File Systems
— Chapter 10

2022年11月

薛哲

xuezhe@bupt.edu.cn

School of Computer Science (National Pilot Software Engineering School)

北京郵電大學

Part Four Storage Management

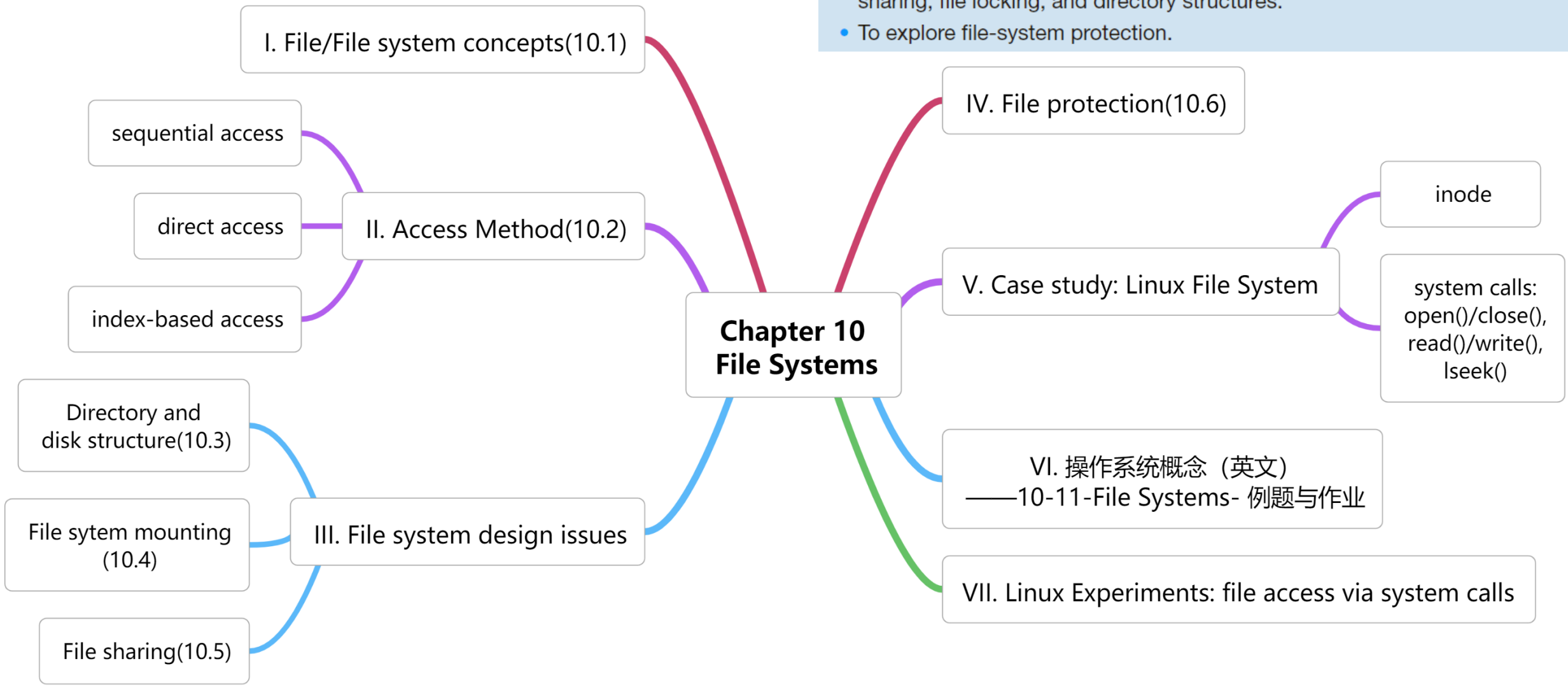
- Chapter 10, 11
  - File Systems
  - Implementing File Systems
- Chapter 12, 13
  - Mass-storage Structures
  - I/O Systems

Introduction

- Secondary storage management, *i.e. file management* , includes
  - data organization
    - files in secondary storage
  - data access
    - files access
- File systems provides two perspectives of secondary storage management
  - user-oriented (Chapter 10)
    - file system **interface**，also known as **logical file system**
    - /*用户看到的文件逻辑/组织结构，用户使用的文件访问方法
  - secondary-storage-oriented (Chapter 11)
    - file system **implementation**, also known as **physical file system**
    - /*文件在secondary storage上的存储、访问方式

CHAPTER OBJECTIVES

- To explain the function of file systems.
- To describe the interfaces to file systems. system call
- To discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures.
- To explore file-system protection.



10.1 File Concept

- **File** Definition(P456 in Edt.9)
  - a named collection of related information that is recorded on secondary storage, i.e. disk or SSD/Flash memory
  - commonly, representing programs and data
  - e.g. C++ source program file, executable file, text file
- **File Definition** in Microsoft Computer Dictionary
  - a complete, named collection of information, such as a program, a set of data used by a program, or a user created document
  - a file is the basic unit of storage that enable a computer to distinguish one set of information from another
  - a file is the “glue” that binds a conglomeration of instructions, numbers, words, or images into a coherent unit that a user can retrieve, change, delete, save, or sent to an output devices

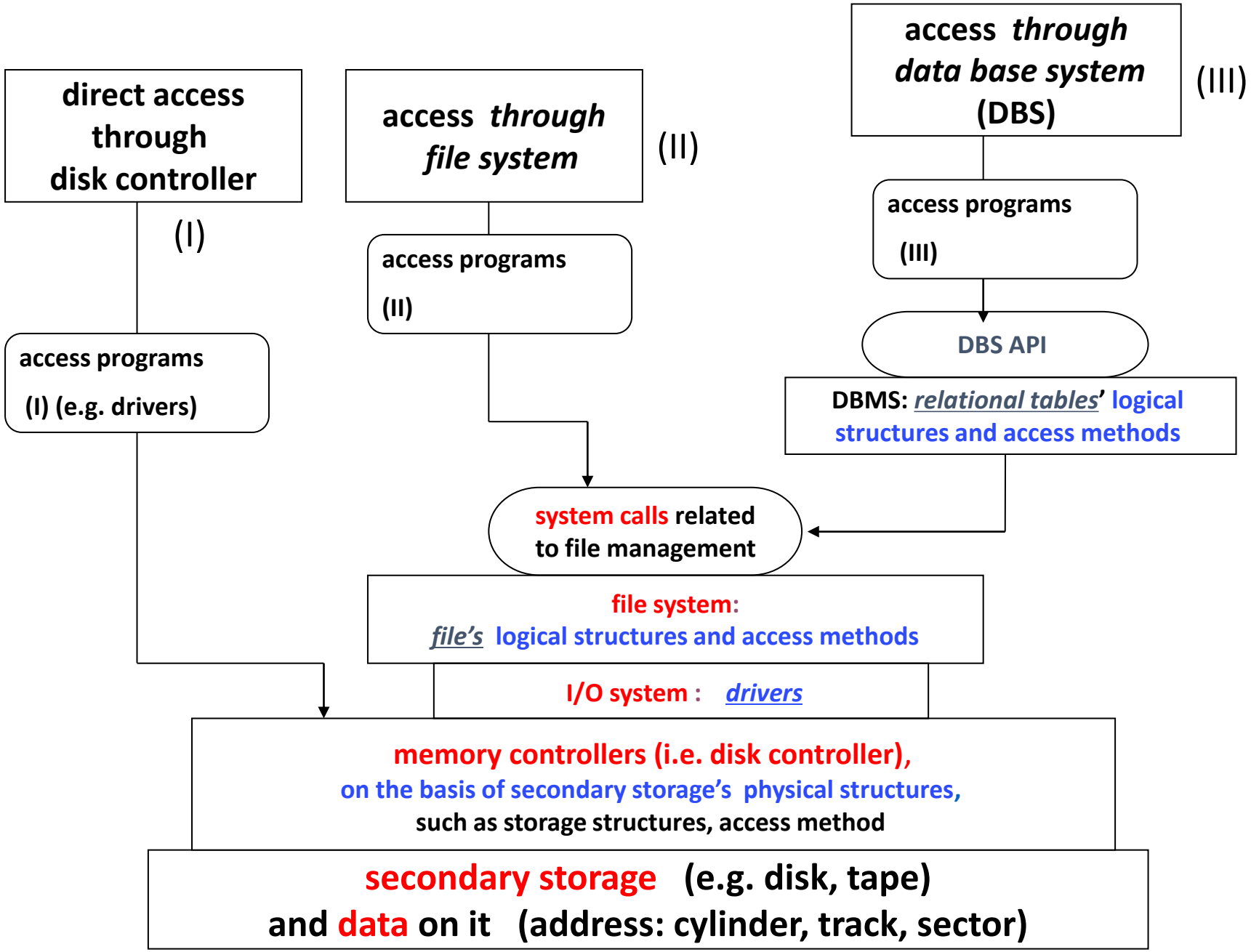
File Concept

- **File System Definition (**P453 in Edt.9**)**
  - a software component in operating systems, providing the mechanism for on-line storage of and access to both data and programs residing on disk/secondary storage
  - in general, also containing
    - a collection of files
    - a directory structure
- **File System *Definition*** in *Microsoft Computer Dictionary*
  - in a operating system, the overall structure in which files are named, stored, and organized
  - file system consists of files, directories, and the information needed to locate and access these items
  - the term *can also refer to the portion of an operating system* that translates requests for file operations from an application program into low-level , sector-oriented tasks that can be understood by the drivers controlling the disk drives

Why file systems needed

- file vs. file system, similar to DB vs. DBMS
- Access Program (I)
  - need to know physical organization access of data on secondary storage
- Access program (II) based on file systems, independent of hardware (i.e. disk, device controller), but related to particular operating systems
  - if OS changes, program (II) also changes
  - file systems provide unified access interfaces between users and data on secondary memory, on the basis of files' logical structures and access methods defined by OS itself
  - the access interfaces may vary with different OS. But for one OS, its access interfaces are identical with respect to different secondary memories and memory controllers

    /*利用OS 的file system, I/O system, 隐藏了不同外存的物理存储和访问差异*/


direct access through disk controller
(I)
access through file system
(II)
access through data base system (DBS)
(III)
access programs (II)
access programs (III)
access programs (I) (e.g. drivers)
DBS API
DBMS: relational tables' logical structures and access methods
system calls related to file management
file system: file's logical structures and access methods
I/O system : drivers
memory controllers (i.e. disk controller), on the basis of secondary storage's physical structures, such as storage structures, access method
secondary storage (e.g. disk, tape) and data on it (address: cylinder, track, sector)

File Concept

- From viewpoints of users, a file consists of a contiguous logical address space, containing
  - a set of structured records reside, record# as logical address

0	1	2	…	k	…		n

//称为有"结构"的记录文件

    - e.g. database file storing relational table, with tuples/rows in table as records in the file
    - e.g. slides in PPT document
  - bytes/words
    - e.g. t*.txt file, executable 0-1 code programs; or

//后两类称为无结构的流式文件

  - storage unit of fixed-size, or logical block
    - such as ***extent*** of 1024-byte in size in Unix/Linux, ext2 and ext4 file system

File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types 
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Many variations, including extended file attributes such as file checksum

- Information about files are kept in the directory entries stored in directory structure, which is maintained on the disk

File Control Block

- Information about files are conceptually organized as FCB (File Control Block) and kept in the directory structure, which is maintained on the disk
  - more details refer to Linux in next chapter

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks

File info Window on Mac OS X

File Operations

- **Create**
- **Write –** at **write pointer** location
- **Read –** at **read pointer** location
- **Reposition within file - seek**
- **Delete(删除)**
- **Truncate(清空)**
- **Open(F_i)** – search the directory structure on disk for entry ***F_i***, and move the content of entry to memory
- **Close (F_i)** – move the content of entry ***F_i*** in memory to directory structure on disk

File Operations

- **Create**
  - allocate secondary storage space to the created file F_i
  - create and insert new FCB entry for F_i corresponding to the file into directory
- Current-file-position-**pointer** for file operations, read/write pointer
  - pointer to the **file data** to be read or wrote
  - e.g. cfo (current file offset) in Linux, refer to Fig.10.4 
- **Write(**file, information to be wrote**) –** at **write pointer** location
- **Read(**file, buffer-address**)–** at **read pointer** location
- **Reposition within file –** also known as file **seek**
  - setup or move current-file-position-pointer for file operations

File Operations

- Delete
  - erase space allocated to the file
  - erase the directory entry of the file
- Truncate
  - erase the contents of a file but keeps its attributes

- The six operations mentioned above comprise the minimal set of required file operations
- Other operations
  - appending, renaming, copy

Example

30. 若目录 dir 下有文件 file1，则为删除该文件内核不必完成的工作是

A. 删除 file1 的快捷方式

B. 释放 file1 的文件控制块

C. 释放 file1 占用的磁盘空间

D. 删除目录 dir 中与 file1 对应的目录项

408-21

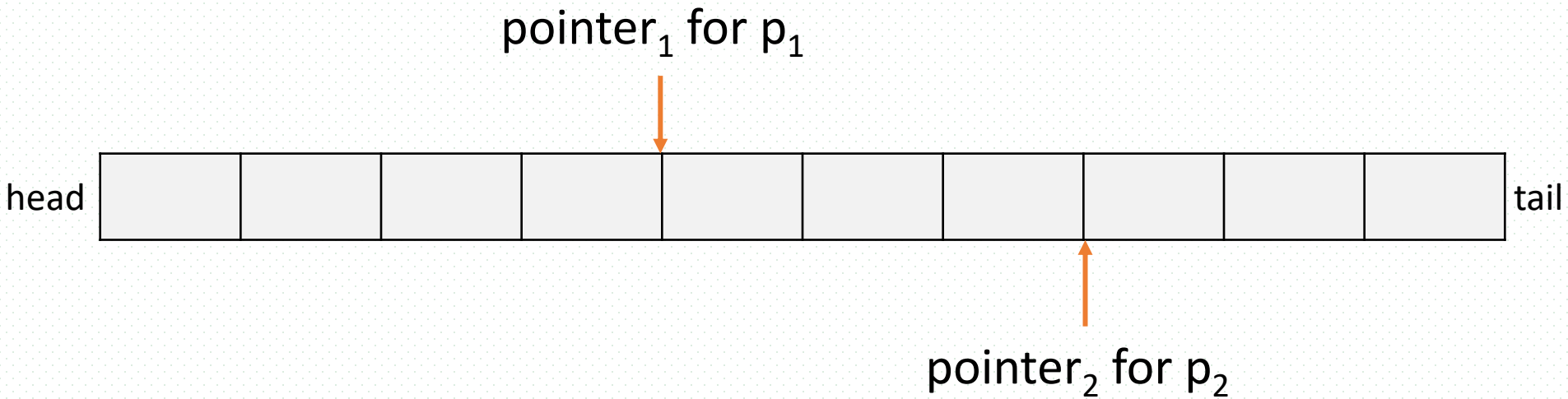
Open/Close Files

- File operations mentioned involve searching the directory for the entry associated with the named file. To avoid searching, an open-file table is introduced in operating systems, which contains information about all opened files
  - open-file table: a data structure in main memory, to record the directory entries of files

- Two operations associated with open-file table are defined
- **Open** (F_i)
  - search the directory structure on the disk for entry F_i, and copy the content of the entry into the open-file table
  - In Linux/Unix, 将文件的读写指针cfo(current file offset)初始化为0，指向文件头部
- **Close** (F_i)
  - remove the entry F_i in the open-file table

Open/Close Files

- In the ***open-file table***, information associated with the opened file
  - the file pointer
    - indicating the last ***read-write*** location in the file
    - unique to each process operating on the opened file
  - the file open count
    - the number of processes which open the file
  - the disk location of the file
  - the access rights



an opened file on disk

Open File Locking

- Some OS provide file locking to control concurrently accessing of files by processes
  - e.g. multiple processes access system log file;

    several transactions write log files in DBS
  - Similar to locking in the **reader-writer problem**
  - **Shared lock** similar to reader lock – several processes can acquire concurrently
  - **Exclusive lock** similar to writer lock
- Mandatory(强制) or advisory(建议):
  - **Mandatory** – access is denied depending on locks held and requested, e.g. in Windows

    //由OS提供完全的locking
  - **Advisory** – processes can find status of locks and decide what to do, in Unix

    //programs与OS一起，实现对文件的locking

File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

Fig.10.3 Common file type

Note:

Operating systems and system programs cooperatively support and implement some complex types of files, especially record-base files

e.g.1 database file supported by DBMS and OS

e.g.2 ppt document implemented by Office

e.g.3 multimedia files

File (Logical) Structure

Three types of file structures

- 1. none-structure/stream-structure
  - file is viewed as a sequence of words, bytes, e.g.
    - executable file viewed as 0/1 strings
    - text file viewed as 8-bit byte strings
- 2. Simple record structure
  - file consists of records with specific structure, e.g.
    - PPT document, consisting of slides
    - database file, storing *tuples*(元组) as record in file
  - the record in file may be
    - fixed length
    - variable length
- 3. Complex structures
  - formatted document
  - relocatable load file

File (Logical) Structure

- Can simulate last two, i.e. simple record structure and complex structure, with first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program

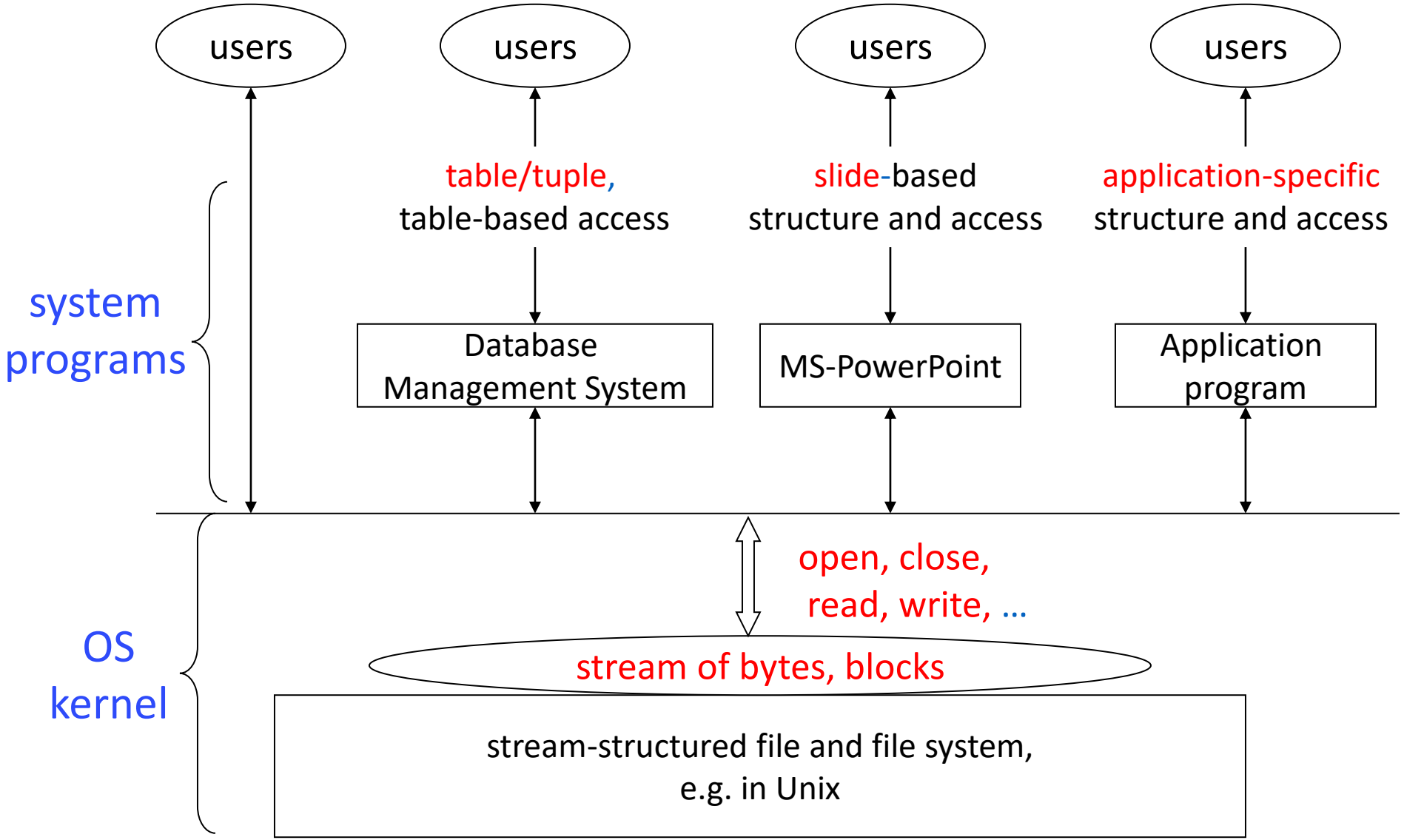


Fig.A.2 High-level interpretation of stream-structured files

instructor: ID(varch) name(varch) dept(varch) **salary(int)**

record 0	10101	Srinivasan	Comp. Sci.	65000

A variable-length record in database file [略]

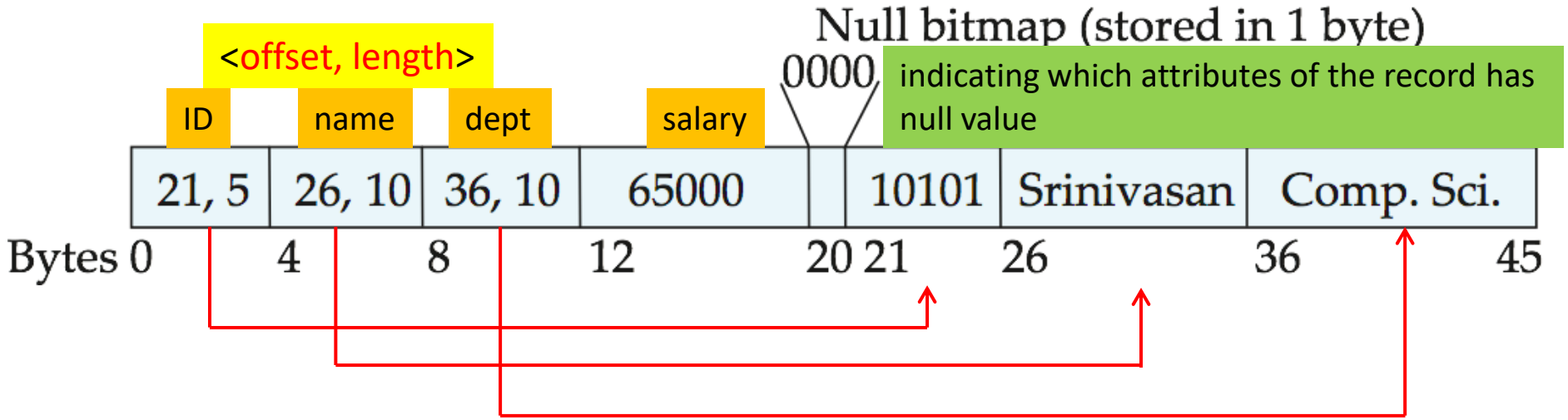


Fig. record 0 is stored in contiguous 46 bytes (0-45)

- Variable length attributes represented by fixed size (offset, length), with actual data stored after all fixed length attributes
- Null values represented by null-value bitmap
- The **slotted page** (e.g. SQL Server2008/201x) structure is often used to organize variable-length records

File Structure

- 为了简化系统，大多数现代操作系统（内核）只/主要提供简单的流式文件结构，记录式文件等更为复杂的数据组织结构由操作系统之上的系统软件（DBMS, MS-Word, MS-Excel）或应用程序通过对流式数据的进一步结构化解释来得到
  - OS支持过多的文件系统类型会复杂化OS的设计

- File's logical structure vs physical structure
  - logical records in files are stored in physical disk ***blocks***

    records ⟺ block

  - a file can also be viewed as a sequence of physical blocks
    - physical structures of the file
- Logical structure determines the user-oriented file system ***interfaces***
  - OS按照文件逻辑结构，e.g. records, bytes, extent, 对用户提供文件访问系统调用

10.2 Access Methods

- Sequential Access
- Direct Access (relative/random access)
- Index-based access
- Example
  - system calls in Linux: open, close, write, read, lseek

Sequential Access

- Sequential access is based on the ***tape*** model of files, commonly-used by editors and compilers
- Data organization
  - information in the file is arranged in order, one byte/word/record after another, according to ***current-position-pointer***, e.g. ***cfo*** in Linux in Fig. 10.4
- File operations
  - *read next* :

    read the next portion of the file, and then advances a file pointer

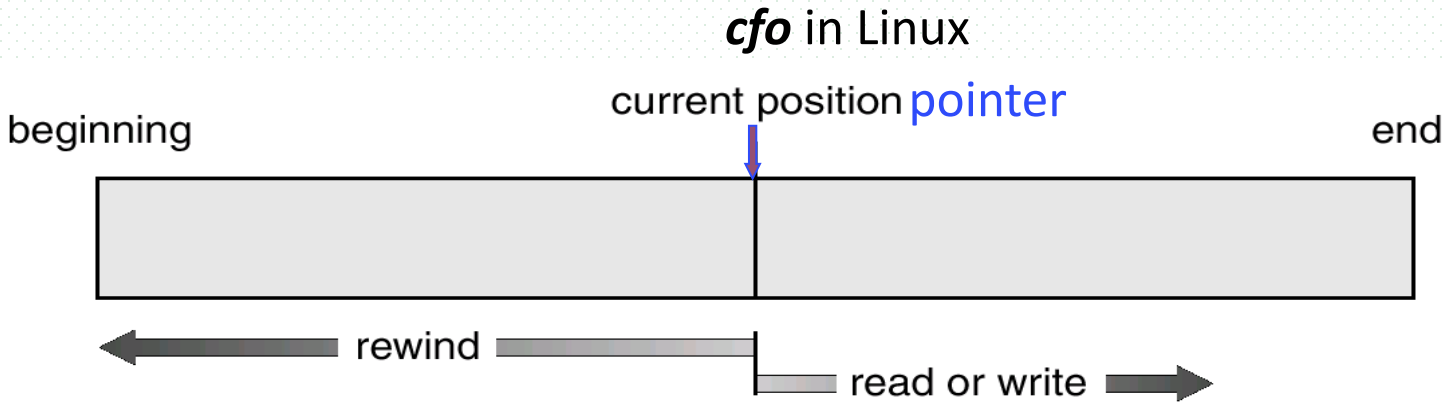
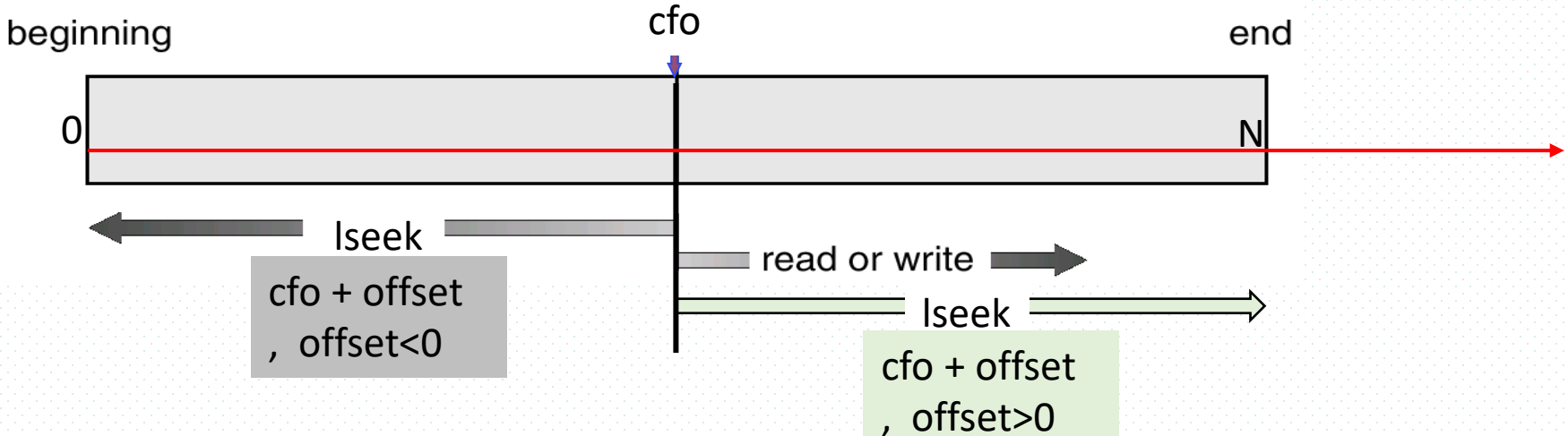


Fig. 10.4 Sequential-access file

当前文件偏移量cfo(current file offset) in Linux

- cfo
  - 表示文件开始处到文件当前位置的字节数，一般是一个非负整数，代表了文件读写指针位置
  - 对文件的read()、write()通常始于cfo，即根据cfo指向的文件位置，进行读写操作，并使cfo值增大，cfo的增量为读写的字节数
- 使用open()操作打开文件时，文件的cfo初始化为0，指向文件开始位置，除非参数flags=O_APPEND
- 使用系统调用lseek()可以改变文件的cfo，即改变文件读写指针位置



Sequential Access

- write next:
  appends to the end of the file, and advances the new end of the file
- *reset:* reset to the beginning, skip forward or backward *n* records

- Example of **sequential access** in DBS: **index-scanning** in B+-tree in next slide【略】
  - select ID, name, department
    from instructor
    where department='Comp.Sci'

【略】

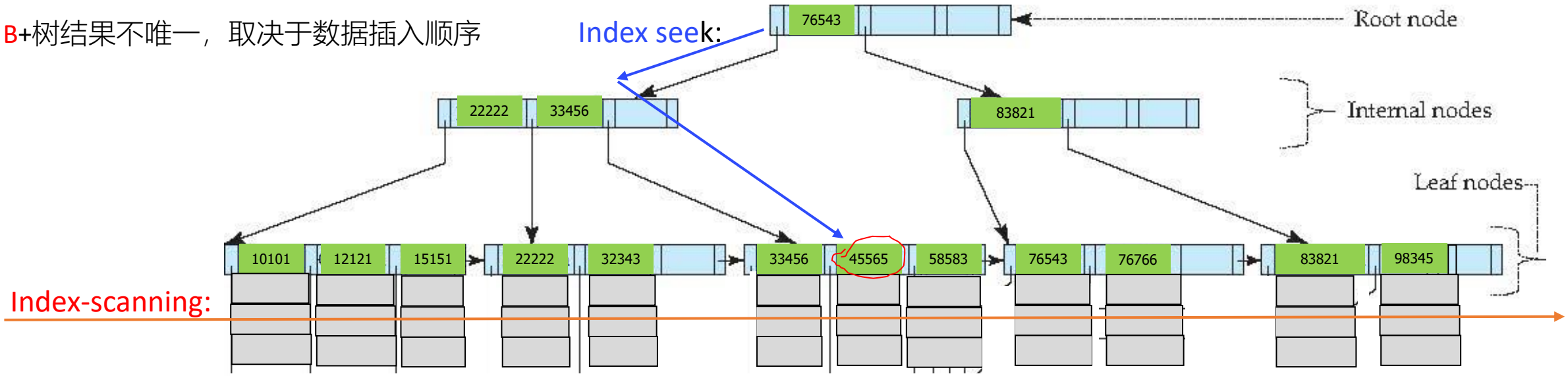
文件记录直接存储在叶节点

B^+-Tree (n=4) Index Sequential File

Primary/Clustered index on ID

n: degree or branching factor

B+树结果不唯一，取决于数据插入顺序




```
create table instructor
      (ID integer, primary key;
                .....
       )
```

10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Index-based access:

index seek

```
select  ID, name, department
from   instructor
where  ID=45565
```

Sequential access:

Index scanning

```
select  ID, name, department
from   instructor
where  department='Comp.Sci'
```

Direct Access (Relative/random access)

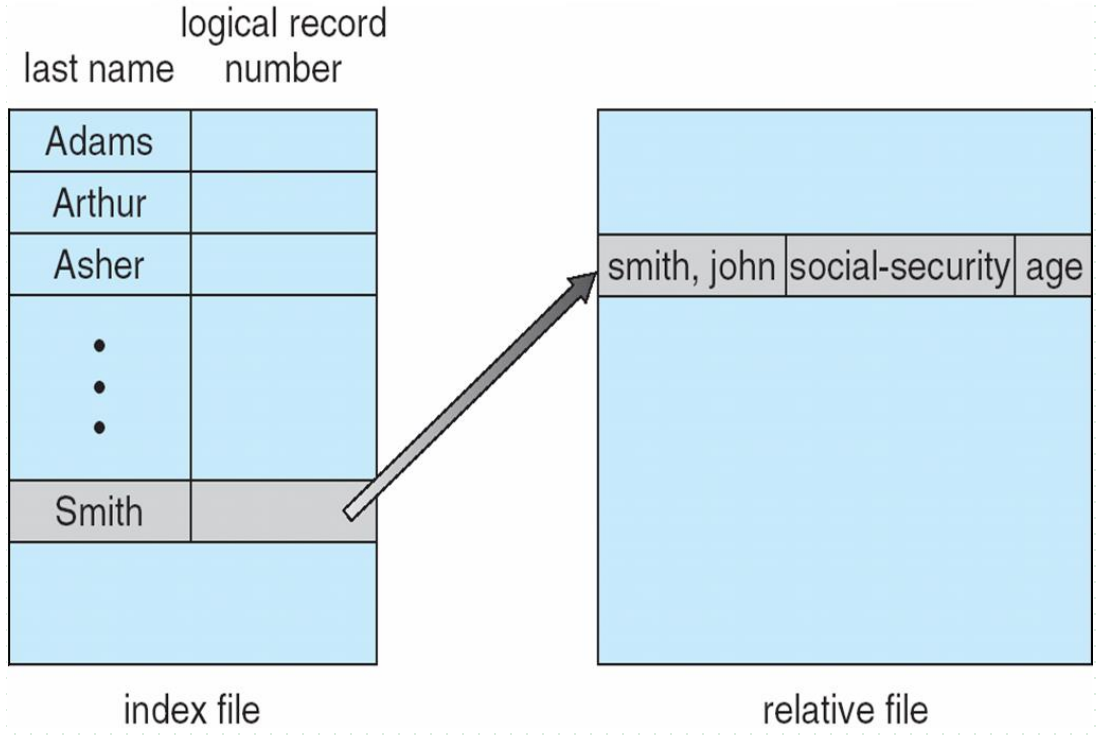
- A file is made up of a numbered sequence of **fixed-sized** logical records or blocks
- Arbitrary records or blocks in the file can be directly accessed, given the number of the records or blocks
- Block number
  - relative block number, an index relative to the beginning of the file
  - e.g. n = relative block number

    read n
    write n
    position to n
    rewrite n
- Relative block numbers allow OS to decide where file should be placed
  - see allocation problem in Ch 12
- Direct access is implemented on the basis of disk-access model which allows random access to any file block

Other Access Methods: Index-based Access

- General involve creation of an index for the file, e.g. B+-tree
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item)
- If too large, muti-level index: index (in memory) of the index (on disk)

- Example of **index-based access**: **Index-seek**
  - select ID, name, department
    from instructor
    where ID=45565



System Calls for File Access in Linux

- open, close, write, read, lseek

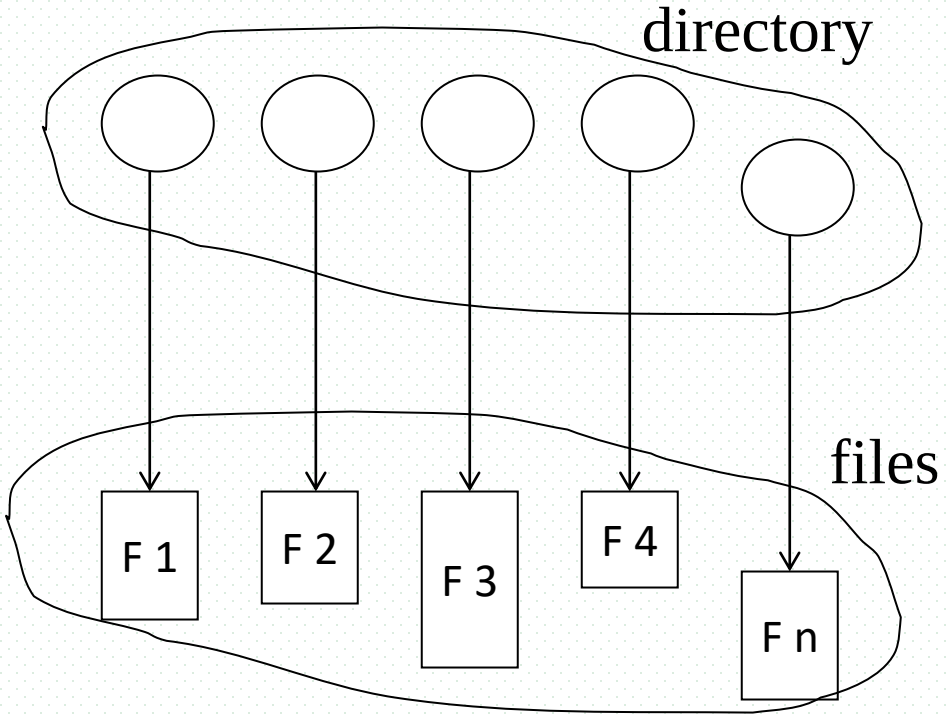
- For more details on the system calls and an example program, refer to

操作系统概念（英文）——11-22-linux文件访问系统调用（open,close,rea

10.3 Directory and Disk Structure

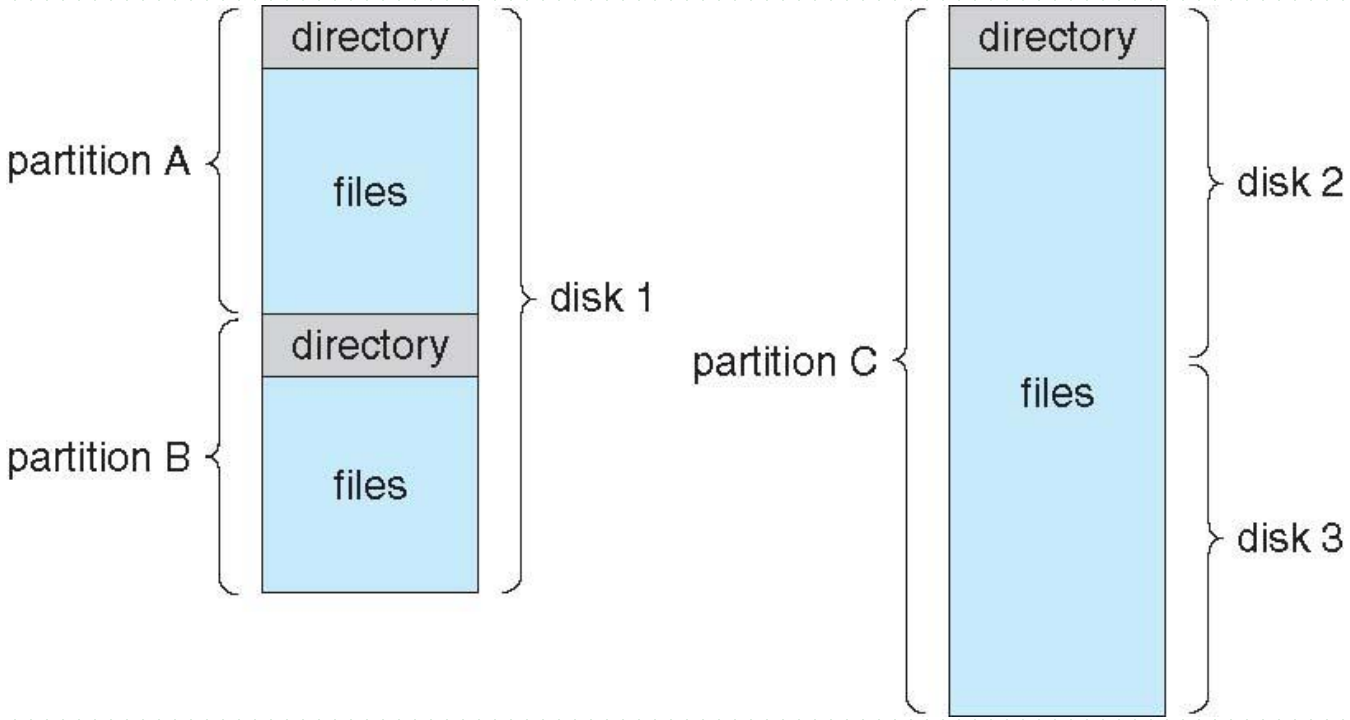
- Directories are used to organize a large number of files in systems, which consists of a collection of nodes containing information about all files
  - set of directory entries (dentry in Linux), set of file-control-block (FCB)

    entry : file-name → file description information (FCB)
- The directory can be viewed as a symbol table that translating file names into their directory entries
- Both the directory structure and the files reside on disk



Disk (Logical) Structure

- Disk can be subdivided into **partitions**
  - partitions also known as minidisks, slices
- Disk or partition can be
  - **raw,** used without a file system, or
    - e.g. create database
  - **formatted** with a file system
- A partition containing file system is known as a **volume**



- Each volume containing file system also tracks that file system's info in **device directory** or **volume table of contents**
- Disks or partitions can be RAID protected against failure

Operations Performed on Directory

- search for a file
- create a file
- delete a file
- list a directory
- rename a file
- traverse (遍历) the file system

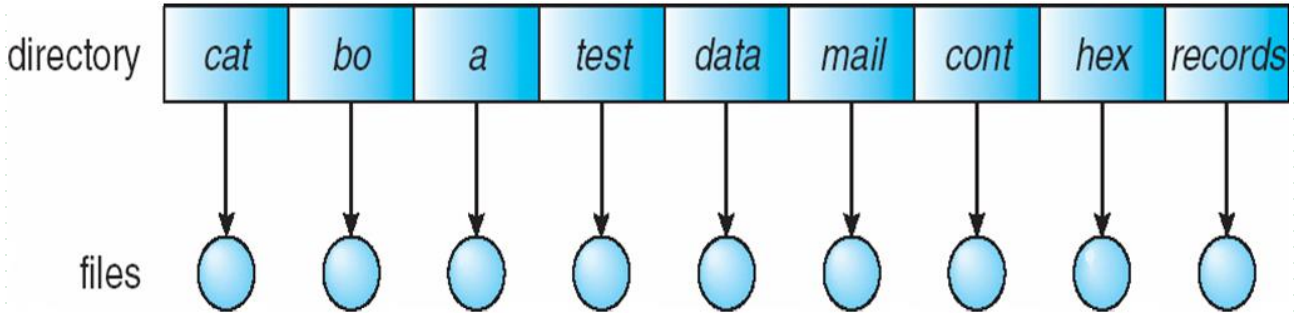
Directory Organization

Organize the directory (logically) to obtain

- Efficiency – locating a file quickly
- Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

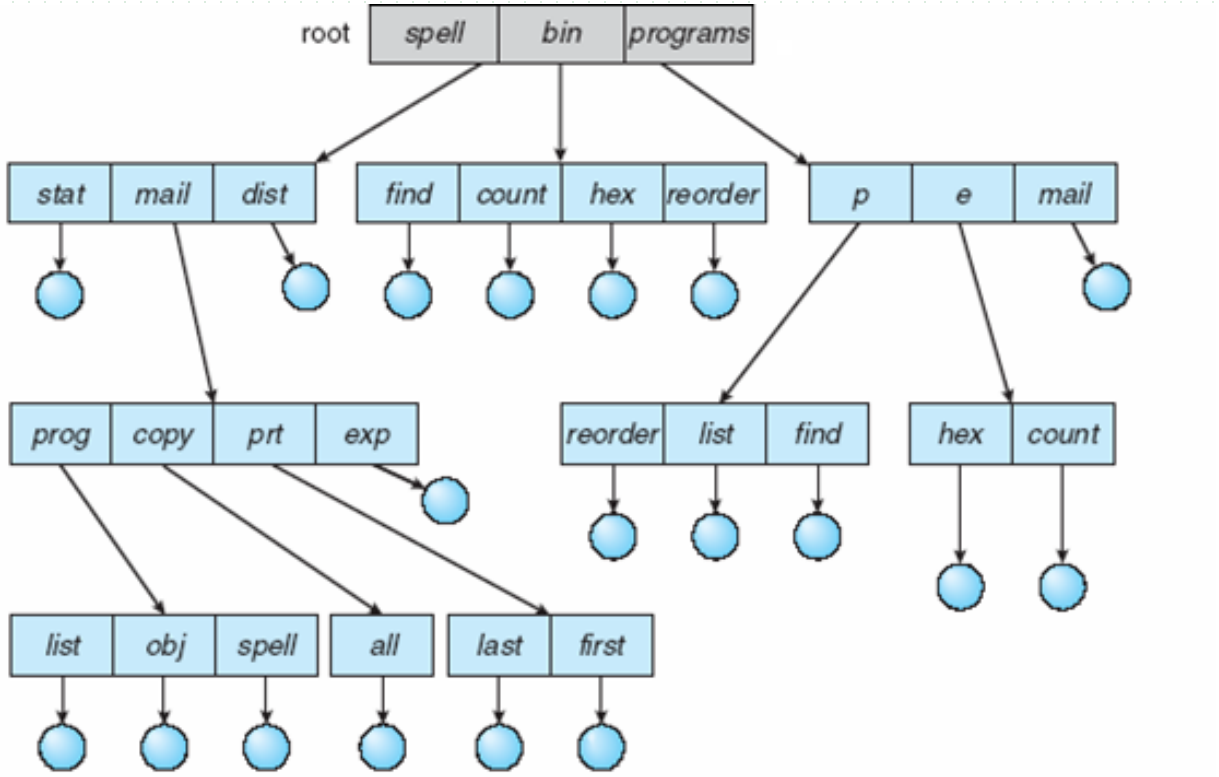
Directory Organization

- Logical structures of the directory
  - single-level directory
  - two-level directory
  - tree-structure directory
  - acyclic-graph directories(非循环图目录)
  - general graph directories

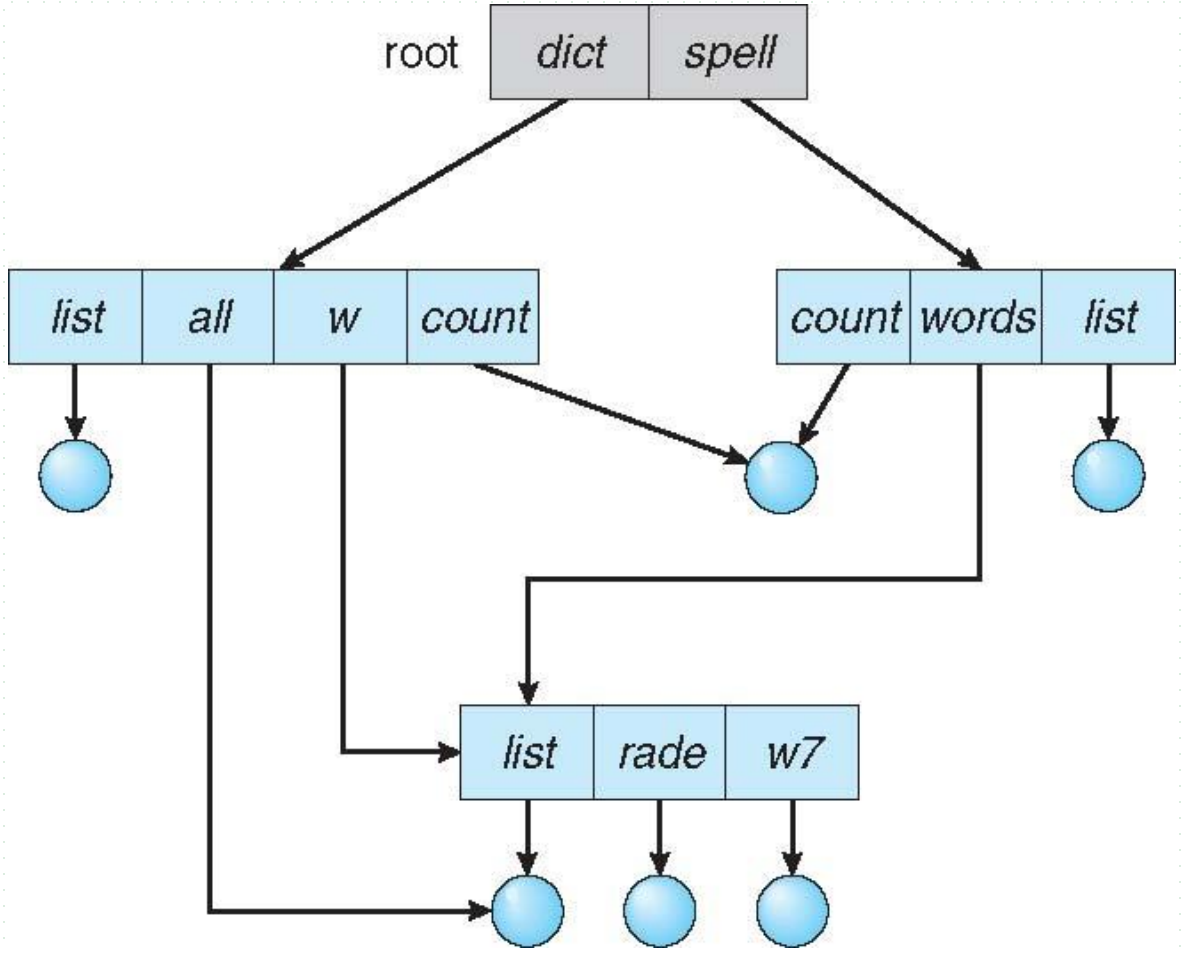


single-level directory

Directory Organization



Tree-Structured Directories

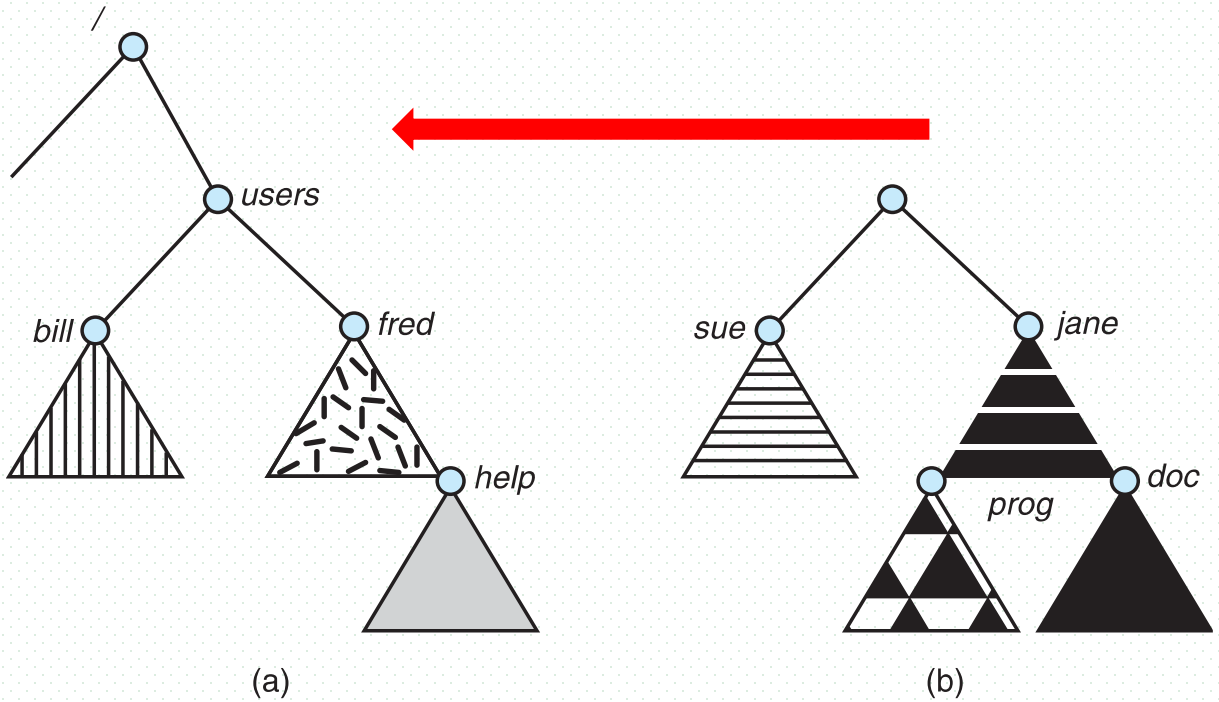


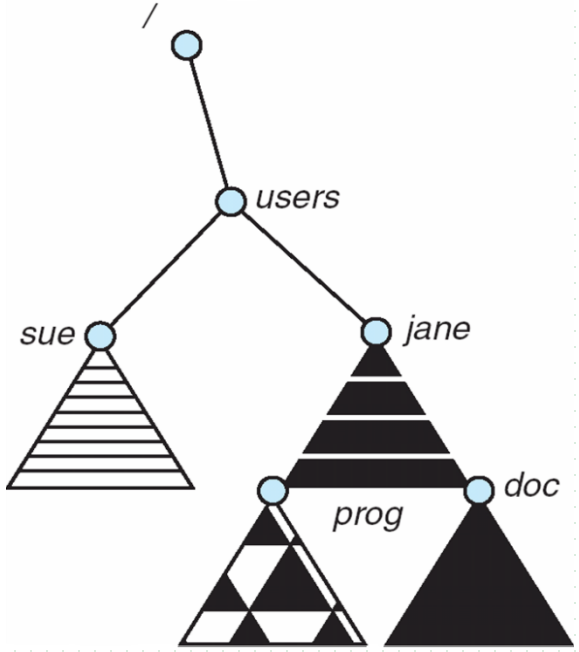
Acyclic-Graph Directories

10.4 File System Mounting

- OS can contain one or more file systems
- Mounting
  - 挂载、安装
- A file system must be **mounted** before it can be accessed
- A unmounted file system (i.e., Fig. 11-11(b)) is mounted at a **mount point,** e.g. /user

Linux启动时，创建文件系统的根目录，之后逐步将各种文件系统挂载到根目录下不同的子目录中；
用户也可以挂载、卸载文件系统





10.5 File Sharing

- Sharing of files on multi-user systems is desirable
- Sharing may be done through a **protection** scheme
  - owner/user of the shared file/directory, who can change attributes of the shared ones, grant access, and has the most control over the shared ones.
  - group of the shared file/directory, who share the access to the file.
  - which operations can be executed by group members is defined by the owners

File Sharing

- On distributed systems, files may be shared across a network
- Network File System (NFS) is a common distributed file-sharing method
- If multi-user system
  - **User IDs** identify users, allowing permissions and protections to be per-user
    Group IDs allow users to be in groups, permitting group access rights
  - Owner of a file / directory
  - Group of a file / directory

Remote File Systems

- Uses networking to allow file system access between systems
  - Manually via programs like FTP
  - Automatically, seamlessly using **distributed file systems**
  - Semi automatically via the **world wide web**
- **Client-server** model allows clients to mount remote file systems from servers
  - Server can serve multiple clients
  - Client and user-on-client identification is insecure or complicated
  - **NFS** is standard UNIX client-server file sharing protocol
  - **CIFS** is standard Windows protocol
  - Standard operating system file calls are translated into remote calls
- Distributed Information Systems **(distributed naming services)** such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing

Protection

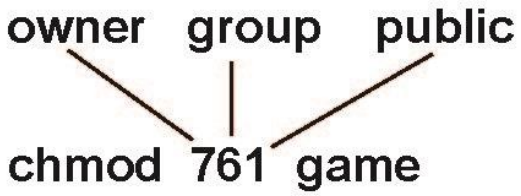
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users on Unix / Linux

			RWX
a) **owner access**	7	⇒	1 1 1
			RWX
b) **group access**	6	⇒	1 1 0
			RWX
c) **public access**	1	⇒	0 0 1

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

```
chgrp      G      game
```

Thanks for your attention

北京郵電大學