MODERN OPERATING SYSTEMS

Third Edition

ANDREW S. TANENBAUM

Chapter 4
File Systems

File Systems (1)

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 Systems (2)

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 Structure

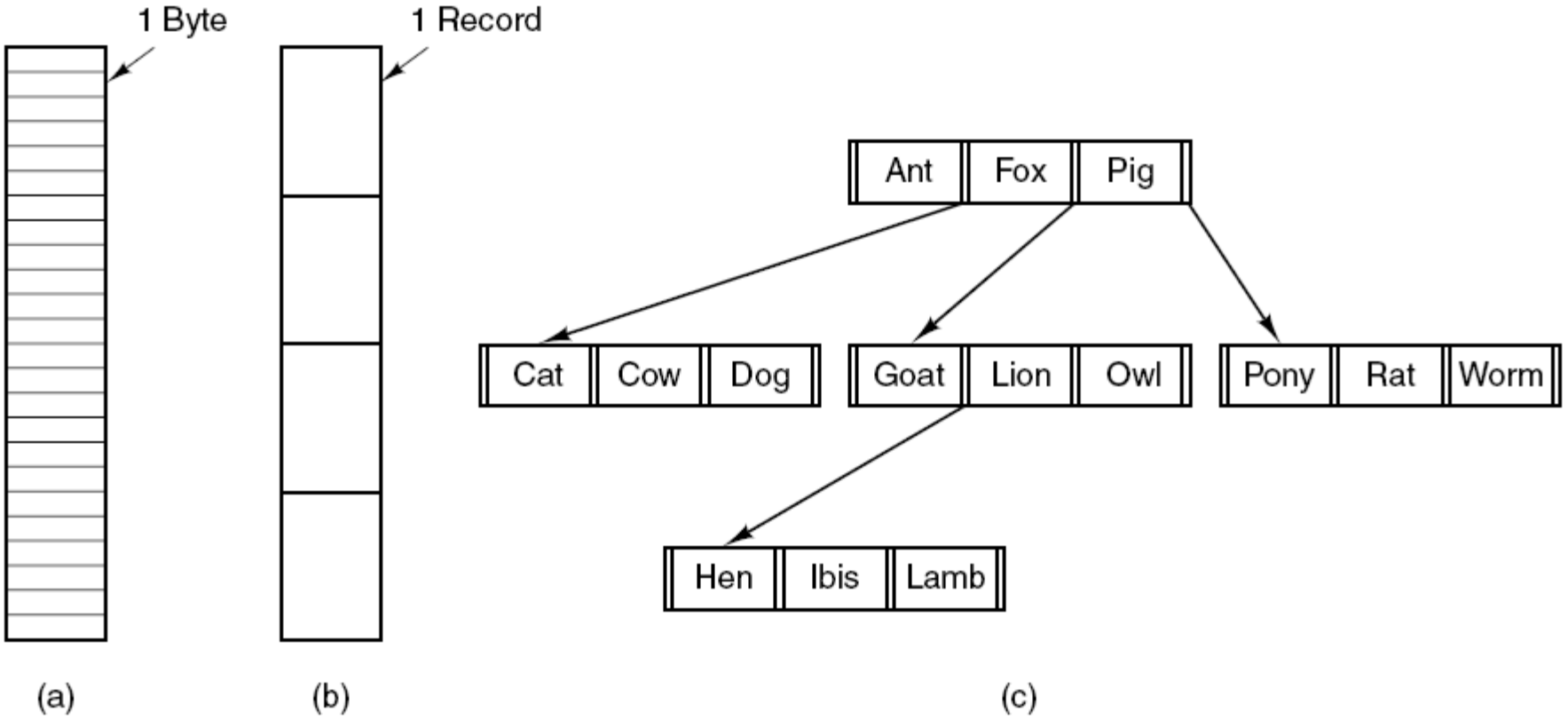

Figure 4-2. Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

File Types

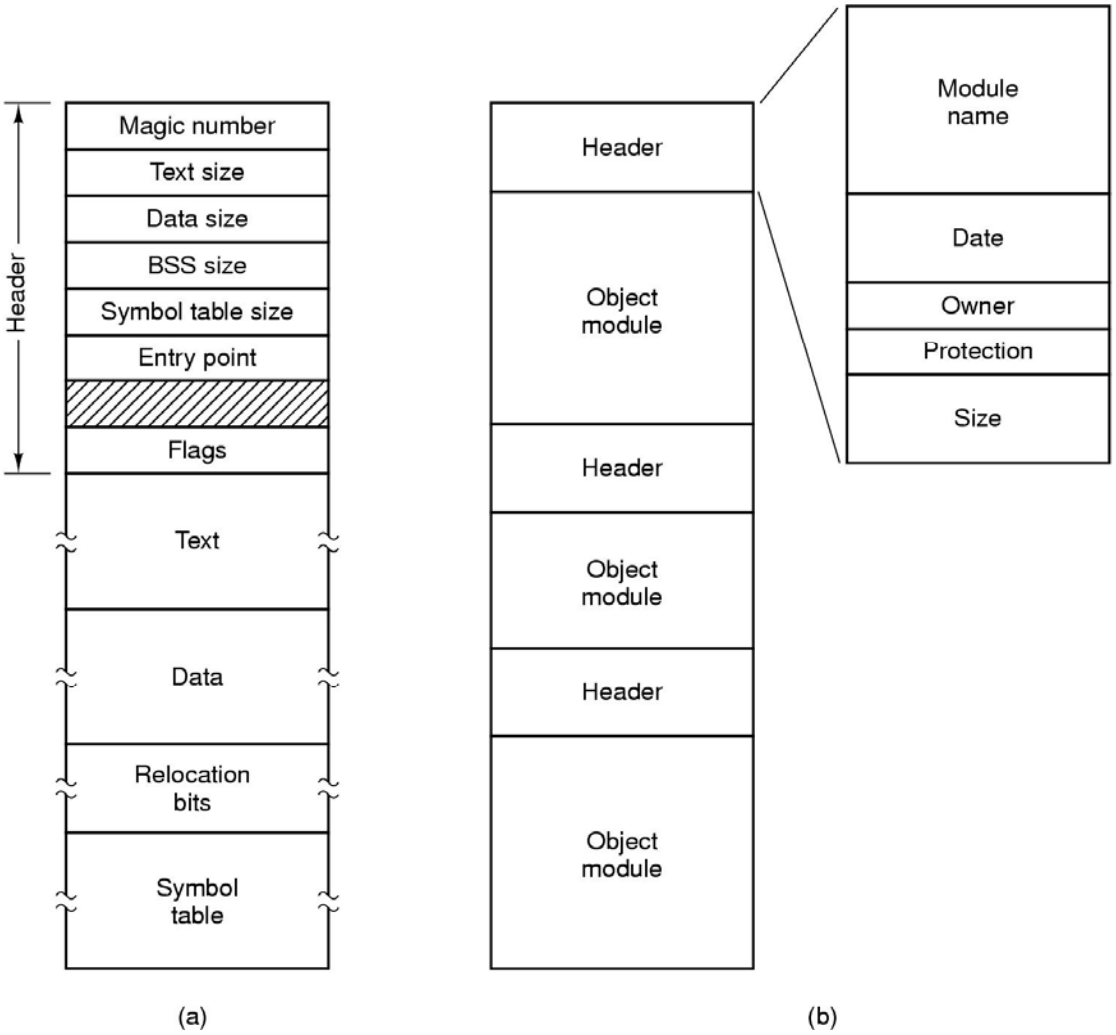

Figure 4-3. (a) An executable file. (b) An archive.

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 Operations

The most common system calls relating to files:

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

Example Program Using File System Calls (1)

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                  /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);       /* ANSI prototype */

#define BUF_SIZE 4096                   /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                /* protection bits for output file */

int main(int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit(1);             /* syntax error if argc is not 3 */

    /* Open the input file and create the output file */
    in_fd = open(argv[1], O_RDONLY);    /* open the source file */
    if (in_fd < 0) exit(2);             /* if it cannot be opened, exit */
    out_fd = creat(argv[2], OUTPUT_MODE); /* create the destination file */
    if (out_fd < 0) exit(3);            /* if it cannot be created, exit */

    . . .
```

Figure 4-5. A simple program to copy a file.

Example Program Using File System Calls (2)

```
        /* Copy loop */
        while (TRUE) {
                rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
        if (rd_count <= 0) break;                       /* if end of file or error, exit loop */
                wt_count = write(out_fd, buffer, rd_count); /* write data */
                if (wt_count <= 0) exit(4);             /* wt_count <= 0 is an error */
        }

        /* Close the files */
        close(in_fd);
        close(out_fd);
        if (rd_count == 0)                              /* no error on last read */
                exit(0);
        else
                exit(5);                                /* error on last read */
}
```

Figure 4-5. A simple program to copy a file.

Hierarchical Directory Systems (1)

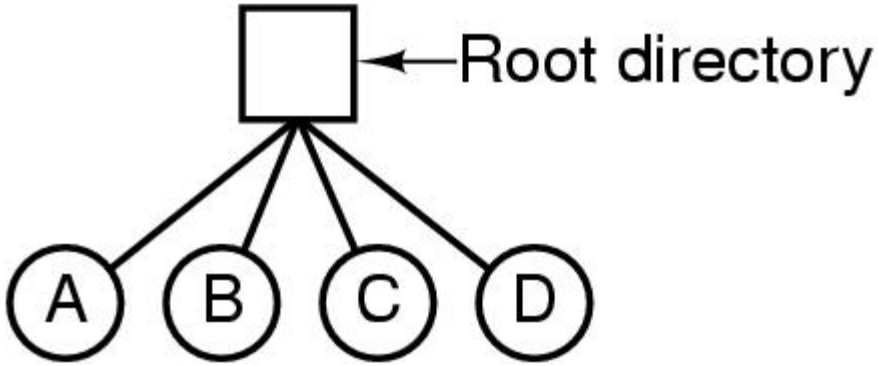

Figure 4-6. A single-level directory system containing four files.

Hierarchical Directory Systems (2)

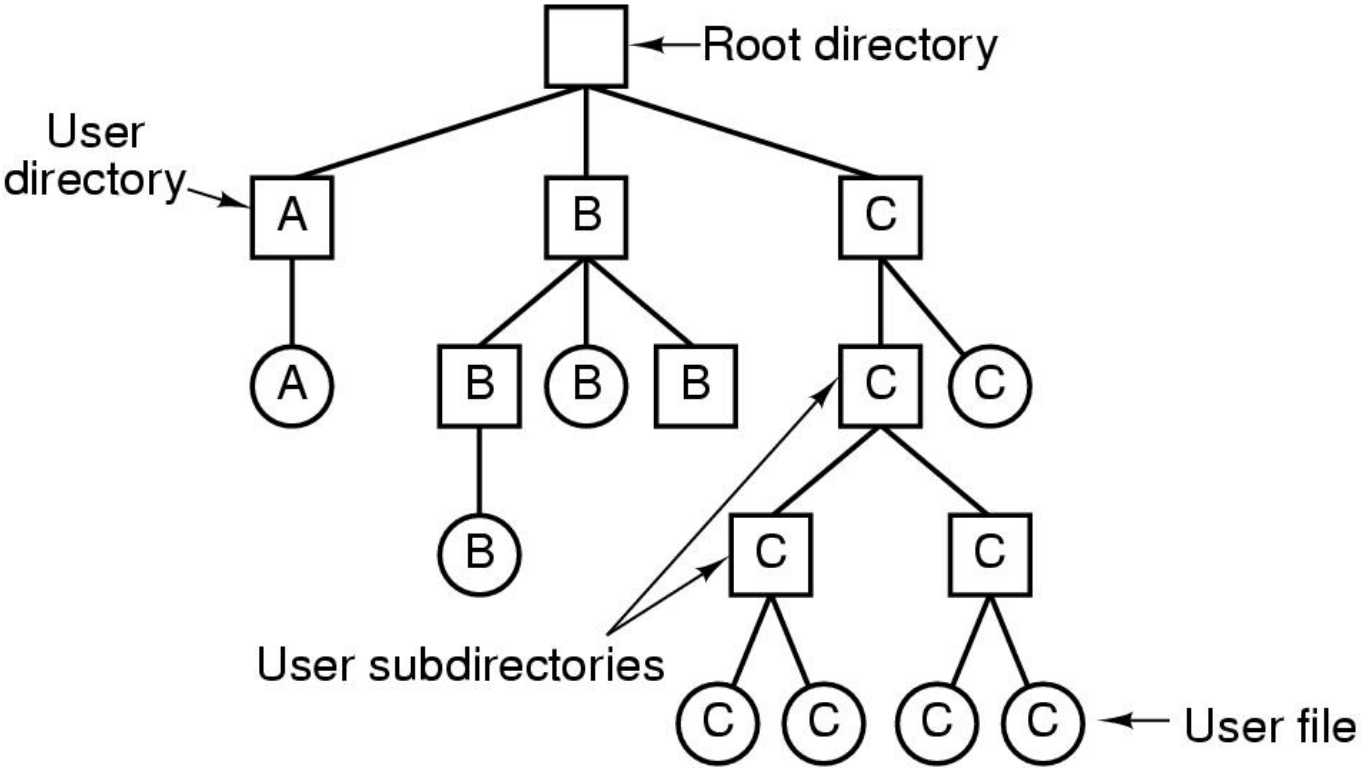

Figure 4-7. A hierarchical directory system.

Path Names

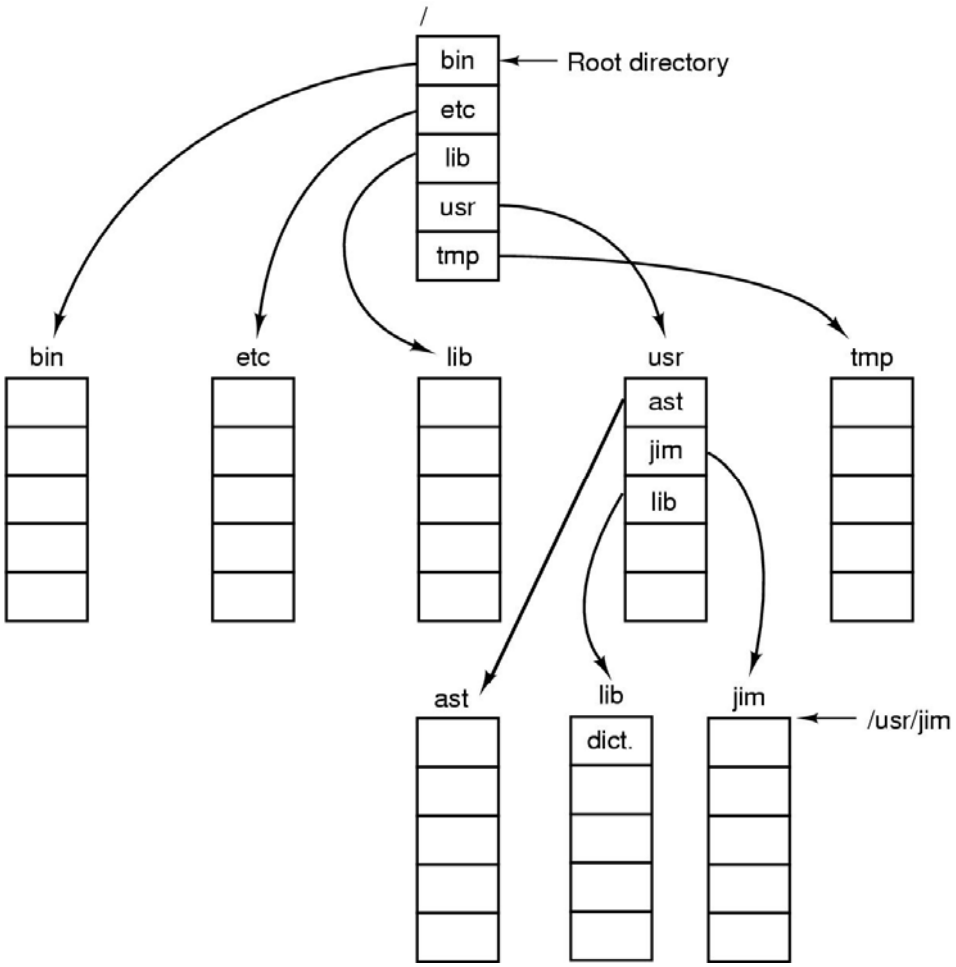

Figure 4-8. A UNIX directory tree.

Directory Operations

System calls for managing directories:

- Create
- Delete
- Opendir
- Closedir
- Readdir
- Rename
- Link
- Uplink

File System Layout

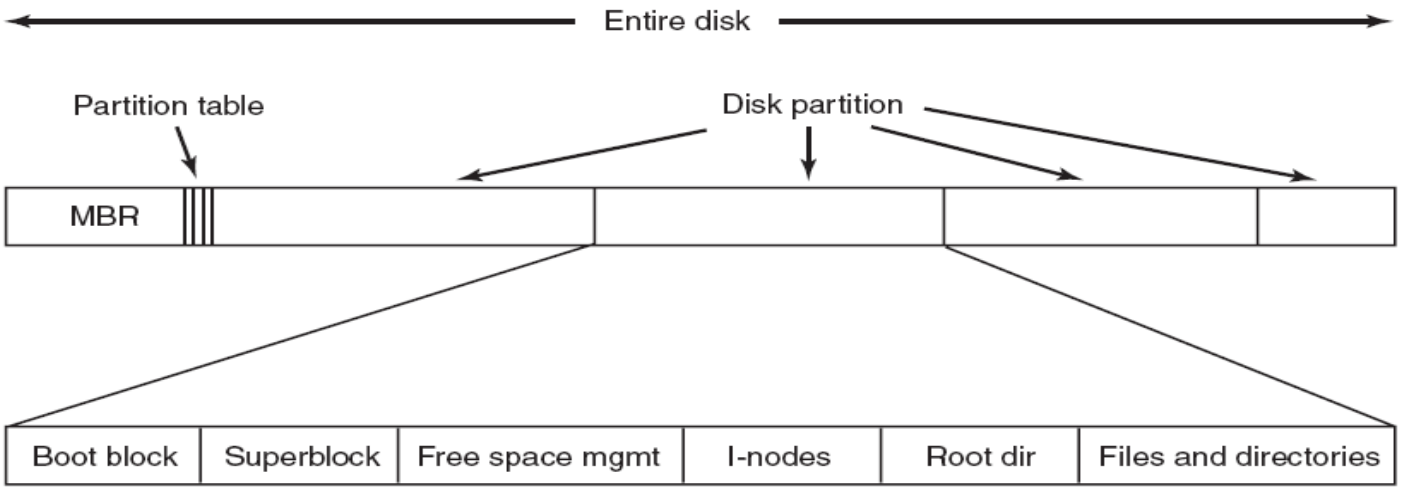

Figure 4-9. A possible file system layout.

Contiguous Allocation

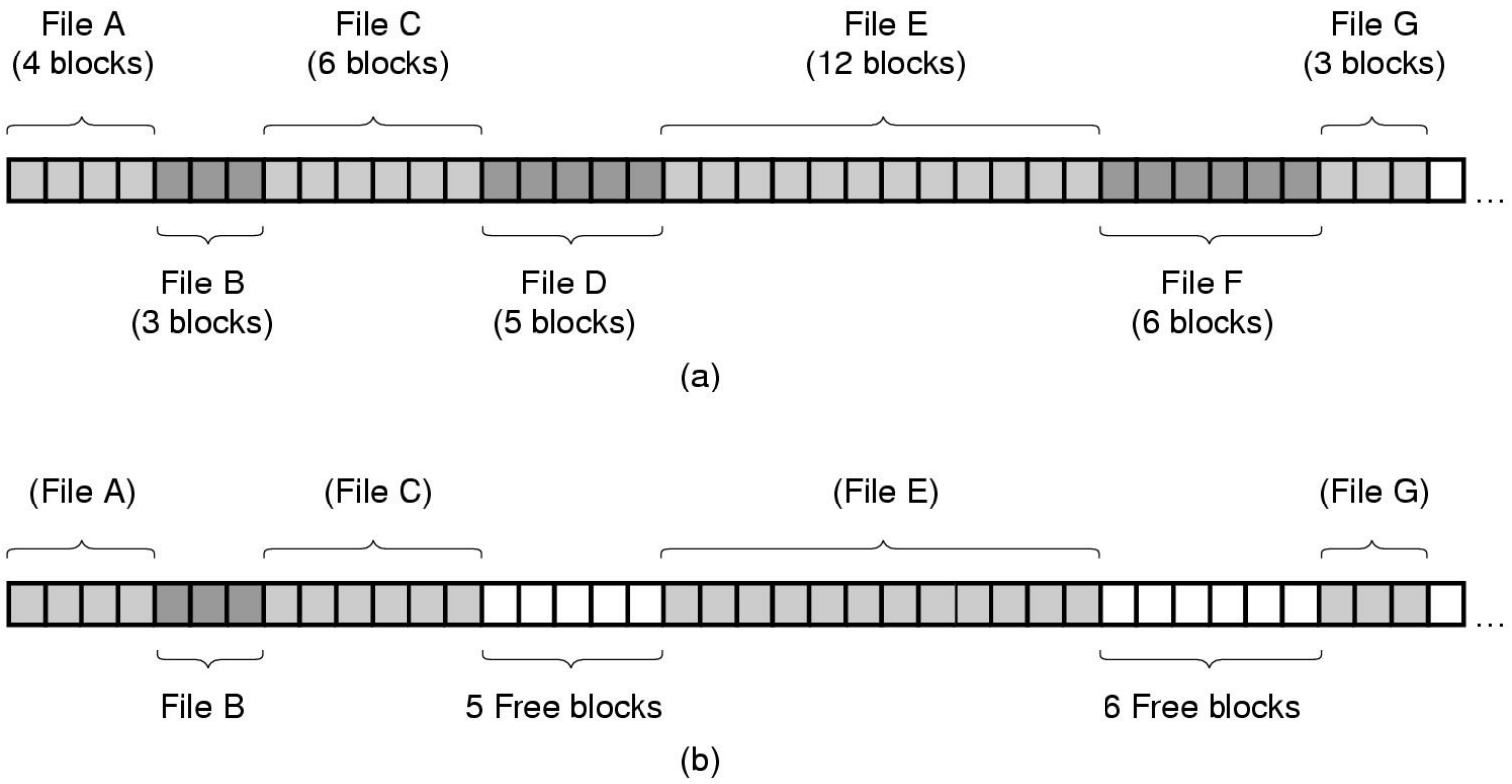

Figure 4-10. (a) Contiguous allocation of disk space for 7 files. (b) The state of the disk after files D and F have been removed.

Linked List Allocation

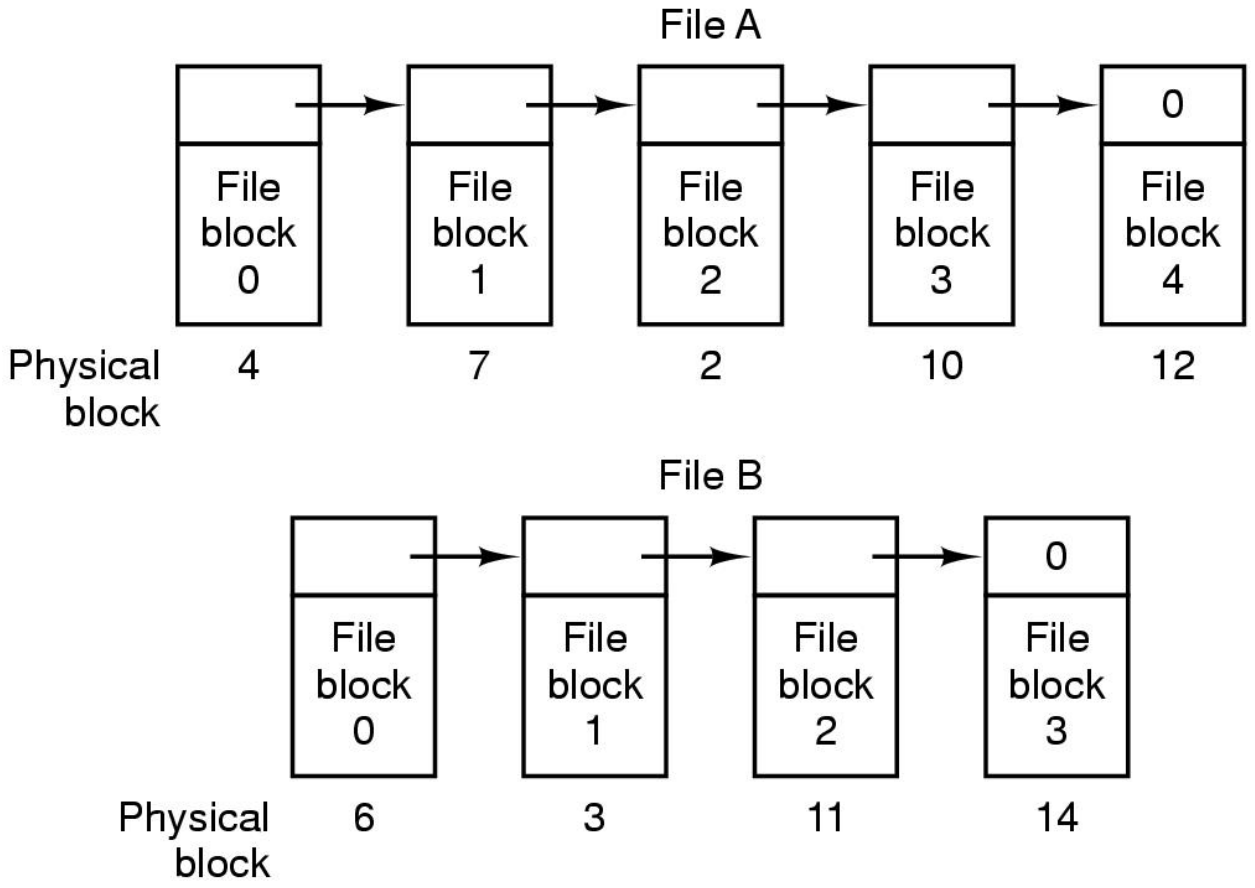

Figure 4-11. Storing a file as a linked list of disk blocks.

Linked List Allocation Using a Table in Memory

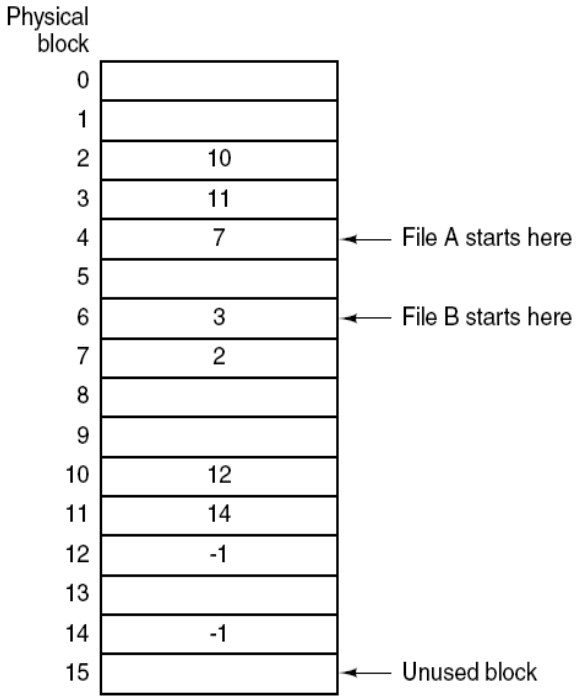

Figure 4-12. Linked list allocation using a file allocation table in main memory.

I-nodes

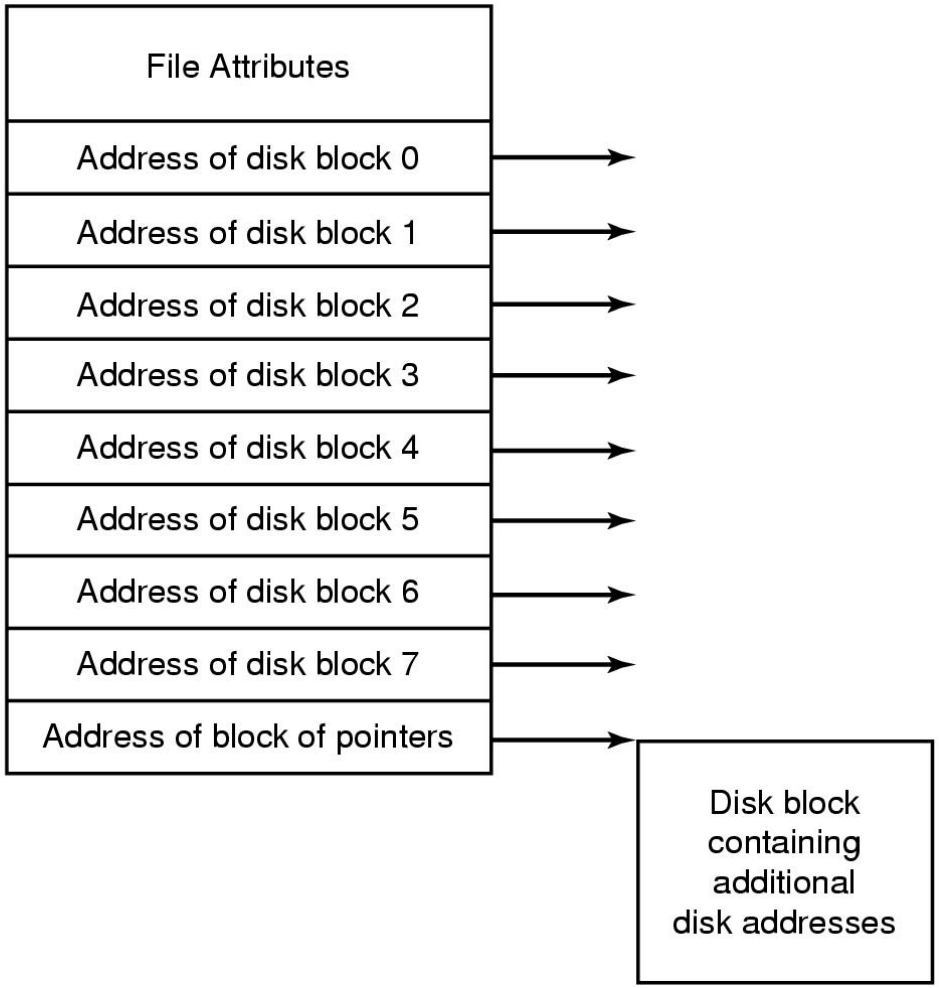

Figure 4-13. An example i-node.

Implementing Directories (1)

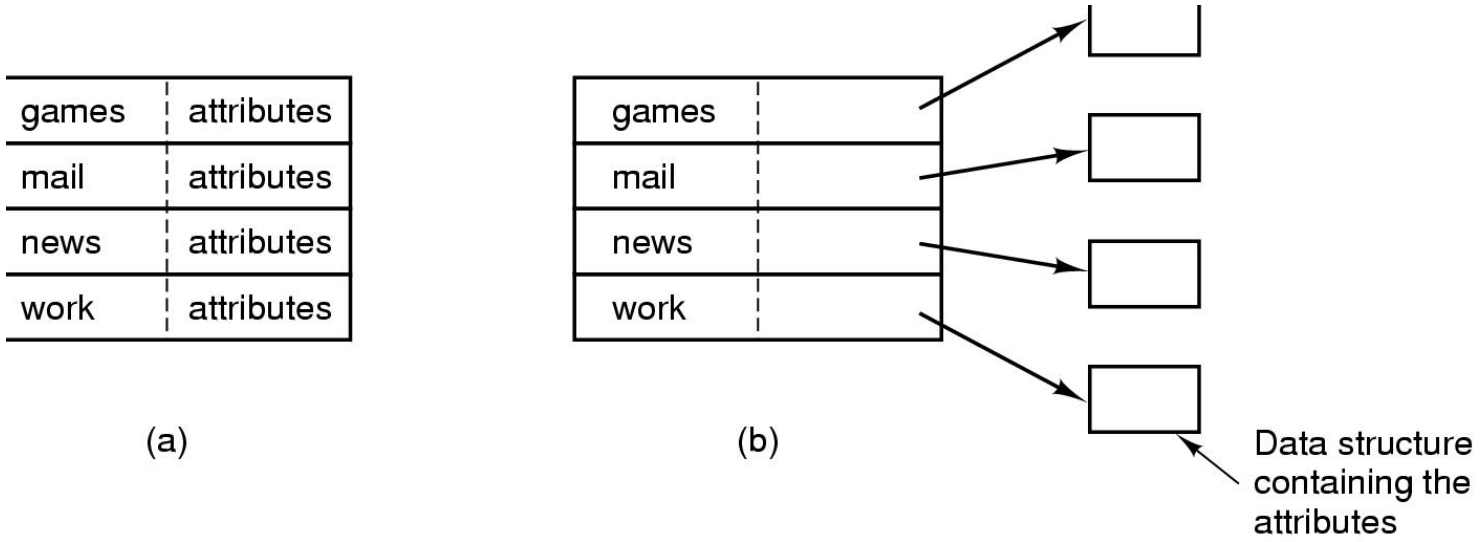

Figure 4-14. (a) A simple directory containing fixed-size entries with the disk addresses and attributes in the directory entry. (b) A directory in which each entry just refers to an i-node.

Implementing Directories (2)

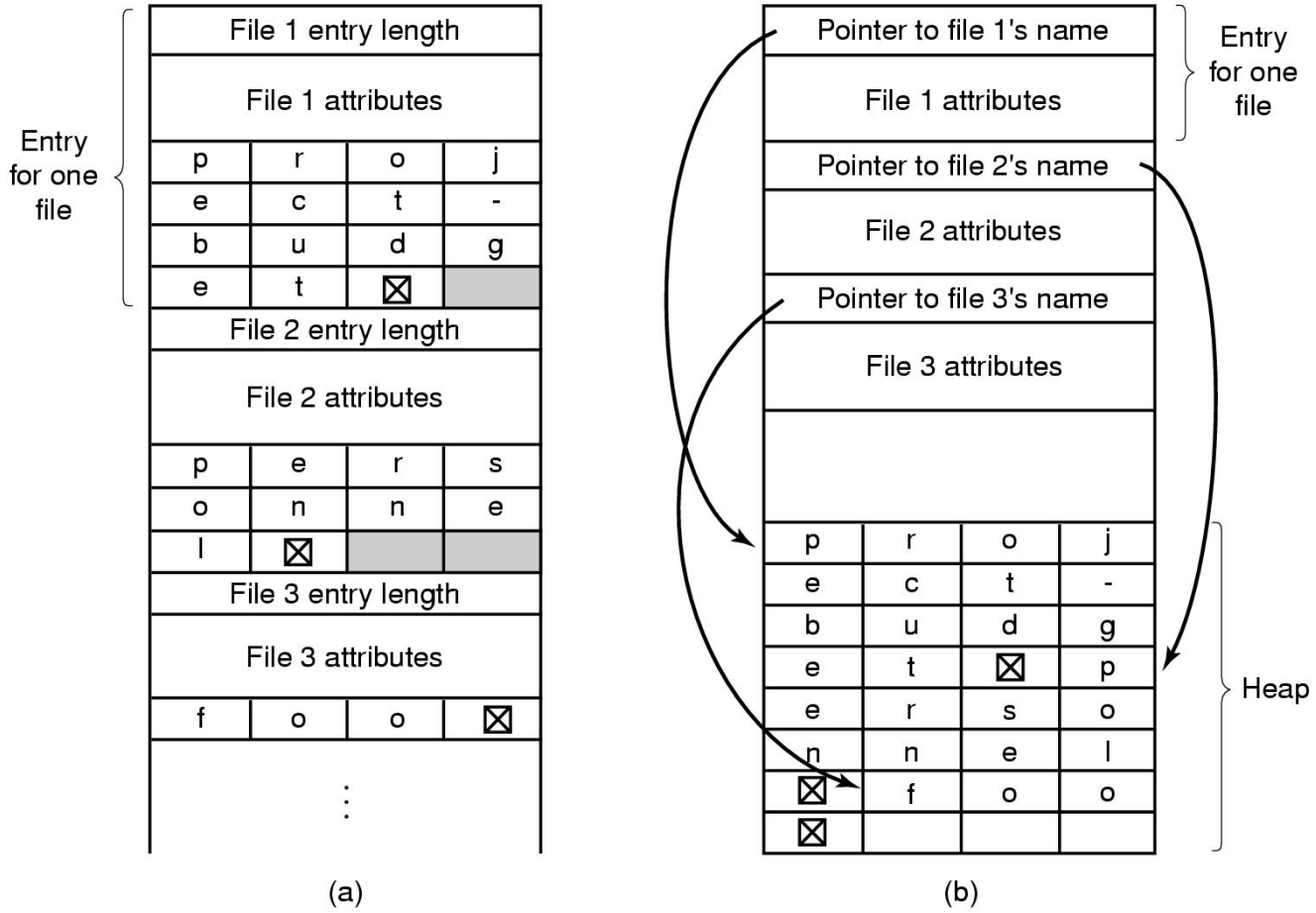

Figure 4-15. Two ways of handling long file names in a directory. (a) In-line. (b) In a heap.

Shared Files (1)

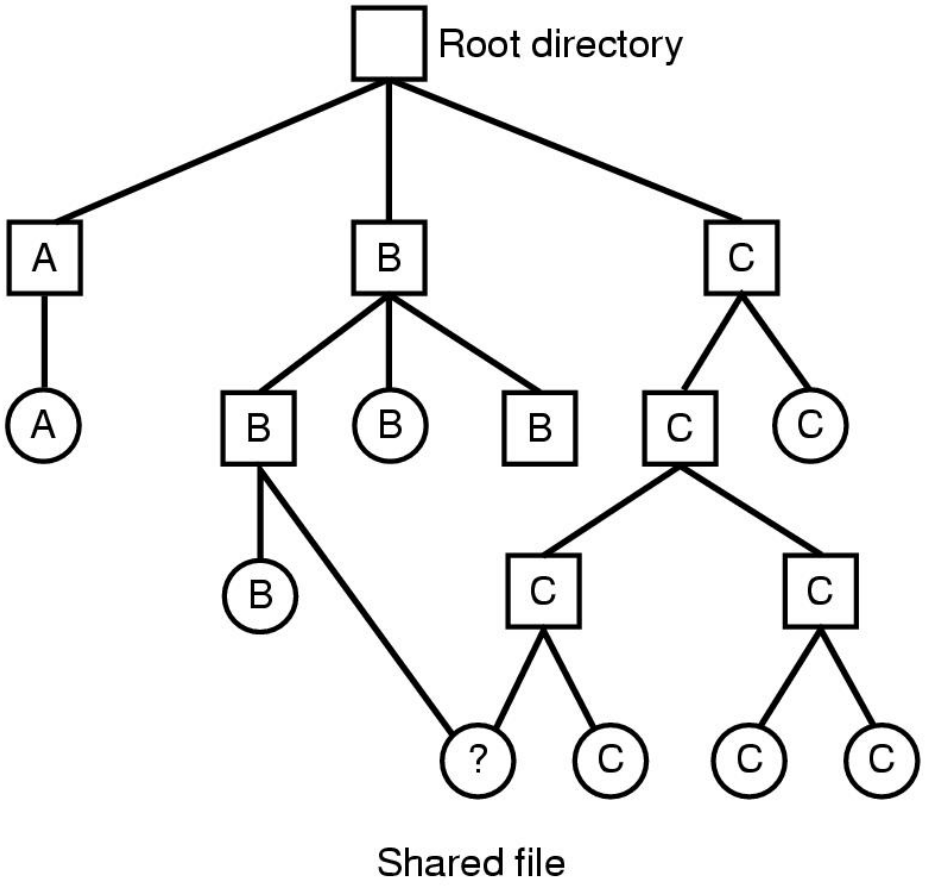

Figure 4-16. File system containing a shared file.

Shared Files (2)

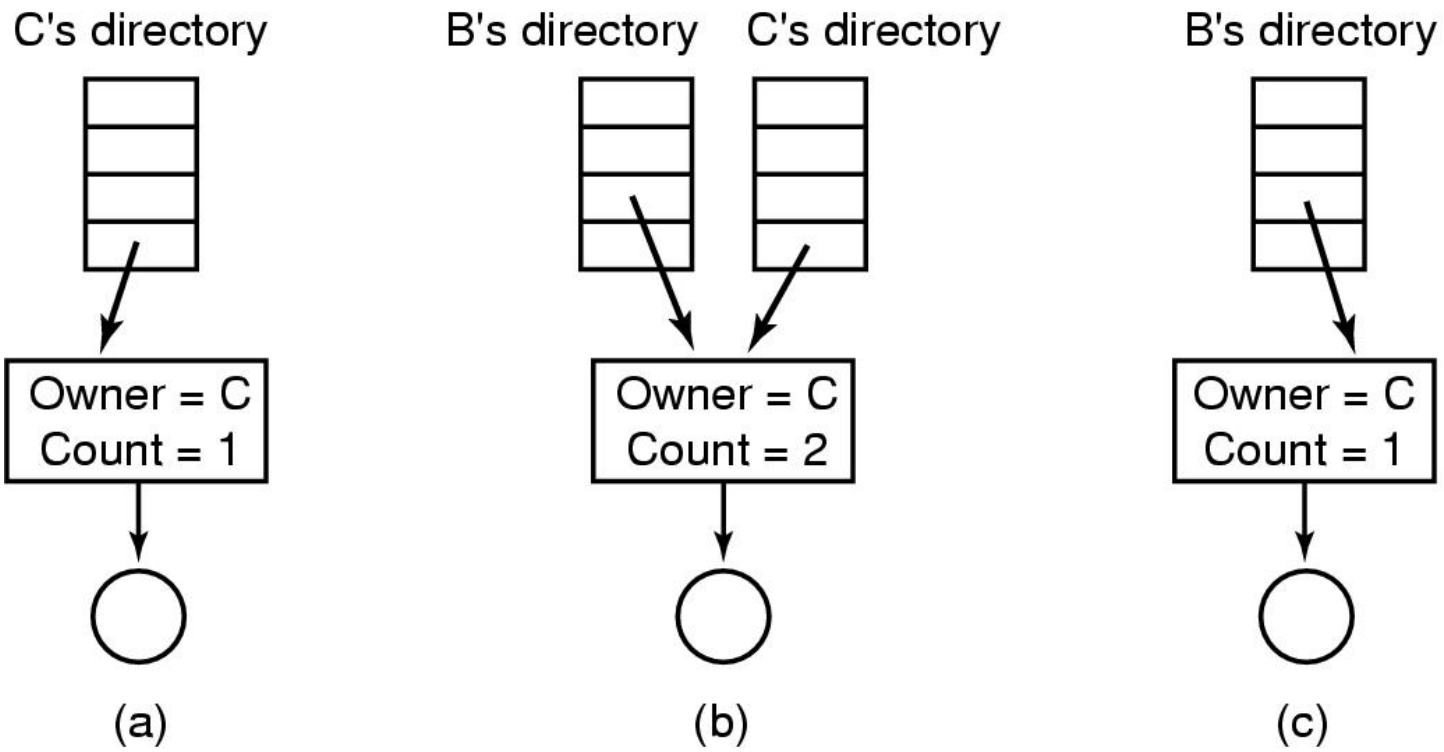

Figure 4-17. (a) Situation prior to linking. (b) After the link is created. (c) After the original owner removes the file.

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.

Virtual File Systems (1)

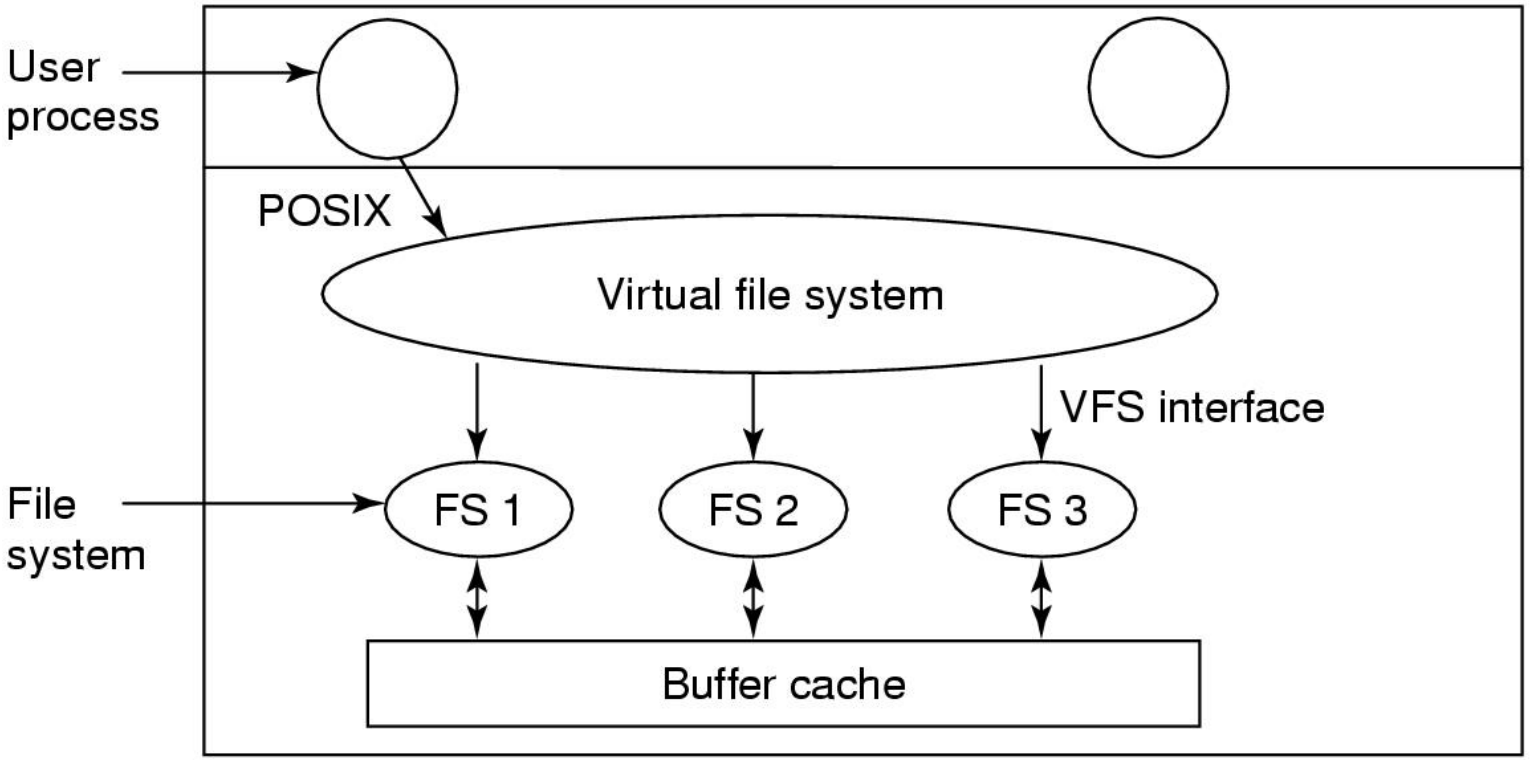

Figure 4-18. Position of the virtual file system.

Virtual File Systems (2)

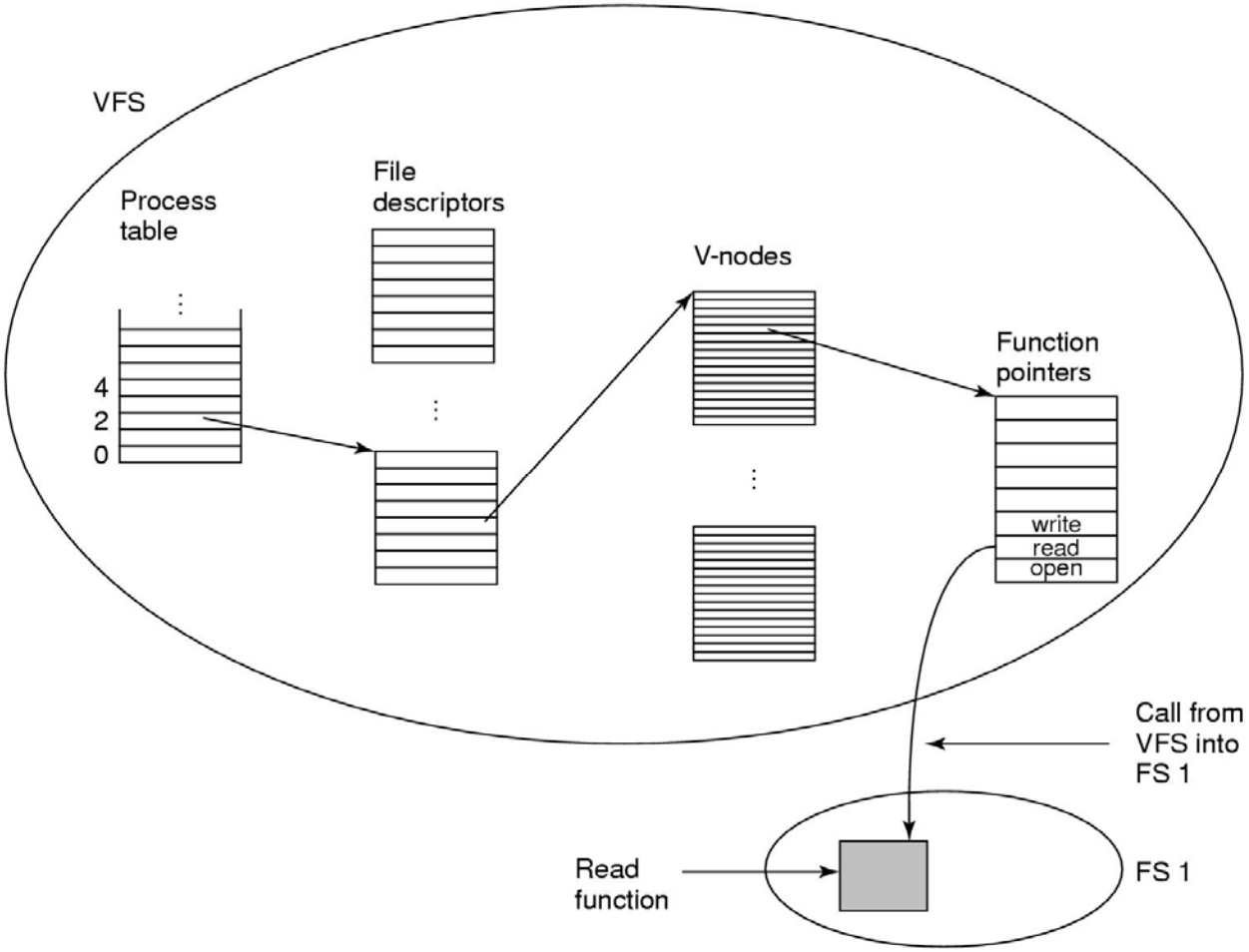

Figure 4-19. A simplified view of the data structures and code used by the VFS and concrete file system to do a read.

Disk Space Management Block Size (1)

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).

Disk Space Management Block Size (2)

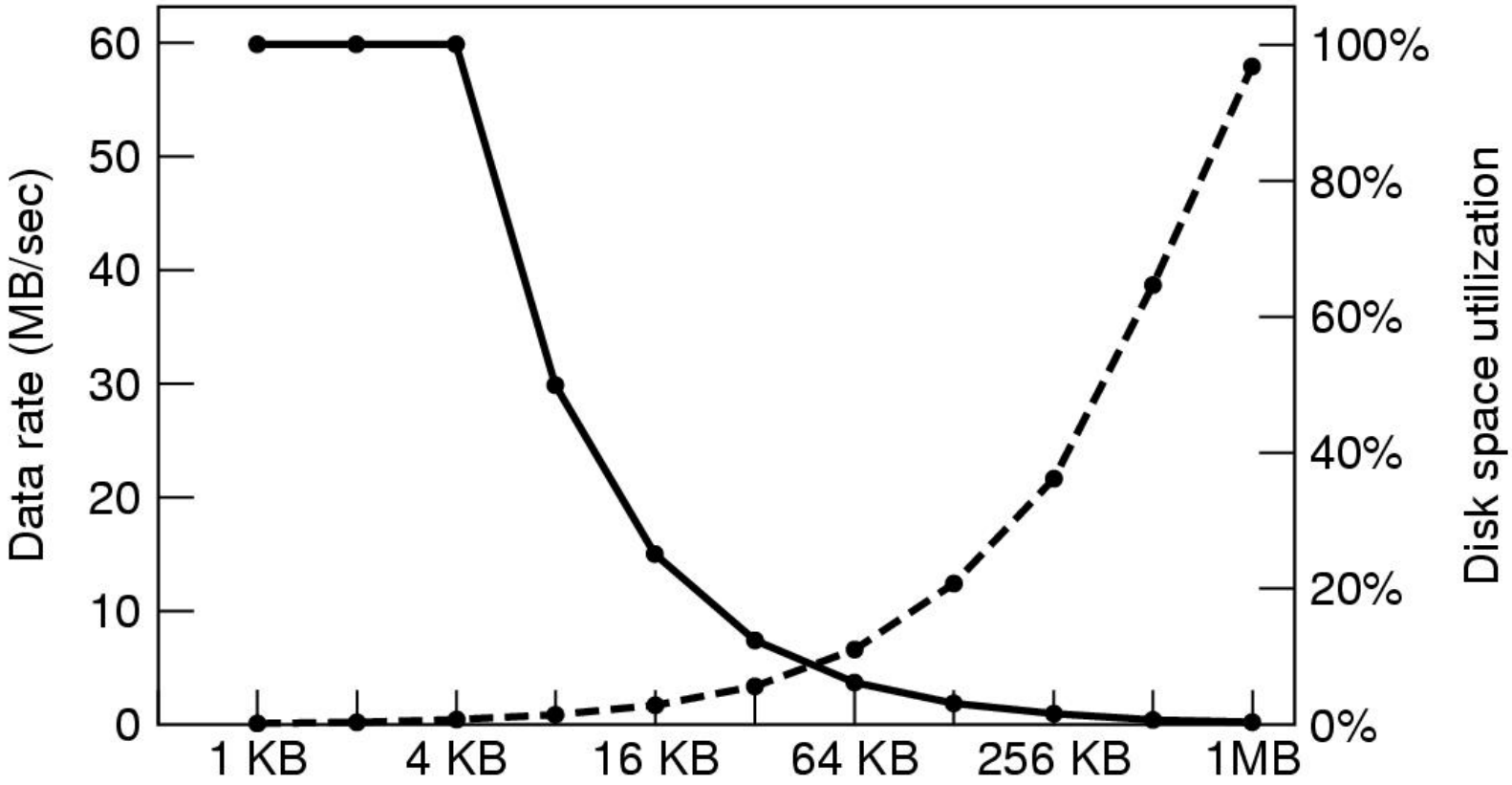

Figure 4-21. The solid curve (left-hand scale) gives the data rate of a disk. The dashed curve (right-hand scale) gives the disk space efficiency. All files are 4 KB.

Keeping Track of Free Blocks (1)

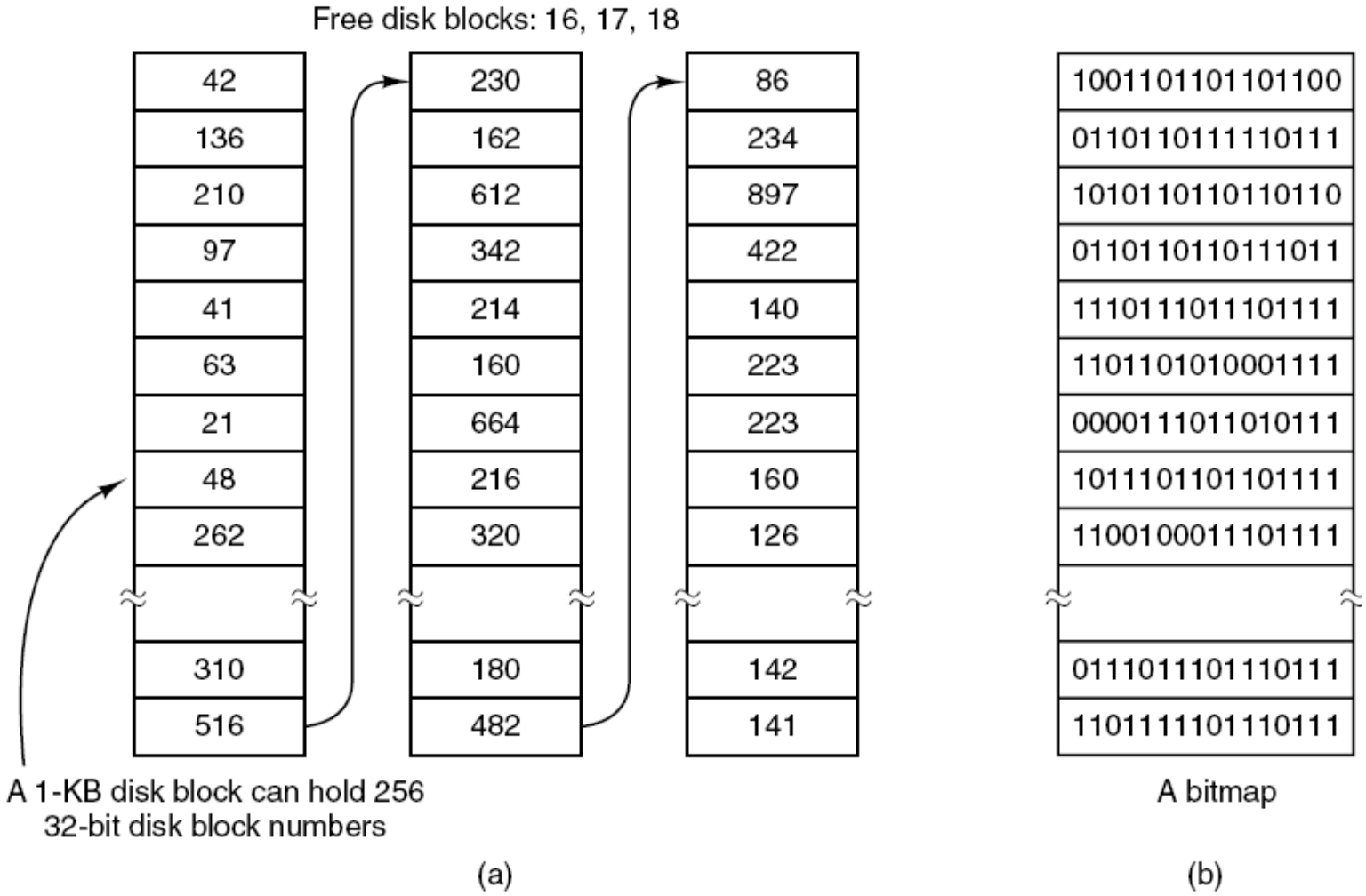

Figure 4-22. (a) Storing the free list on a linked list. (b) A bitmap.

Keeping Track of Free Blocks (2)

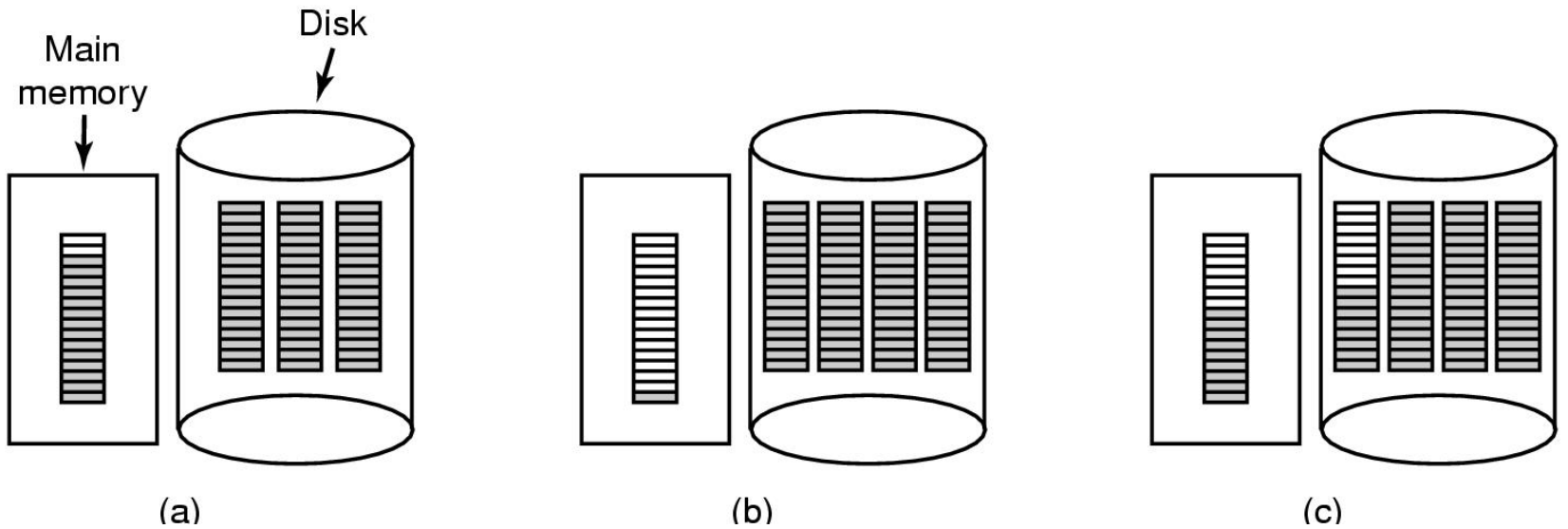

Figure 4-23. (a) An almost-full block of pointers to free disk blocks in memory and three blocks of pointers on disk. (b) Result of freeing a three-block file. (c) An alternative strategy for handling the three free blocks. The shaded entries represent pointers to free disk blocks.

Disk Quotas

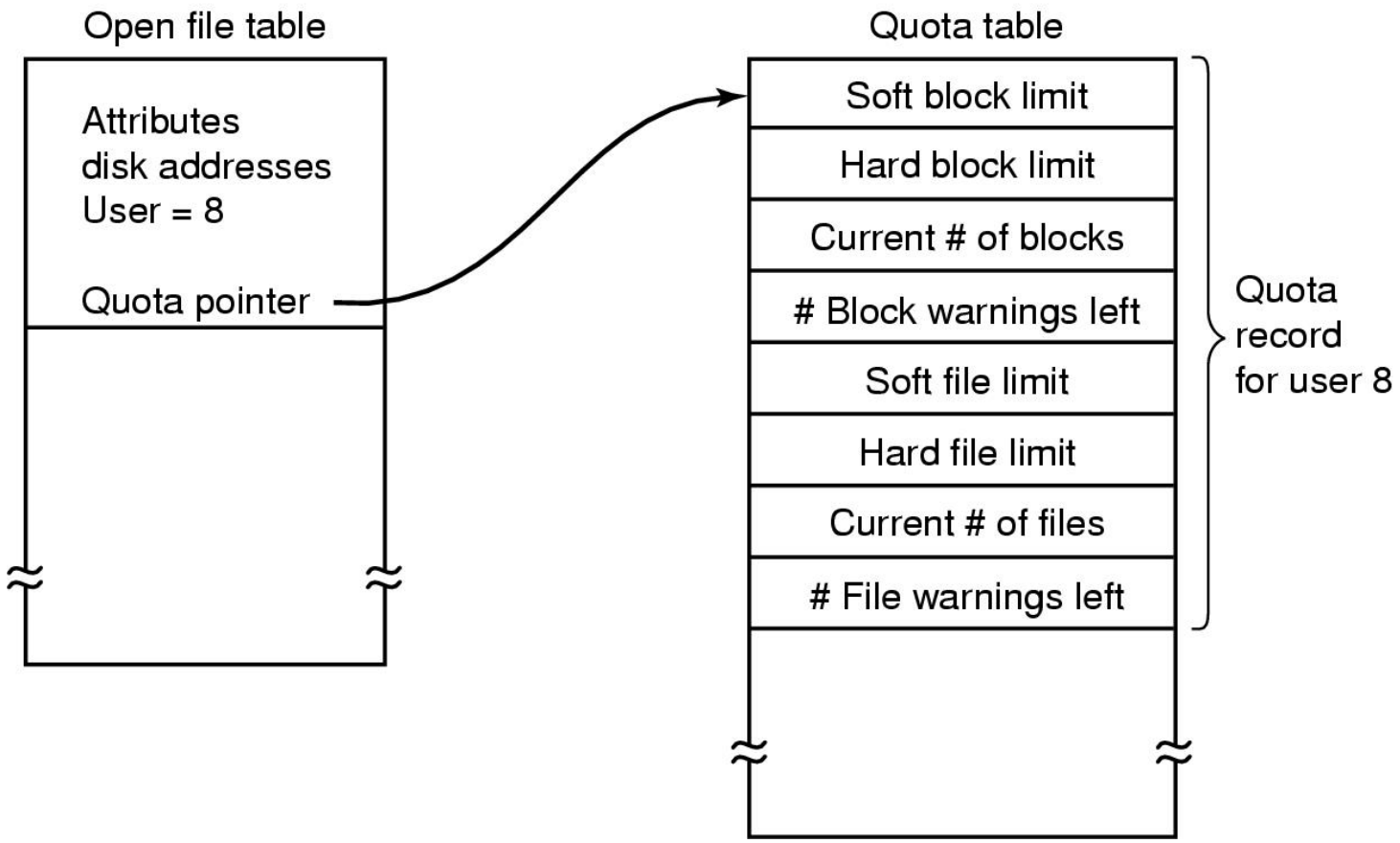

Figure 4-24. Quotas are kept track of on a per-user basis in a quota table.

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 (2)

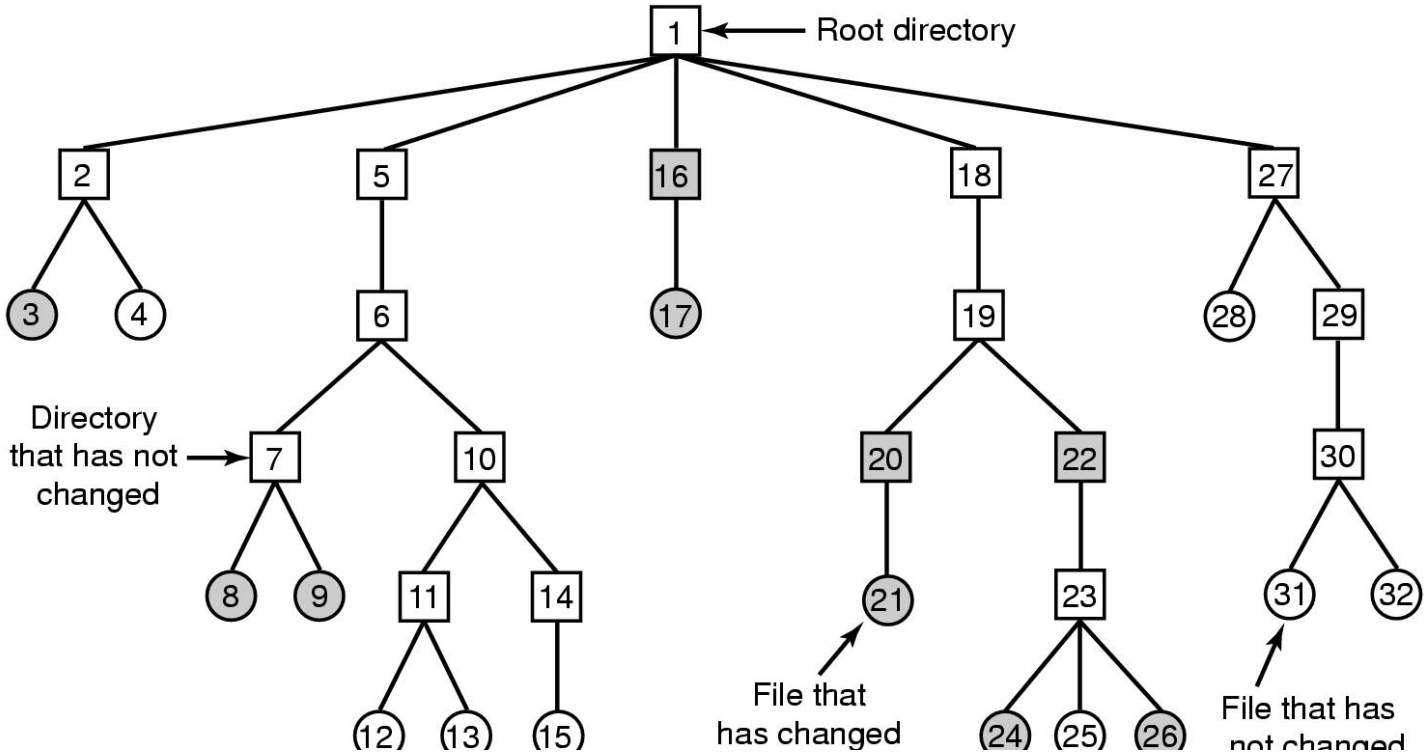

Figure 4-25. A file system to be dumped. Squares are directories, circles are files. Shaded items have been modified since last dump. Each directory and file is labeled by its i-node number.

File System Backups (3)

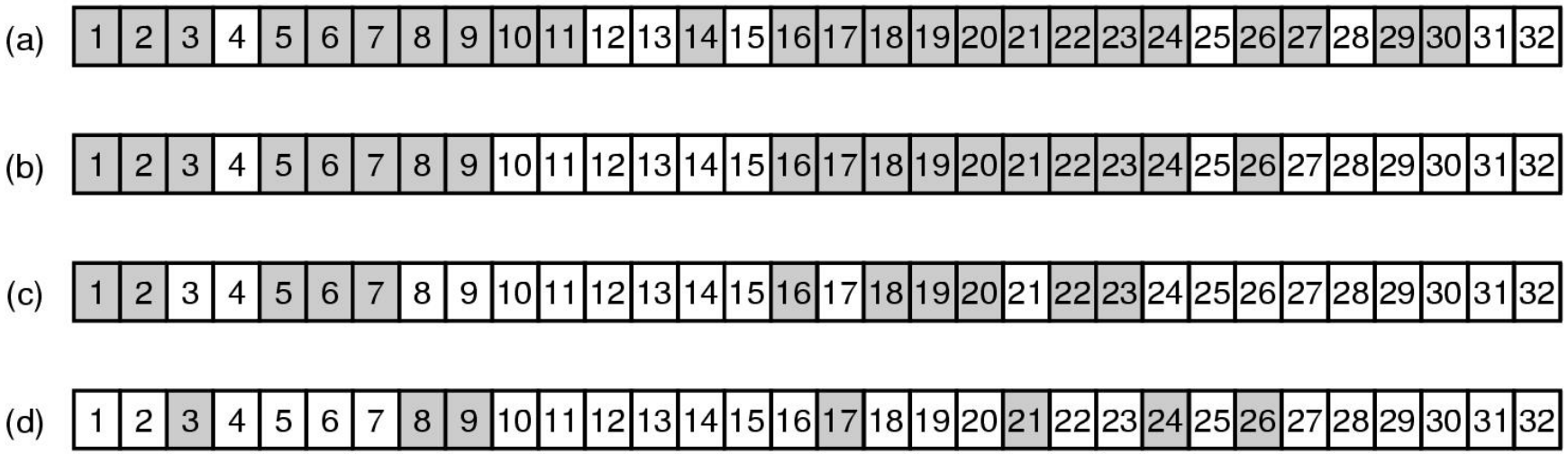

Figure 4-26. Bitmaps used by the logical dumping algorithm.

File System Consistency

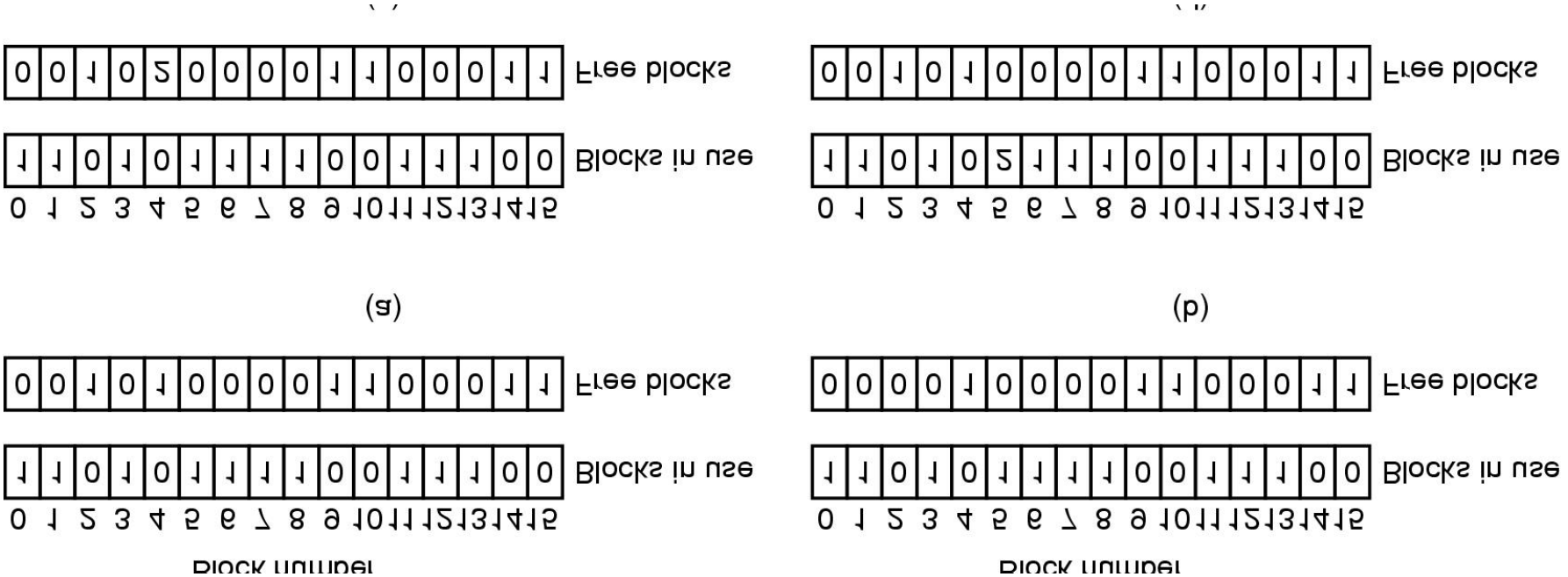

Figure 4-27. File system states. (a) Consistent. (b) Missing block. (c) Duplicate block in free list. (d) Duplicate data block.

Caching (1)

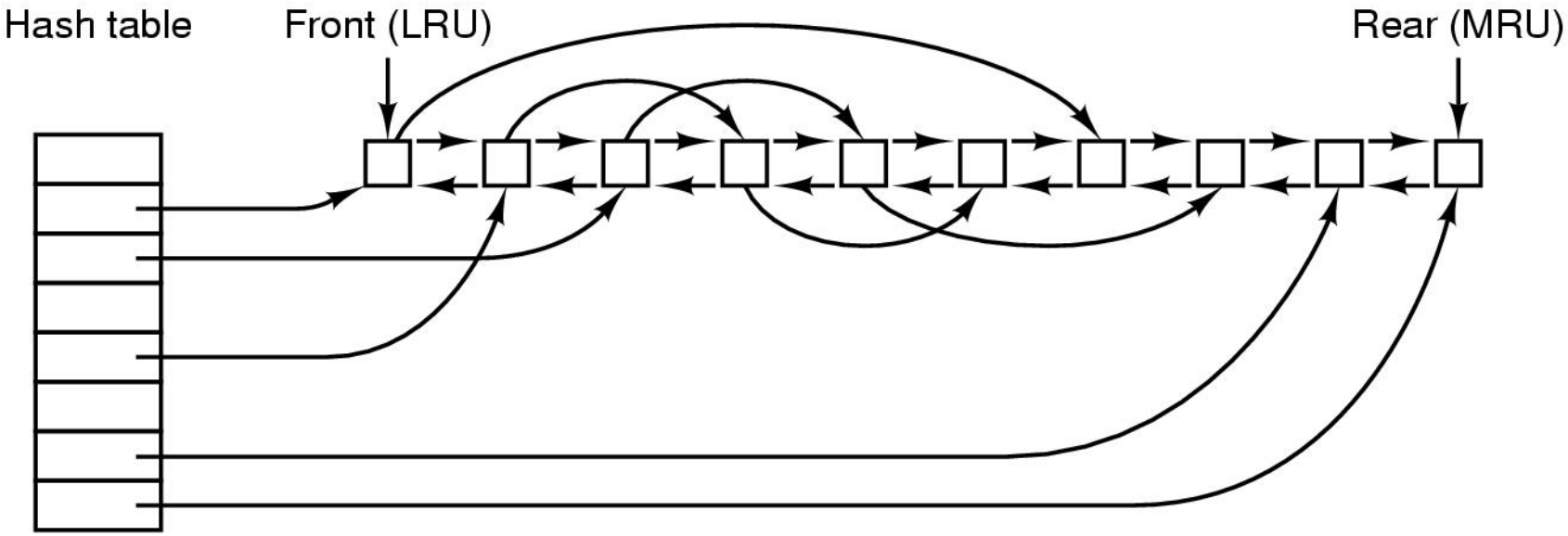

Figure 4-28. The buffer cache data structures.

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?

Reducing Disk Arm Motion

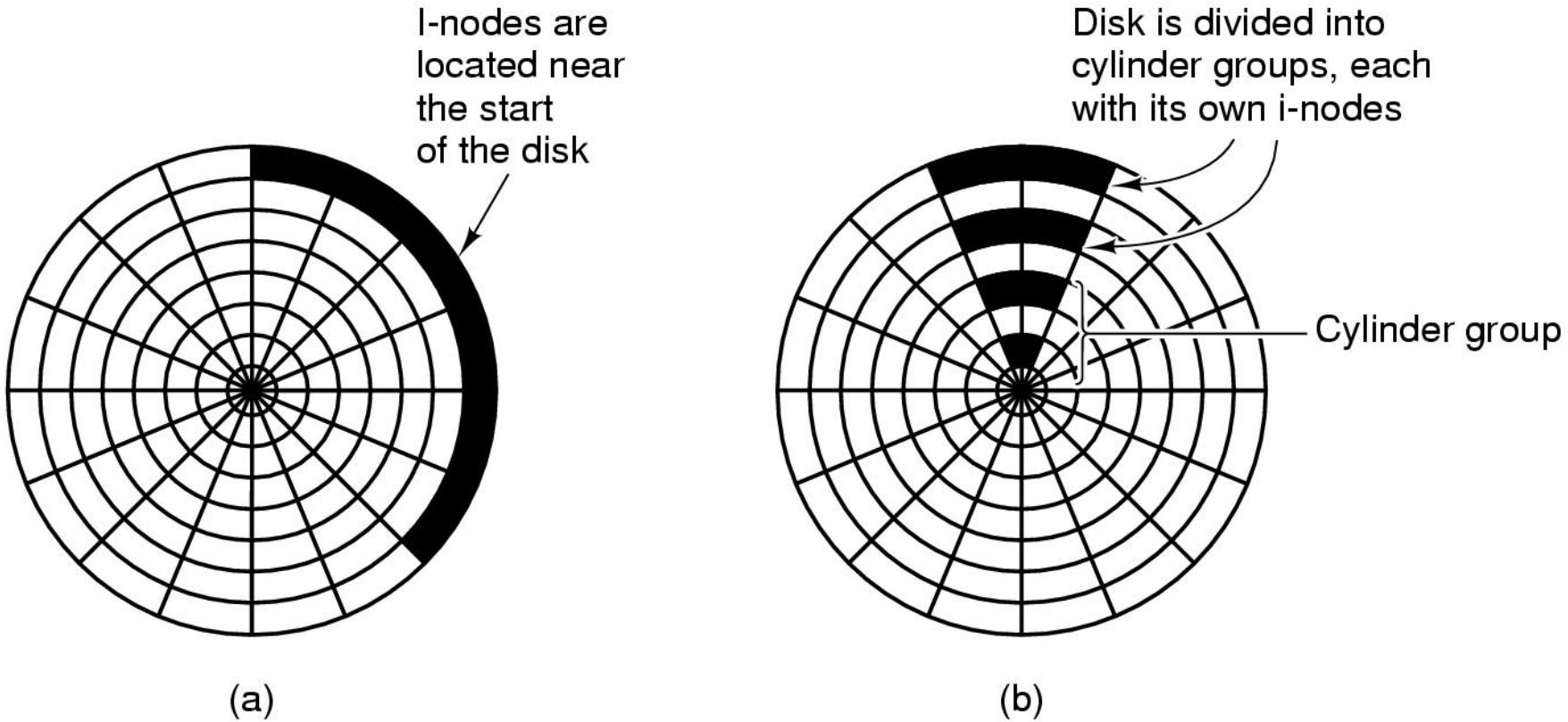

Figure 4-29. (a) I-nodes placed at the start of the disk. (b) Disk divided into cylinder groups, each with its own blocks and i-nodes.

The ISO 9660 File System

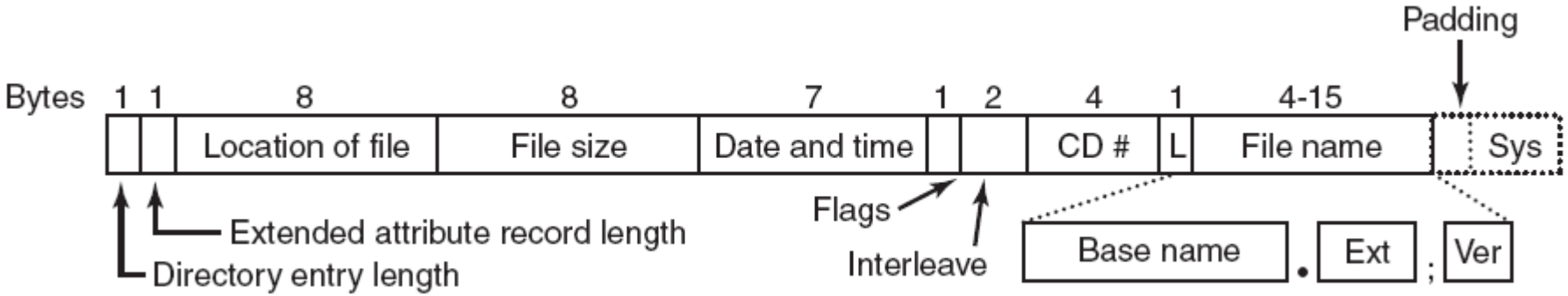

Figure 4-30. The ISO 9660 directory entry.

Rock Ridge Extensions

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.

Joliet Extensions

Joliet extension fields:

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

The MS-DOS File System (1)

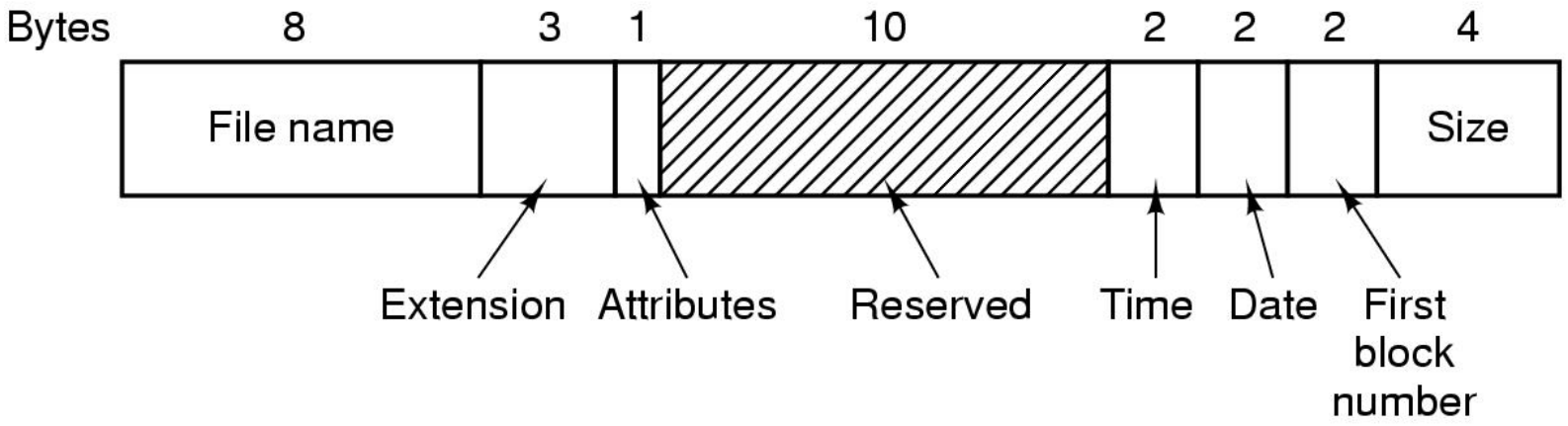

Figure 4-31. The MS-DOS directory entry.

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.

The UNIX V7 File System (1)

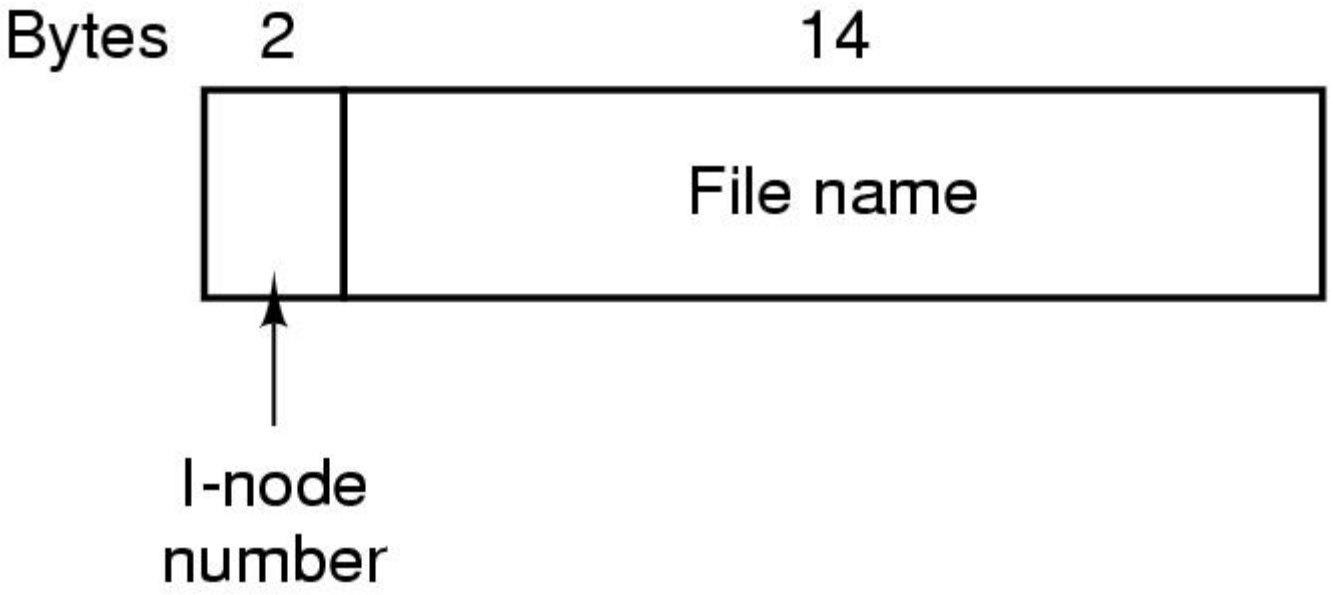

Figure 4-33. A UNIX V7 directory entry.

The UNIX V7 File System (2)

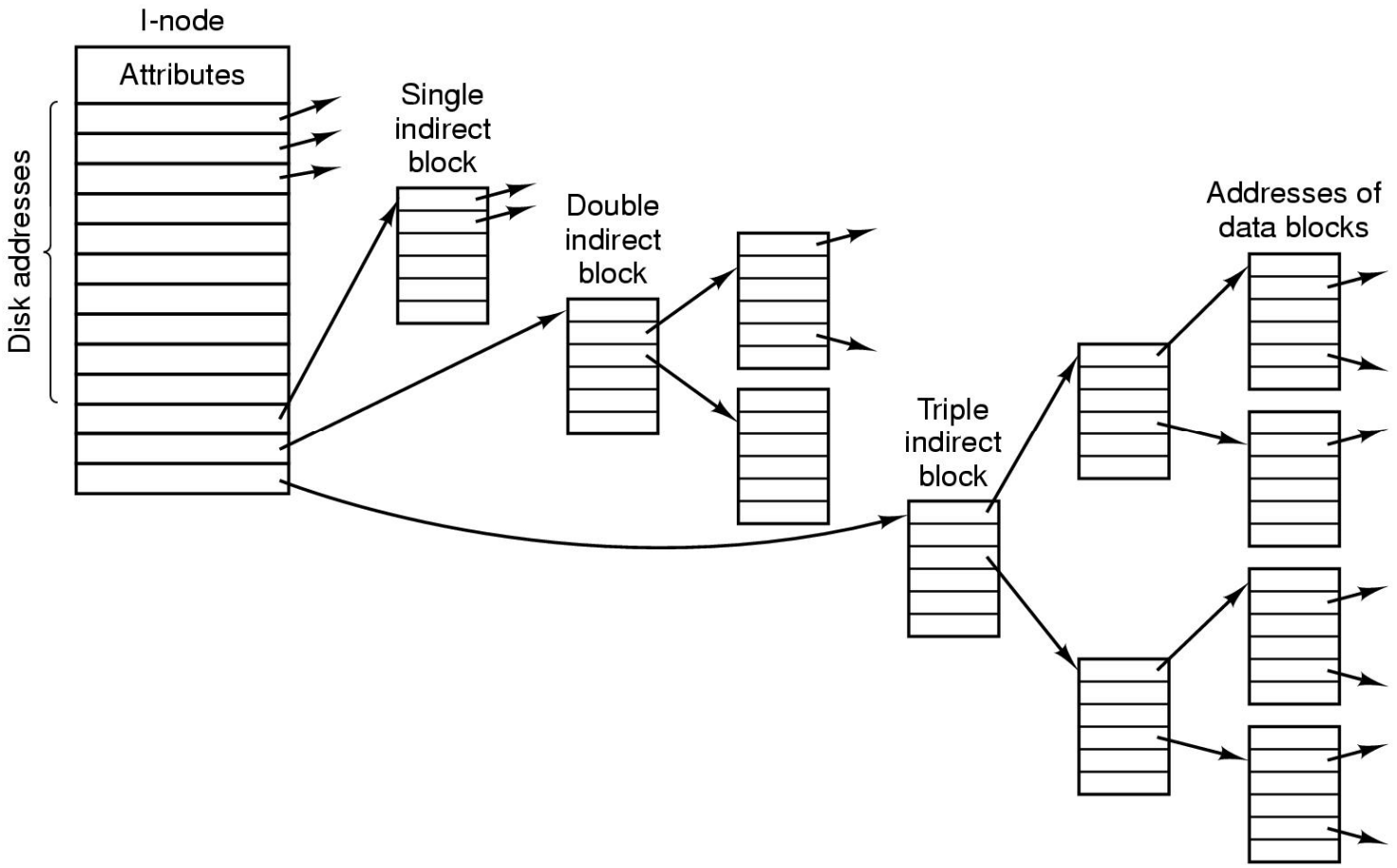

Figure 4-34. A UNIX i-node.

The UNIX V7 File System (3)

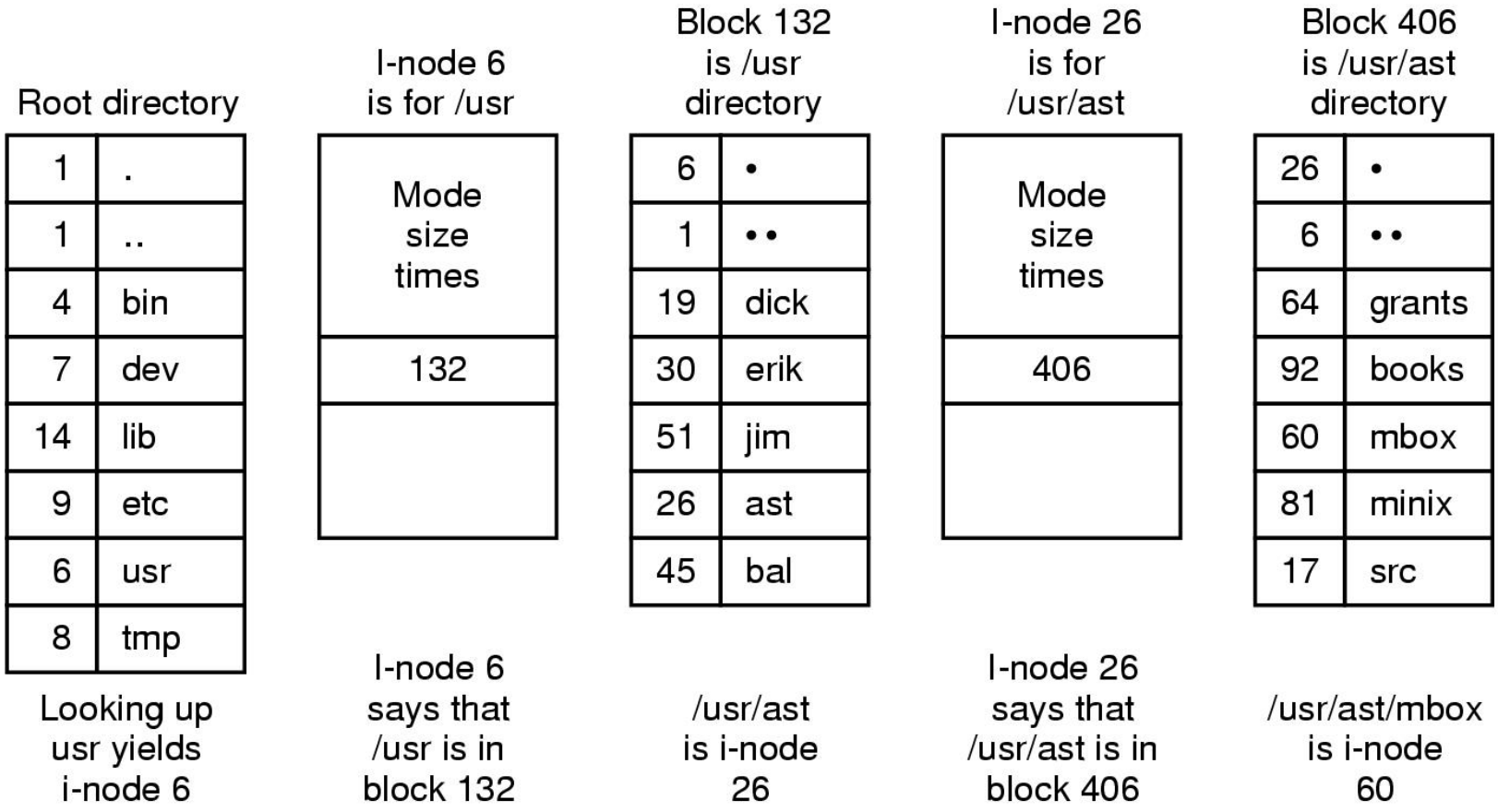

Figure 4-35. The steps in looking up */usr/ast/mbox*.