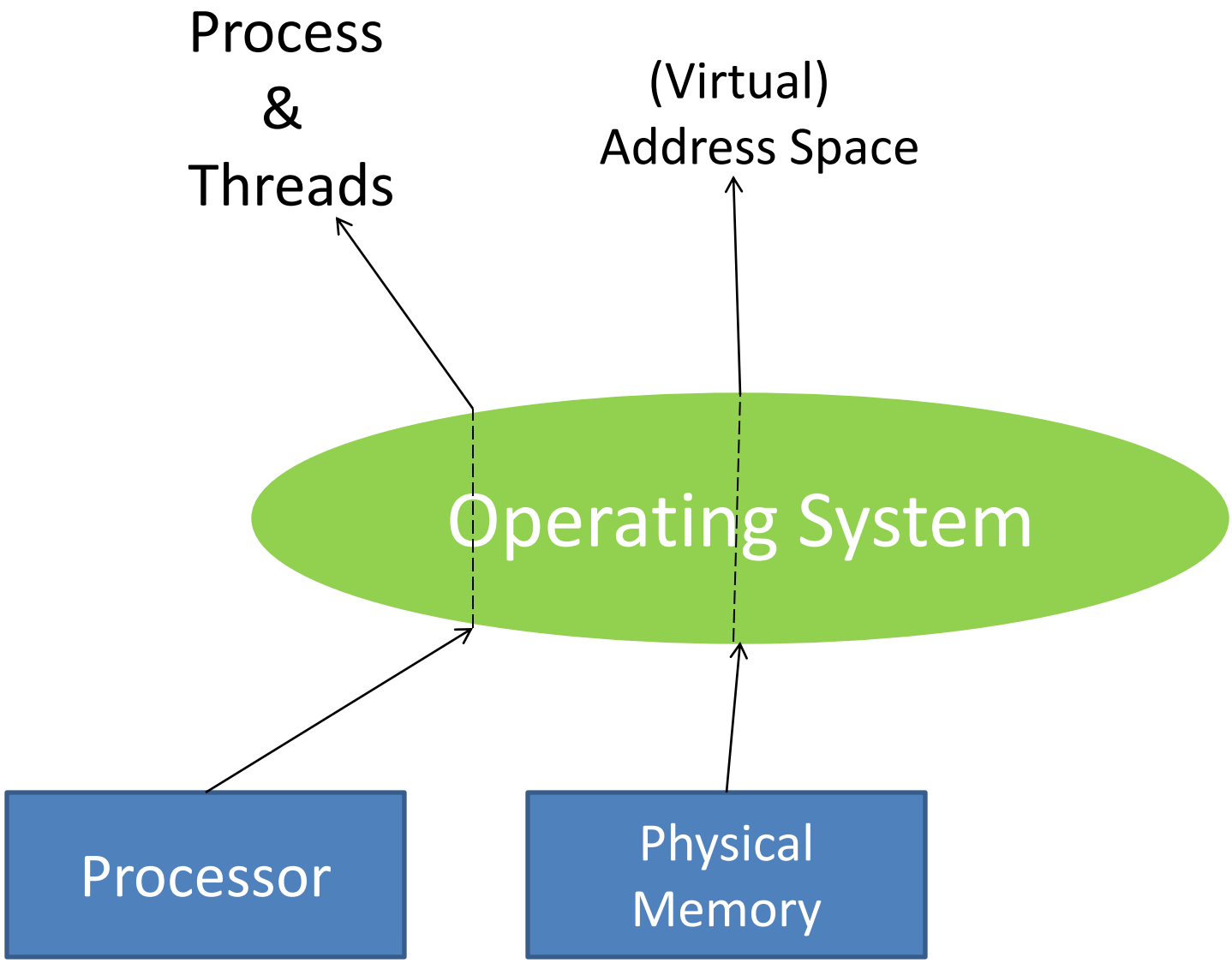
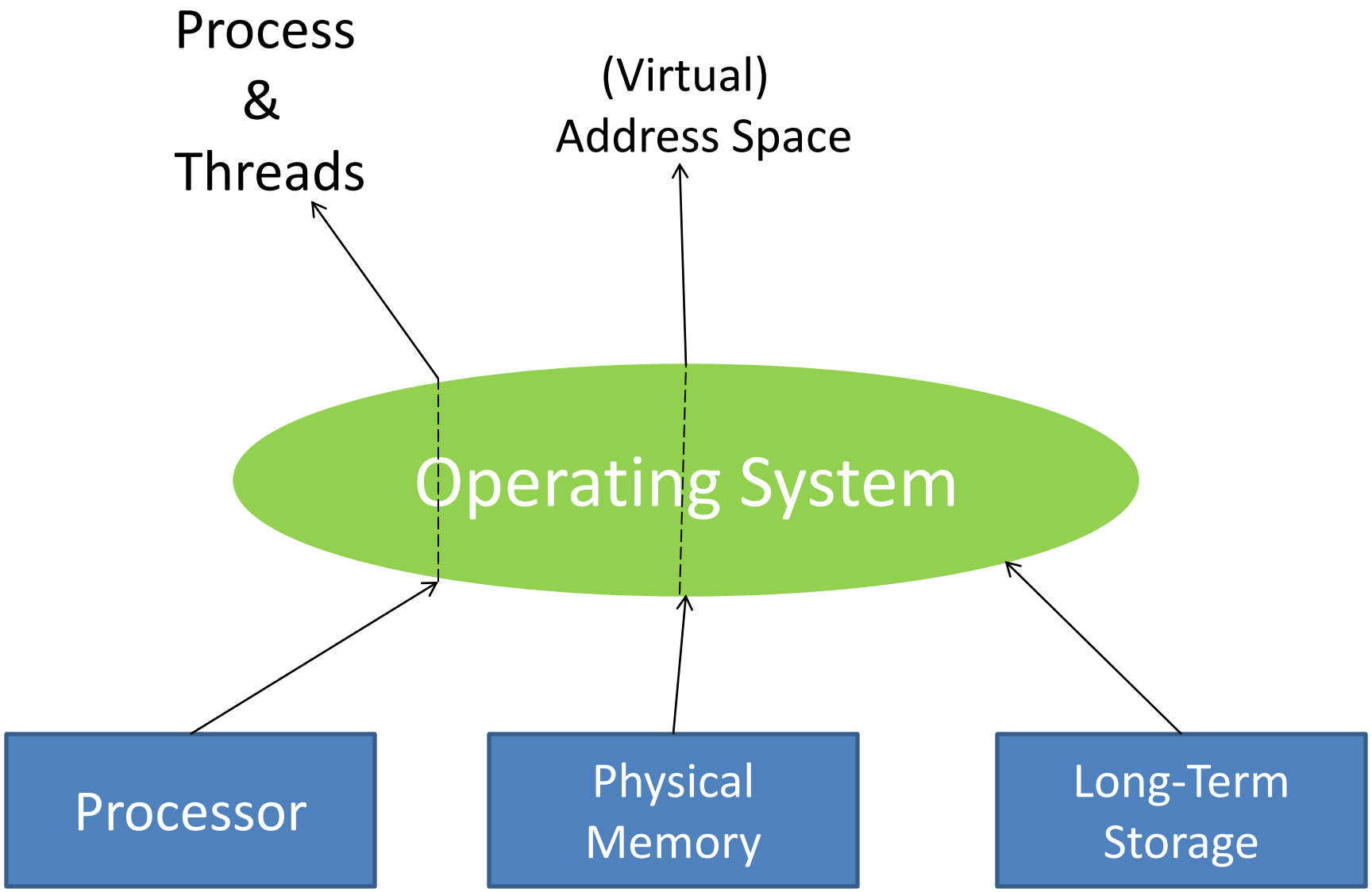
CSCI-GA.2250-001

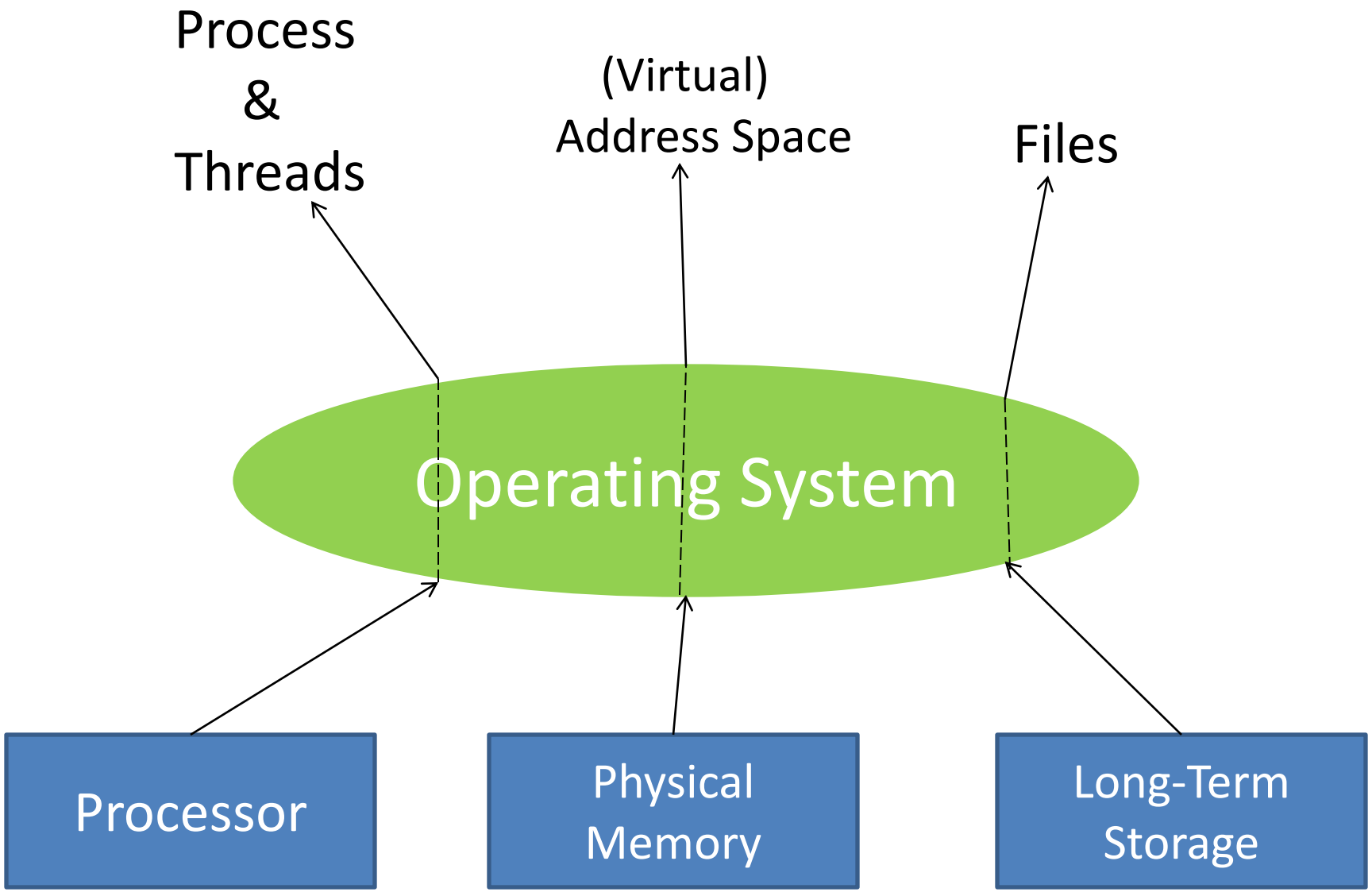
Operating Systems

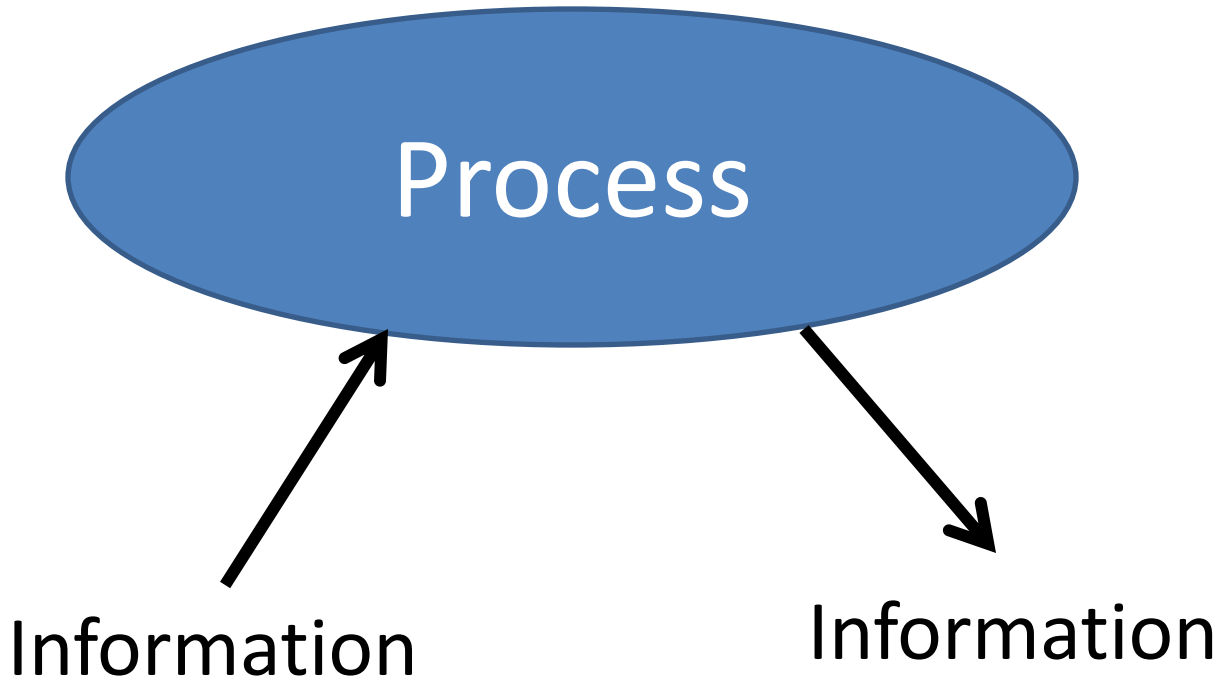
Lecture 7: File Systems I

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com


Process
&
Threads
(Virtual)
Address Space
Operating System
Processor
Physical
Memory


Process
&
Threads
(Virtual)
Address Space
Operating System
Processor
Physical
Memory
Long-Term
Storage


Process
&
Threads
(Virtual)
Address Space
Files
Operating System
Processor
Physical
Memory
Long-Term
Storage



Is it OK to keep this information only in the process address space?

Shortcomings of Process Address Space

- Virtual address space may not be enough storage for all information
- Information is lost when process terminates, is killed, or computer crashes.
- Multiple processes may need the information at the same time

Requirements for Long-term Information Storage

- Store very large amount of information
- Information must survive the termination of the process using it
- Multiple processes must be able to access the information concurrently

Files

- Data collections created by users
- The File System is one of the most important parts of the OS to a user
- Desirable properties of files:

Long-term existence

- files are stored on disk or other secondary storage and do not disappear when a user logs off

Sharable between processes

- files have names and can have associated access permissions that permit controlled sharing

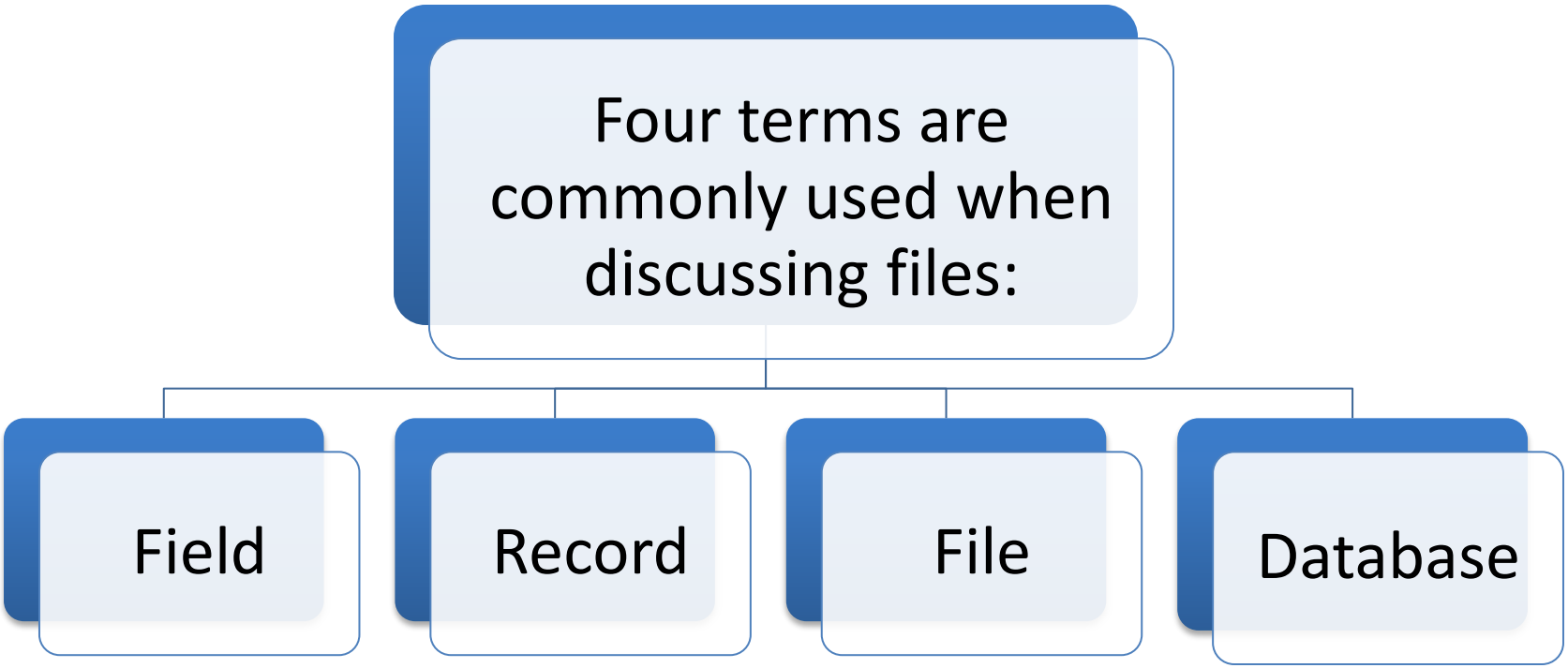
Structure

- files can be organized into hierarchical or more complex structure to reflect the relationships among files

Files

- Logical units of information created by processes
- Used to model disks instead of RAM
- Information stored in files must be persistent (i.e. not affected by processes creation and termination)
- Managed by OS
- The part of OS dealing with files is known as the file system

File Structure



Structure Terms

Field

- basic element of data
- contains a single value
- fixed or variable length

Record

- collection of related fields that can be treated as a unit by some application program
- fixed or variable length

Database

- collection of related data
- relationships among elements of data are explicit
- designed for use by a number of different applications
- consists of one or more types of files

File

- collection of similar records
- treated as a single entity
- may be referenced by name
- access control restrictions usually apply at the file level

Issues

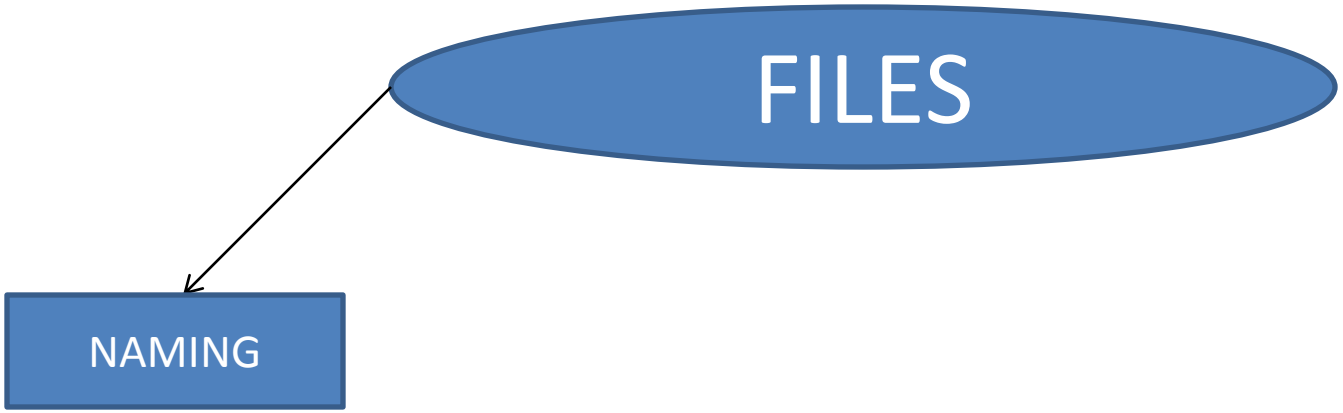
- How do you find information?
- How do you keep one user from reading another user's data?
- How do you know which blocks are free?

Files from The User's point of View

- Files
  - Naming
  - Structure
  - Types
  - Access
  - Attributes
  - Operations
- Directories
  - Single-level
  - Hierarchical
  - Path names
  - Operations

File Systems

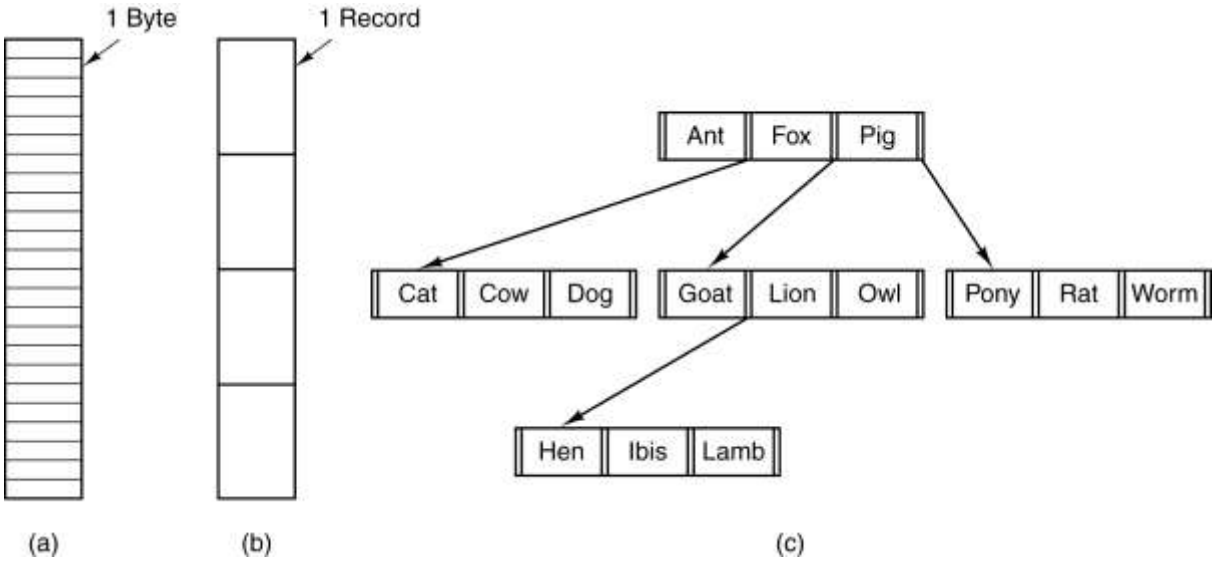
- Provide a means to store data organized as files as well as a collection of functions that can be performed on files
- Maintain a set of attributes associated with the file
- Typical operations include:
  - Create
  - Delete
  - Open
  - Close
  - Read
  - Write



- Shields the user from details of file storage.
- In general, files continue to exist even after the process that creates them terminates.

FILES

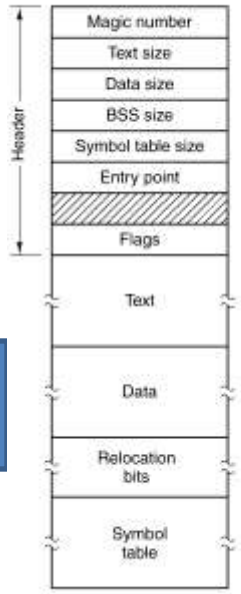
- NAMING
- Structure

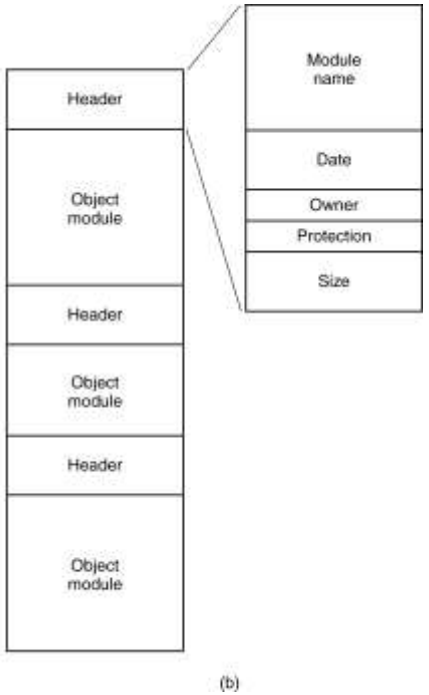


OS view Historical Still Used in Some Mainframes

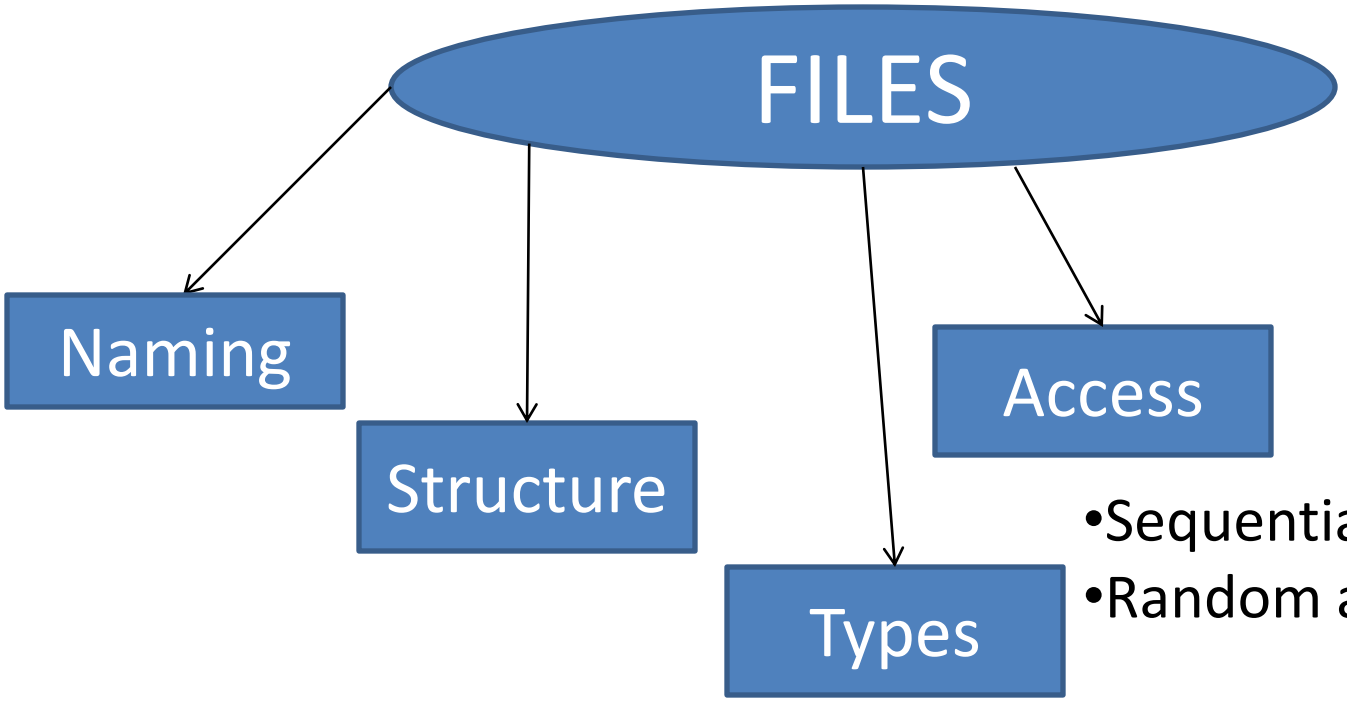
FILES

- Naming
- Structure
- Types

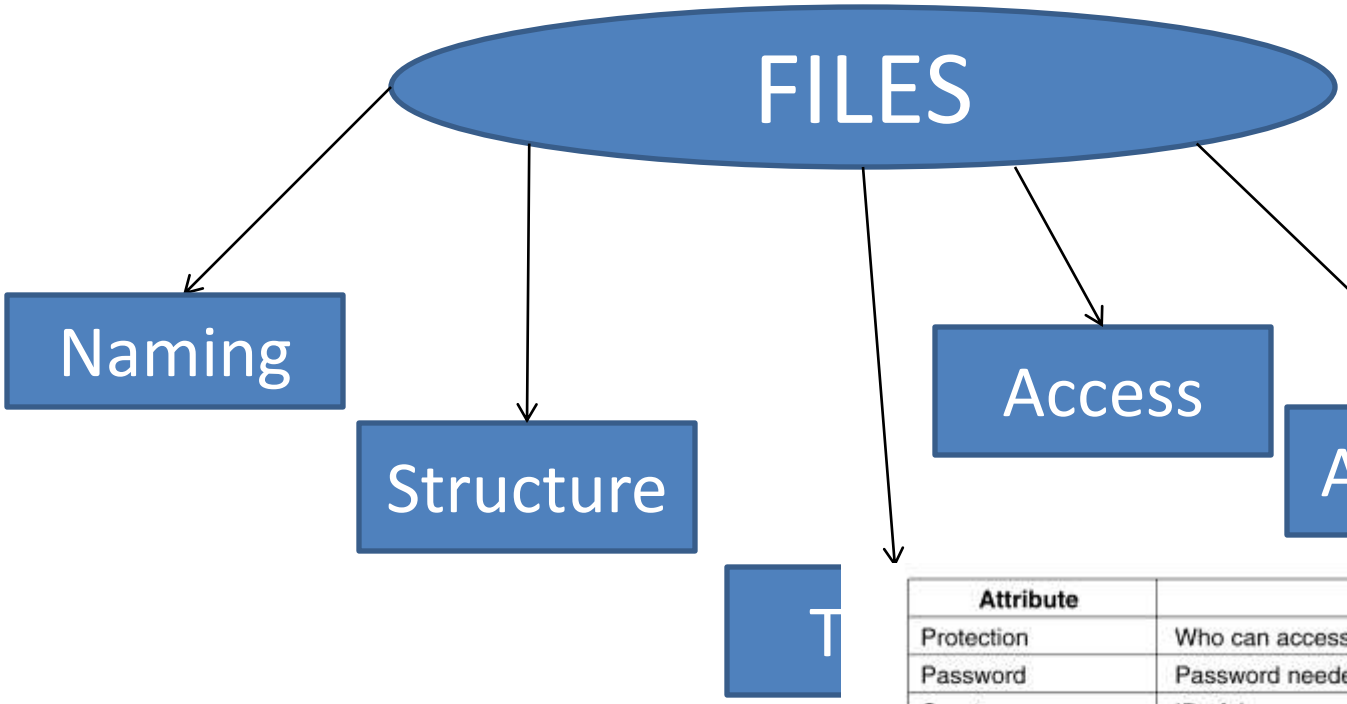




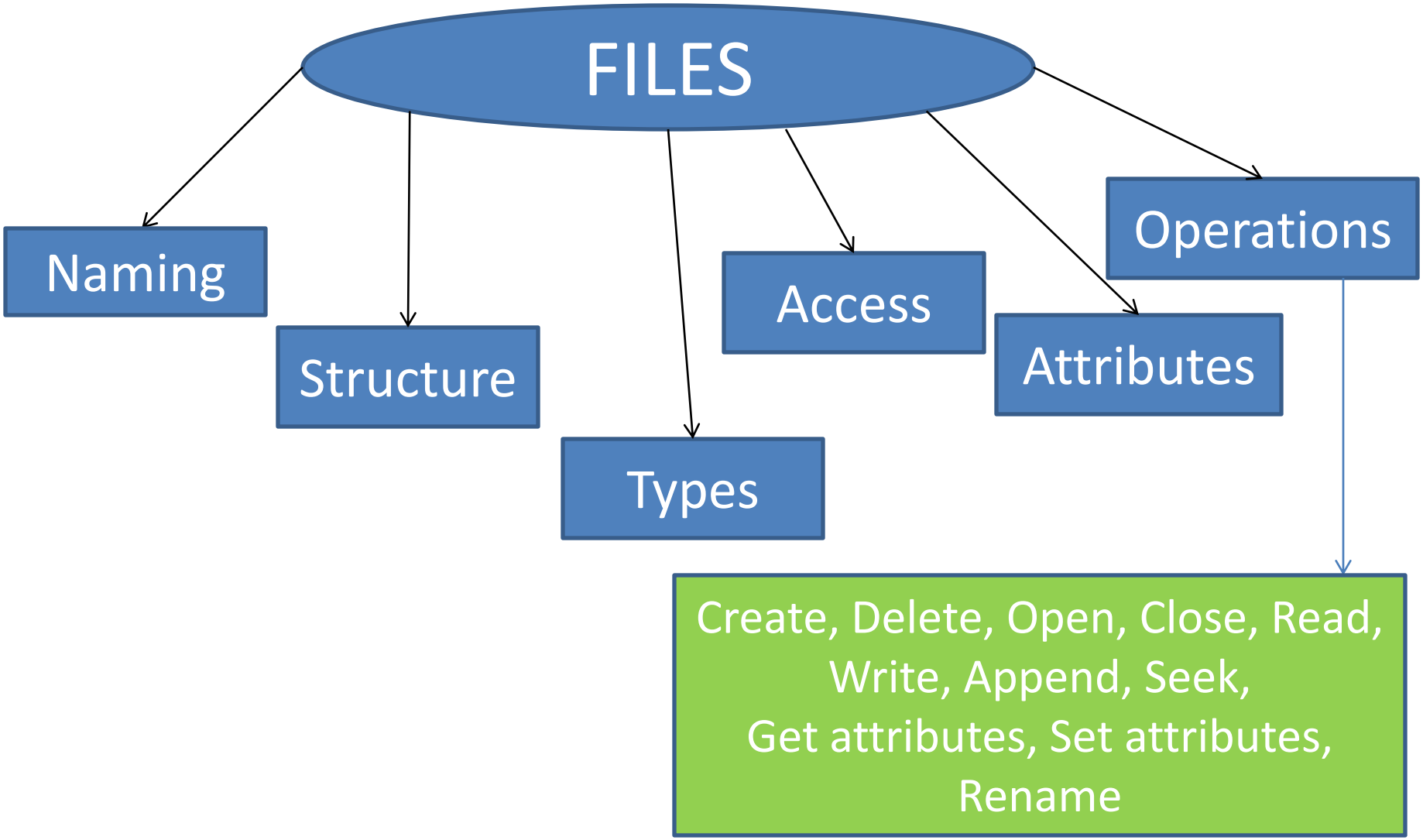
- Regular files
  - ASCII
  - Binary
- Character special
  - to model serial devices (printers, networks, …)
- Block special
  - to model disks



- Sequential
- Random access

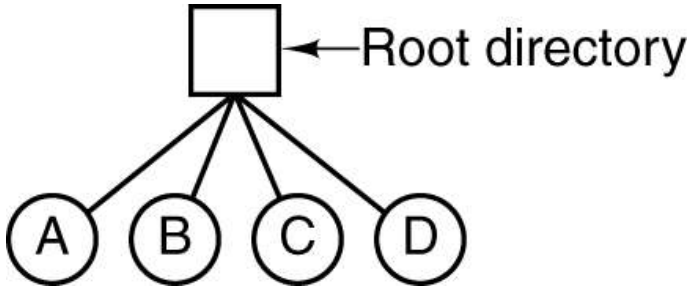


Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 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


FILES
Naming
Structure
Types
Access
Attributes
Operations
Create, Delete, Open, Close, Read,
Write, Append, Seek,
Get attributes, Set attributes,
Rename

```
/* 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 */

    /* 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: Single-Level Directory Systems



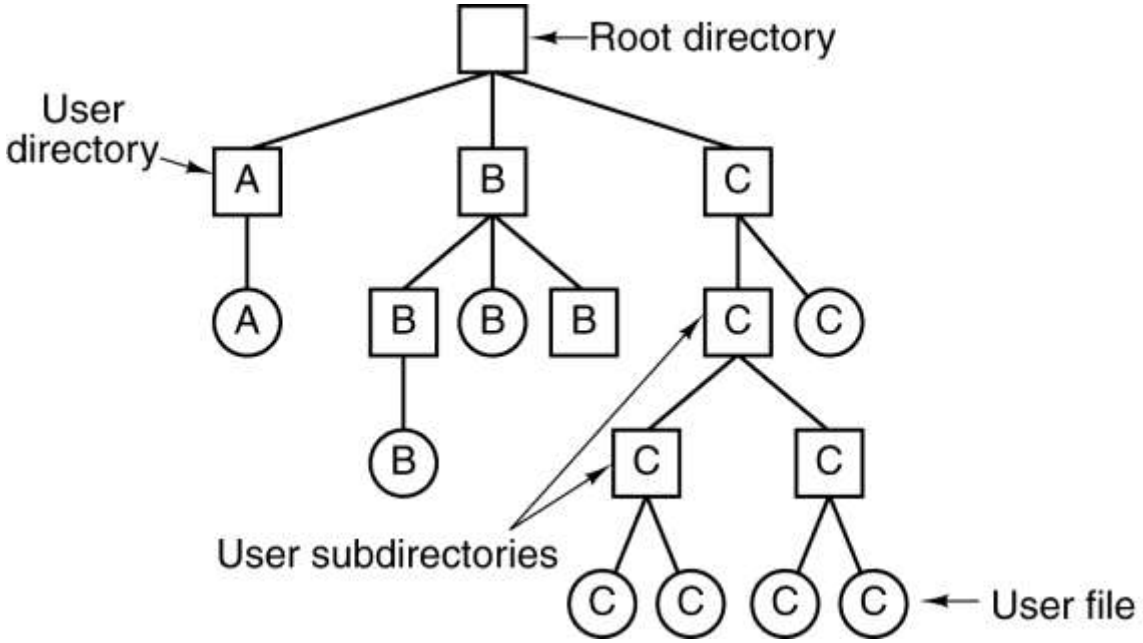
+ Simplicity

-Not adequate for large number of files.

Used in simple embedded devices

Directories: Hierarchical Directory Systems

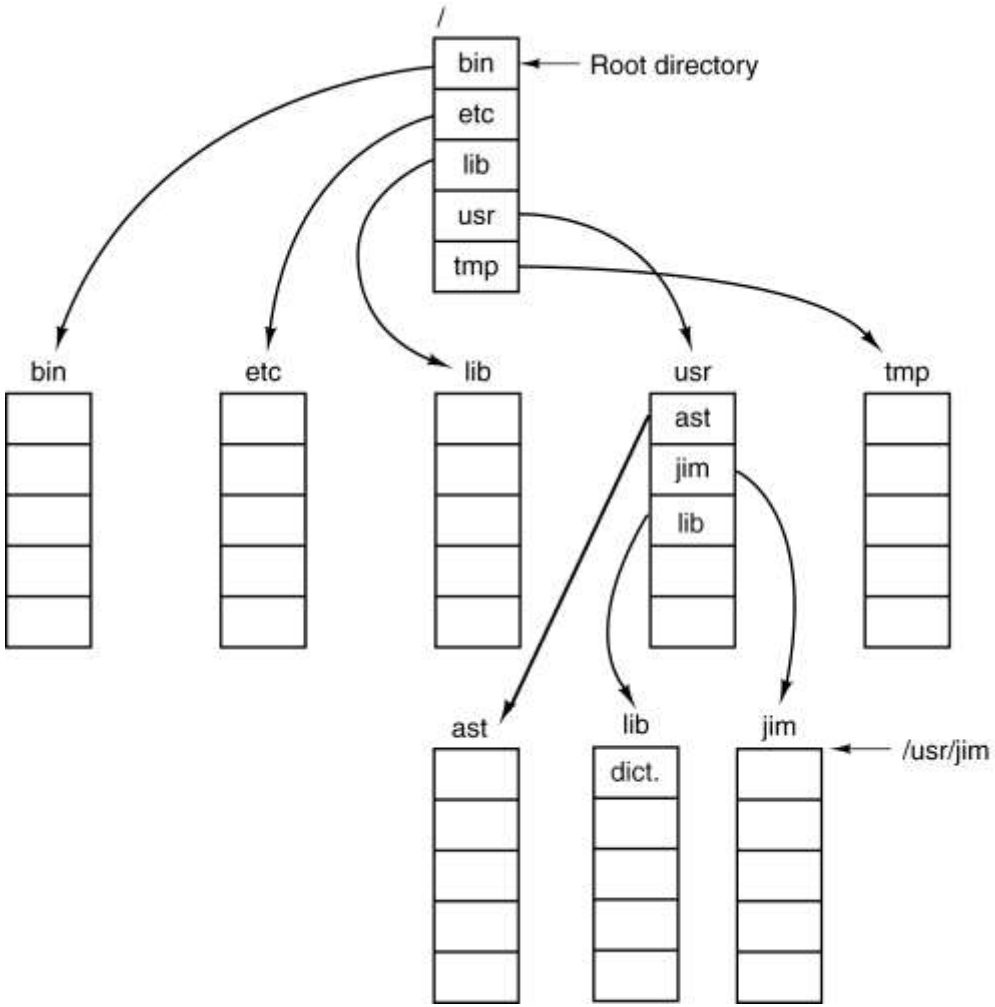
- Group related files together
- Tree of directories



Directories: Path Names

- Needed when directories are used
- Absolute path names
  - Always start at the root
  - A path from the root to the specified file
  - The first character is the separator
- Relative path names
  - Relative to the working directory
  - Each process has its own working directory

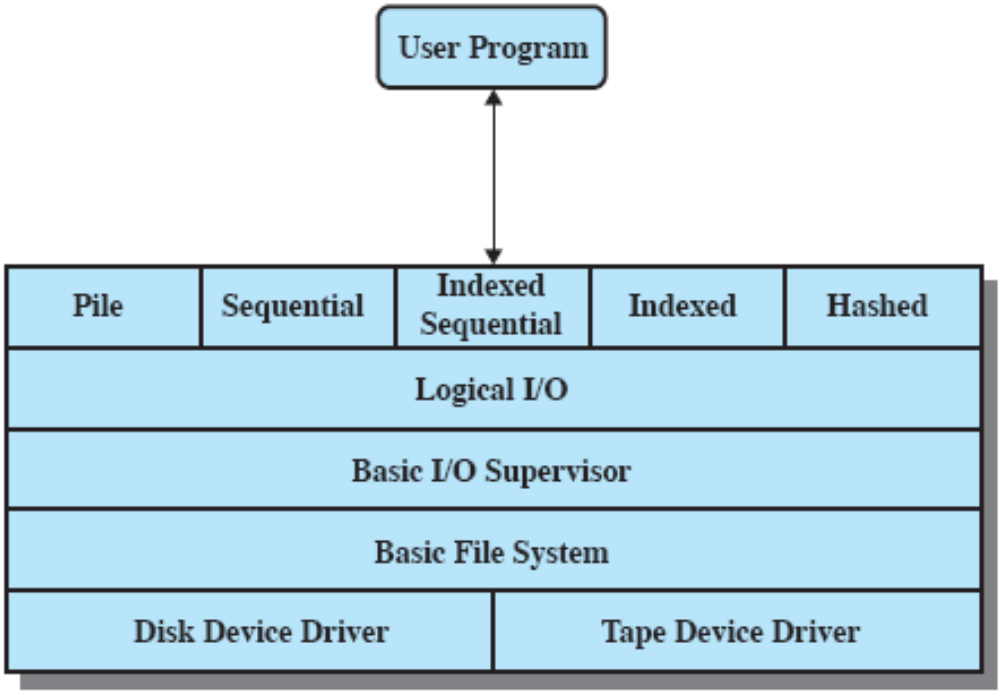
Directories: Path Names



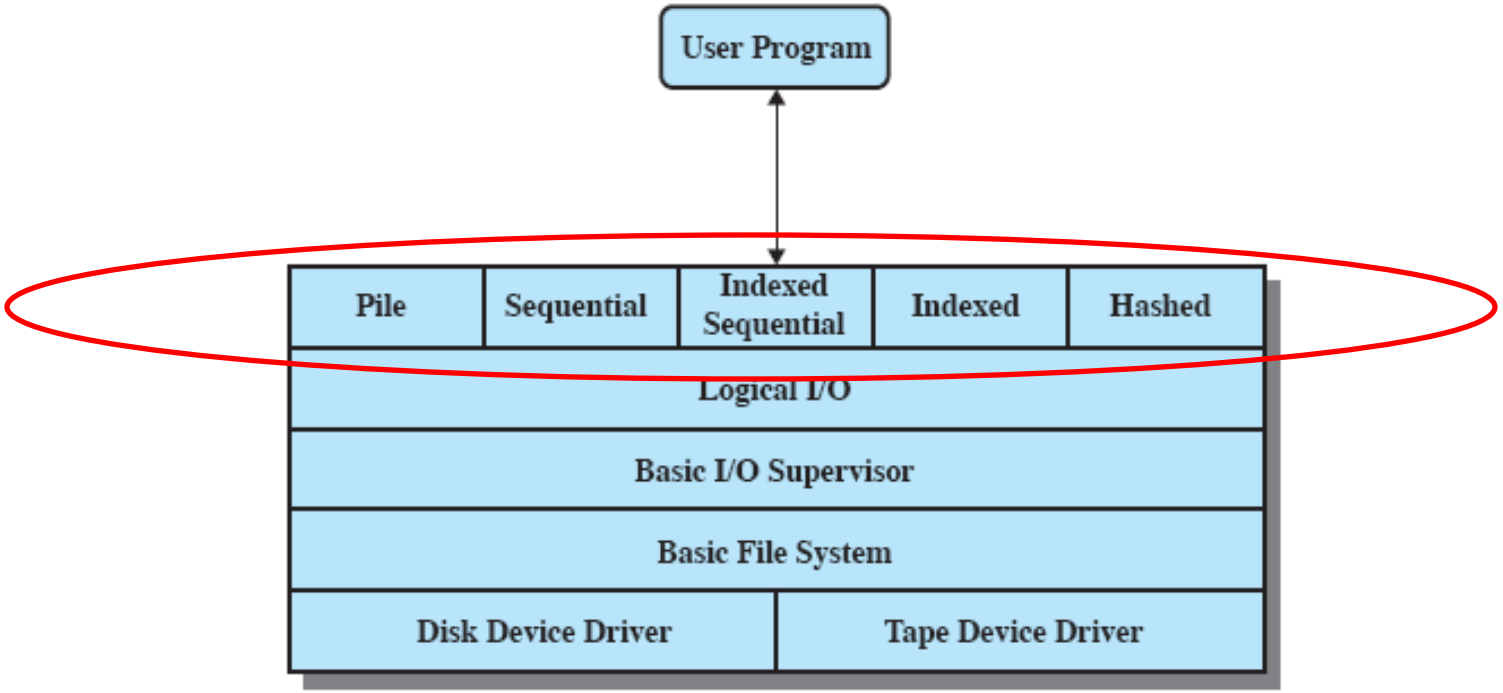
Directories: Operations

- More variations among OSes than file operations
- Examples (from UNIX):
  - Create, deleted
  - Opendir, closedir
  - Readdir
  - Rename
  - Link, unlink

Typical Organization

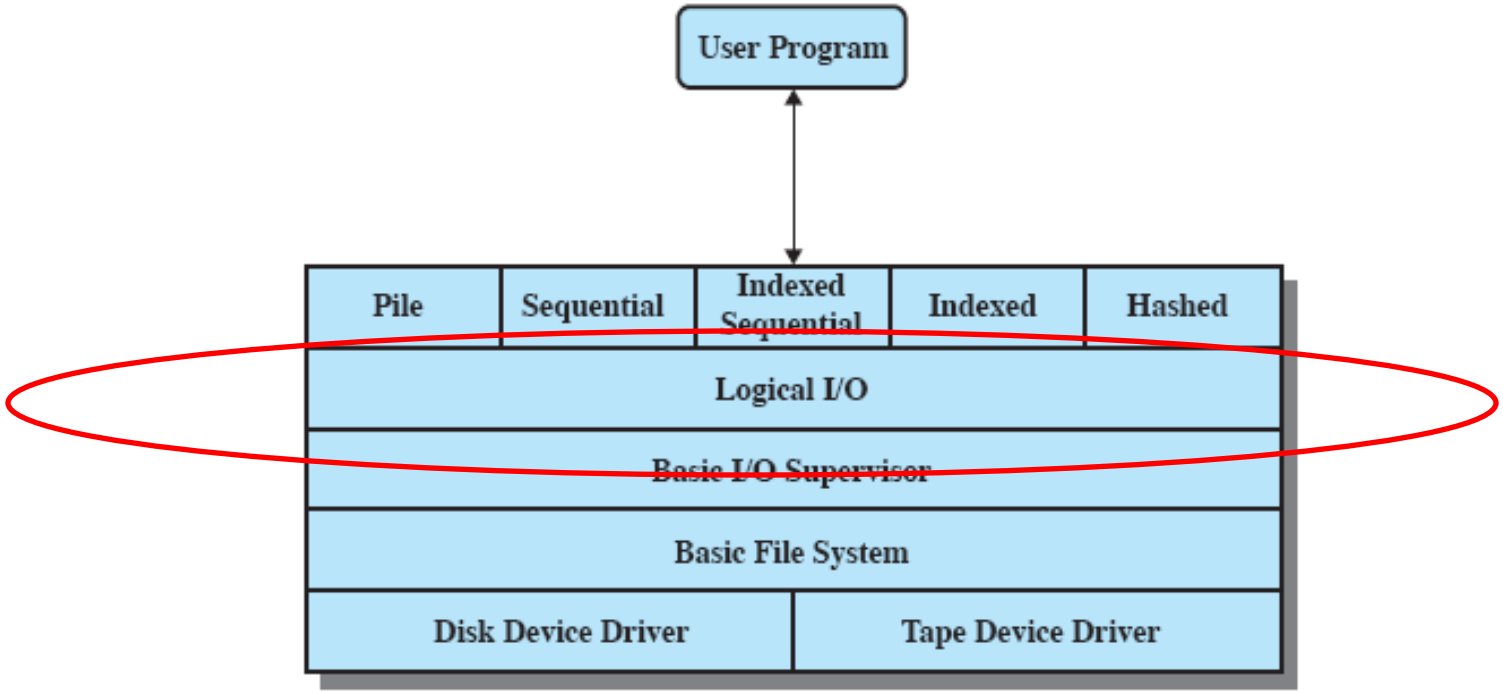


Typical Organization



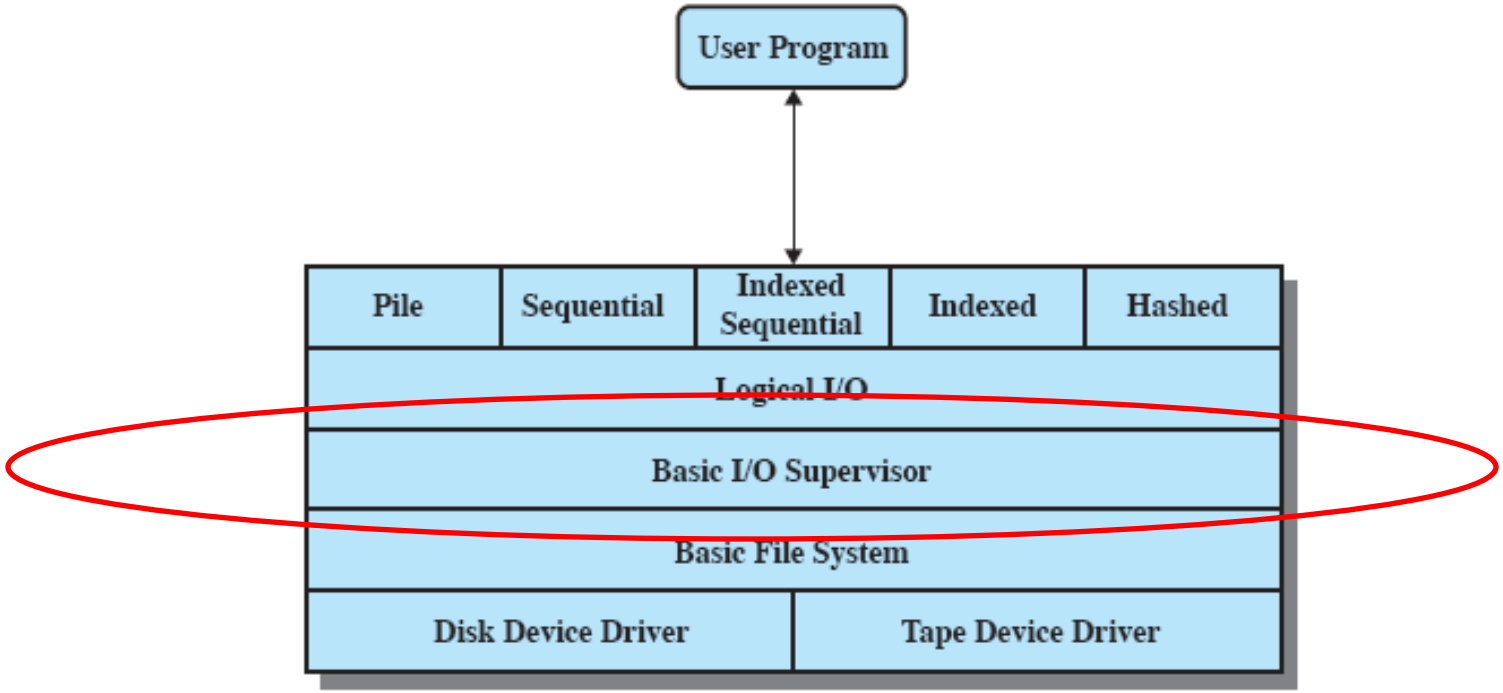
- The level closest to application and provides the access method to file system,
- The figure above shows some access methods, each of which reflects different file structures and different ways of accessing them.

Typical Organization



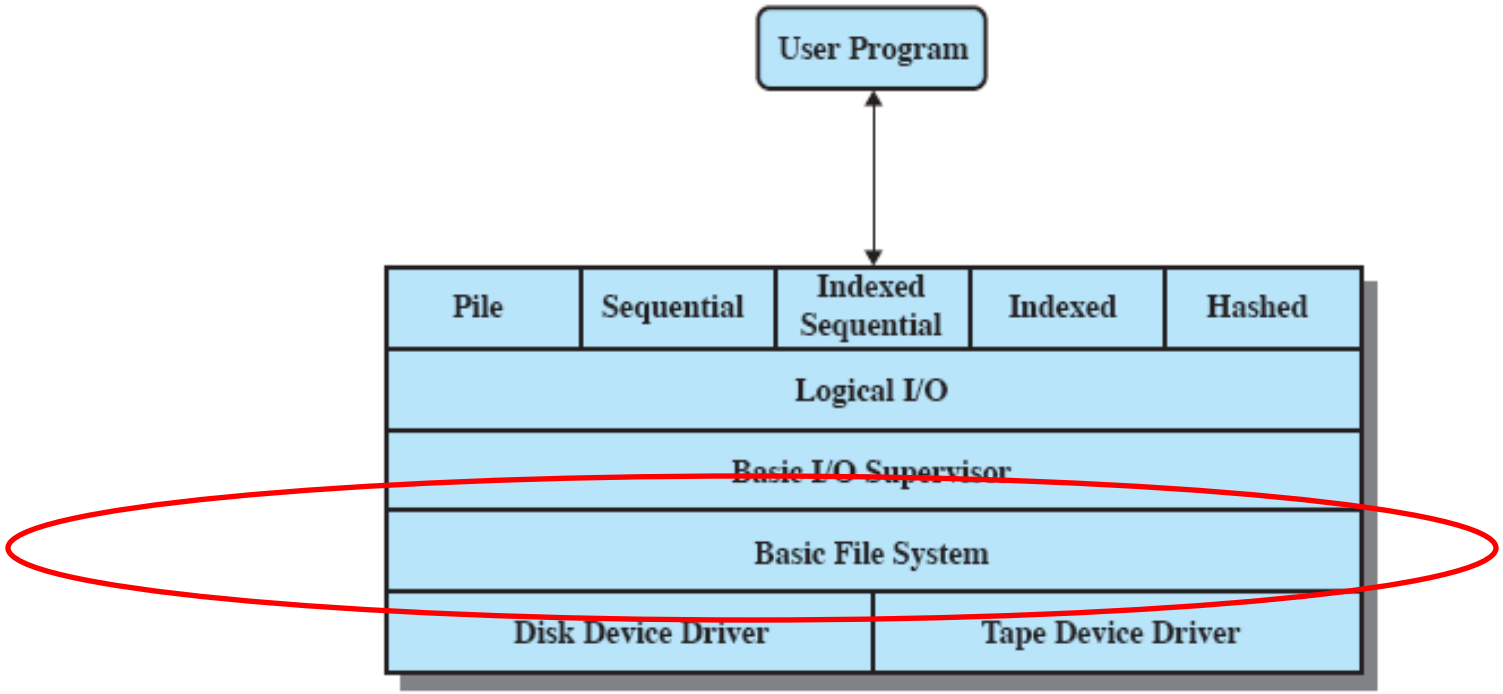
- Enables users and application to access records.
- Deals with file records

Typical Organization

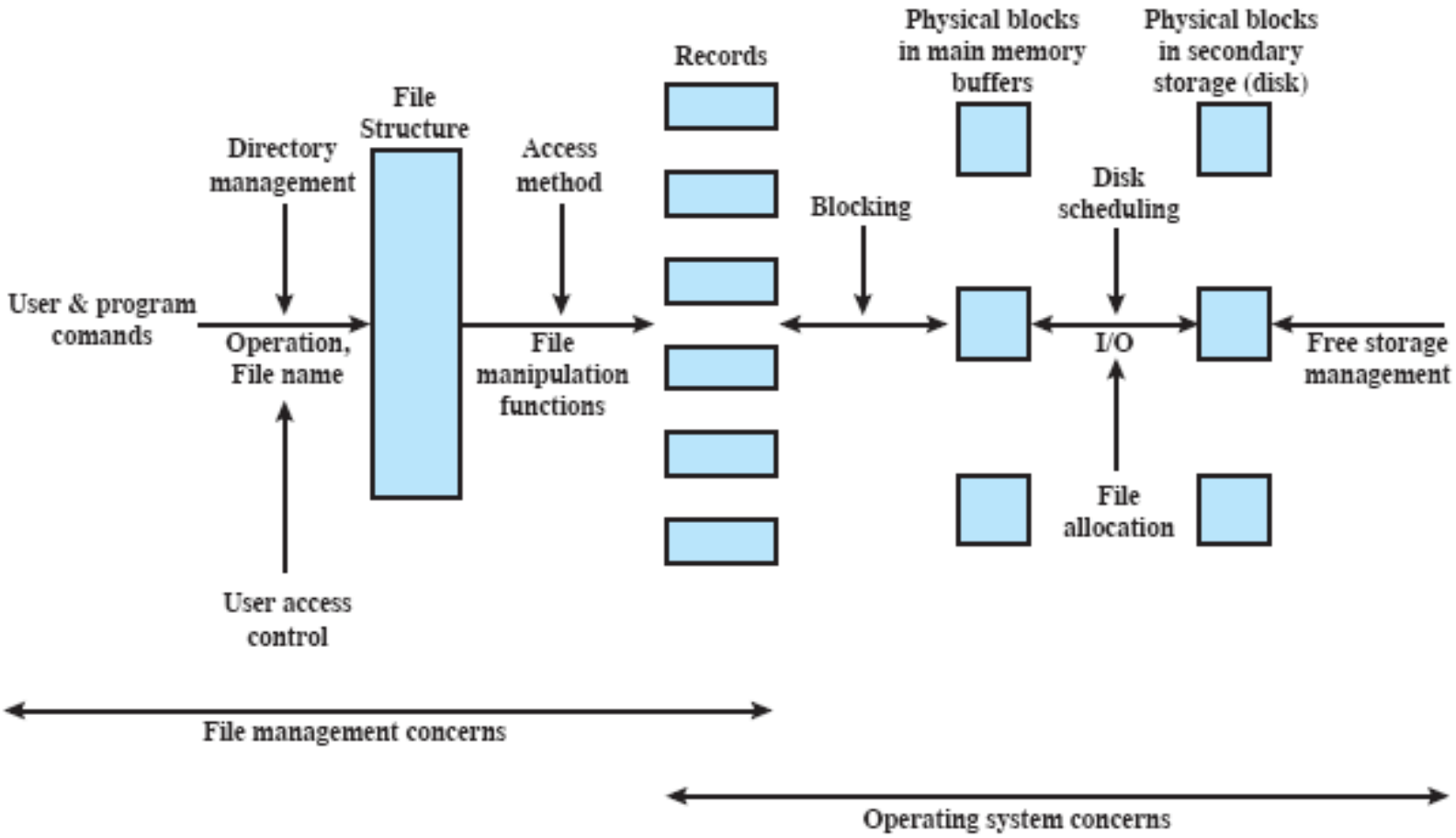


- Responsible for all File I/O initiation and termination
- Selects the device on which File I/O is to be performed.
- Concerned with scheduling disk accesses to optimize performance.
- I/O buffers are assigned at that level.

Typical Organization



- Primary interface with the environment outside the computer system.
- Deals with blocks of data
- Concerned with placement of those blocks in secondary storage
- Concerned with buffering those blocks in main memory.
- Does not understand the structure of data of files involved.


Physical blocks in main memory buffers
Physical blocks in secondary storage (disk)
Records
File Structure
Directory management
Access method
Blocking
Disk scheduling
User & program comands
Operation, File name
File manipulation functions
I/O
Free storage management
File allocation
User access control
File management concerns
Operating system concerns

Conclusions

- Files are OS abstraction for storage, same as address space is OS abstraction for physical memory, and processes (& threads) are OS abstraction for CPU.
- In this lecture we discussed files from user perspective. Next lecture we will discuss the implementation.