《 操 作 系 统 》

文件系统

首都师范大学
信息工程学院
霍其润

Long-term Information Storage

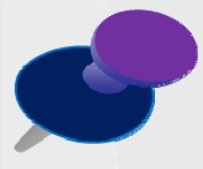
1. Must store large amounts of data
2. Information stored must survive after the termination of the process using it
3. Multiple processes must be able to access the information concurrently

Long-term Information Storage

- Files
- Managed by OS——File system

FILE SYSTEM

- FILES
- DIRECTORIES
- FILE SYSTEM IMPLEMENTATION
- EXAMPLE FILE SYSTEMS



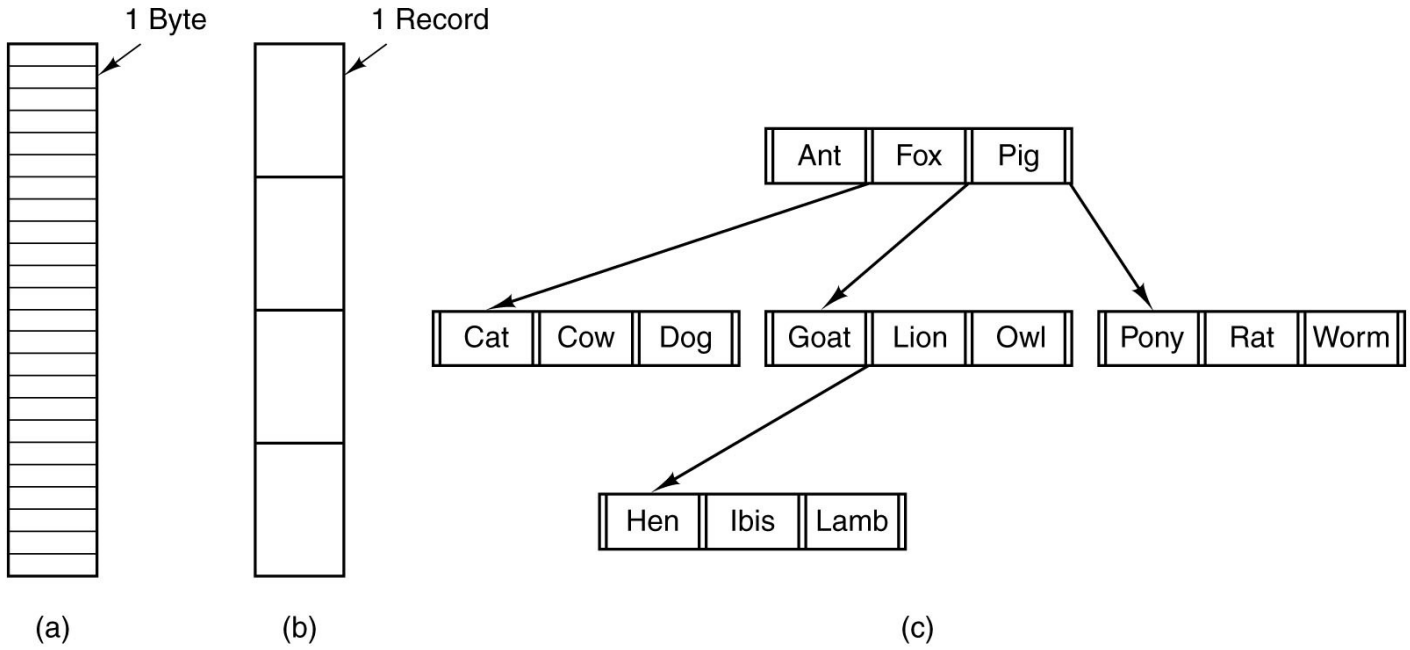
一 文件（FILES）

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
- The difference of UNIX

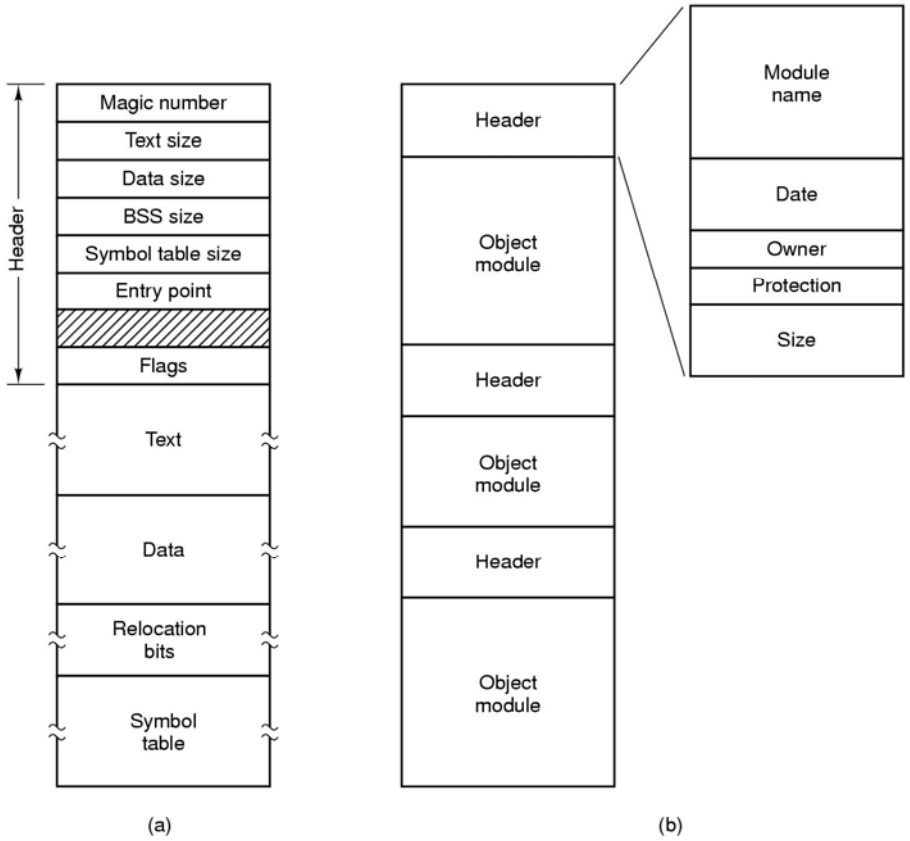
File Structure



- Three kinds of files
  - byte sequence； record sequence； tree

File Types

- regular file
  - ASCII file
  - binary file
- dirctory
- special file
  - character special file
  - block special file



(a) An executable file (b) An archive

File Access

- Sequential access
  - read all bytes/records from the beginning
  - cannot jump around, could rewind or back up
  - convenient when medium was magnetic tape
- Random access
  - bytes/records read in any order
  - essential for data base systems

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

File Attributes

- Base information
  - Name
  - type
- Space information
  - Length
- Access & control information
  - Owner
  - Authority
- Using information
  - Date & time

File Operations

We use these functions by system call always.

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

            . . .
```

A simple program to copy a file (copyfile abc xyz)

Example Program Using File System Calls (2)

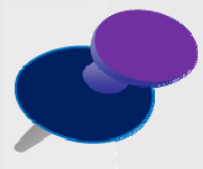
```
        /* Copy loop */
        while (TRUE) {
                rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
        if (rd_count <= 0) break;                         /* if end of file or error, exit loop */
                wt_count = write(out_fd, buffer, rd_count); /* write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

A simple program to copy a file (copyfile abc xyz)

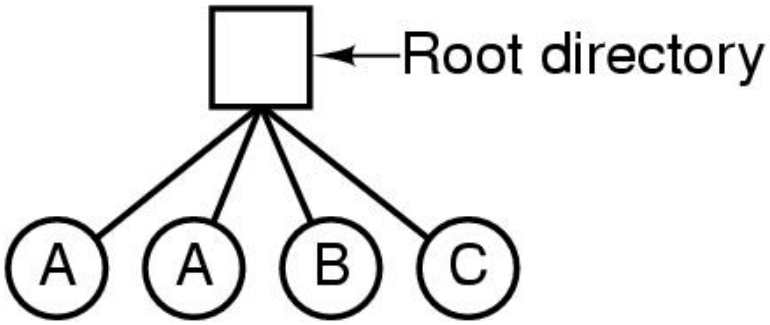
小 结

- File Naming
- File Structure
- File Types
- File Access
- File Attributes
- File Operations



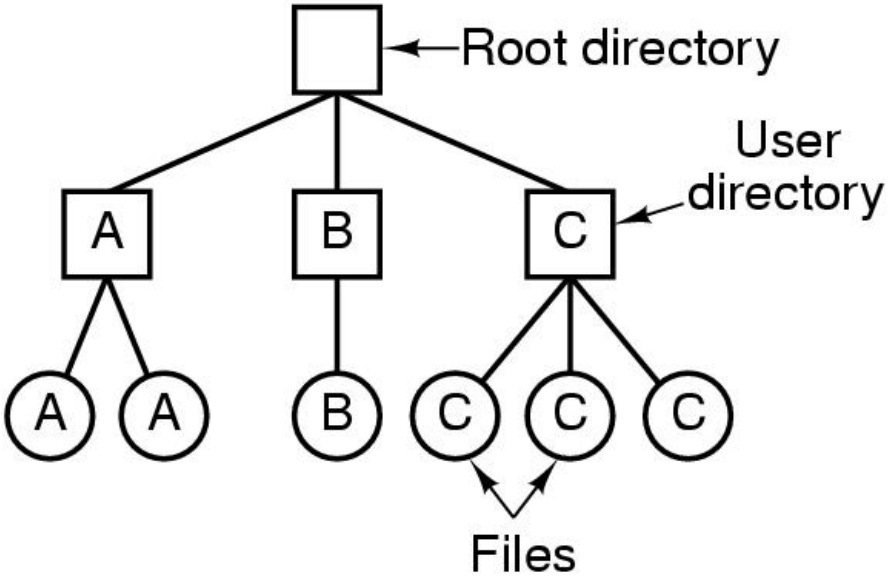
二 目录（DIRECTORIES）

Single-Level Directory Systems



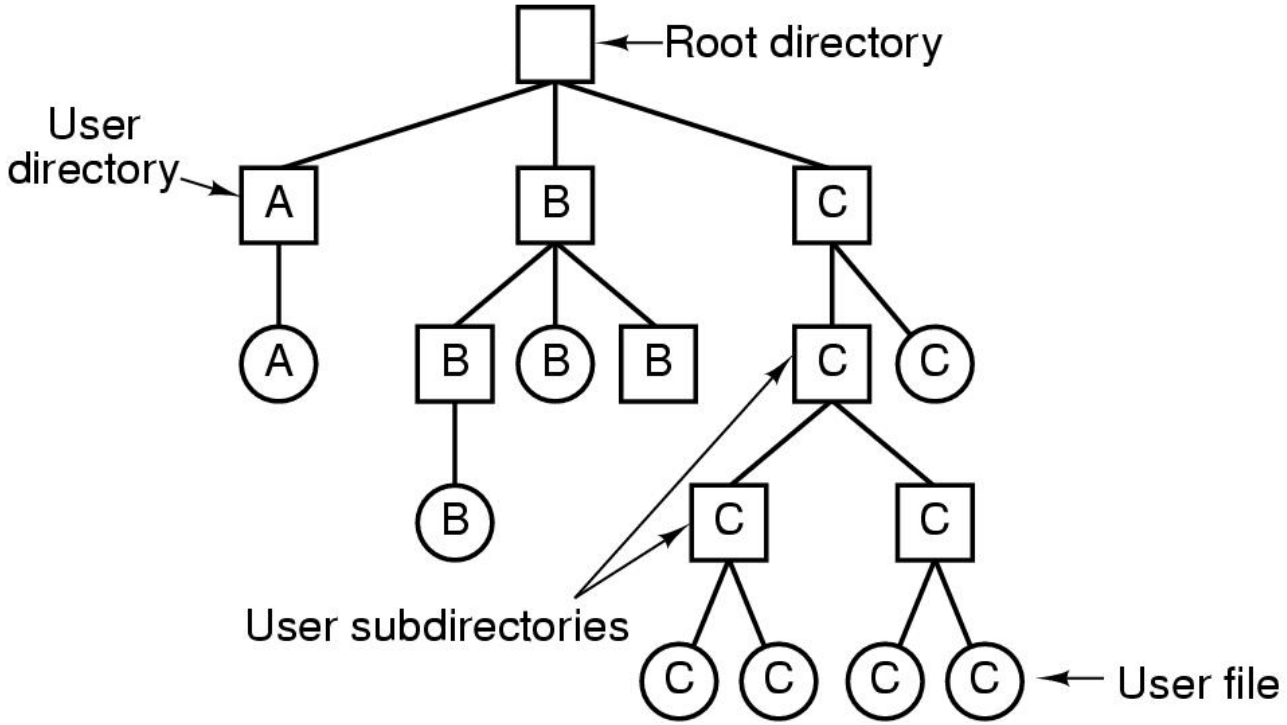
- A single level directory system
  - contains 4 files
  - owned by 3 different people, A, B, and C

Two-level Directory Systems



- Letters indicate *owners* of the directories and files

Hierarchical Directory Systems



- A hierarchical directory system

Path Names

/usr/lib/dictionary

cp /usr/lib/dictionary /usr/lib/dictionary.bak

当前目录为 /usr/lib

cp dictionary dictionary.bak

“.”：当前目录

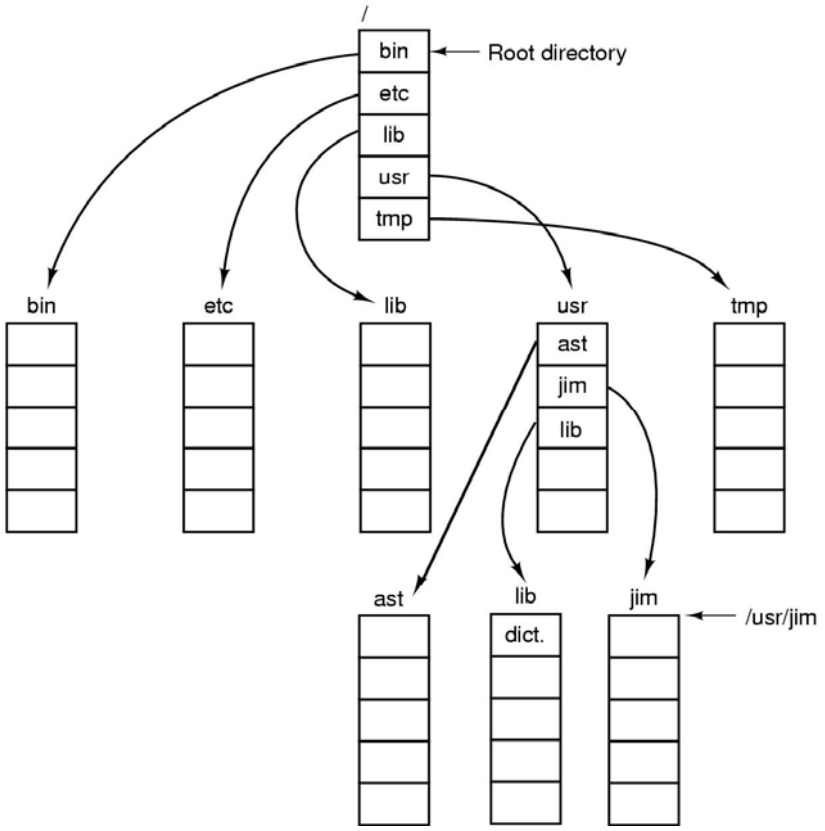
“..”：父目录

当前目录为 /usr/jim

cp ../lib/dictionary .

cp /usr/lib/dictionary dictionary

cp /usr/lib/dictionary /usr/jim/dictionary



- A UNIX directory tree

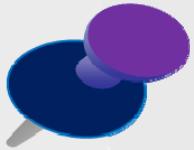
Directory Operations

We use these functions by system call always.

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

小 结

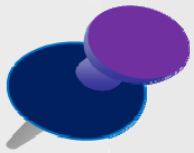
- Single-Level Directory Systems
- Two-Level Directory Systems
- Hierarchical Directory Systems
- Path Names
- Directory Operations



三 文件系统的实现
（FILE SYSTEM IMPLEMENTATION）

FILE SYSTEM IMPLEMENTATION

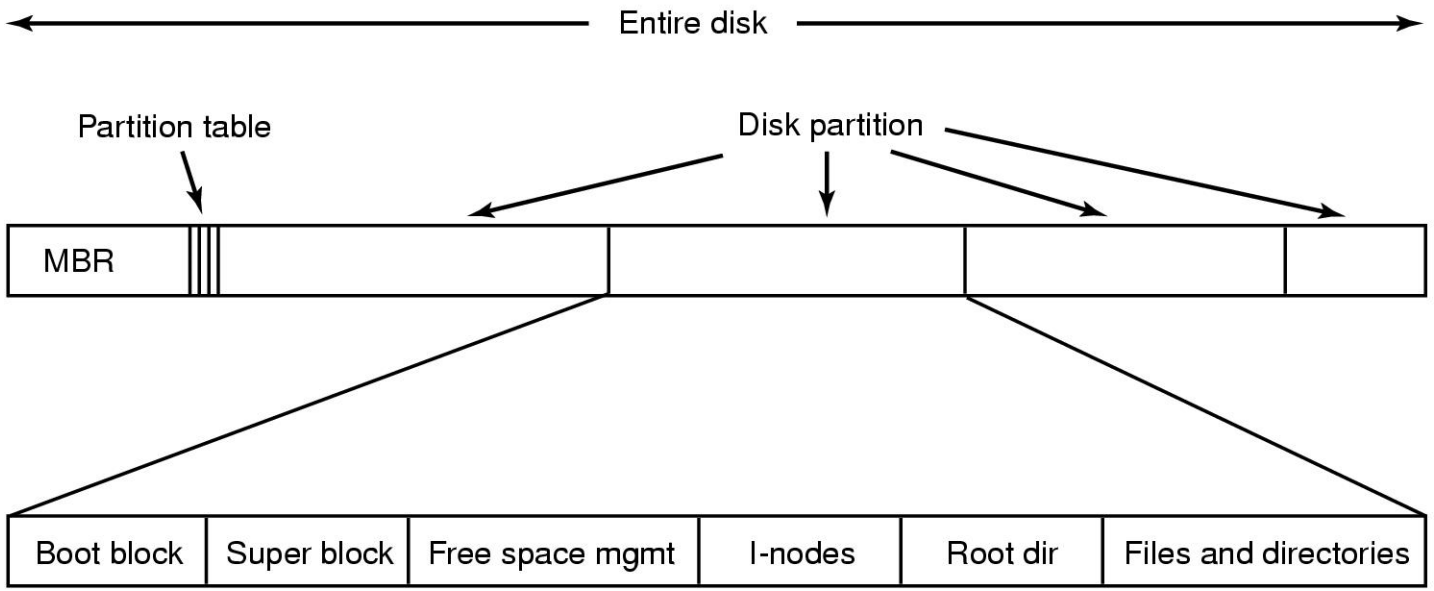
- File System Layout
- Implementing Files
- Implementing Directories
- Shared Files
- Disk Space Management



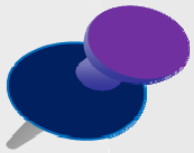
三 文件系统的实现（FILE SYSTEM IMPLEMENTATION）

1、文件系统布局

File System Implementation



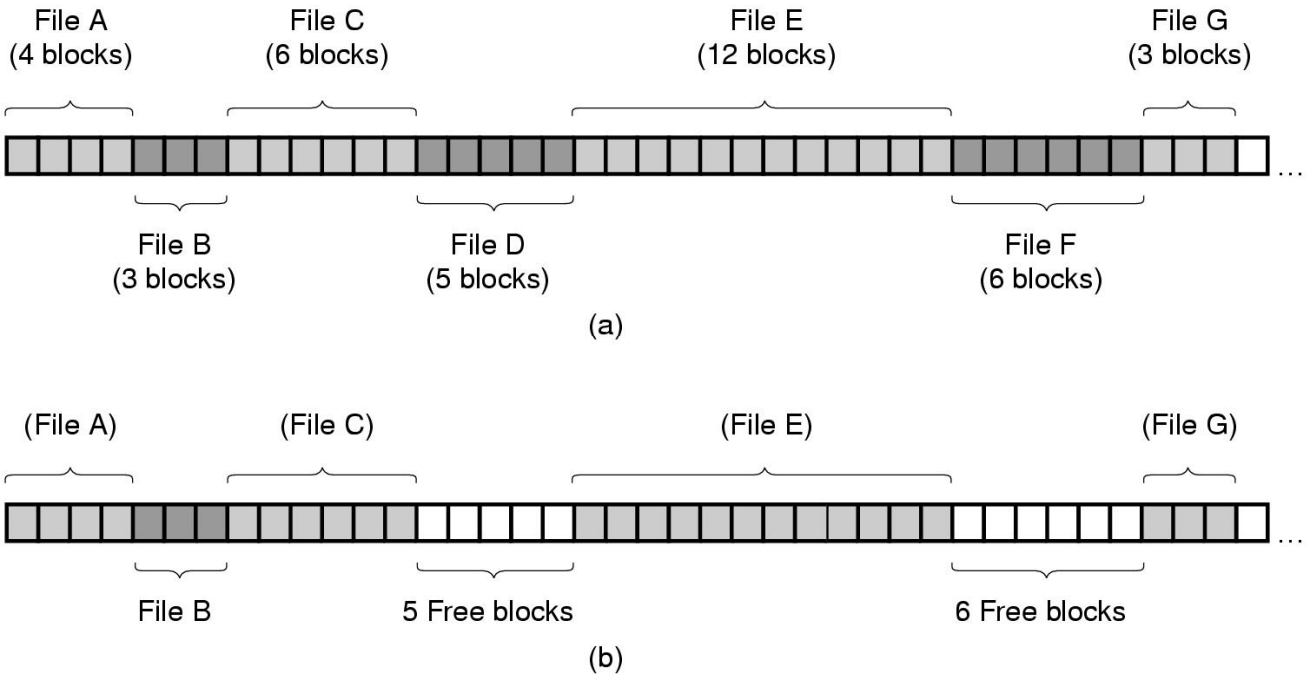
- A possible file system layout



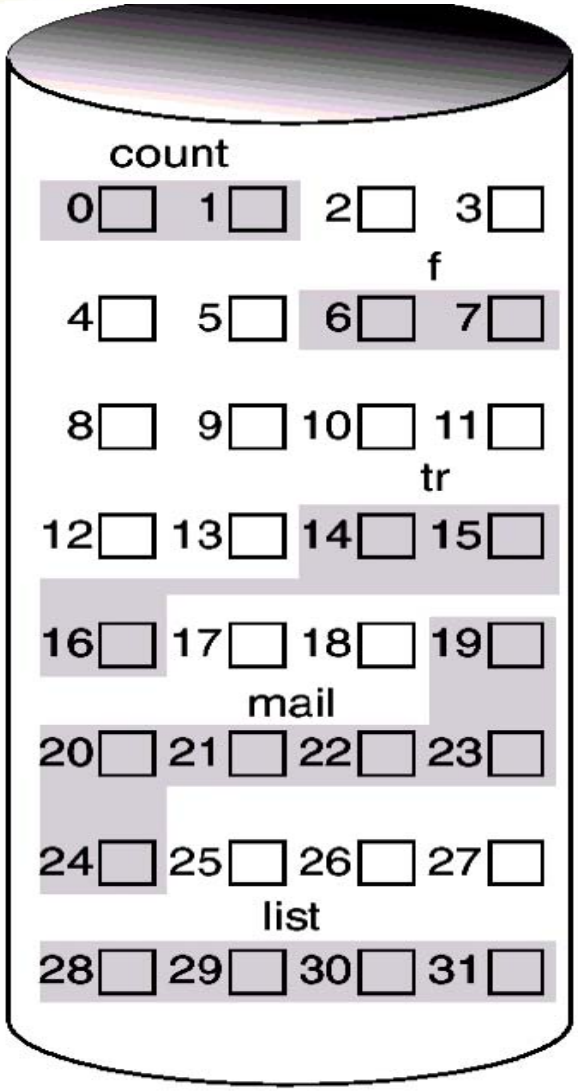
三 文件系统的实现
（FILE SYSTEM IMPLEMENTATION）

2、文件的实现

Implementing Files (1)



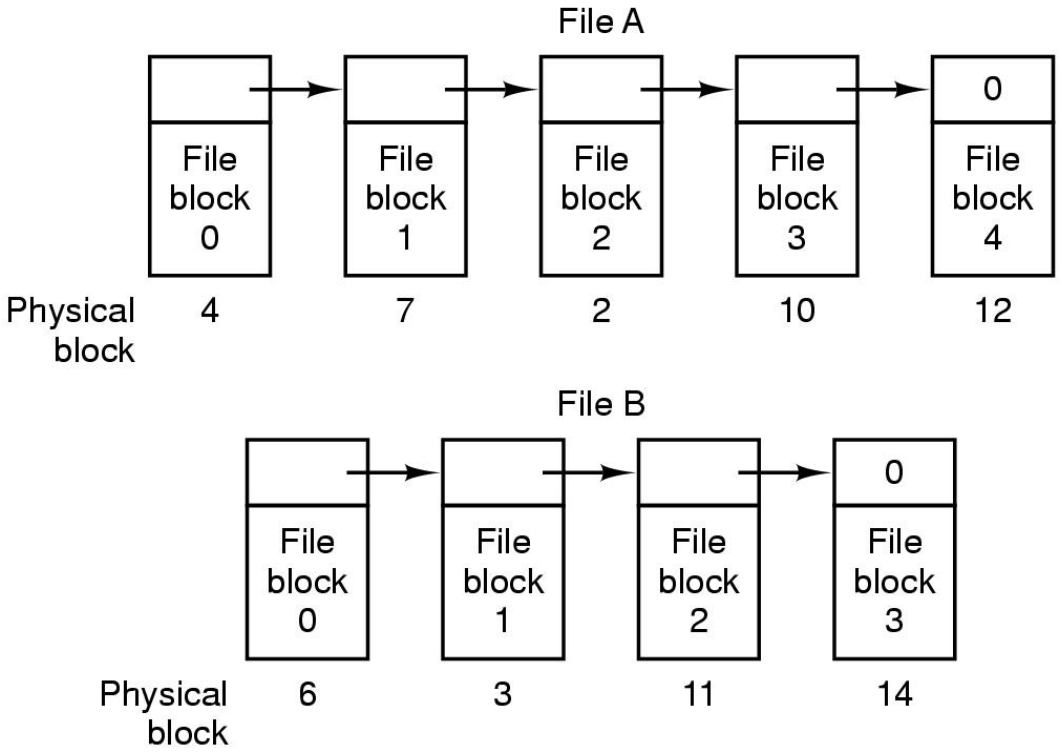
- (a) Contiguous allocation of disk space for 7 files
- (b) State of the disk after files *D* and *F* have been removed



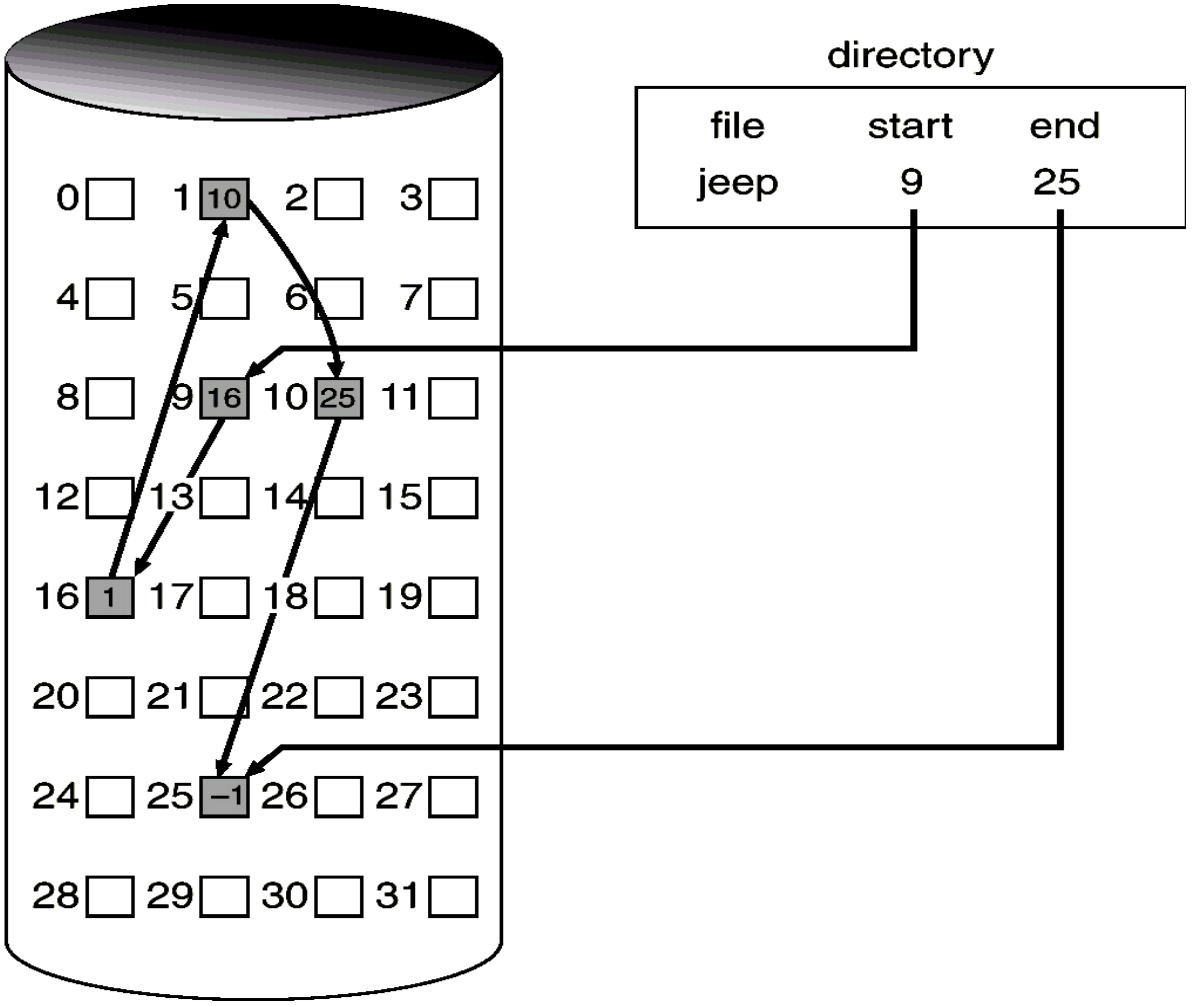
directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

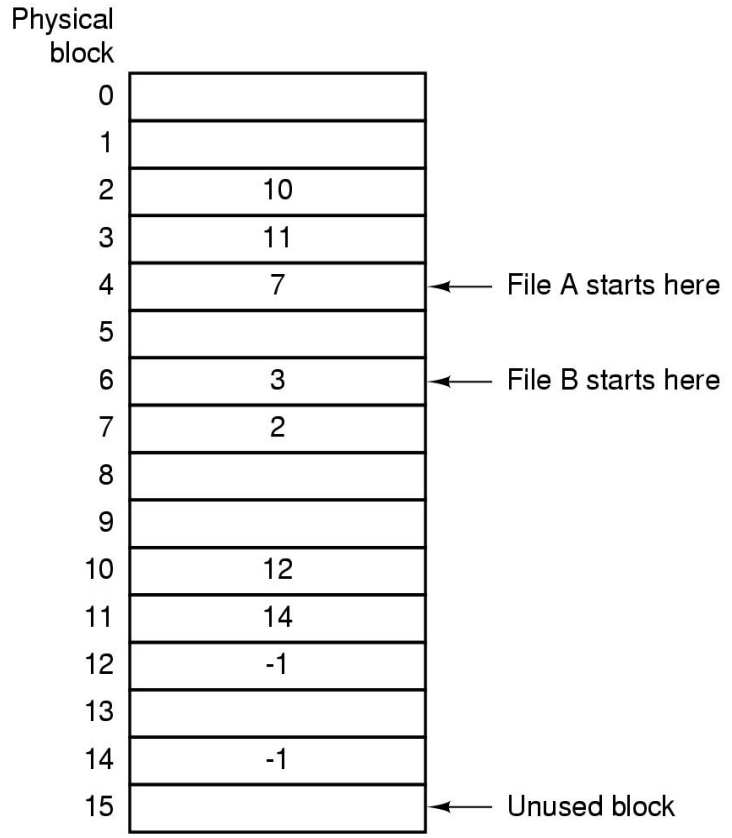
Implementing Files (2)



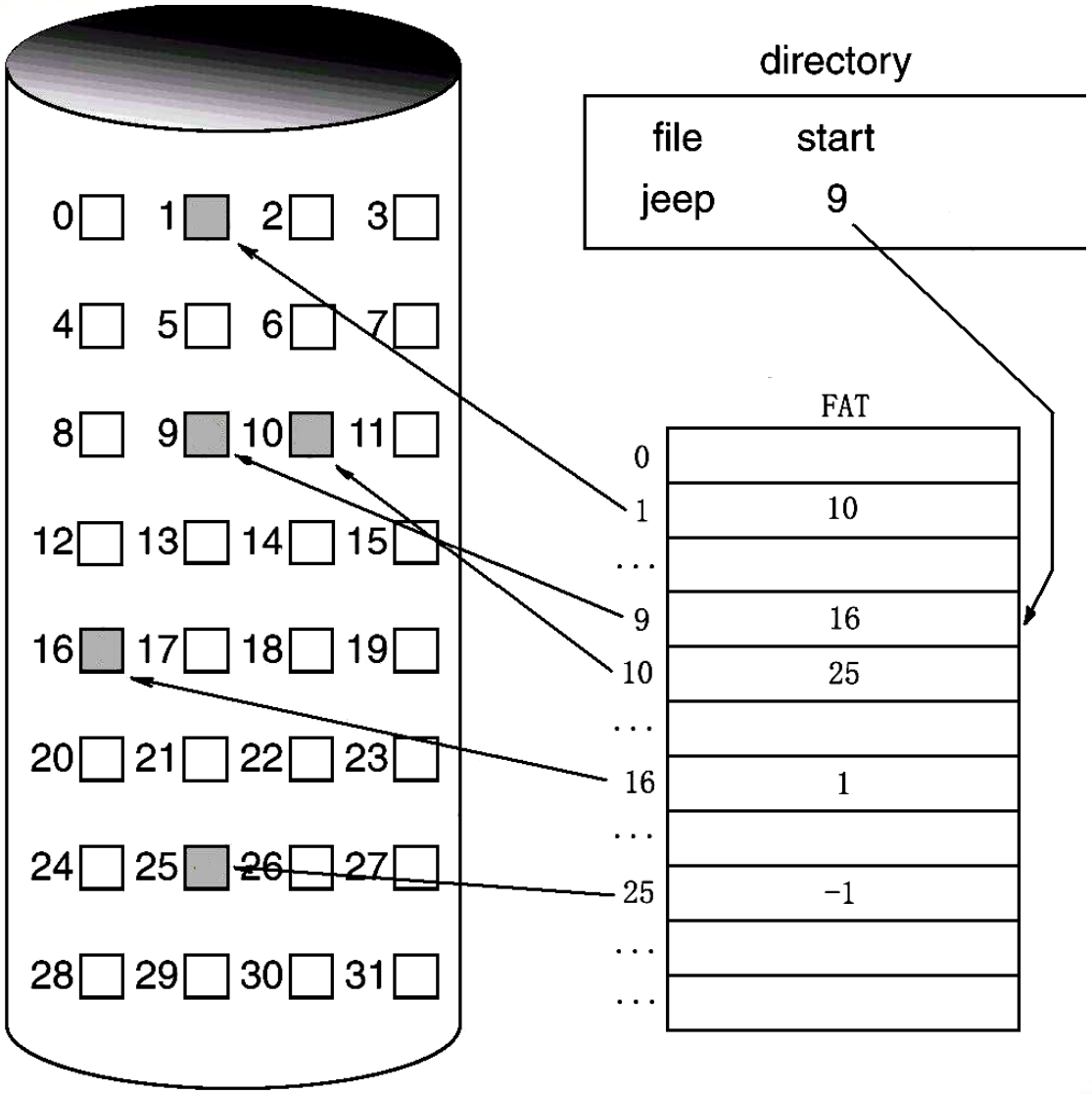
- Storing a file as a linked list of disk blocks


directory
file
start
end
jeep
9
25
0
1
10
2
3
4
5
6
7
8
9
16
10
25
11
12
13
14
15
16
1
17
18
19
20
21
22
23
24
25
−1
26
27
28
29
30
31

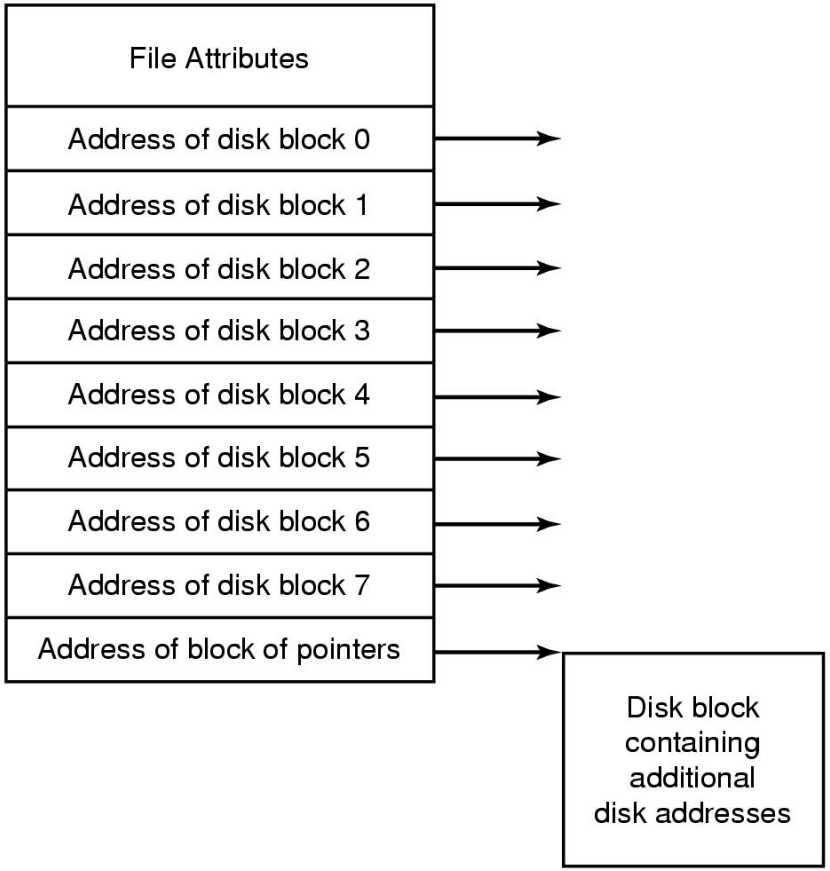
Implementing Files (3)



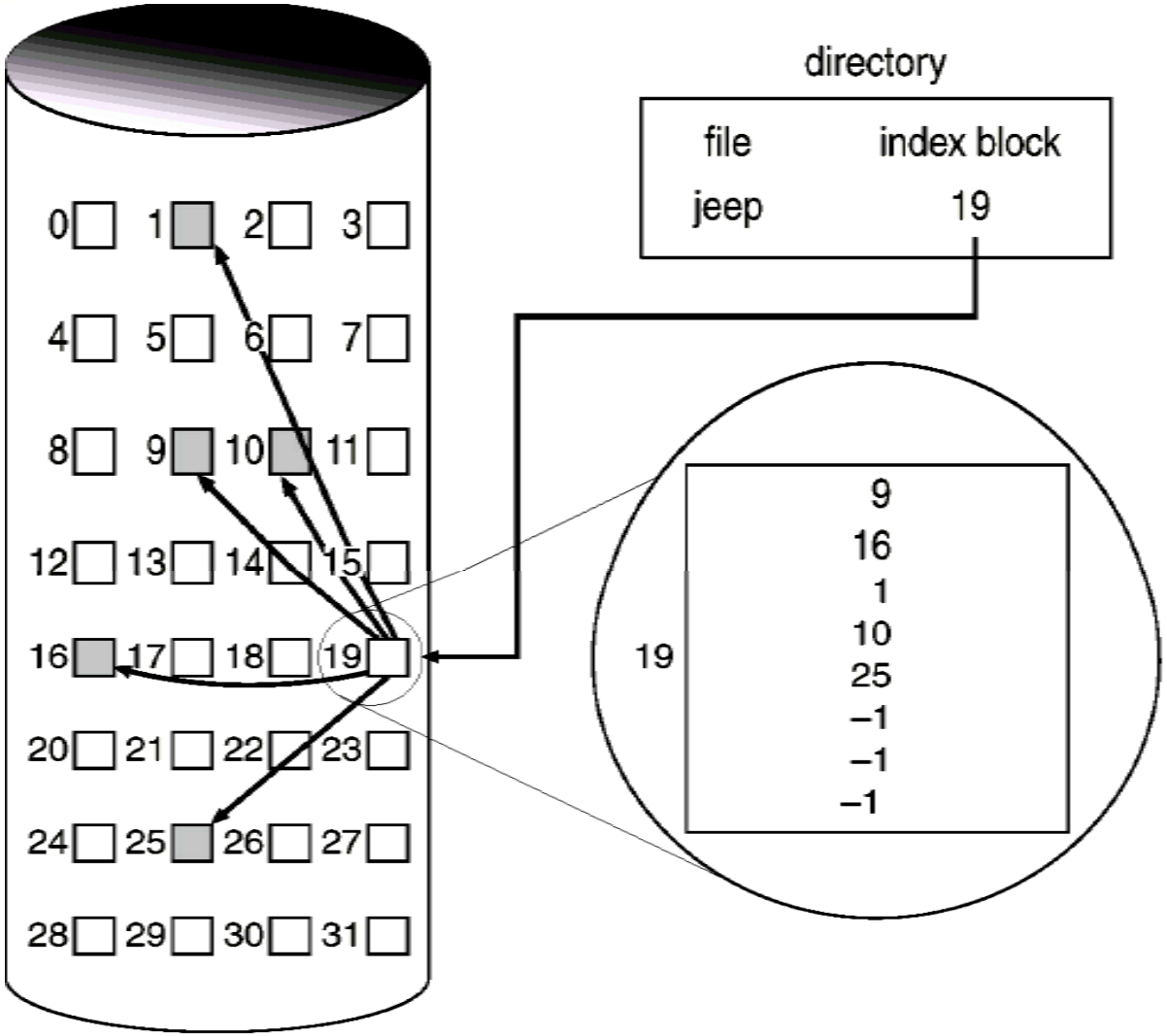
- Linked list allocation using a file allocation table in RAM

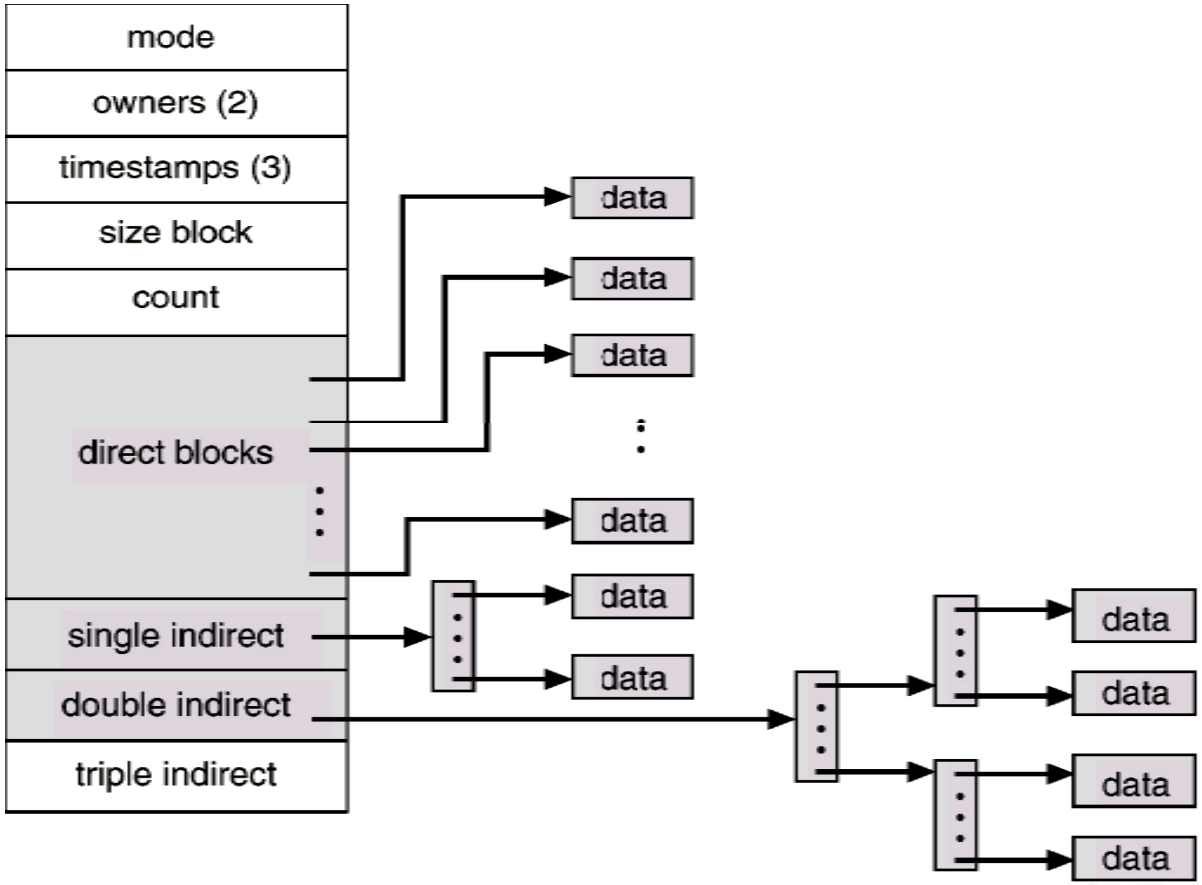

directory
file
start
jeep
9
FAT
0
1
10
9
16
10
25
16
1
25
-1
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31

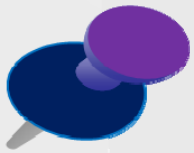
Implementing Files (4)



- An example i-node


directory
file
index block
jeep
19
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31
19
9
16
1
10
25
−1
−1
−1

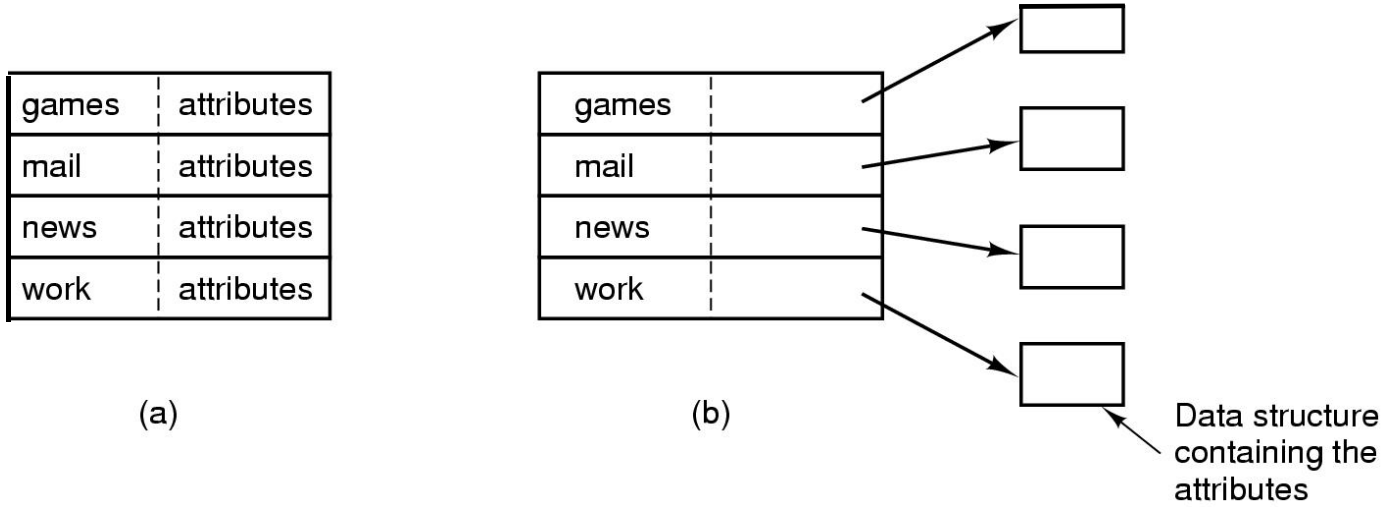

mode
owners (2)
timestamps (3)
size block
count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data



三 文件系统的实现（FILE SYSTEM IMPLEMENTATION）

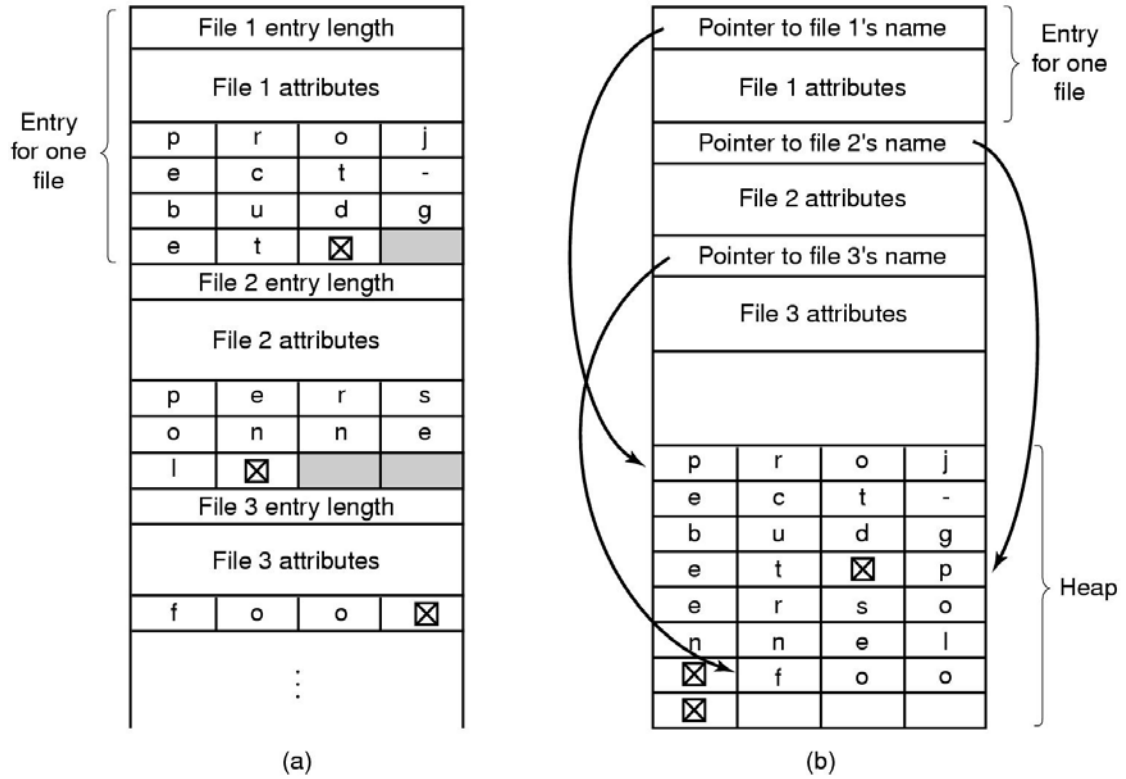
3、目录的实现

Implementing Directories (1)



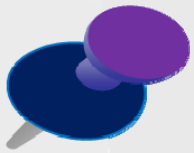
- (a) A simple directory
  - fixed size entries
  - disk addresses and attributes in directory entry
- (b) Directory in which each entry just refers to an i-node

Implementing Directories (2)



- Two ways of handling long file names in directory

(a) In-line (b) In a heap



三 文件系统的实现
（FILE SYSTEM IMPLEMENTATION）

4、共享文件

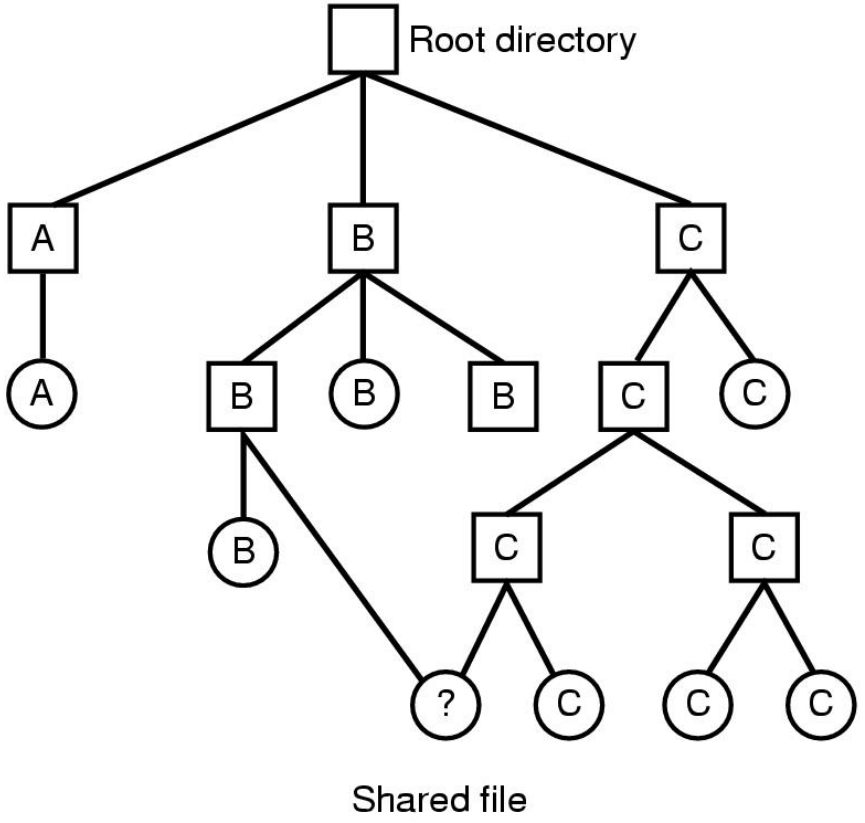
Shared Files

- 一处存储

  路径查找；文件命名

- 物理复制
- 链接技术

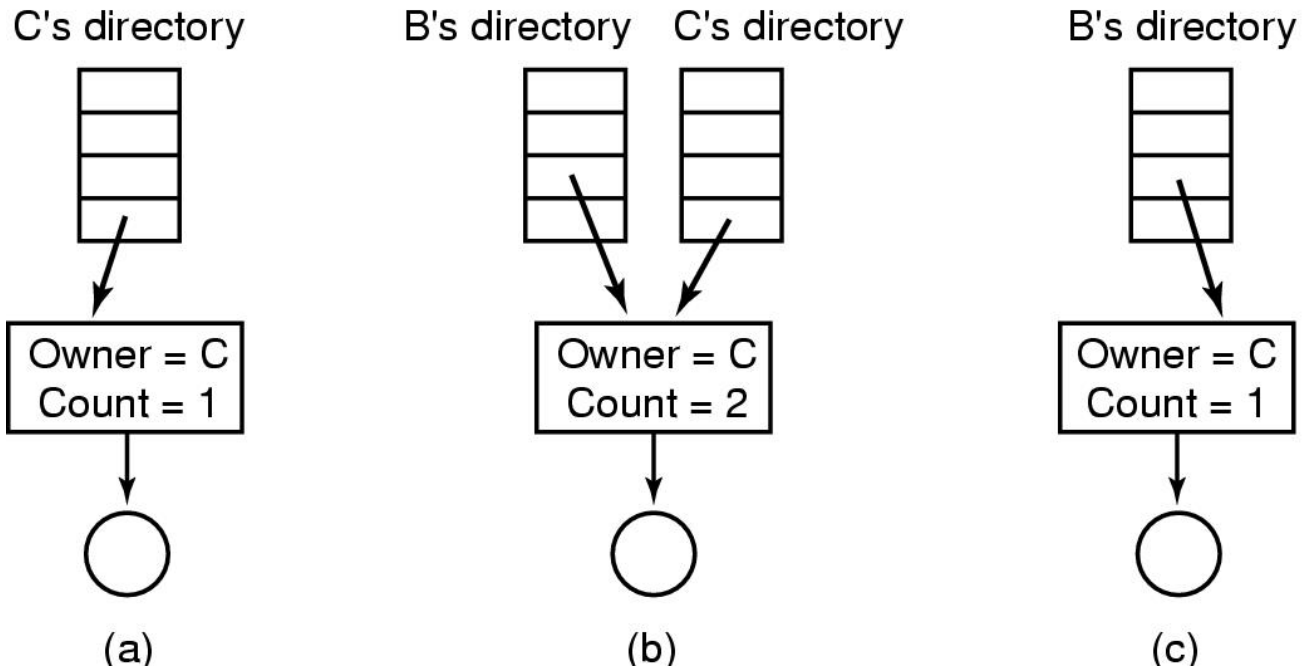
  硬链接；符号链接



Shared file

- File system containing a shared file

hard link



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

hard link

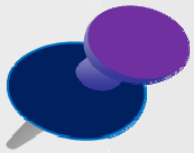
基于索引节点的文件别名：通过多个文件名链接(link)到同一个索引结点，可建立同一个文件的多个彼此平等的别名。别名的数目记录在索引结点的链接计数中，若其减至0，则文件被删除。

- UNIX举例：“ln source target”;当“rm source”则该文件还存在，文件名为target。
- 限制：不能跨越不同文件卷；通常不适用于目录（在UNIX中只对超级用户允许），否则由树状变为网状。

symbolic linking

基于符号链接的文件别名：它是一种LINK类型的文件，其内容是到另一个目录或文件的路径。建立符号链接文件，并不影响原文件，实际上它们各是一个文件。可以建立任意的别名关系，甚至原文件是在其他计算机上。

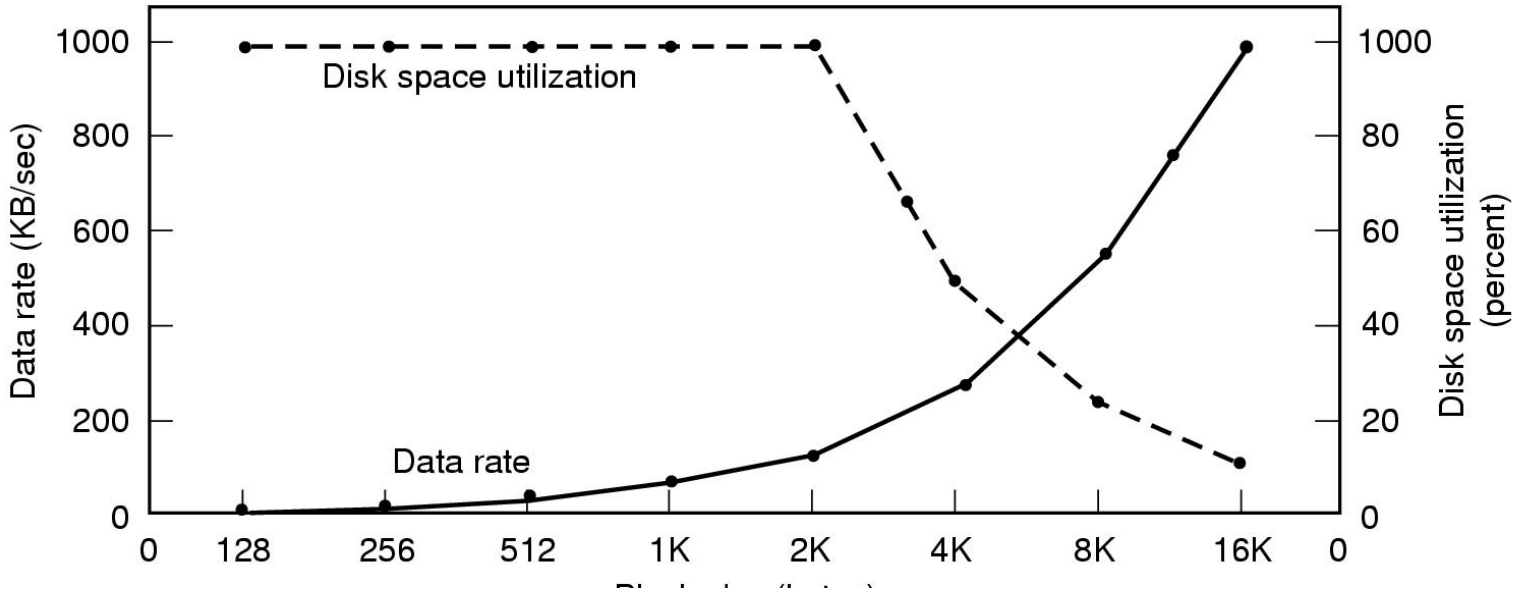
- UNIX举例：“ln -s a b”；当“rm a”则文件a不存在，b便无法访问。
- 适用范围和灵活性较大。



三 文件系统的实现（FILE SYSTEM IMPLEMENTATION）

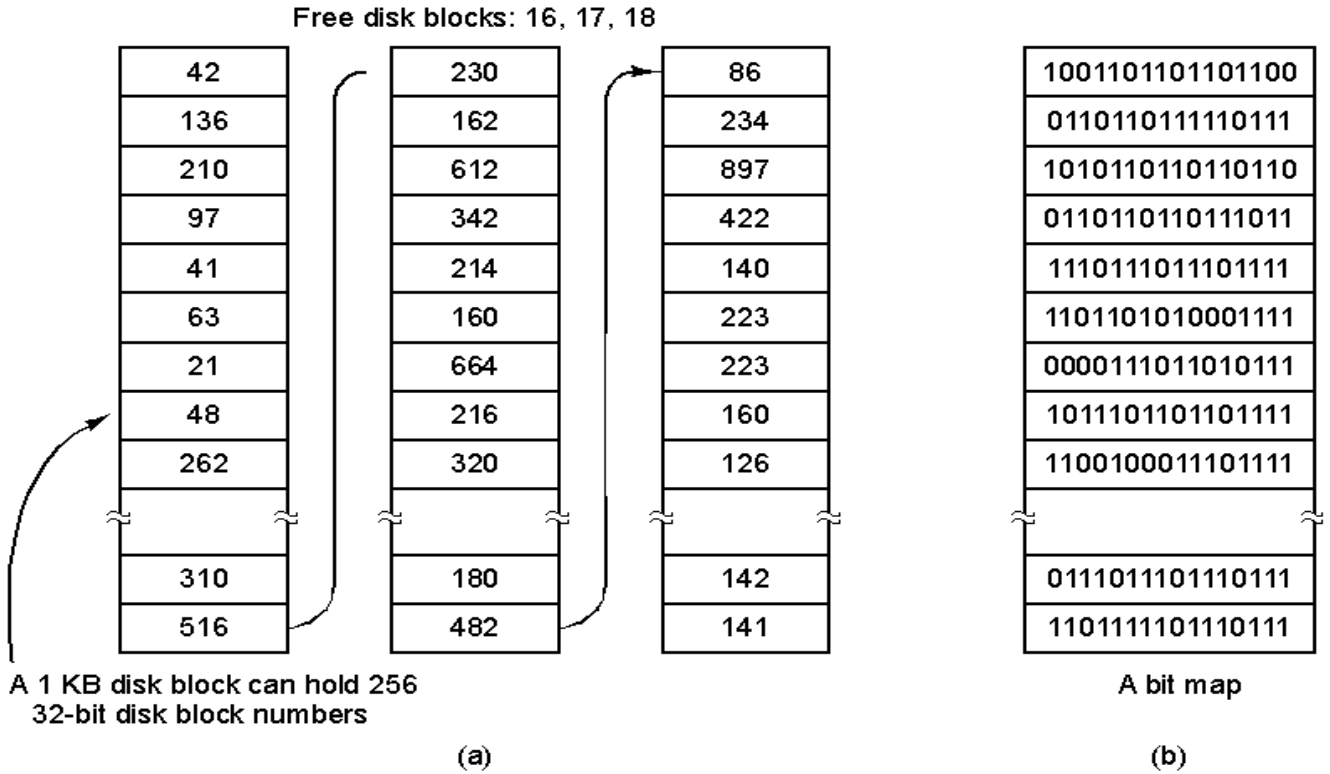
5、磁盘空间管理

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

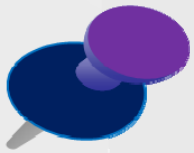
Disk Space Management (2)



- (a) Storing the free list on a linked list
- (b) A bit map

小 结

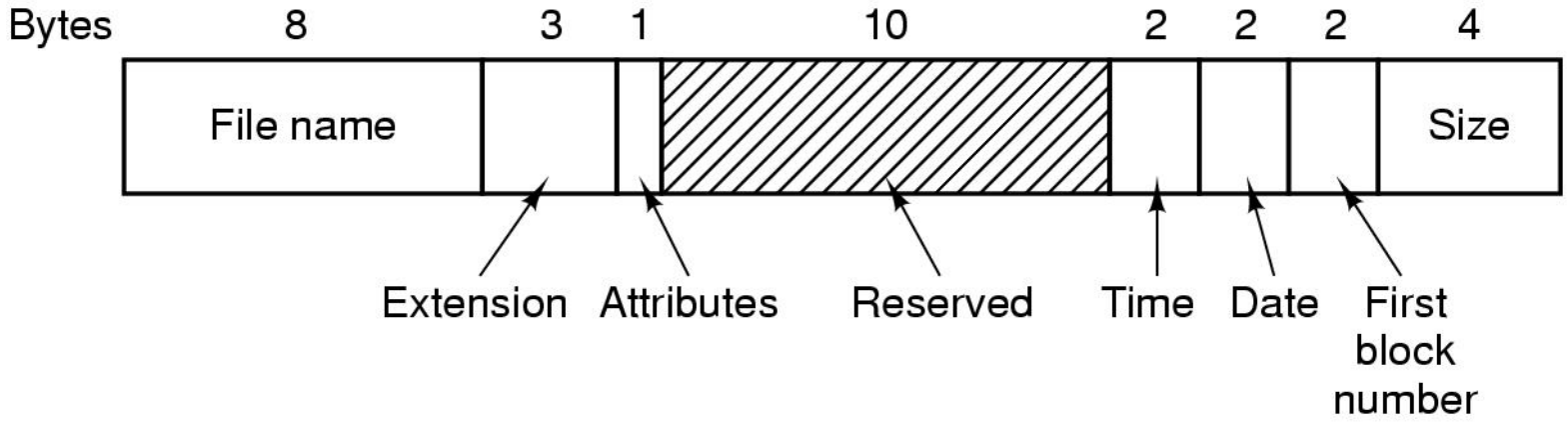
- 文件系统的布局
- 文件的物理结构
  - 连续、链式、i节点
- 目录的两种结构
- 文件的共享
- 磁盘空间管理



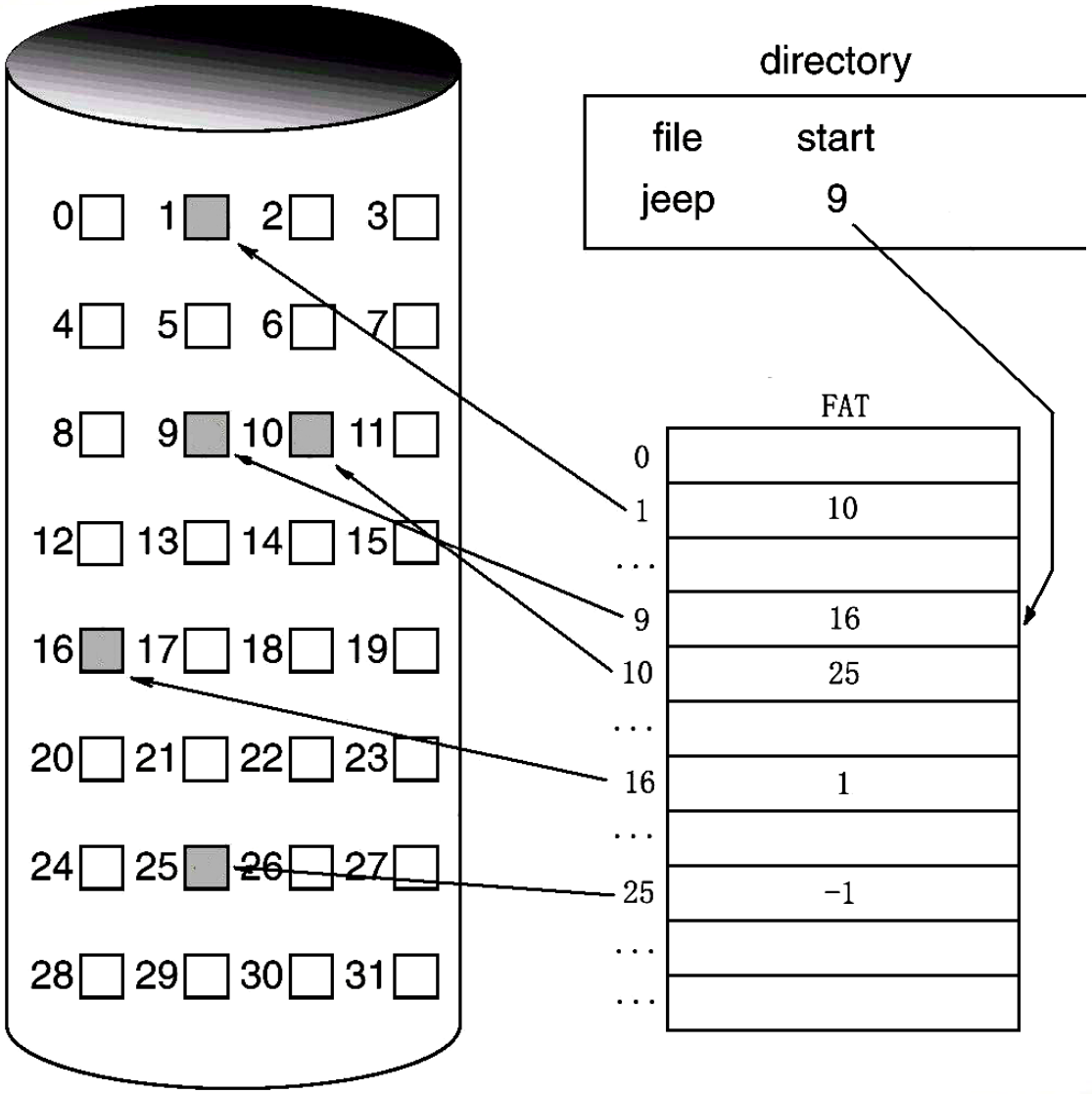
四 文件系统实例
（EXAMPLE FILE SYSTEMS）

1、MS-DOS 文件系统

The MS-DOS File System (1)



- The MS-DOS directory entry


directory
file
start
jeep
9
FAT
0
1
10
9
16
10
25
16
1
25
-1
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31

The MS-DOS File System (2)

FAT12:

4K*2^12=2^(12+12)=2^24

=16M

FAT16:

32K*2^16=2^(15+16)=2^31

=2048M

FAT32:

32K*2^28=2^(15+28)=2^43

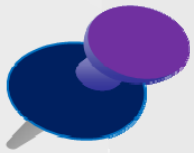
=8T（理论上）

512*2^32=2^(9+32)=2^41

=2T（实际上）

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

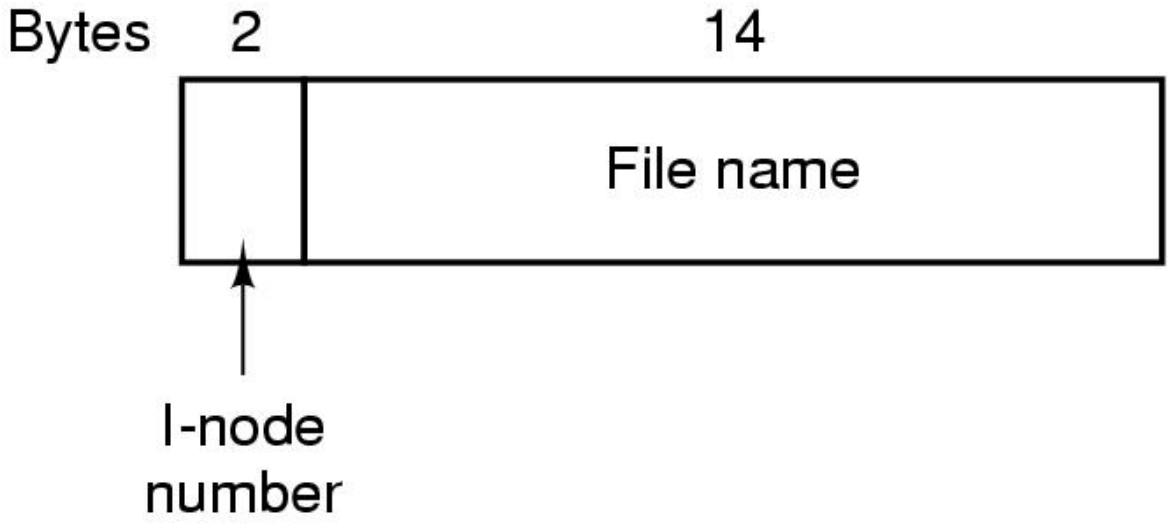
Maximum partition size for different block sizes.
The empty boxes represent forbidden combinations.



四 文件系统实例
（EXAMPLE FILE SYSTEMS）

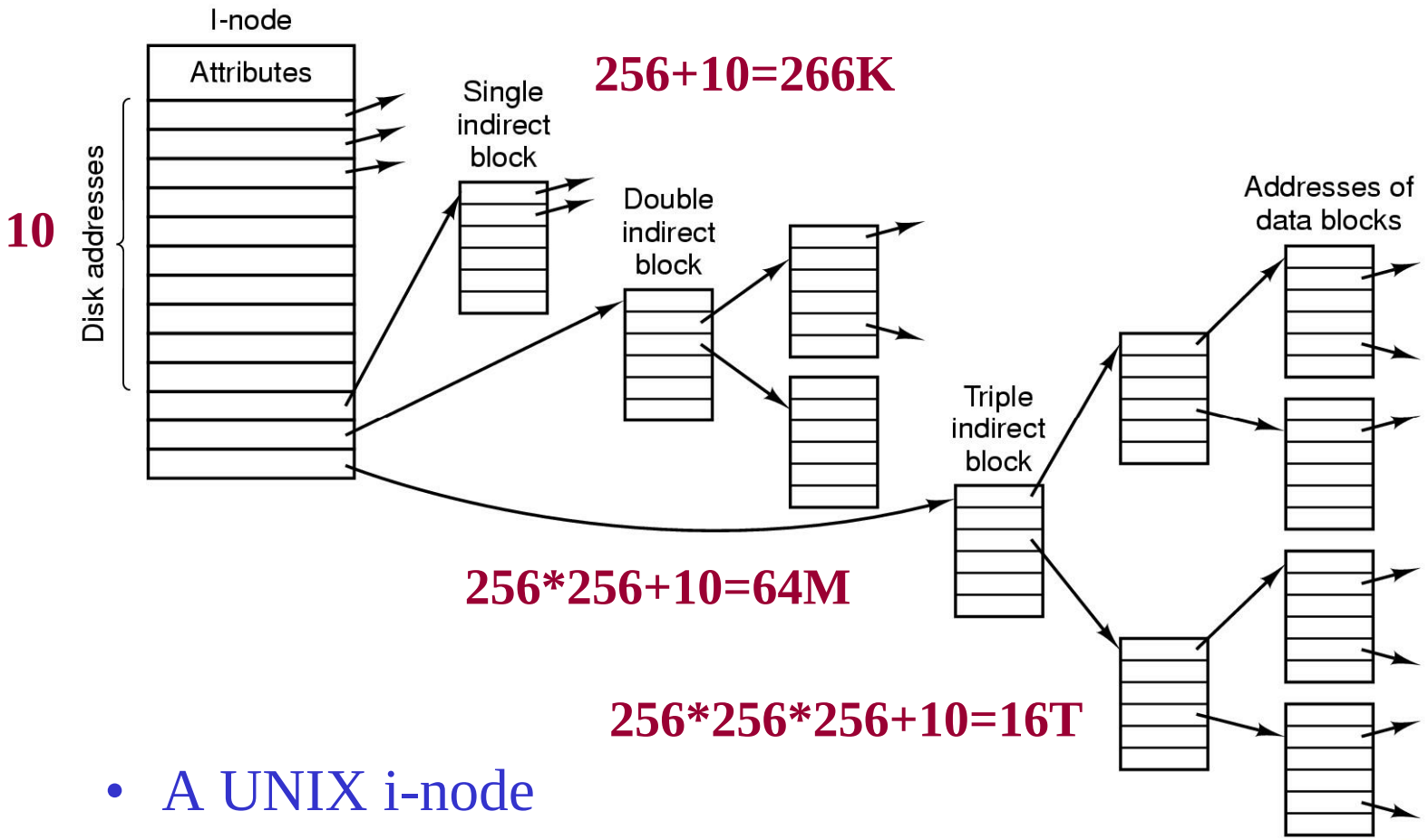
2、UNIX V7 文件系统

The UNIX V7 File System (1)



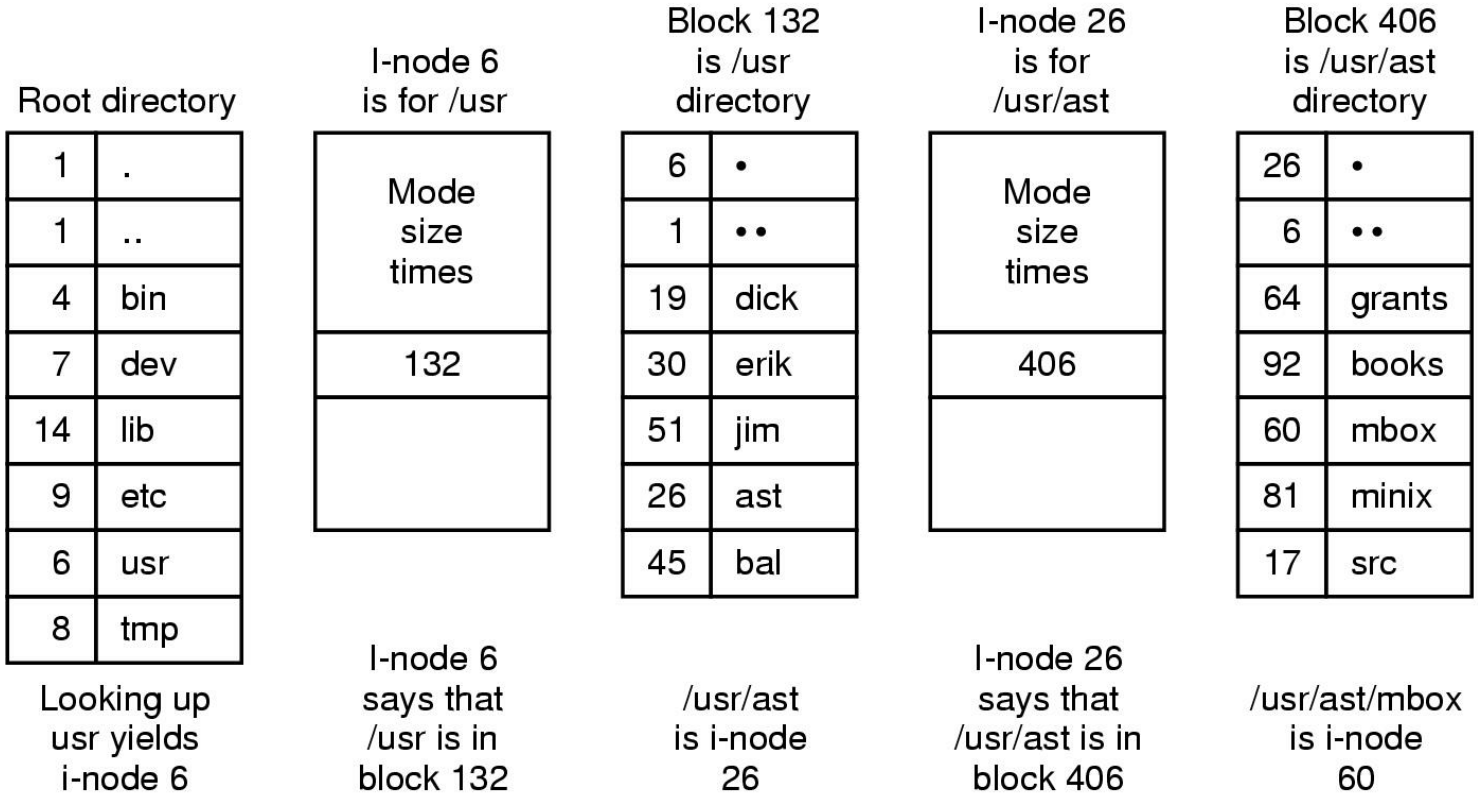
- A UNIX V7 directory entry

The UNIX V7 File System (2)

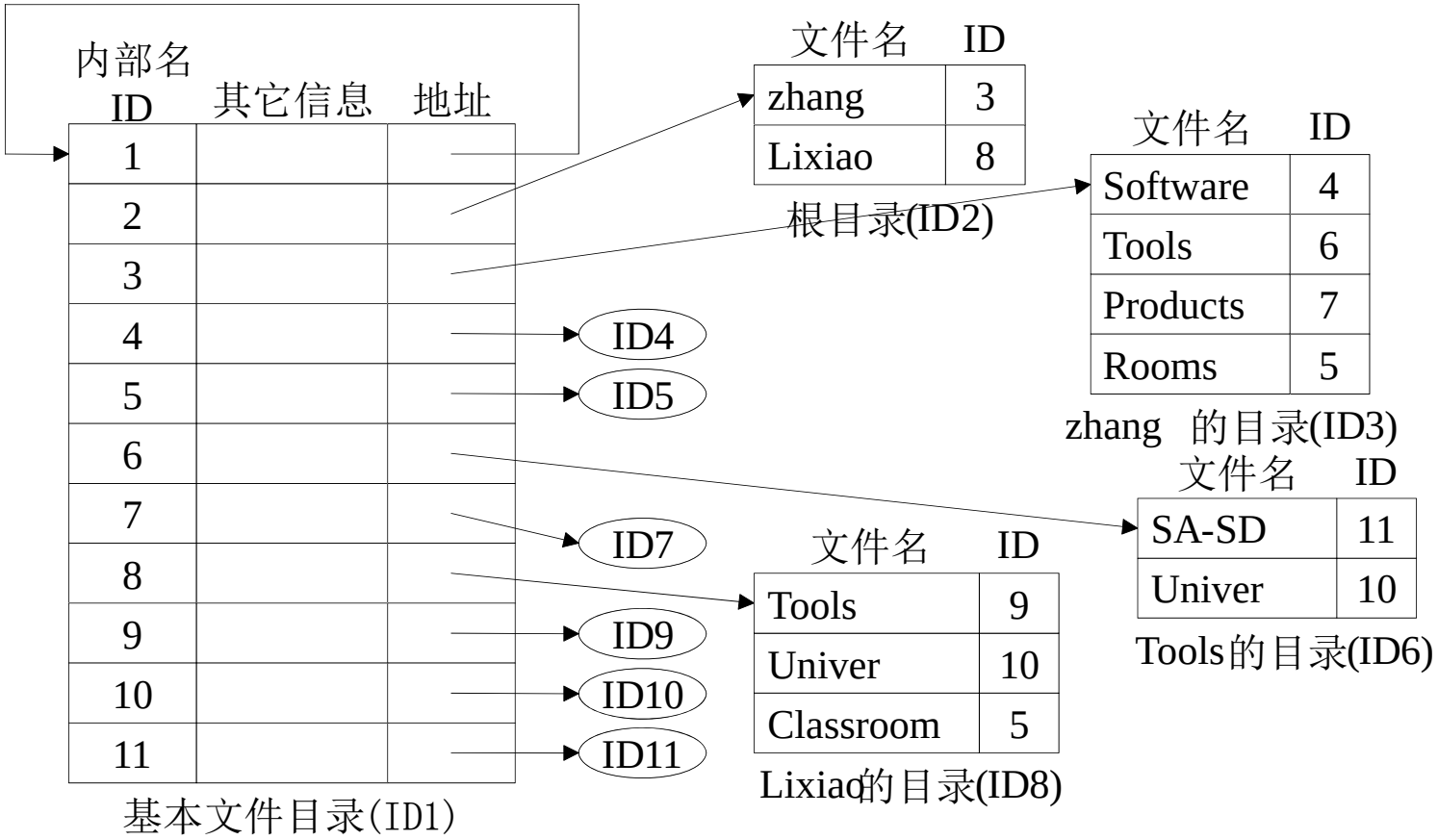


- A UNIX i-node

The UNIX V7 File System (3)



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


内部名
ID
其它信息
地址
1
2
3
4
5
6
7
8
9
10
11
ID4
ID5
ID7
ID9
ID10
ID11
基本文件目录(ID1)
文件名
ID
zhang
3
Lixiao
8
根目录(ID2)
文件名
ID
Software
4
Tools
6
Products
7
Rooms
5
zhang 的目录(ID3)
文件名
ID
SA-SD
11
Univer
10
Tools的目录(ID6)
文件名
ID
Tools
9
Univer
10
Classroom
5
Lixiao的目录(ID8)

小 结

- The MS-DOS File System
- The UNIX V7 File System