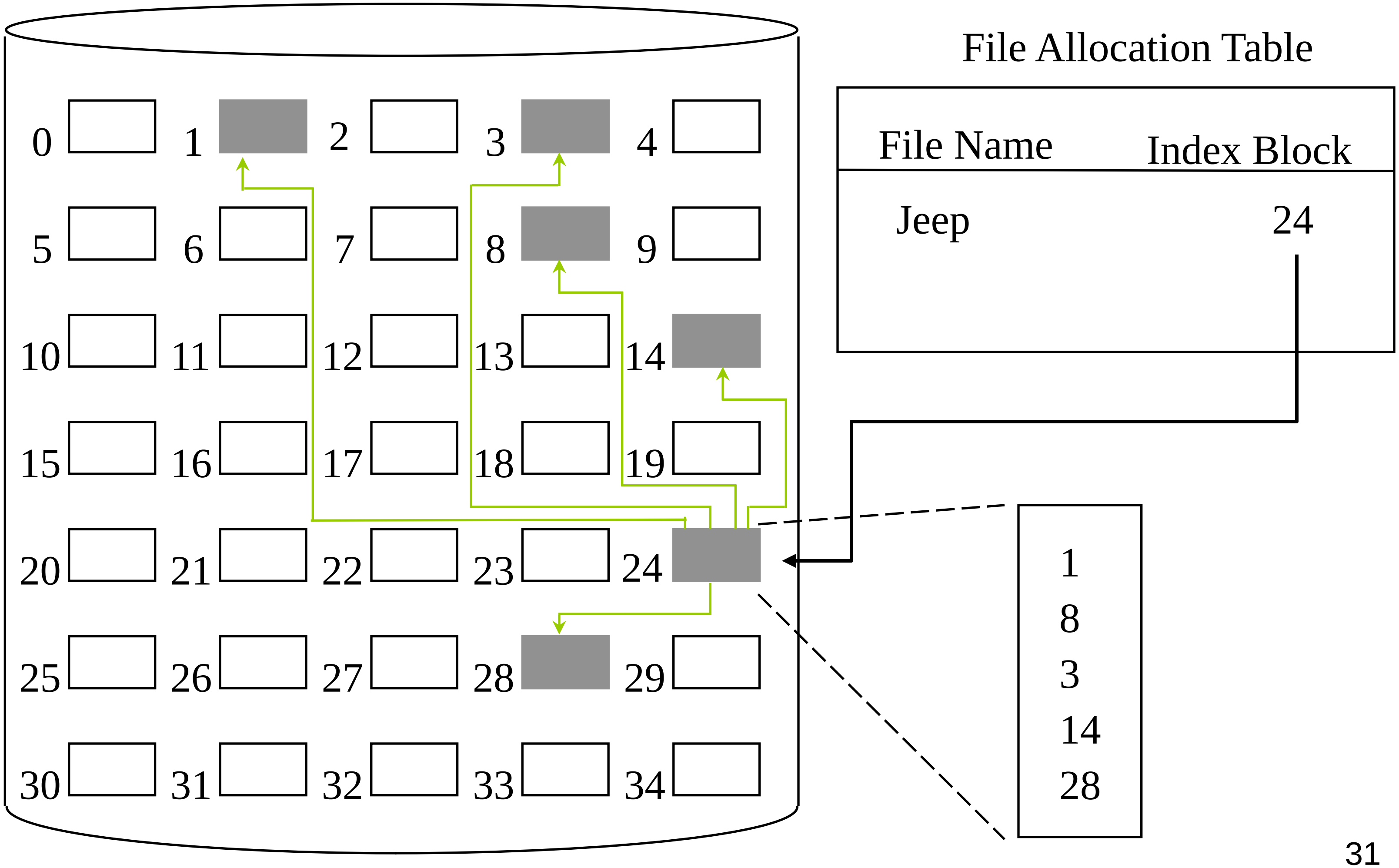


- Advantages:
  - Random Access is easy
  - No external fragmentation
- Disadvantages
  - Overhead of index block

Indexed Allocation



Indexed Allocation



Linked List Allocation Using a Table in Memory

- FAT-File Allocation Table
- Advantage
  - Can take use of the whole block
  - Random access is easy
  - only to store the starting block number
- Disadvantage
  - To keep the entire table in memory
  - Can’t scale well

I-nodes

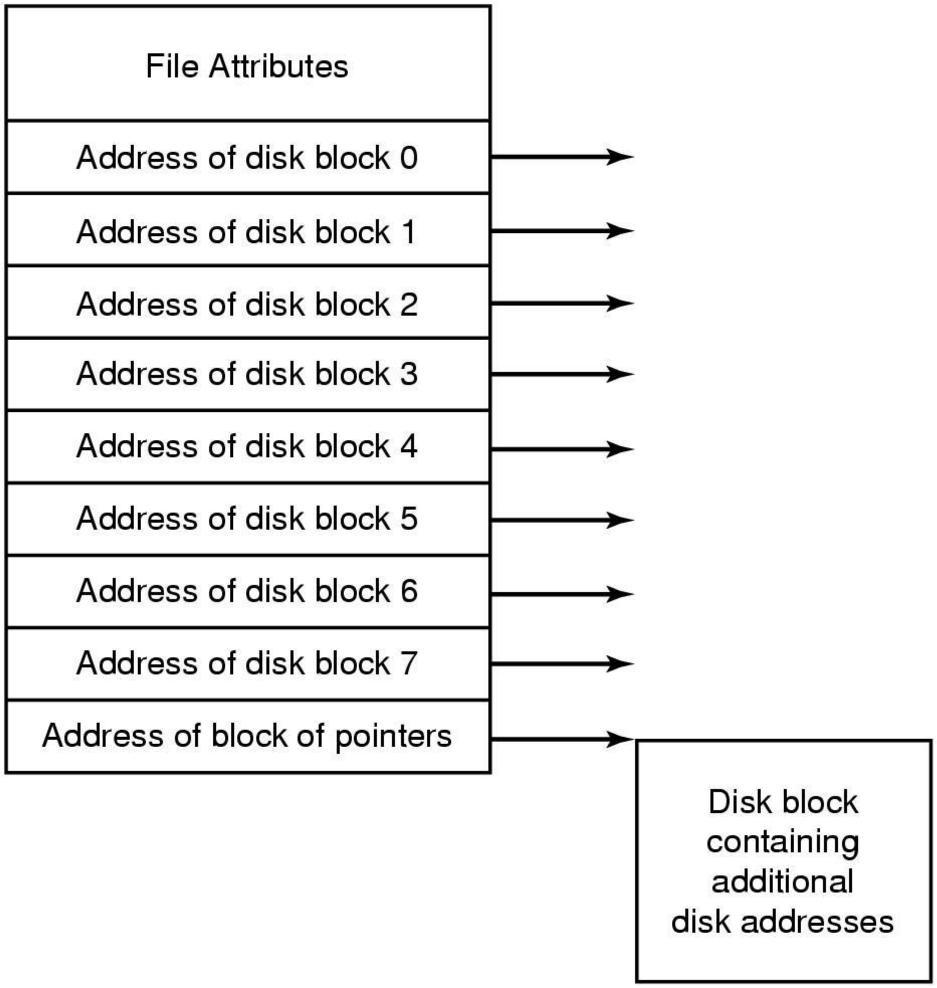


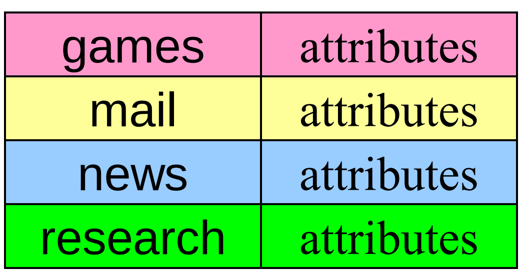
Figure 4-13. An example i-node.

i-nodes

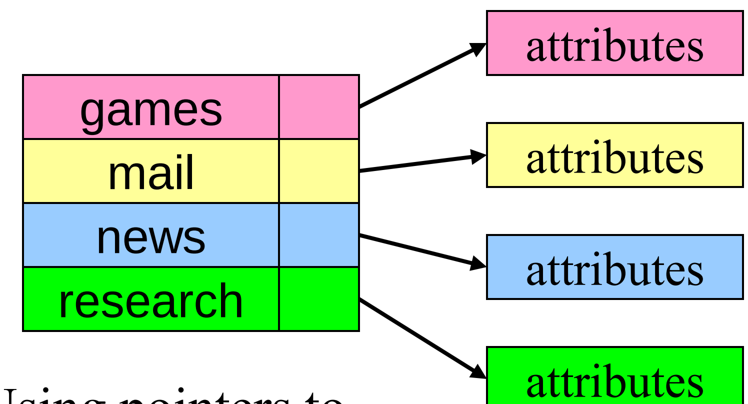
- Advantage
  - i-node need only be in memory when the corresponding file is open; file table grows linearly with the disk
- Disadvantage
  - Each i-node has fixed size

What's in a directory?

- Two types of information
  - File names
  - File metadata (size, timestamps, etc.)
- Basic choices for directory information
  - Store all information in directory
    - Fixed size entries
    - Disk addresses and attributes in directory entry
  - Store names & pointers to index nodes (i-nodes)



Storing all information in the directory



Using pointers to index nodes

Implementing Directories (1)

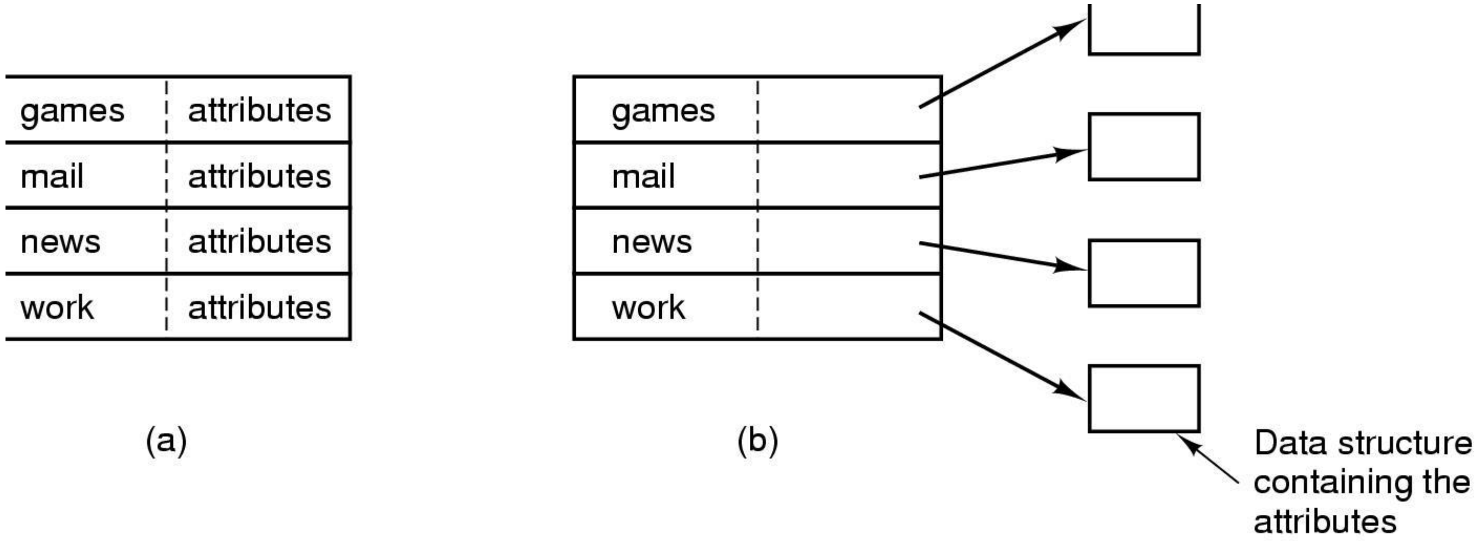


Figure 4-14. (a) A simple directory containing fixed-size entries with the disk addresses and attributes in the directory entry. (b) A directory in which each entry just refers to an i-node.

Implementing Directories (2)

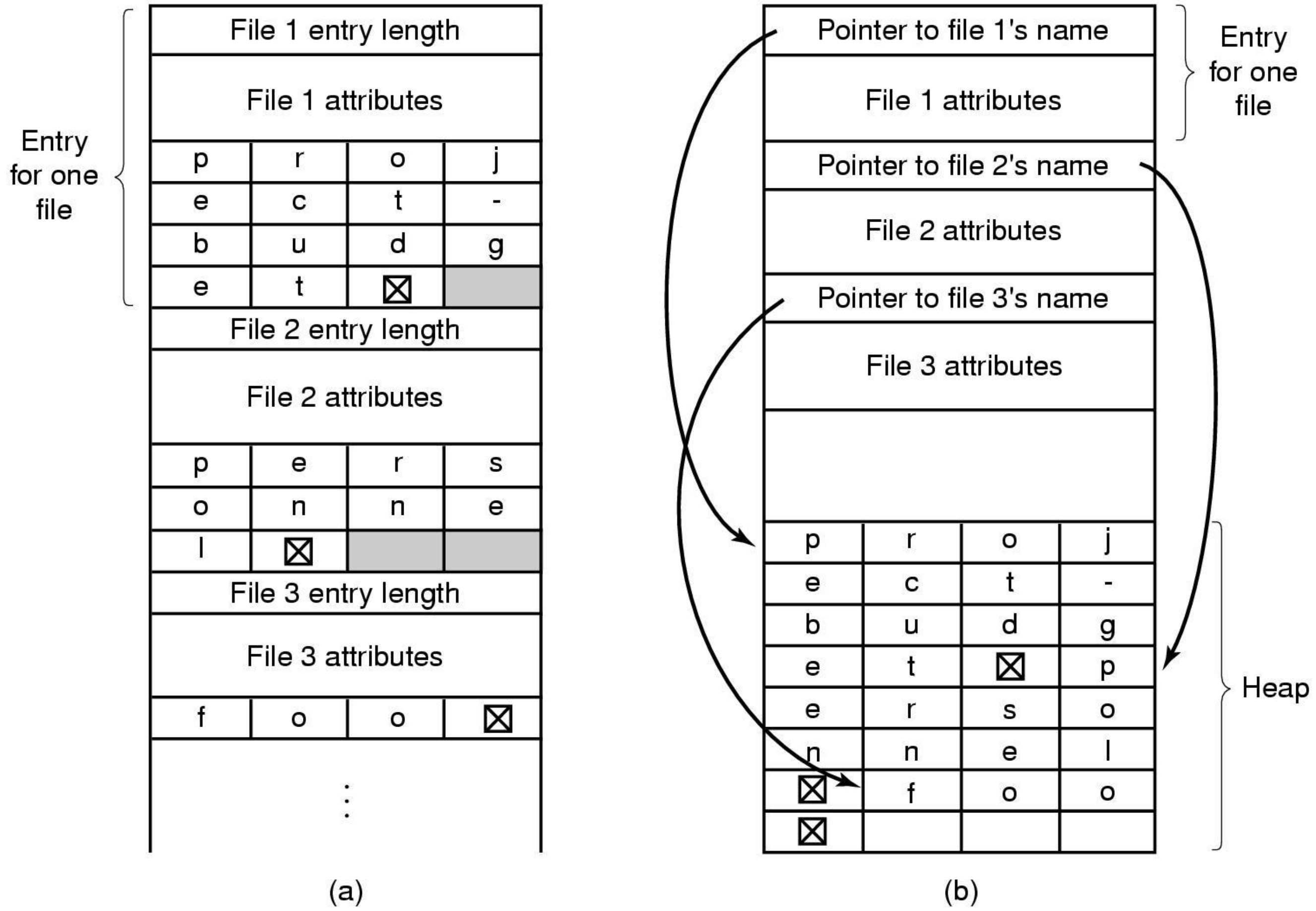


Figure 4-15. Two ways of handling long file names in a directory. (a) In-line. (b) In a heap.

Shared Files (2)

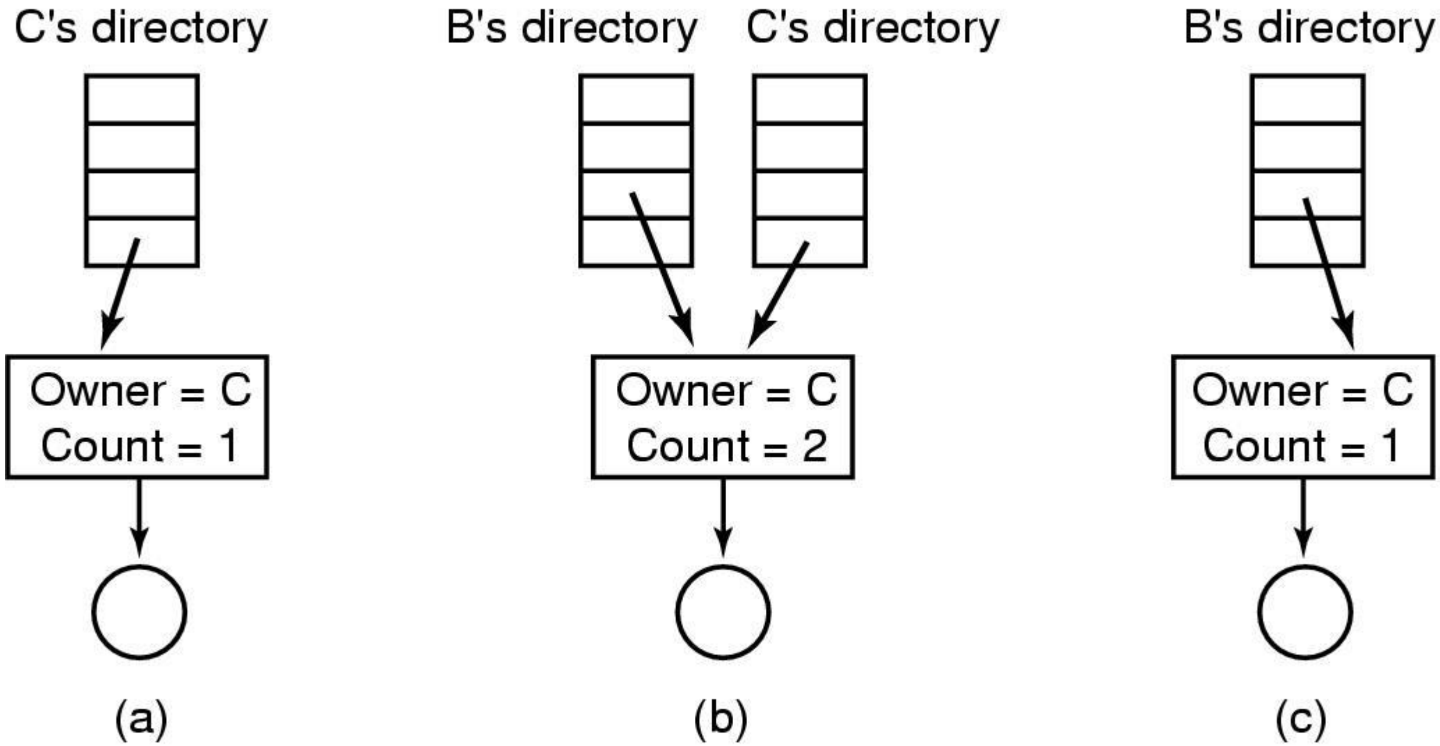
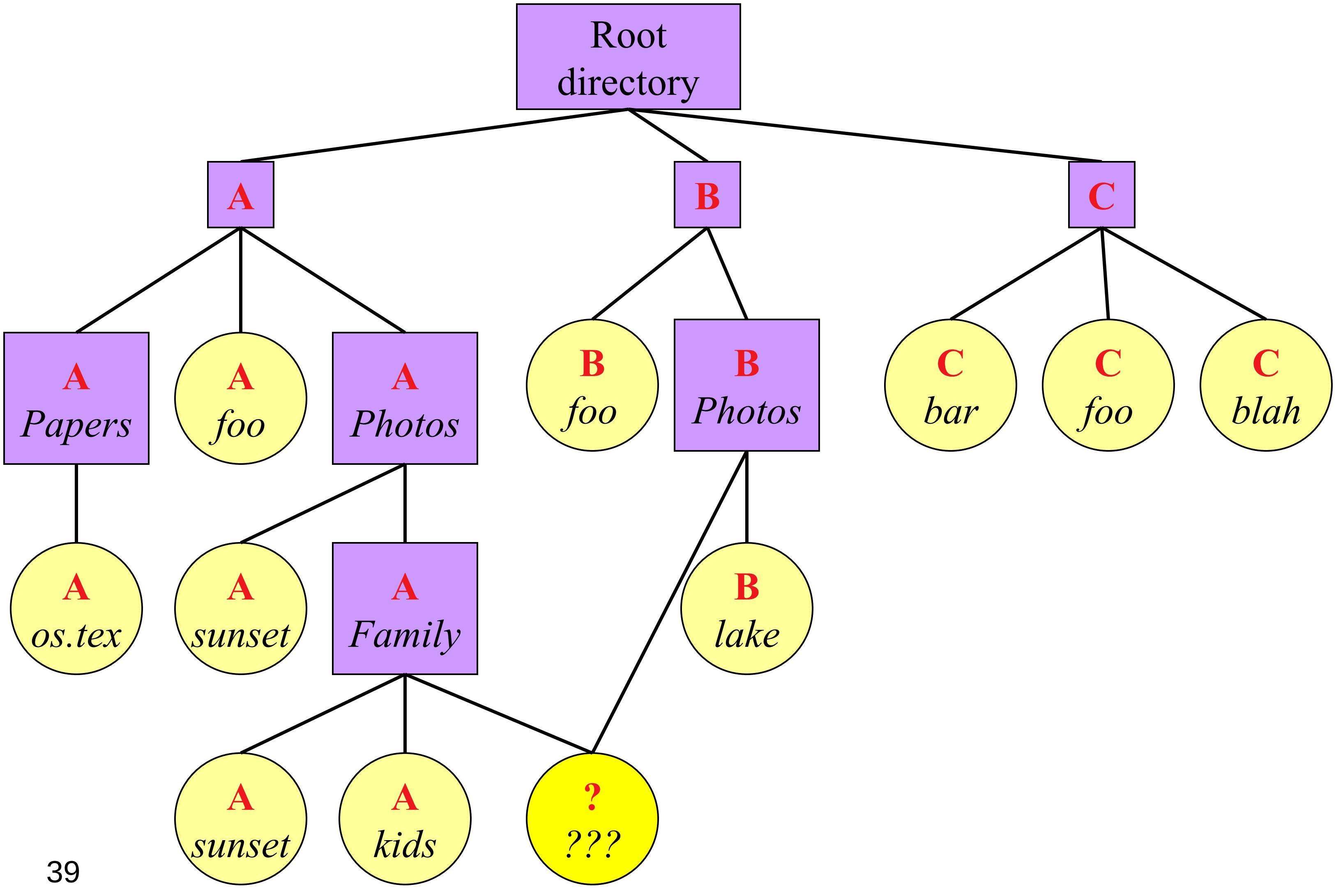


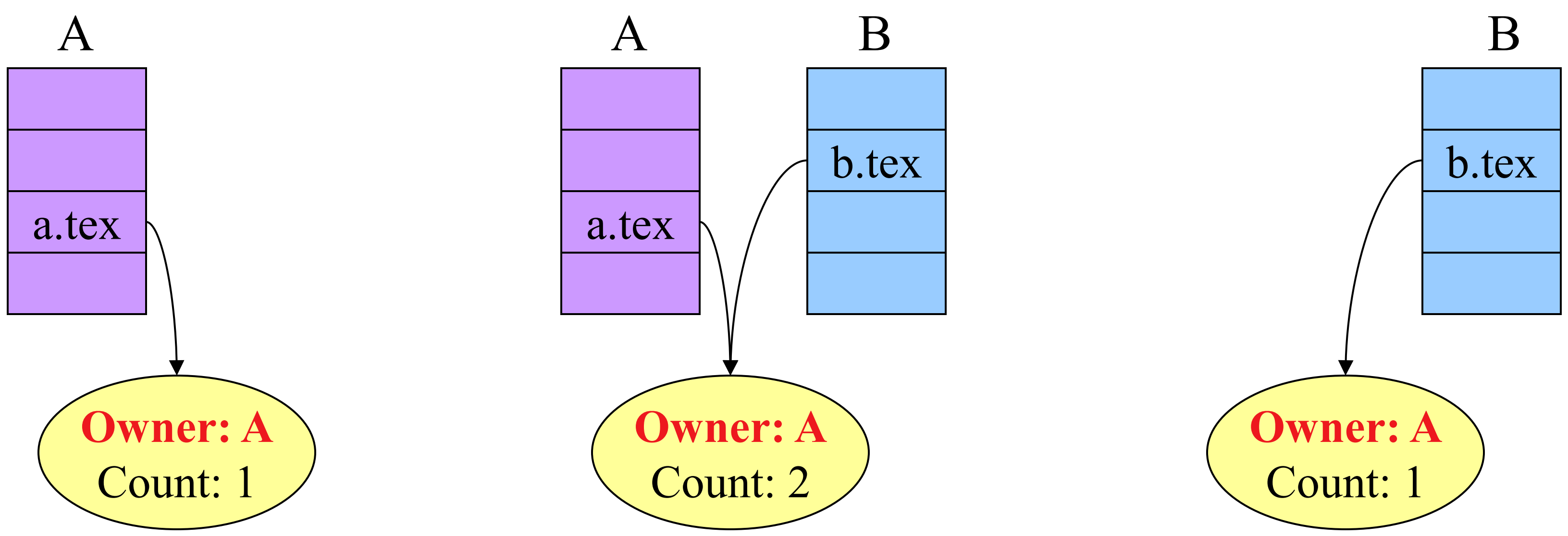
Figure 4-17. (a) Situation prior to linking. (b) After the link is created. (c) After the original owner removes the file.

Sharing files



Solution: use links

- A creates a file, and inserts into her directory
- B shares the file by creating a link to it
- A unlinks the file
  - B still links to the file
  - Owner is still A (unless B explicitly changes it)



The MS-DOS File System (1)

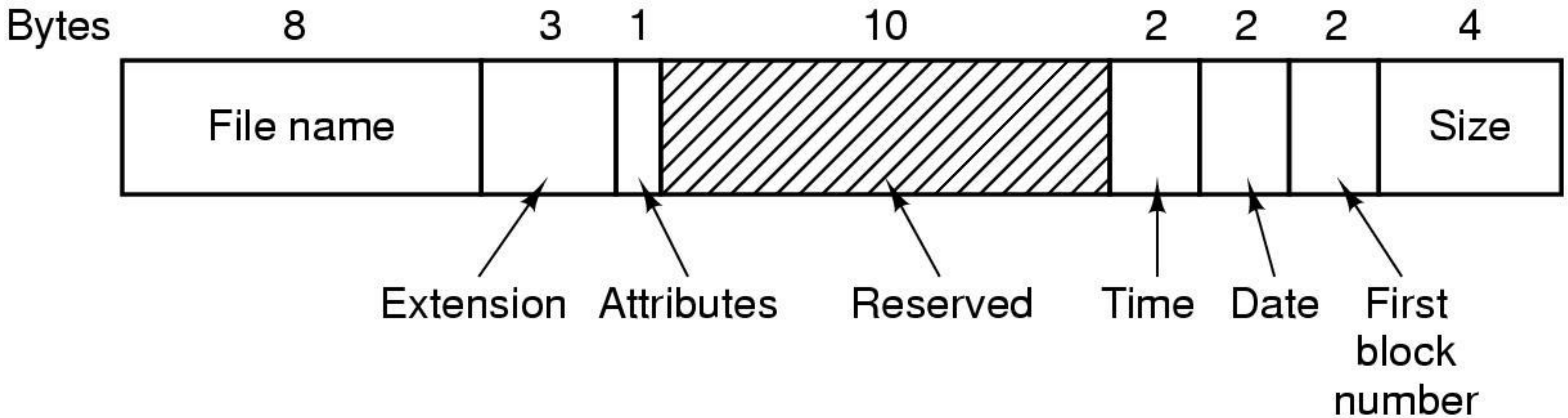


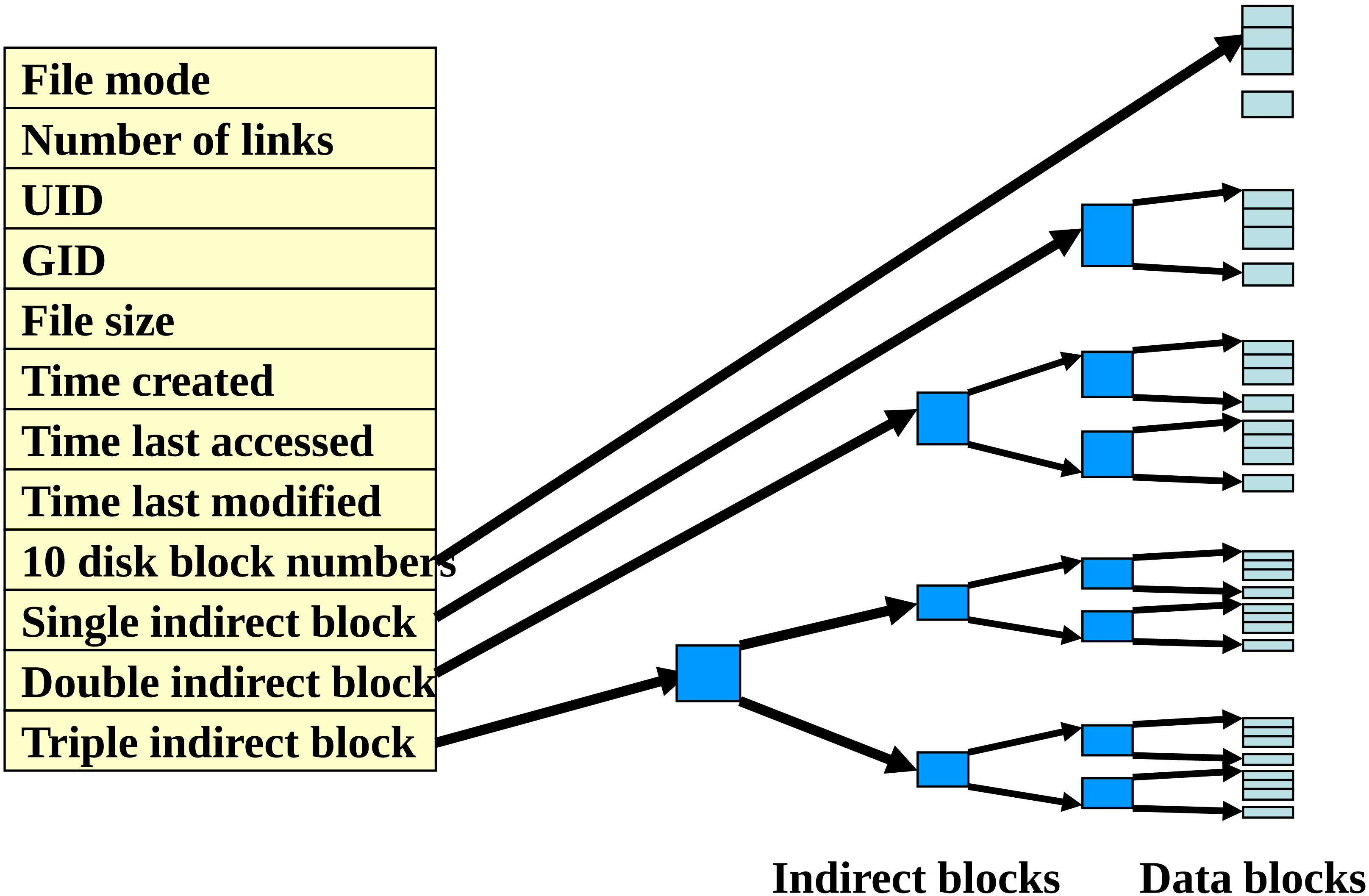
Figure 4-31. The MS-DOS directory entry.

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

Figure 4-32. Maximum partition size for different block sizes. The empty boxes represent forbidden combinations.

i (index)-nodes (UNIX)



i-nodes (Cont.)

- Assume each block is 1 KB in size and 32 bits (4 bytes) are used as block numbers
- Each indirect block holds 256 block numbers
- ***First 10 blocks*** : file size <= 10 KB
- ***Single indirect*** : file size <= 256+10 = 266 KB
- ***Double indirect*** : file size <= 256*256 +266 = 65802 KB = 64.26 MB
- ***Triple indirect***: file size <= 256*256*256 + 65802= 16843018 KB = ~16 GB

EXT Details

- Directory Structure
  - The improved byte allocation is as follows:
    - 0-3 Inode value
    - 4-5 Length of entry
    - 6 Length of name (up to 255 now)
    - 7 File type
      - » 0 unknown
      - » 1 regular file
      - » 2 directory
      - » 3 character device
      - » 4 block device
      - » 5 FIFO
      - » 6 Unix Socket
      - » 7 Symbolic link
    - 8- Name in ASCII

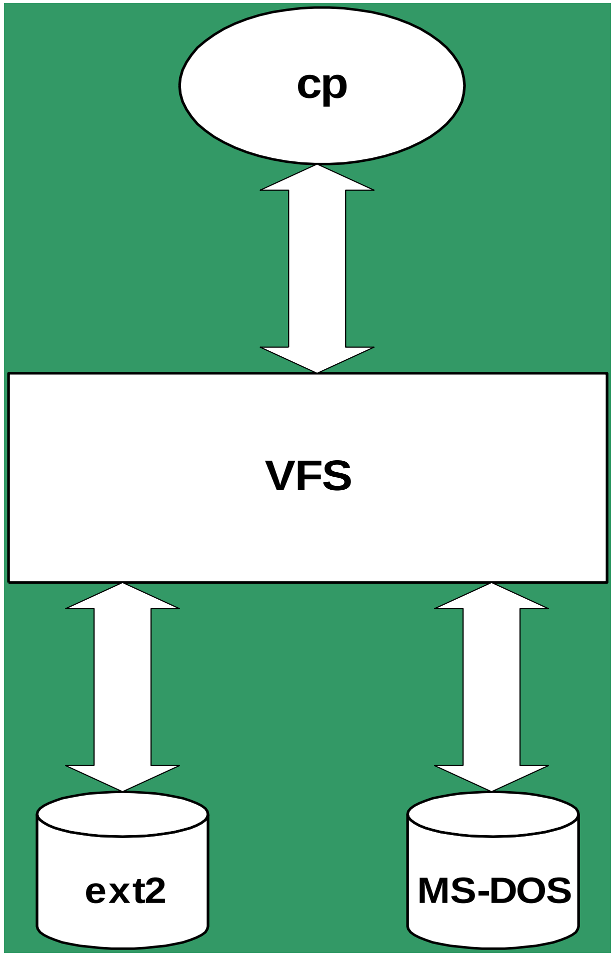
Linux File System Structure

- Linux uses a Virtual File System (VFS)
  - Defines a file object
  - Provides an interface to manipulate that object
- Designed around OO principles
  - File system object
  - File object
  - Inode object (index node)
- Primary File System - ext2fs
  - Supports (or maps) several other systems (MSDOS, NFS (network drives), VFAT (W95), HPFS (OS/2), etc.

Virtual Filesystem

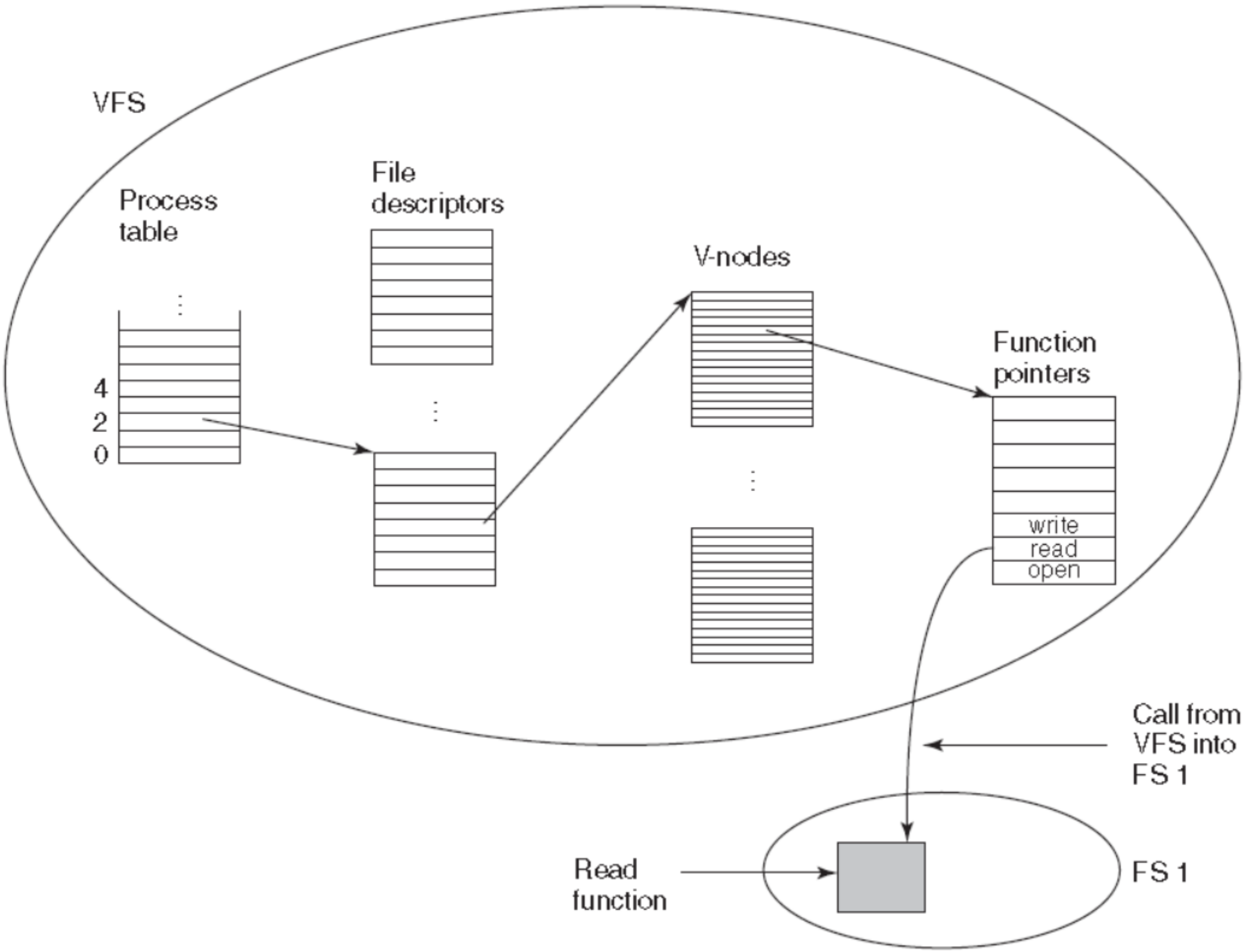
- A kernel software layer that handles all system calls related to a standard UNIX filesystem.
- Supports:
  - Disk-based filesystems
    - IDE Hard drives (UNIX, LINUX, SMB, etc.)
    - SCSI Hard drives
    - floppy drives
  - Network filesystems
    - remotely connected filesystems
  - Special filesystems
    - /proc

VF Example




```
inf =  open ("/floppy/test",
   O_RDONLY, 0);
outf = open ("/tmp/test",
   O_WRONLY|O_CREATE|O_TRUNC,
   0600);
do {
   cnt = read(inf, buf, 4096);
   write (outf, buf, cnt);
} while (cnt);
close (outf);
close (inf);
```

Virtual File system and Processes



Disk Space Management

- Files are normally stored on disk, so management of disk space is a major concern to file-system designers.
- Two general strategies are possible for storing an *n* byte file
  - *n* consecutive bytes of disk space are allocated, or the file is split up into a number of (not necessarily) contiguous blocks
  - chop files up into fixed-size blocks that need not be adjacent

Disk Space Management

- Block size
  - the question arises how big the block should be

Disk space Management

Length	VU 1984	VU 2005	Web
1	1.79	1.38	6.67
2	1.88	1.53	7.67
4	2.01	1.65	8.33
8	2.31	1.80	11.30
16	3.32	2.15	11.46
32	5.13	3.15	12.33
64	8.71	4.98	26.10
128	14.73	8.03	28.49
256	23.09	13.29	32.10
512	34.44	20.62	39.94
1 KB	48.05	30.91	47.82
2 KB	60.87	46.09	59.44
4 KB	75.31	59.13	70.64
8 KB	84.97	69.96	79.69

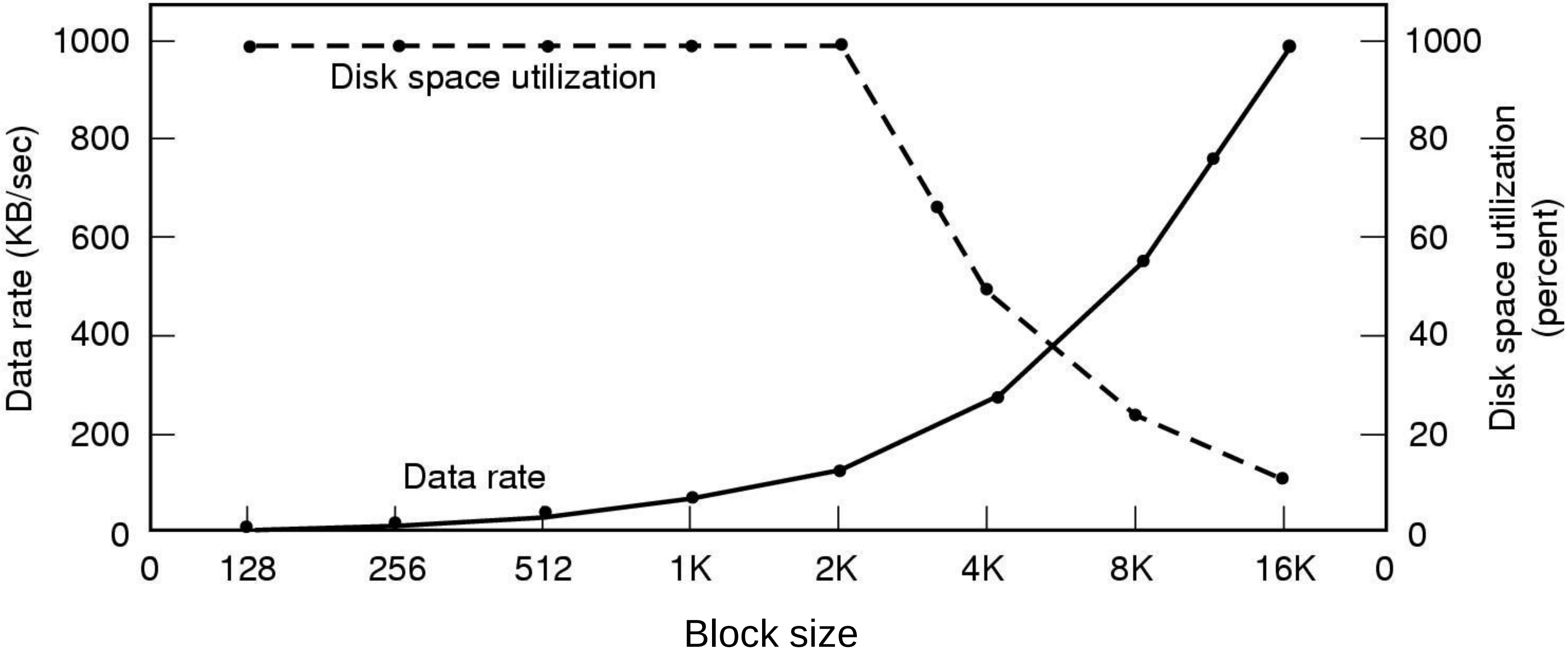
Length	VU 1984	VU 2005	Web
16 KB	92.53	78.92	86.79
32 KB	97.21	85.87	91.65
64 KB	99.18	90.84	94.80
128 KB	99.84	93.73	96.93
256 KB	99.96	96.12	98.48
512 KB	100.00	97.73	98.99
1 MB	100.00	98.87	99.62
2 MB	100.00	99.44	99.80
4 MB	100.00	99.71	99.87
8 MB	100.00	99.86	99.94
16 MB	100.00	99.94	99.97
32 MB	100.00	99.97	99.99
64 MB	100.00	99.99	99.99
128 MB	100.00	99.99	100.00

Figure 4-20. Percentage of files smaller than a given size (in bytes).

- As an example, consider a disk with 1 MB per track, a rotation time of 8.33 msec, and an average seek time of 5 msec. The time in milliseconds to read a block of *k* bytes is then the sum of the seek, rotational delay, and transfer times:

$$5 + 4.165 + (k/1000000) \times 8.33$$

Disk Space Management



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

Free Space Management

- Bit Vector management

0	1	1	0	.....	1

- One bit for each block
  - 0 = free; 1 = occupied
- Use bit manipulation commands to find free block
- Bit vector requires space
  - block size = 4096 = 2^{12}
  - disk size = 1 gigabyte = 2^{30}
  - bits = $2^{(30-12)} = 2^{18}$ = 32k bytes

Free Space Management

- Bit vector (advantages):
  - Easy to find contiguous blocks
- Bit vector (disadvantages):
  - Wastes space (bits allocated to unavailable blocks)
- Issues:
  - Must keep bit vector on disk (reliability)
  - Must keep bit vector in memory (speed)

Keeping Track of Free Blocks (1)

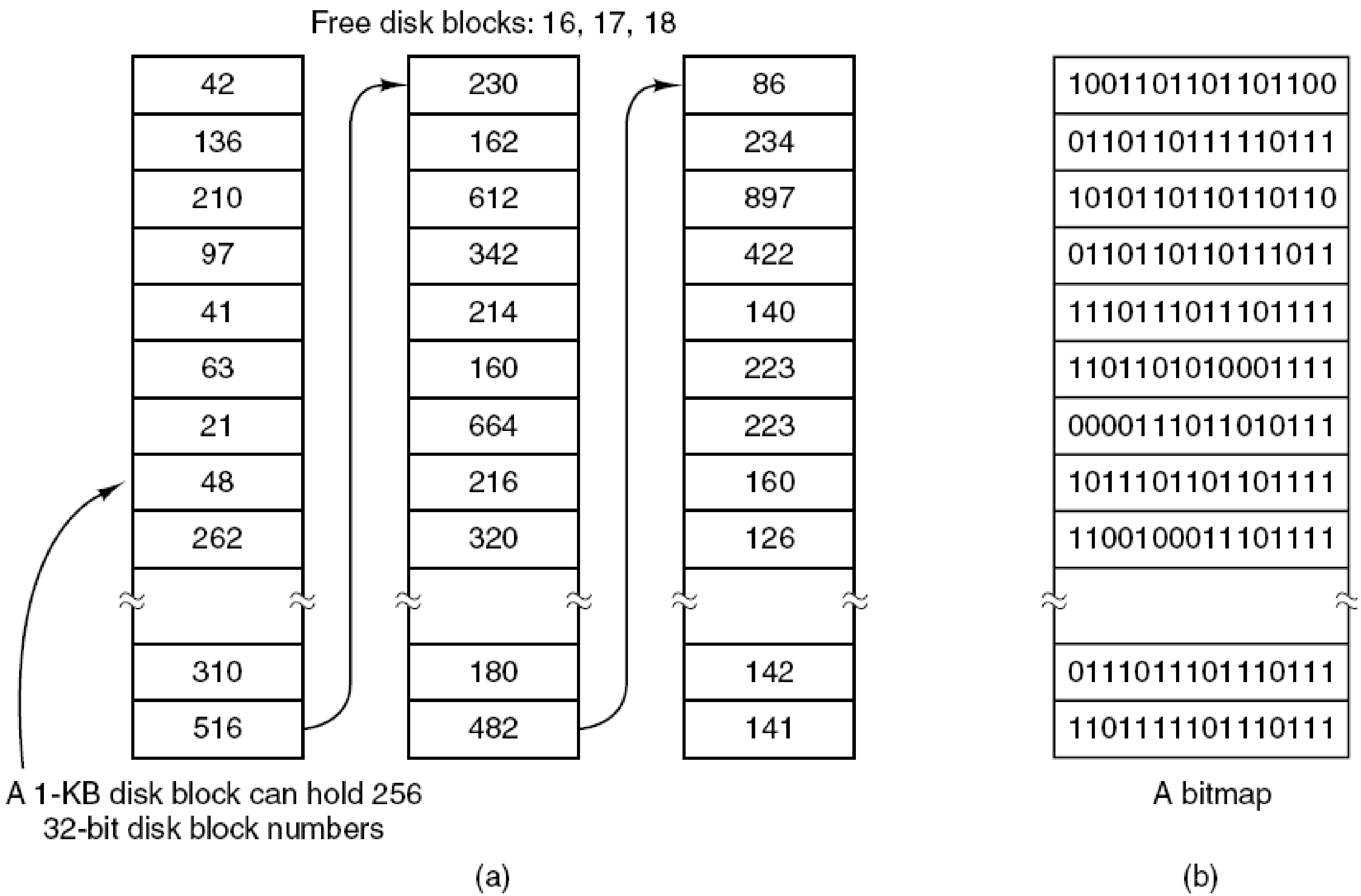


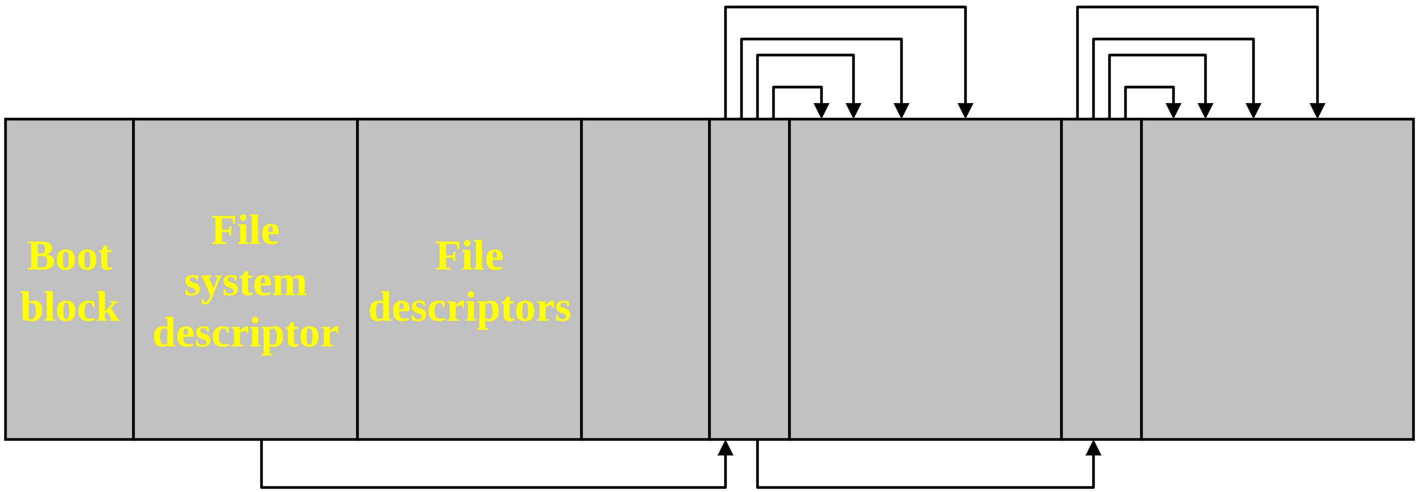
Figure 4-22. (a) Storing the free list on a linked list. (b) A bitmap.

Free Space Management

- Linked List management
  - Use linked list to identify free space
- Advantages:
  - no wasted space
- Disadvantages:
  - harder to identify contiguous space.
- Issues:
  - Must protect pointer to free list

Free Space Management

- Grouping of blocks



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 Backups (1)

- should the entire file system be backed up or only part of it?
- it is wasteful to back up files that have not changed since the previous backup
- since immense amounts of data are typically dumped, it may be desirable to compress the data before writing them to tape
- it is difficult to perform a backup on an active file system
- making backups introduces many nontechnical problems into an organization

File System Backups (1)

- Tw o strategies can be used for dumping a disk to a backup disk
  - Physical dump
    - Issues
      - there is no value in backing up unused disk blocks.
      - dumping bad blocks
    - The main advantages of physical dumping are simplicity and great speed
  - Logical dump

File System Backups (2)

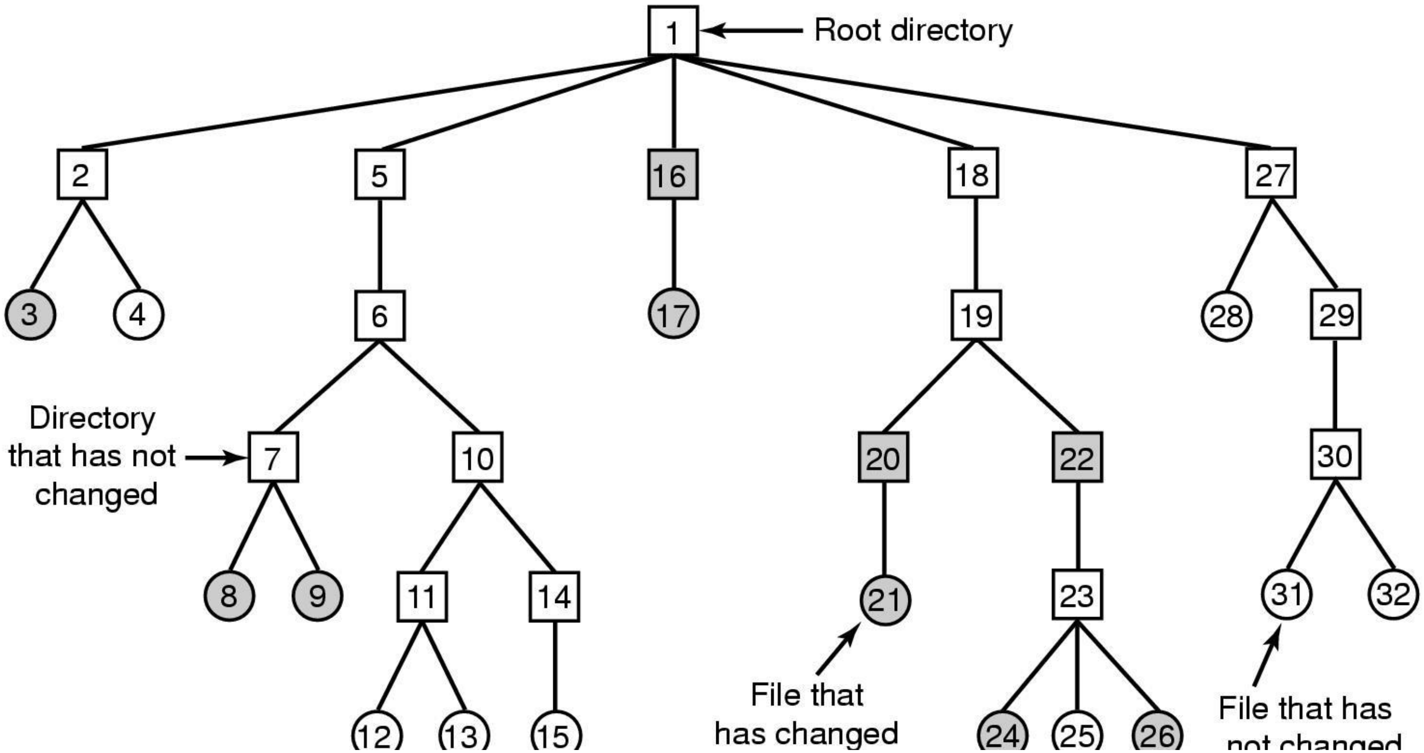


Figure 4-25. A file system to be dumped. Squares are directories, circles are files. Shaded items have been modified since last dump. Each directory and file is labeled by its i-node number.

File System Backups (3)

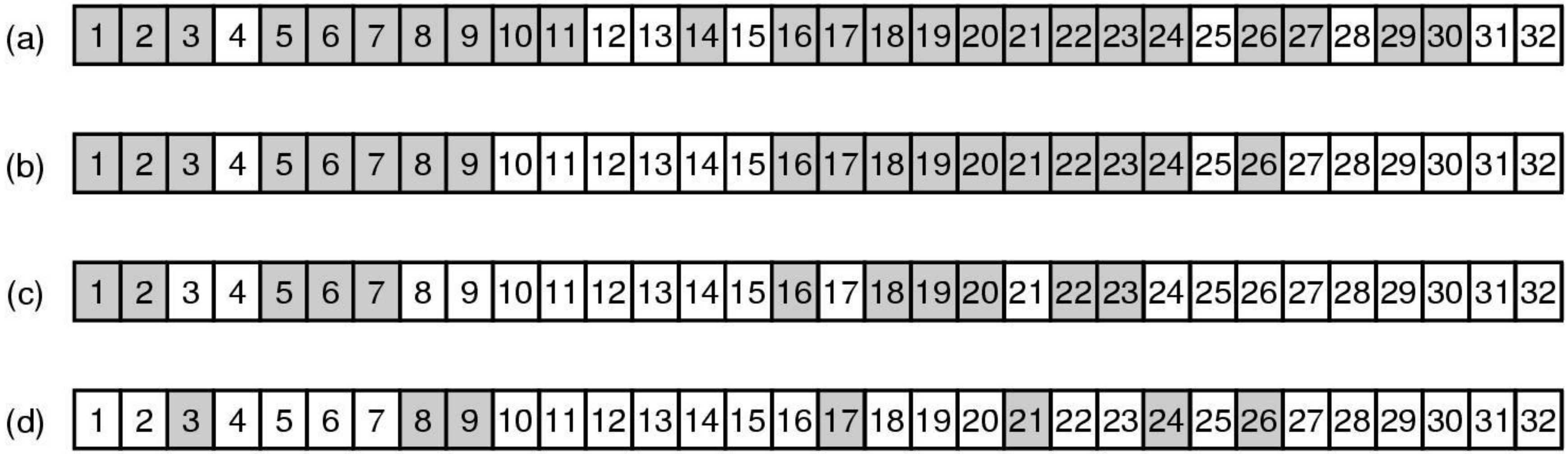


Figure 4-26. Bitmaps used by the logical dumping algorithm.

File System Consistency

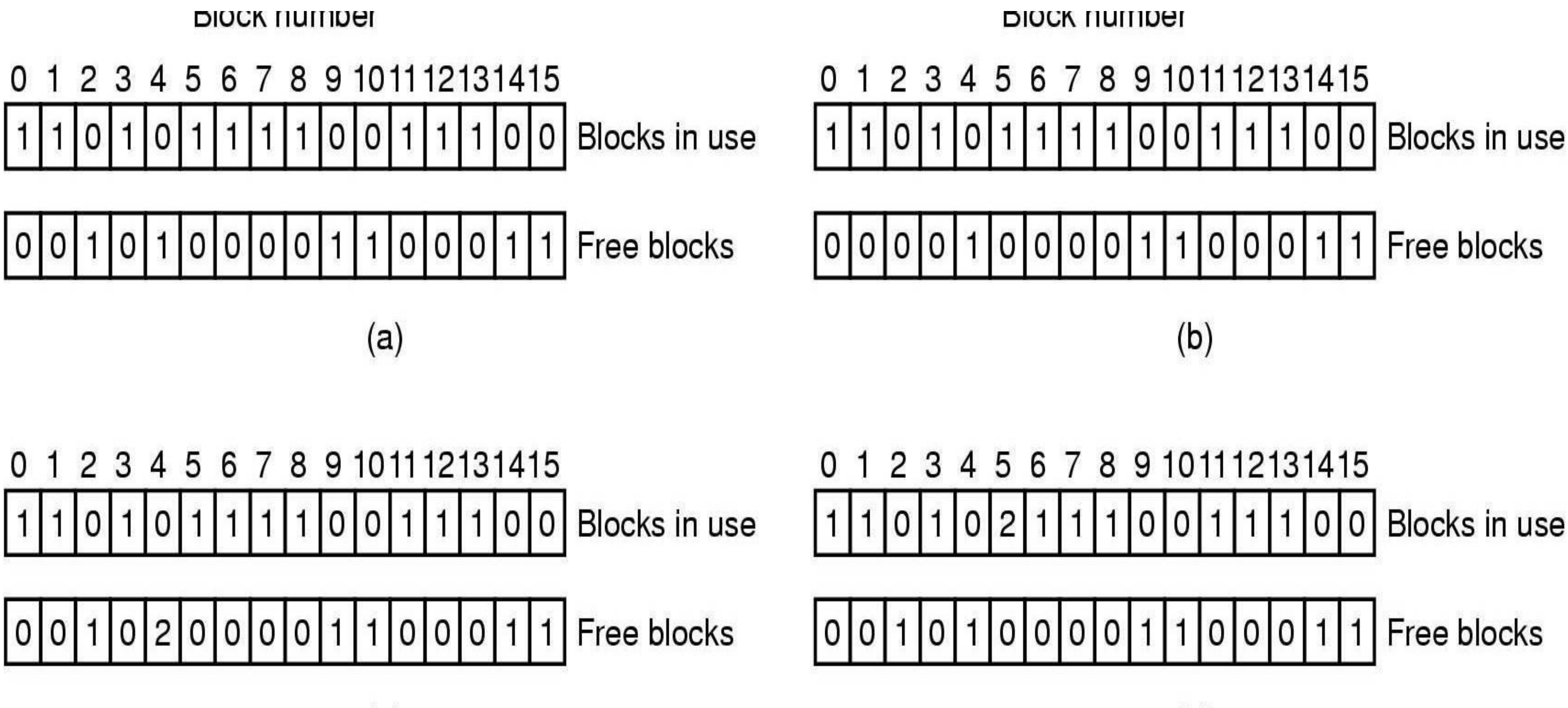


Figure 4-27. File system states. (a) Consistent. (b) Missing block. (c) Duplicate block in free list. (d) Duplicate data block.

Caching (1)

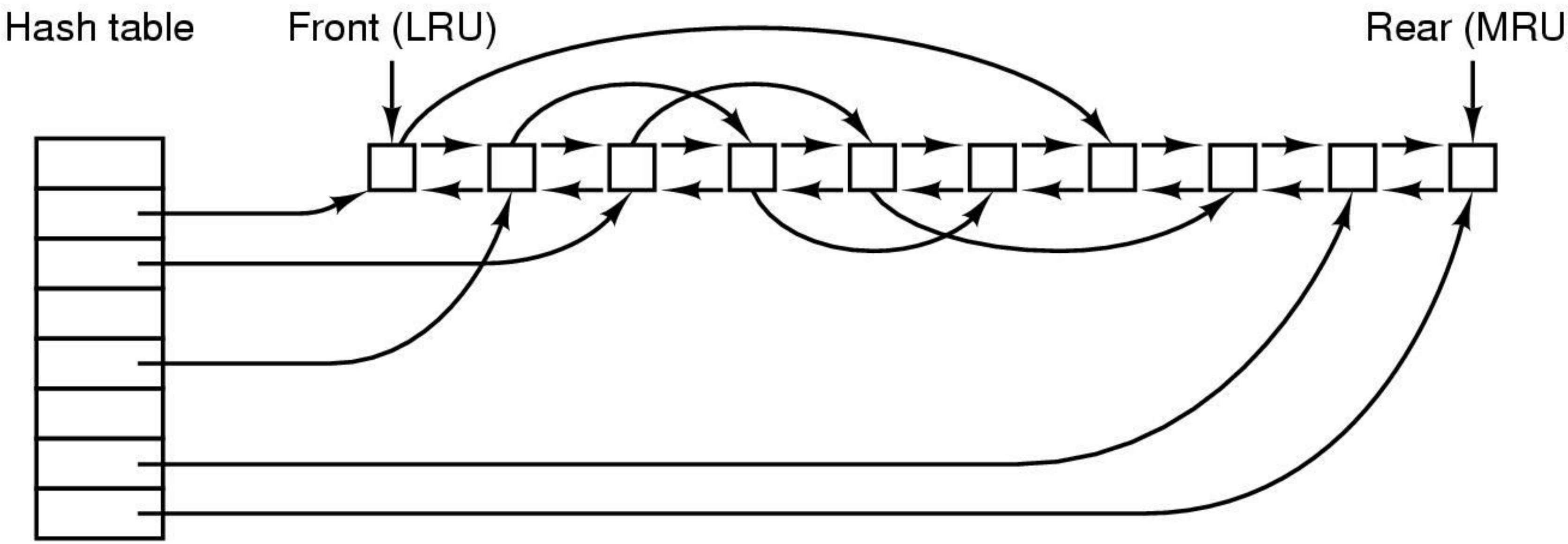


Figure 4-28. The buffer cache data structures.