Operating Systems

Jinghui Zhong （钟竞辉）
Office： B3-515
Email ： jinghuizhong@scut.edu.cn

Review

Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB. Which hole is taken for successive segment requests of

(a) 12 MB
(b) 10 MB
(c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit.

If an instruction takes 1 nsec and a page fault takes an additional *n* nsec, give a formula for the effective instruction time if page faults occur every *k* instructions.

You are given the following data about a virtual memory system:

(a)The TLB can hold 1024 entries and can be accessed in 1 clock cycle (1 nsec).
(b) A page table entry can be found in 100 clock cycles or 100 nsec.
(c) The average page replacement time is 6 msec.

If page references are handled by the TLB 99% of the time, and only 0.01% lead to a page fault, what is the effective address-translation time?

Review

A computer with an 8-KB page, a 64-MB main memory, and a 64-GB virtual address space uses an inverted page table to implement its virtual memory. How big should the hash table be to ensure a mean hash chain length of less than 1? Assume that the hash-table size is a power of two.

It has been observed that the number of instructions executed between page faults is directly proportional to the number of page frames allocated to a program. If the available memory is doubled, the mean interval between page faults is also doubled. Suppose that a normal instruction takes 1 microsec, but if a page fault occurs, it takes 2001 μsec (i.e., 2 msec) to handle the fault. If a program takes 60 sec to run, during which time it gets 15,000 page faults, how long would it take to run if twice as much memory were available?

How long does it take to load a 64-KB program from a disk whose average seek time is 5 msec, whose rotation time is 5 msec, and whose tracks hold 1 MB

(a) for a 2-KB page size?
(b) for a 4-KB page size?

The pages are spread randomly around the disk and the number of cylinders is so large that the chance of two pages being on the same cylinder is negligible.

Long-term information storage

- Must store large amounts of data
  - ✓Gigabytes -> terabytes -> petabytes
- Stored information must survive the termination of the process using it
  - ✓Lifetime can be seconds to years
  - ✓Must have some way of finding it
- Multiple processes must be able to access the information concurrently
- **Solution:** Store information on disk or other external media in units called **files**.

Naming files

- Important to be able to *find* files after they're created
- Every file has at least one name
- Name can be

  "foo.c", "my photo",

  "4502",
- Case may or may not matter (Depends on the file system)
- Name may include information about the file's contents, e.g.,

The name should make it easy to figure out what's in it;

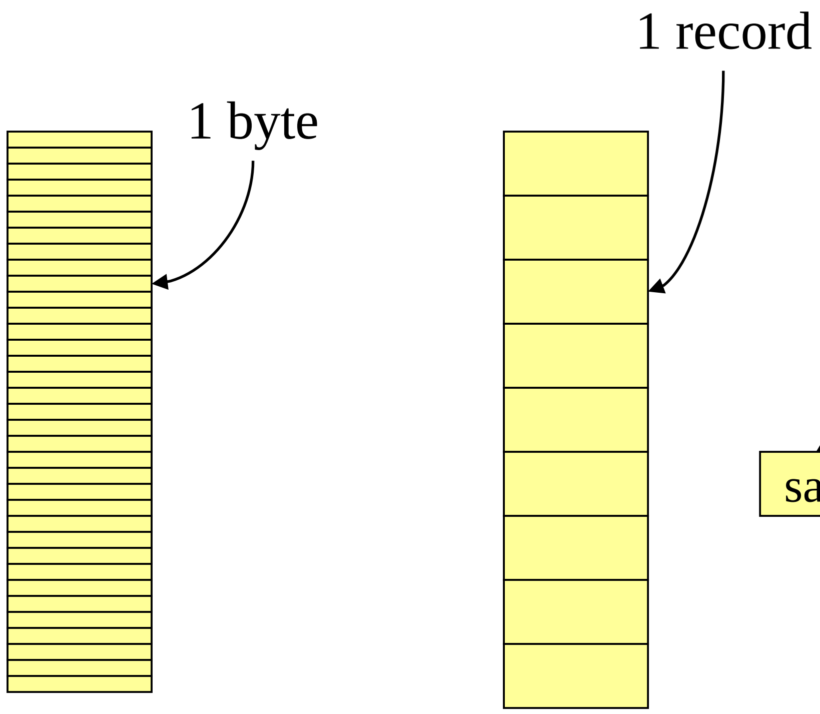
Computer may use part of the name to determine the file type

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

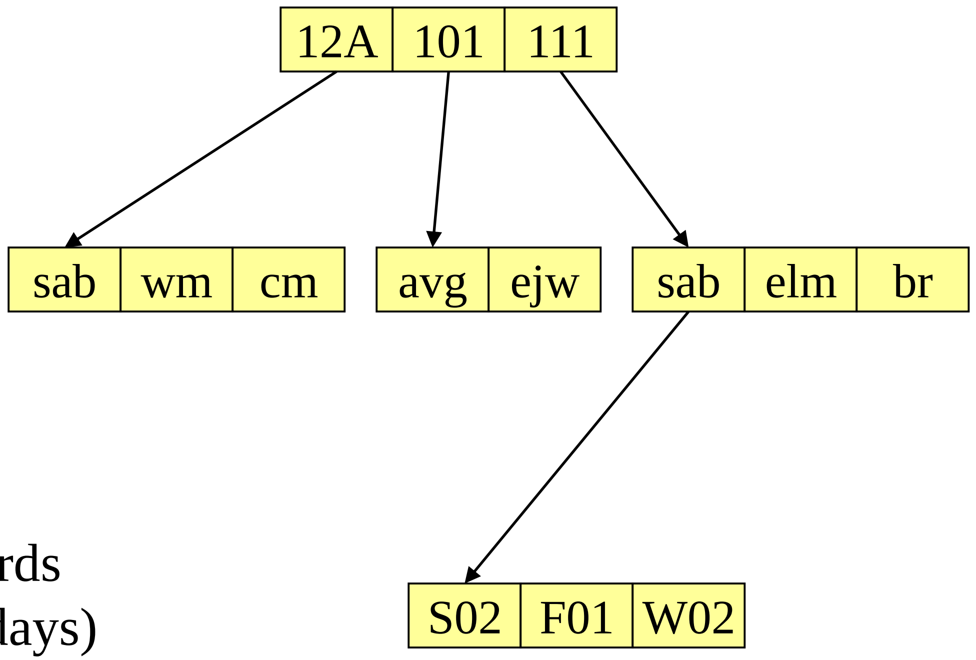
Typical file extensions.

File Structure



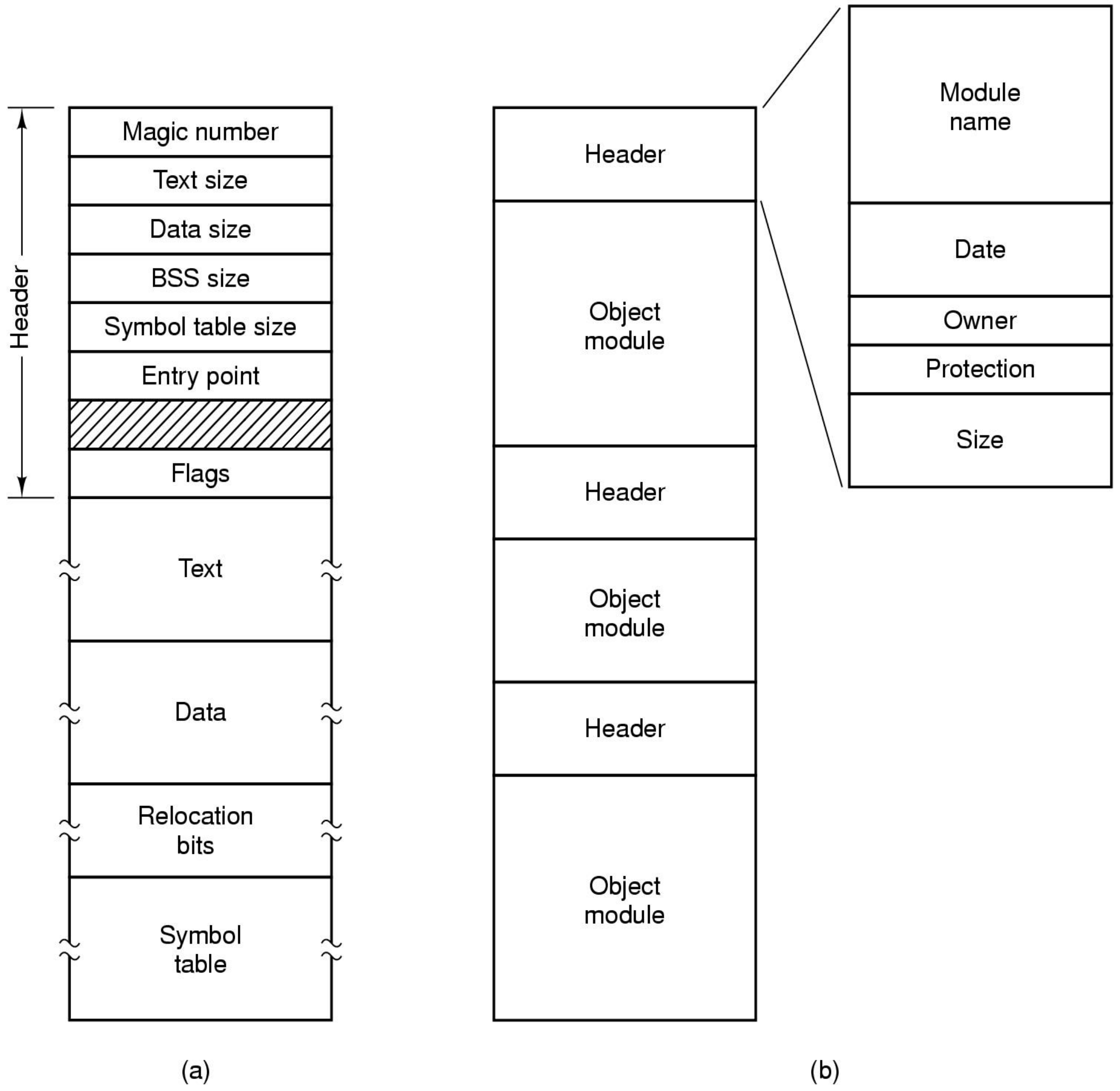
Sequence of bytes
(Windows & UNIX)

Sequence of records
(common in old days)



Tree
(large mainframe computers)

File Types



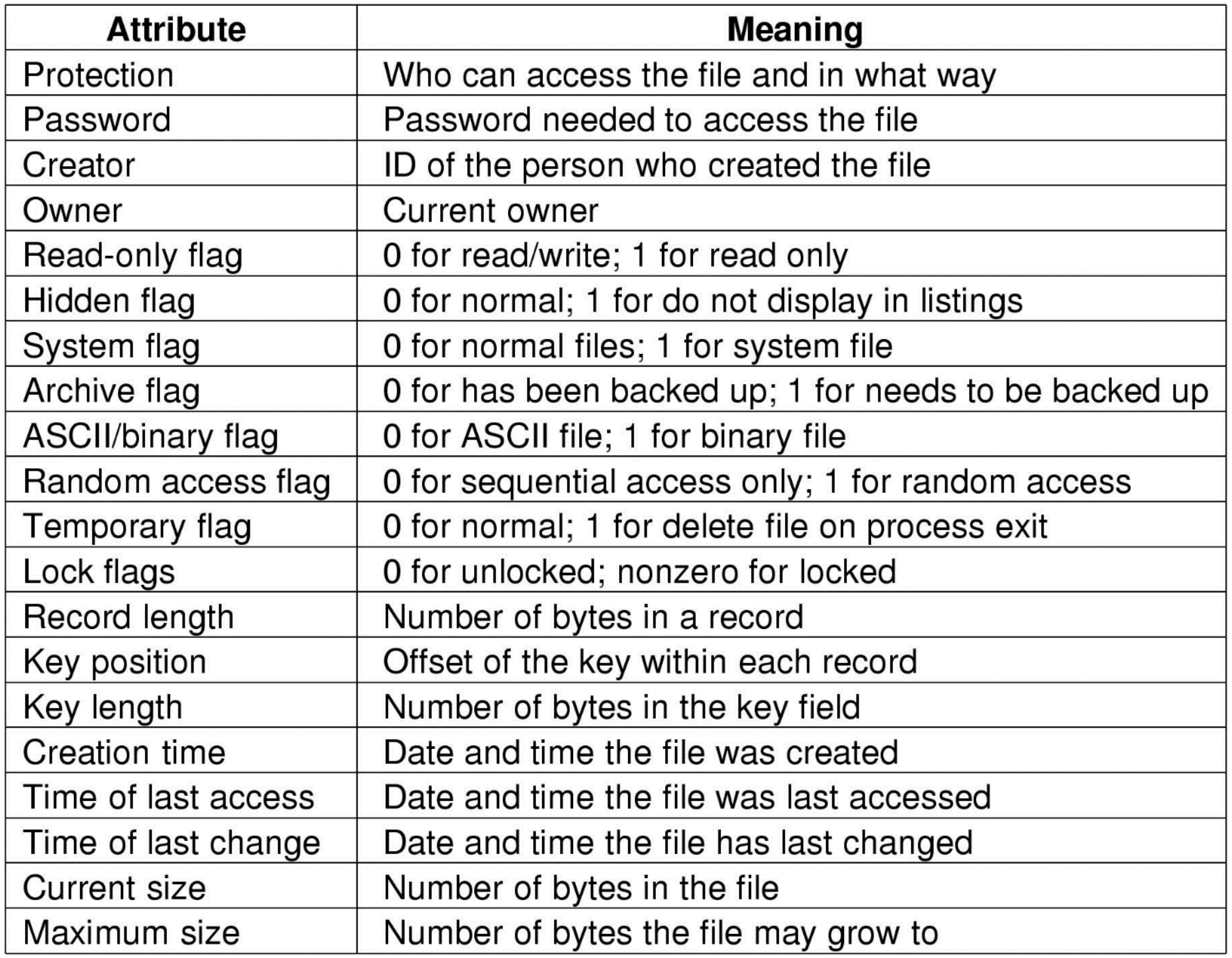
(a) An executable file (b) An archive

File Access

- **Sequential access**
  - ✓read all bytes/records from the beginning
  - ✓cannot jump around
  - ✓convenient when medium was magnetic tape
- **Random access**
  - ✓bytes/records read in any order
  - ✓essential for database systems

File Attributes

- Operating systems associate extra information with each file, called **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 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

An Example Program Using File System Calls

```
/* 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 */
```

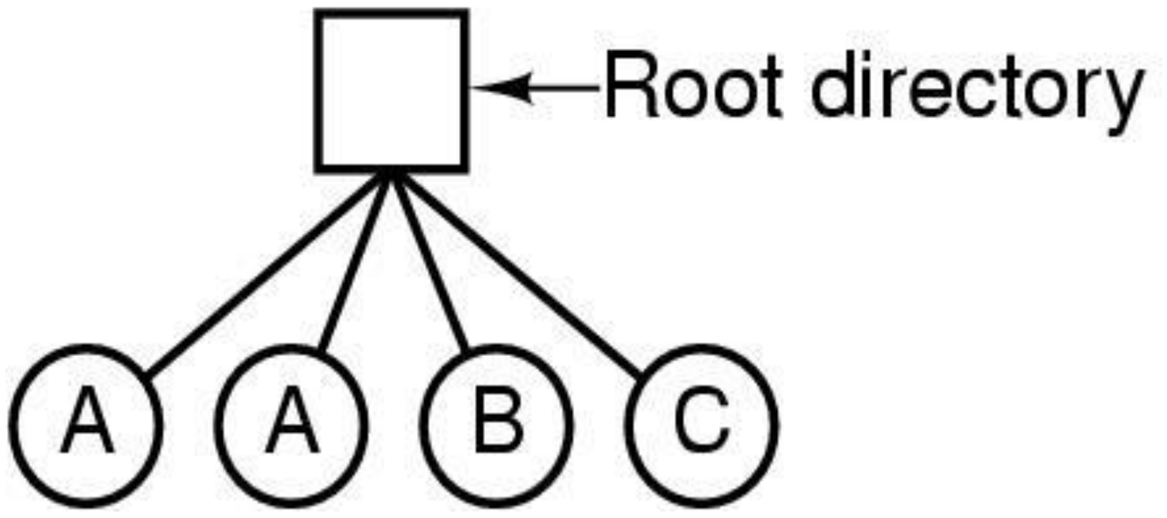
An Example Program Using File System Calls

```
    /* 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 */

    /* 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 */
}
```

Directories

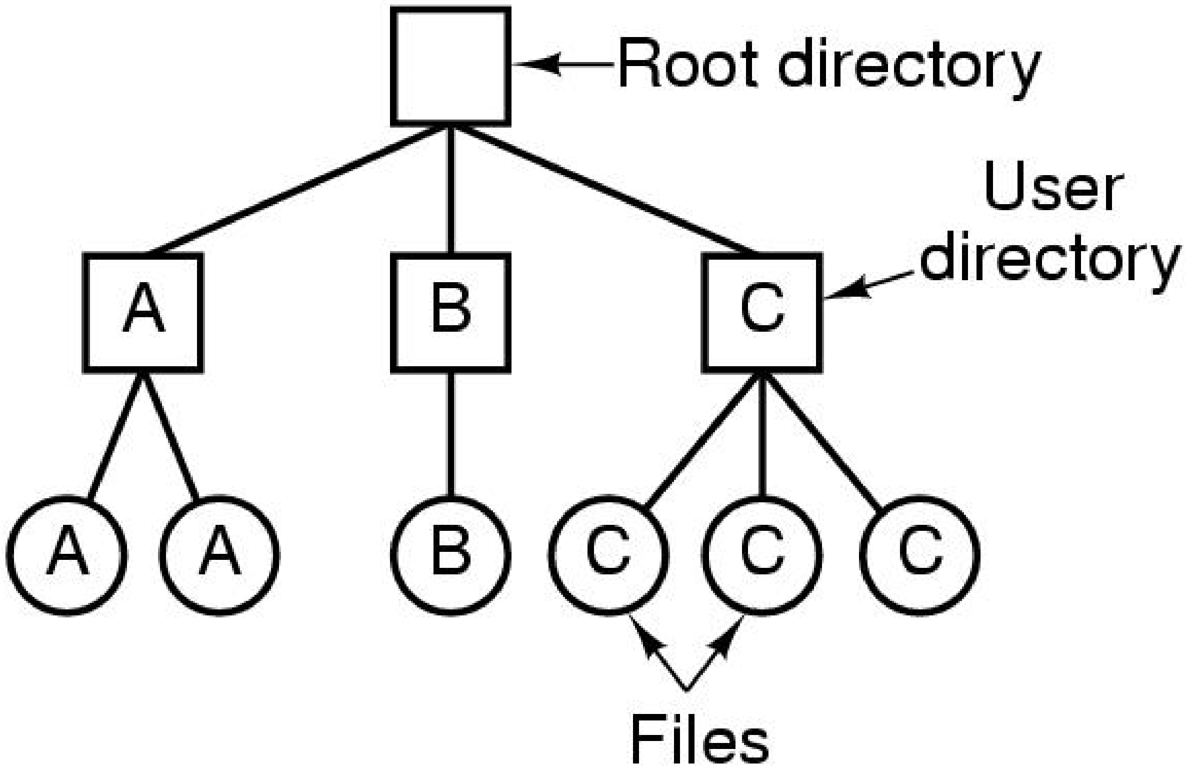
- **File systems have directories or folders to keep track of files.**
  - ① A single-level directory has one directory (**root**) containing all the files.
  - ② A two-level directory has a root directory and user directories.
  - ③ A hierarchical directory has a root directory and arbitrary number of subdirectories.

Directories - A single level directory system



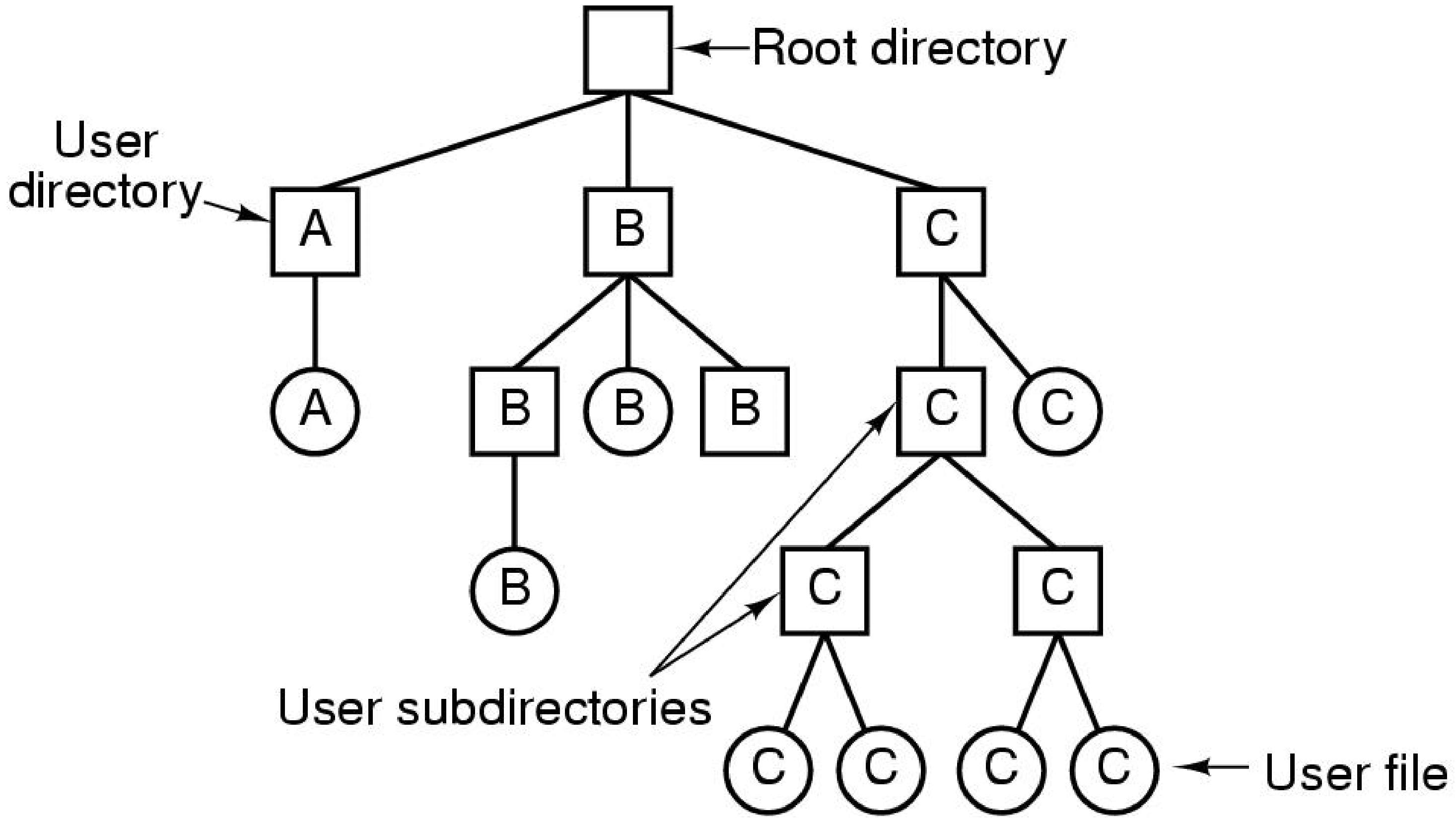
- A single level directory system
  - ✓contains 4 files
  - ✓owned by 3 different people, A, B, and C
- Common in early personal computers.

Two-level Directory Systems



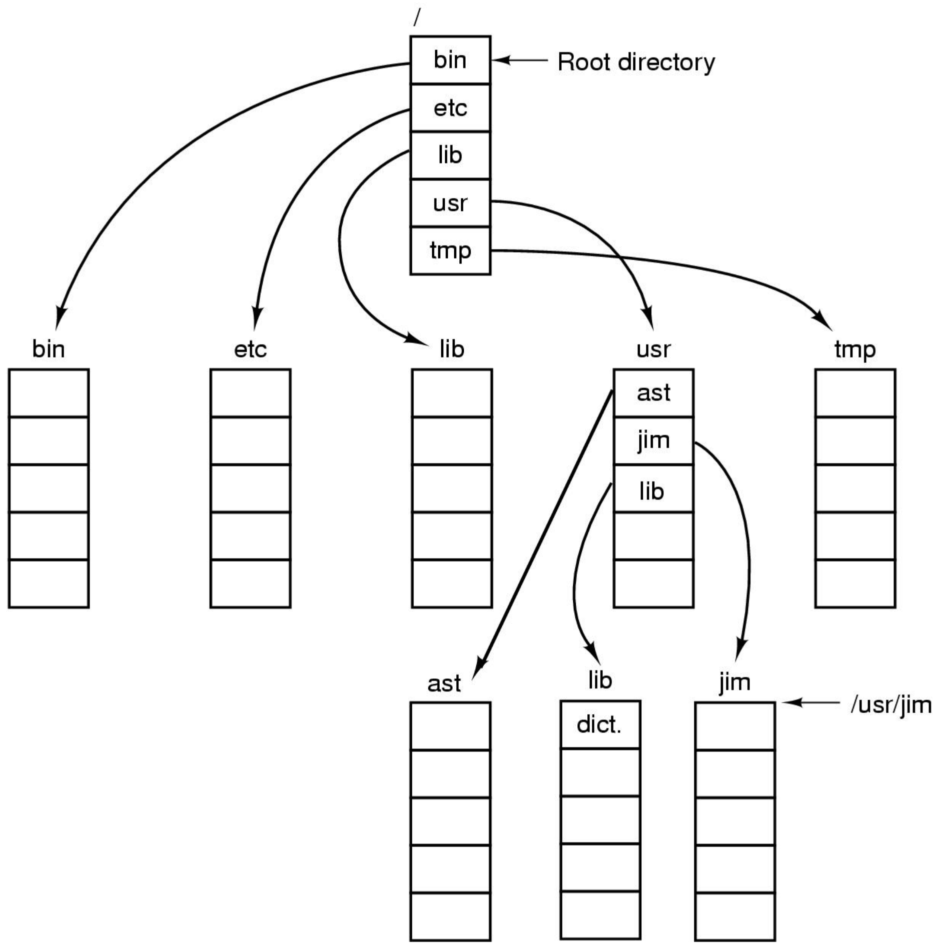
Letters indicate *owners* of the directories and files

Hierarchical Directory Systems



All modern file systems are organized in this manner.

A UNIX directory tree



Directories

- **Two different methods are used to specify file names in a directory tree:**
  ① **Absolute path name** consists of the path from the root directory to the file. e.g., cp /usr/ast/mailbox /usr/ast/mailbox.bak

  ② **Relative path name** consists of the path from the current directory (**working directory**). e.g,
  cp ../lib/dictionary ➔ cp /usr/lib/dictionary

Directories

- The path name are different in different systems:
  - ✓Winodws: \usr\ast\mailbox
  - ✓UNIX: /usr/ast/mailbox
  - ✓MULTICS: >usr>ast>mailbox
- "." and ".." are two special entries in the file system.
  - ✓Dot (.) refers to the current directory (working directory).
  - ✓Dot dot (..) refers to its parent.

Directory Operations

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

Check points

① List three reasons for long-term information storage;

② What is sequence of bytes file structure?

③ Please list at least five common file attributes.

④ What is relative path name and why do we need relative path name?