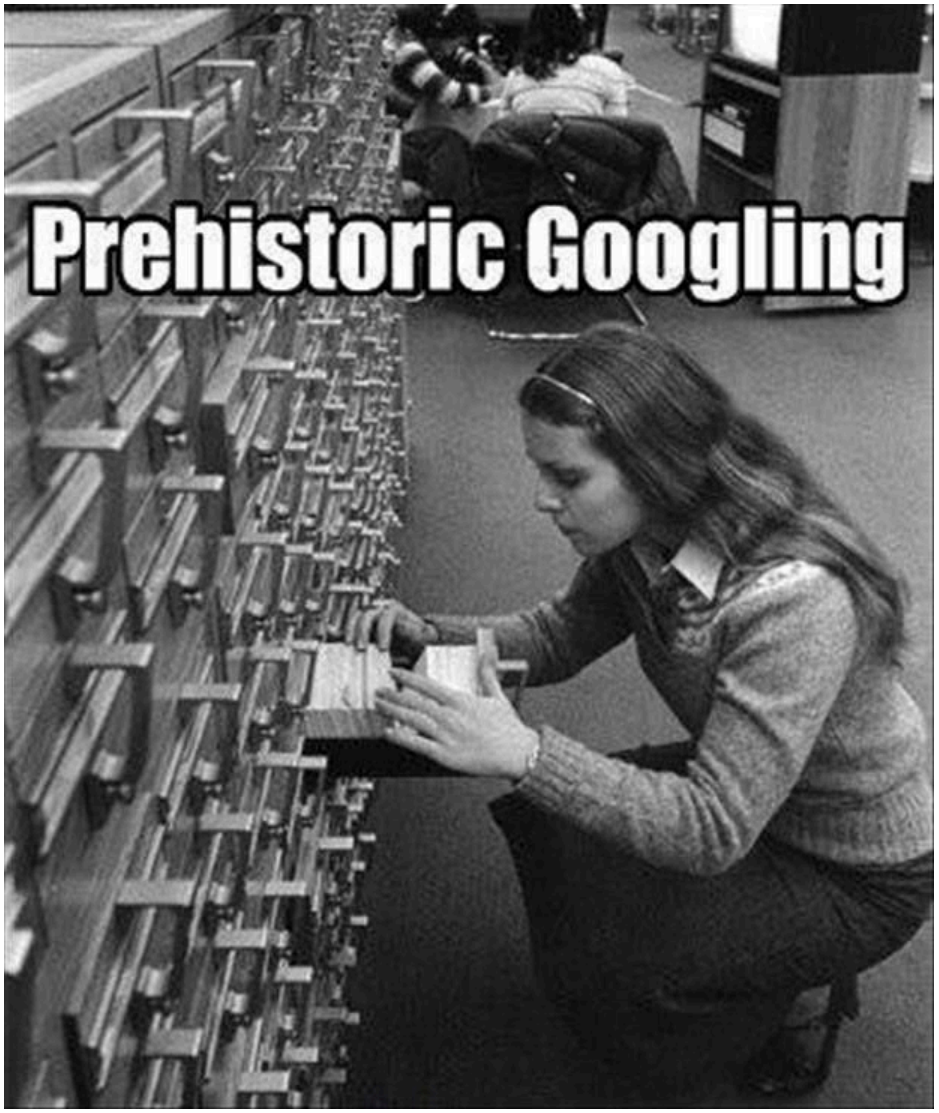
CS 492: Operating Systems

File Systems

Instructor: Iraklis Tsekourakis

Email: itsekour@stevens.edu

Why File Systems?



File Systems (1)

Essential requirements for long-term information storage:

1. It must be possible to store a very large amount of information.
2. Information must survive termination of process using it.
3. Multiple processes must be able to access information concurrently.

File Systems (2)

Think of a disk as a linear sequence of fixed-size blocks and supporting two operations:

1. Read block *k*.
2. Write block *k*

File Systems (3)

Questions that quickly arise:

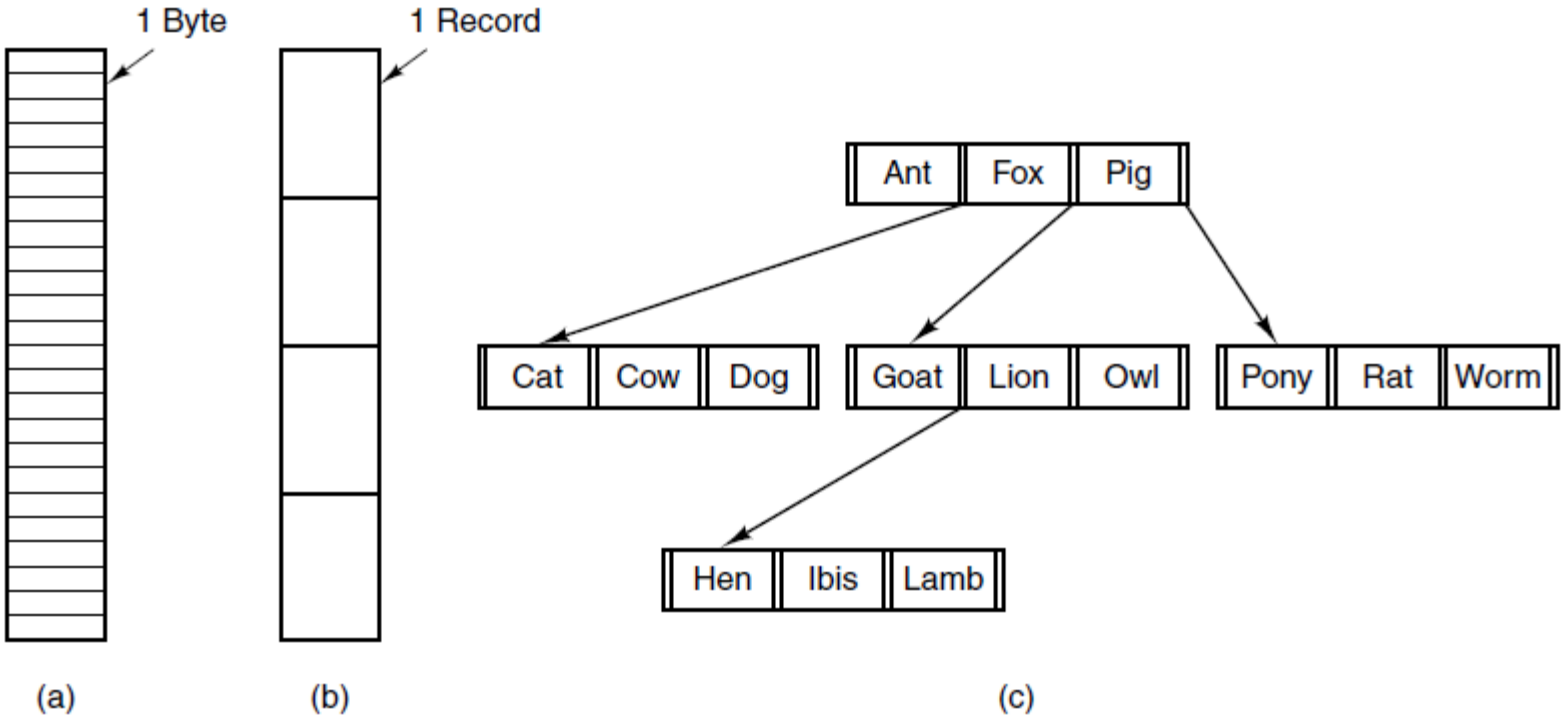
1. How do you find information?
2. How do you keep one user from reading another user's data?
3. How do you know which blocks are free?

File Naming

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

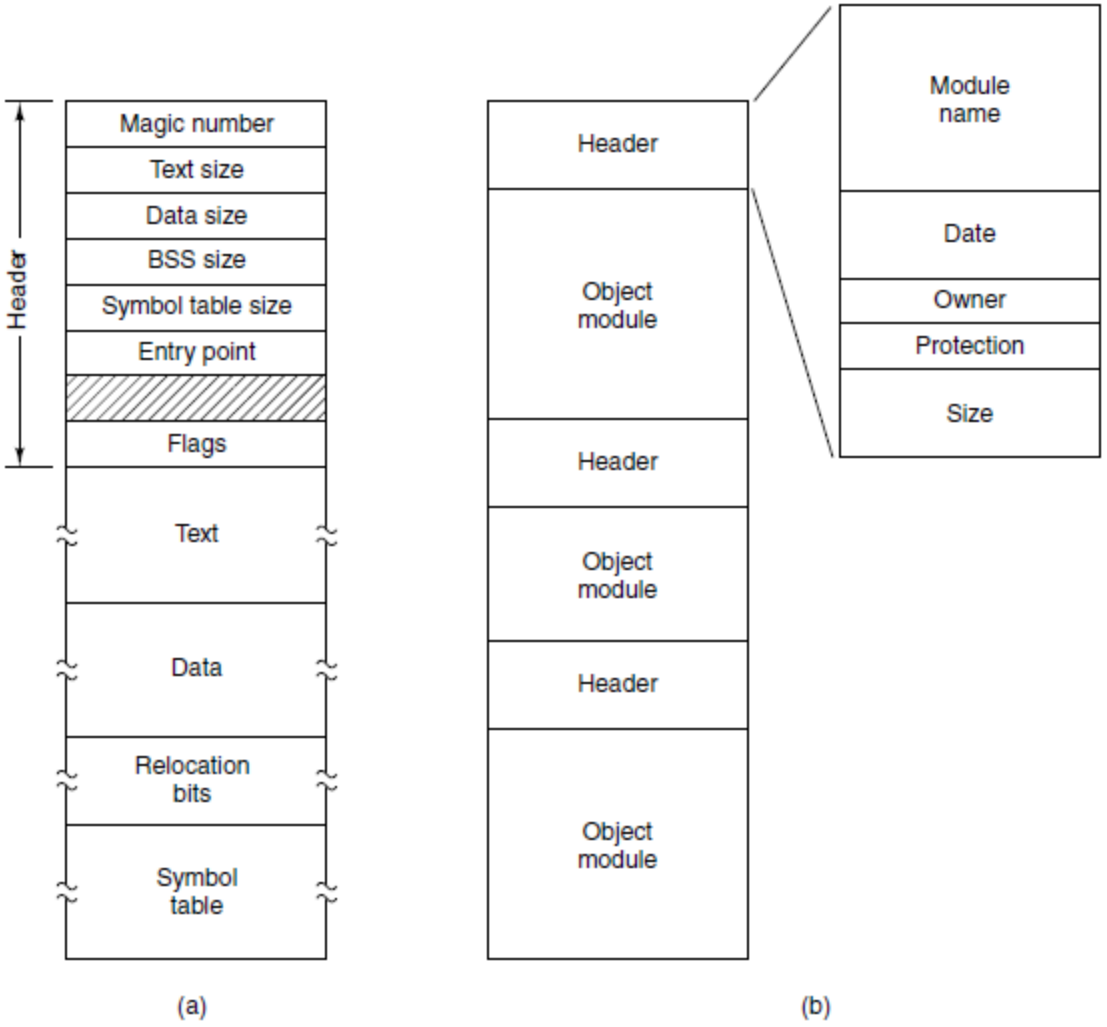
Some typical file extensions.

File Structure



Three kinds of files. (a) Byte sequence.
(b) Record sequence. (c) Tree.

File Types



(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

Some possible file attributes.

Question

In Windows, when a user double clicks on a file listed by Windows Explorer, a program is run and given that file as a parameter. List two different ways the operating system could know which program to run.

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

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 */
```

A simple program to copy a file.

Example Program Using File System Calls (2)

```
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 */

/* 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 */
```

A simple program to copy a file.

Example Program Using File System Calls (3)

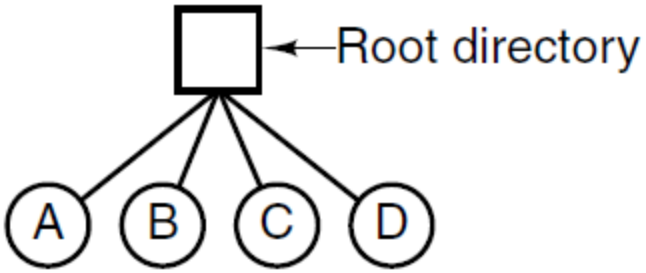
```
        /* Copy loop */
        while (TRUE) {
                rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
        if (rd_count <= 0) break;                         /* if end of file or error, exit loop */
                wt_count = write(out_fd, buffer, rd_count); /* write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

Question

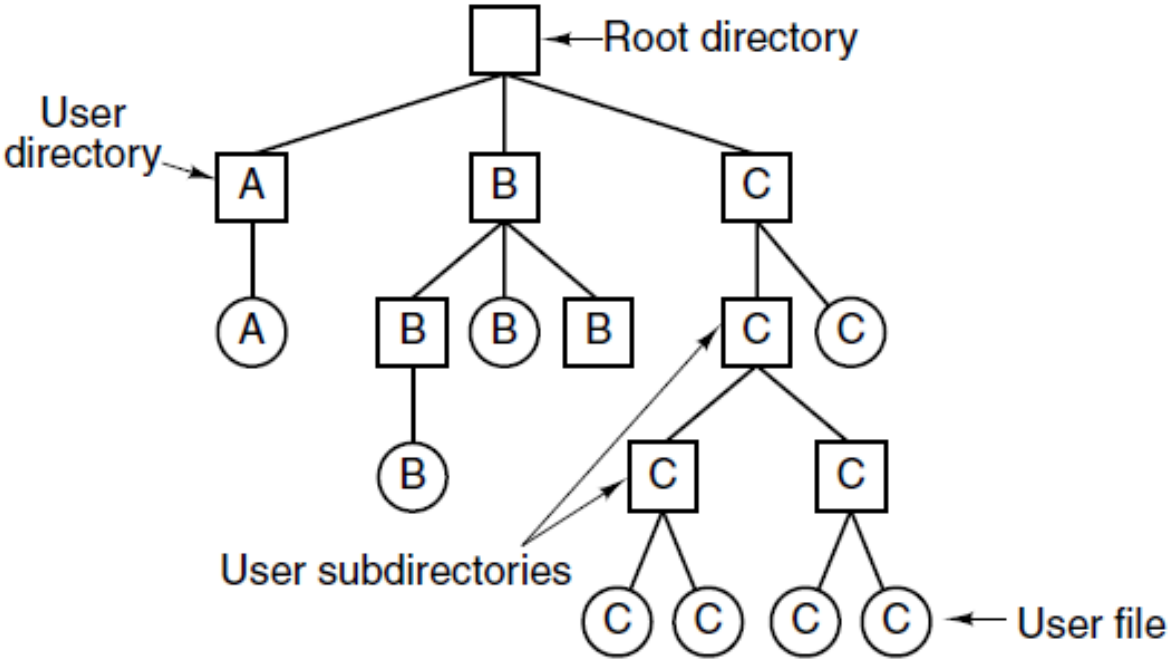
In some systems it is possible to map part of a file into memory. What restrictions must such systems impose? How is this partial mapping implemented?

Single-Level Directory Systems



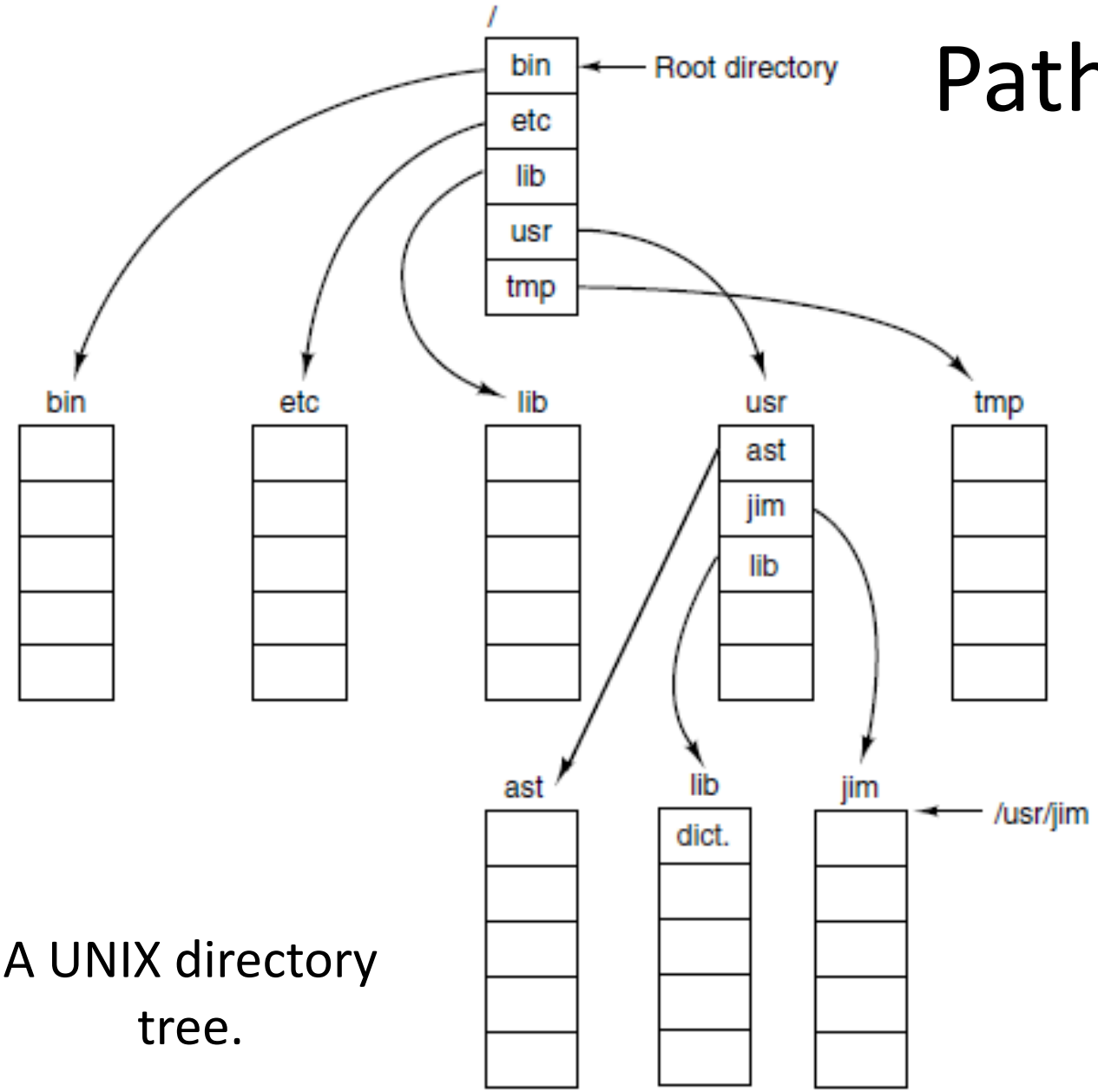
A single-level directory system containing four files.

Hierarchical Directory Systems



A hierarchical directory system.

Path Names



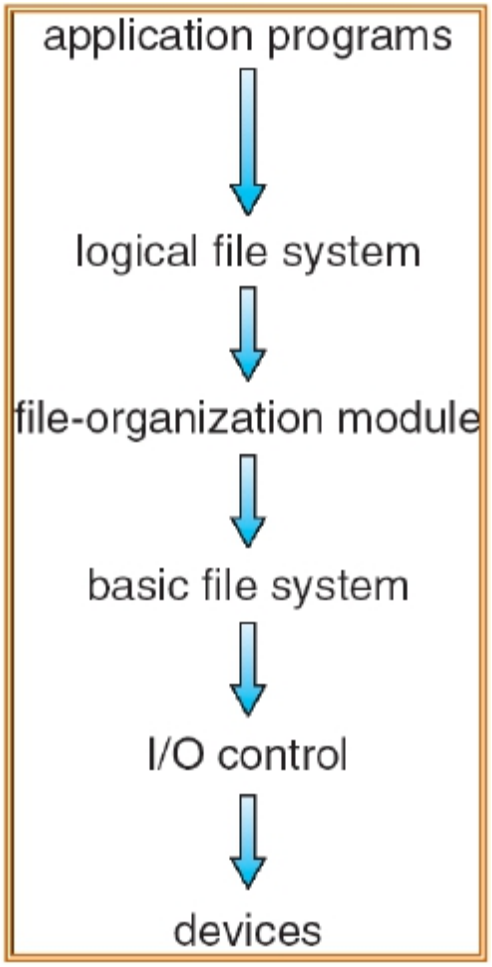
A UNIX directory tree.

Directory Operations

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

Implementation of File System

File-System Structure

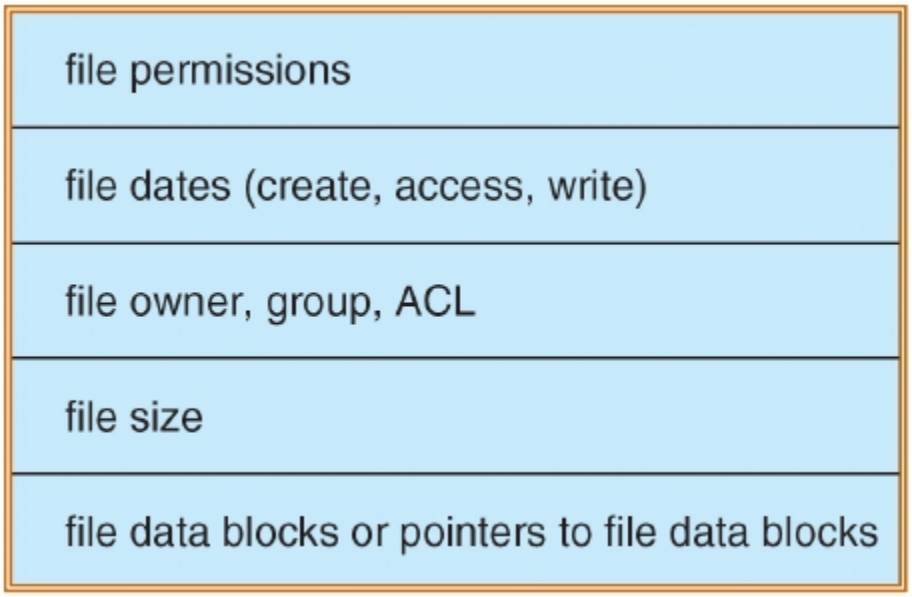


Layered file system structure

- File structure concept:
  - A logical storage unit
  - A collection of related information
- **File system** implements file structures
  - File system resides on secondary storage (disks)
  - File system structure is organized into layers

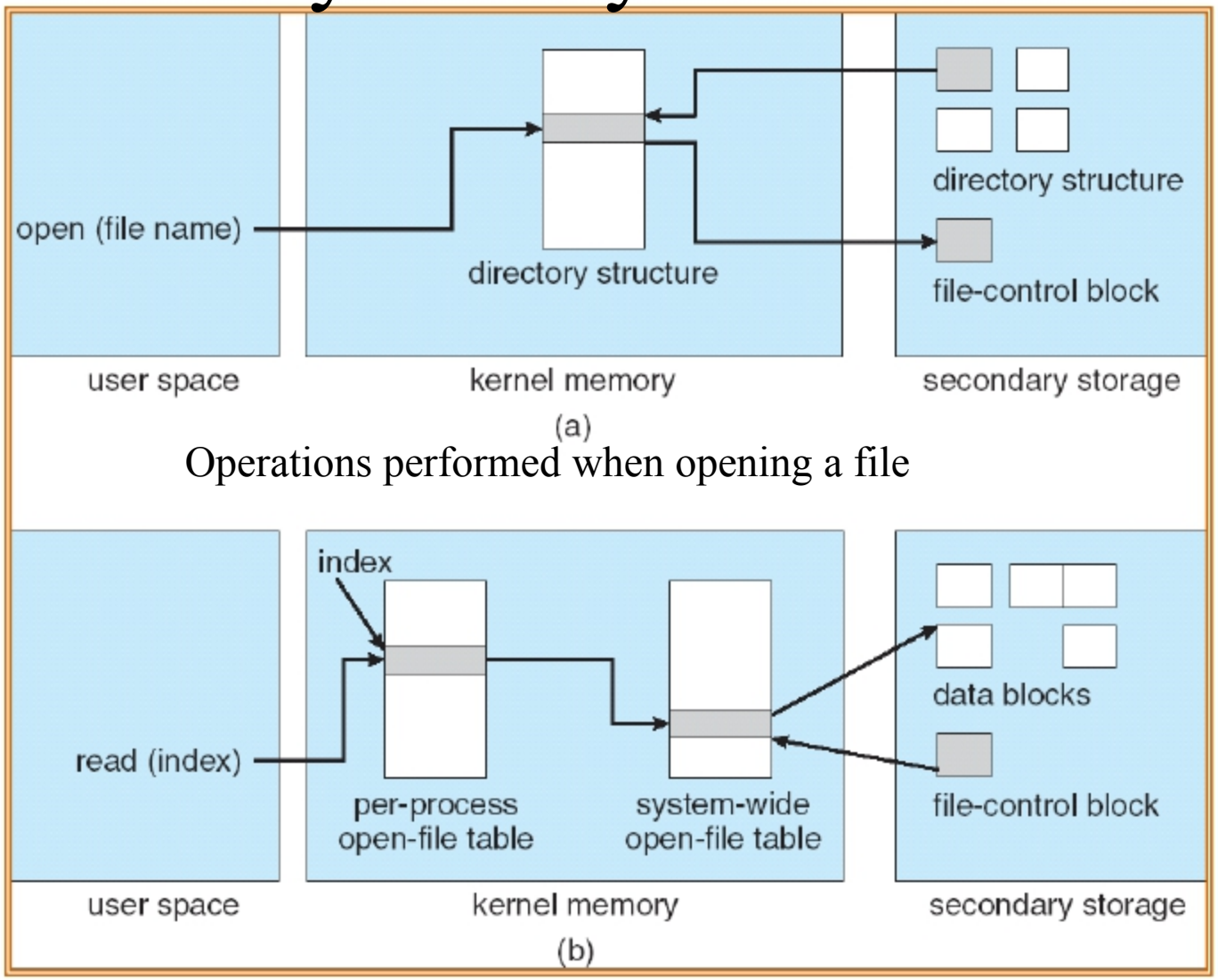
File-Control Blocks

- The logical file system maintains structures consisting of information about a file: **file-control block** (FCB)



Typical FCB

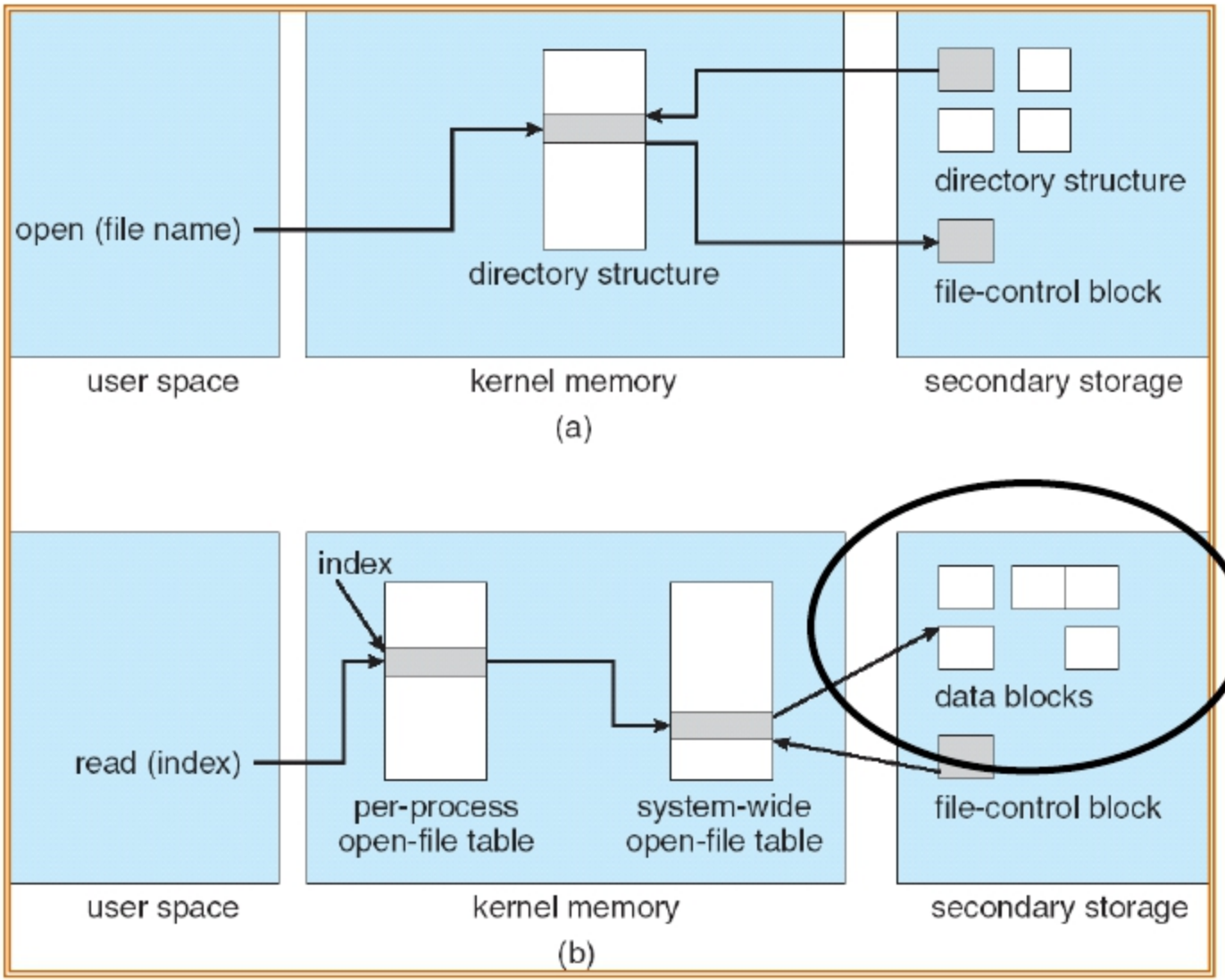
In-Memory File System Structures



Operations performed when opening a file

Operations performed when reading the file after opening

Allocation Methods



How to efficiently allocate data blocks for files on disk?

Allocation Methods

Three allocation methods:

1. **Contiguous allocation**
2. **Linked allocation**
3. **Indexed allocation**