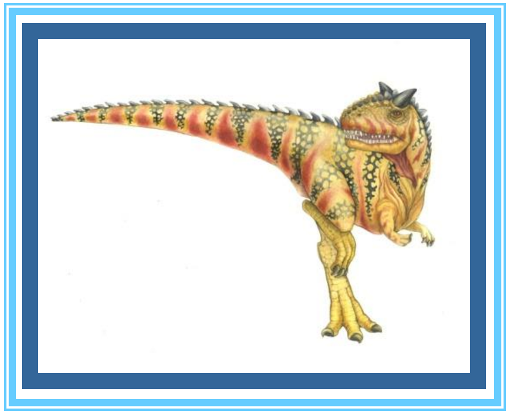
Chapter 13: File-System Interface



[illegible] : ? calculate effective bandwith

? calculate average access time

Disk scheduling

Chapter 13: File System Interface

how os do that ?

- File Concept what is this?
- Access Methods sequencial / random access
- Disk and Directory Structure some methods -low level with trade-offs
- File-System Mounting
- File Sharing not protected
- Protection not sharing

Objectives

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

why we need this?
- provide convenience
- efficient to manage !

- 下一課会发現

pros and cons

understand
what is protection,
how is works

不然会 suffer from linux!

File Concept

logically continuous!

- Contiguous logical address space

instructions and this!

implementation!

- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

不同 files!

- Contents defined by the file's creator
  - Many types, consider text file, source file, executable file

high flexibility!

File Attributes

- Name – information kept in human-readable form
- Identifier – unique tag (number) identifies files within a file system
- Type – needed by systems that support different types
- Location – pointer to file location on device
- Size – current file size
- Protection – controls who can do reading, writing, executing, etc.
- Time, date, and user identification – data for protection, security, and usage monitoring
- Information about files are kept in a directory structure, maintained on the disk - part of which currently in use can be cached in main memory for fast access
- Many variations, including extended file attributes such as file checksum

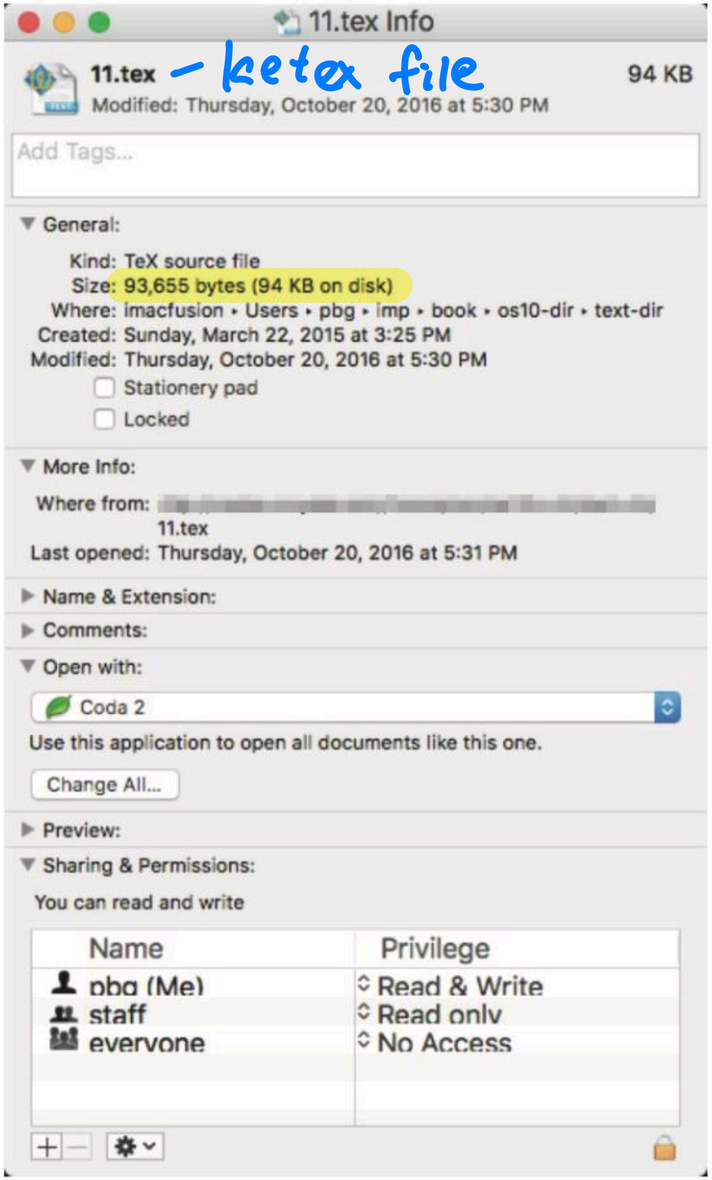
let OS know. how to open? Sometimes encoded!

not embedded in content itself!

cache this in memory!

ensure that not changed by someone checksum in MacOS

File info Window on Mac OS X



11.tex - latex file

94 KB

Modified: Thursday, October 20, 2016 at 5:30 PM

▼ General:

Kind: TeX source file
Size: 93,655 bytes (94 KB on disk)
Where: imacfusion ▸ Users ▸ pbg ▸ imp ▸ book ▸ os10-dir ▸ text-dir
Created: Sunday, March 22, 2015 at 3:25 PM
Modified: Thursday, October 20, 2016 at 5:30 PM

□ Stationery pad
□ Locked

▼ More Info:

Where from: [illegible]
11.tex
Last opened: Thursday, October 20, 2016 at 5:31 PM

▼ Open with:

Coda 2

Use this application to open all documents like this one.

▼ Sharing & Permissions:

You can read and write

Name	Privilege
pbg (Me)	Read & Write
staff	Read only
everyone	No Access

file information

File Operations

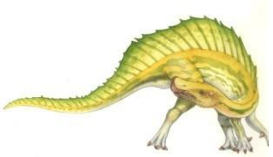
- File is an ADT or abstract data type
- Create – create a file
- Write – at write pointer location
- Read – at read pointer location

important!

- Reposition within file - seek
- Delete
- Truncate 缩减！
- Open(F_i) – search the directory structure on disk for entry F_i, and move the content of entry to memory, preparing file for subsequent access
- Close (F_i) – move the content of entry F_i in memory to directory structure on disk

structure maintained by the kernel!

- Such operations involve the changes of various OS kernel data structures



Open Files

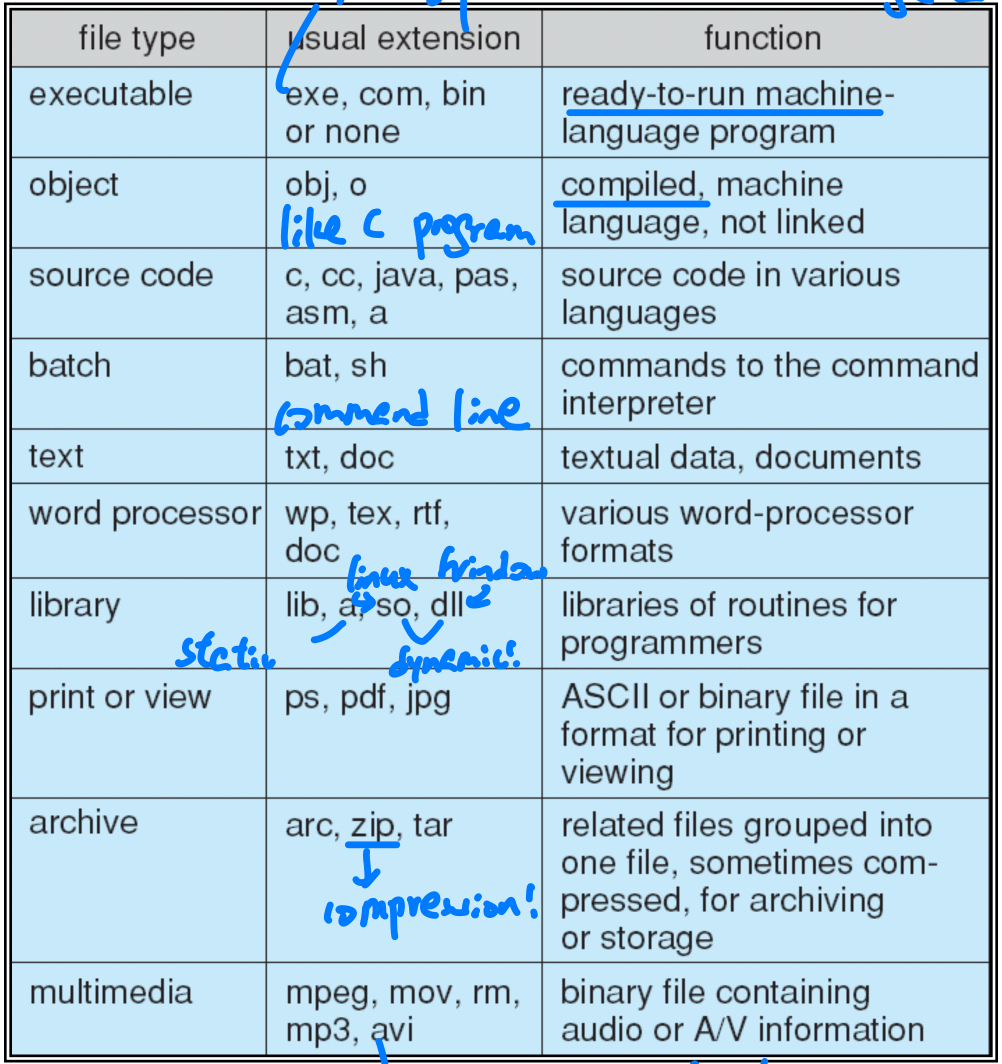
fork!

- Several data structures are needed to manage open files:
  - Open-file tables: tracks open files, system-wide open-file table, and per-process open-file table ← copy this table!
  - File pointer: pointer to last read/write location, per process that has the file open → how open change the table?
  - File-open count: counting the number of processes that the file has been opened – to allow removal of data from the open-file table when the last processes closes it (when file-open count is zero)
  - Disk location of a file: cache of data access information (Fast access)
  - Access rights: per-process access mode information

change the files : get the pointer!

File Types – Name, Extension

ready to run → lodge a process



file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o like C program	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh command line	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	linux windows lib, a, so, dll static dynamic!	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar compression!	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

videos!

modify by you

.docx
↑
extension of this file
↓
OS use this to disthguish

Access Methods

Simple!

beginning

- **Sequential Access** – simplest access method

```
read next
write next
reset
```

→ ptr move to beginning

read 完後 move!

no read after last write
(rewrite)

implemented by sequential access method

- **Direct Access** – file is fixed length logical records

blocks!,

why need the blocks!

```
read n
write n
position to n
        read next
        write next
rewrite n
```

← relative block number!

start index 0/1 depends on os!

n = relative block number

block so small ⇒ lots of overhead!

- Relative block numbers allow OS to decide where file should be placed
  - See disk block allocation problem in Chapter 14

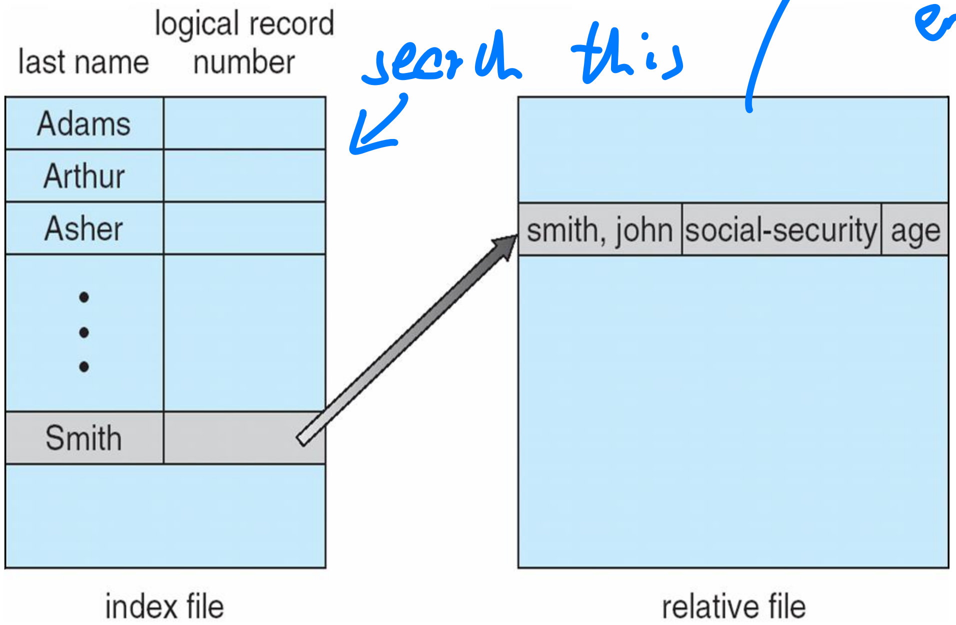
Other Access Methods

widely used nowadays

- Other file access methods can be built on top of direct-access method
- Generally, involve creation of an **index** for a file
  - Keep index in memory for fast location of the data to be operated on
  - If too large, index (in memory) of the index (on disk)
- IBM indexed sequential-access method (ISAM) is an example
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS ✓
- VMS operating system provides index and relative files as another example

Memory scan
→ improve performance

if all search then long time, every entry is much longer!



search this

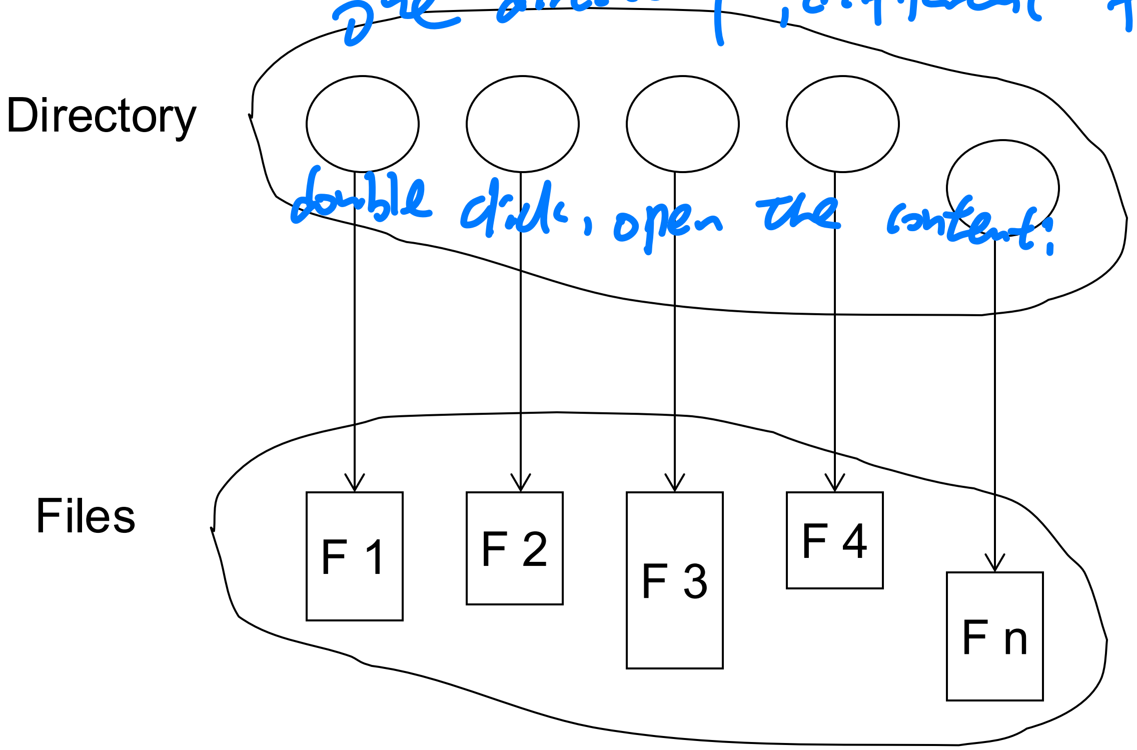
Index is large ⇒ put in disk ⇒ & sequential access

fetch complex information!

Directory Structure

- A collection of nodes containing information about all files

one directory, different files:



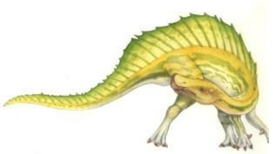
double click, open the content:

Both the directory structure and files reside on disk

physical

Disk Structure

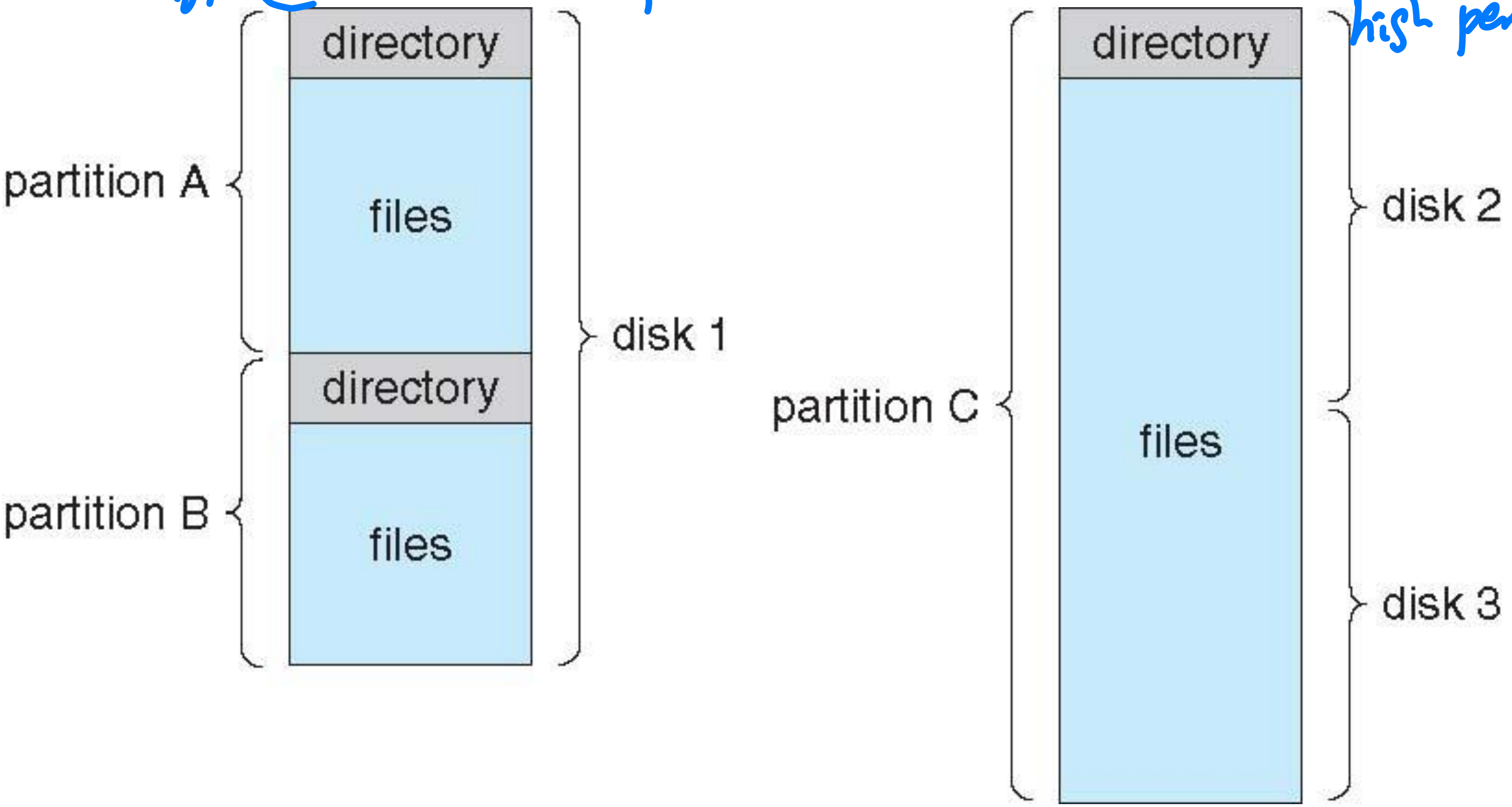
- Disk can be subdivided into partitions — CDE partitions
- Disks or partitions can be RAID protected against failure
  2個disk，2個SSD 隨組 performance/reliability
- Disk or partition can be used raw – without a file system, or formatted with a file system
- Partitions are also known as minidisks, slices ← nickname
- An entity on a disk containing a file system known as a volume → Mac OS
- Each volume containing a file system also keeps track of the file system info in device directory or volume table of contents
- Other than general-purpose file systems, there are many special-purpose file systems, frequently within the same operating system or computing systems



A Typical File-system Organization

one disk 多個 partitions! build Raid 0

high performance



Operations Performed on Directory

- Search for a file ✓
- Create a file
- Delete a file 不想要!
- List a directory ls . . .
- Rename a file rn
- Traverse the file system dfdn . . .

Organize the Directory (Logically) to Obtain

- **Efficiency** – locating a file quickly

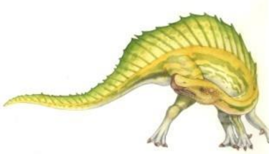
for human!
↓
directory convenient to me, 不想 put every file to one single file!

- **Naming** – convenient to users

file 不可撞名!

  - Two users can have same name for different files
  - The same file can have several different names

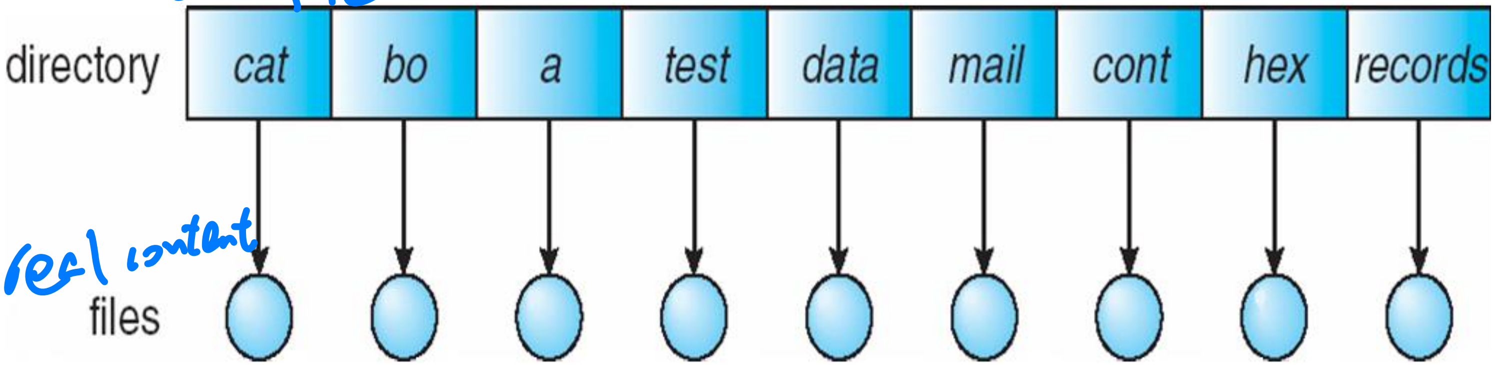
- **Grouping** – logical grouping of files by properties, (e.g., all Java programs, all games, my comp3511, …)



Single-Level Directory

- A single directory for all users

one file



real content

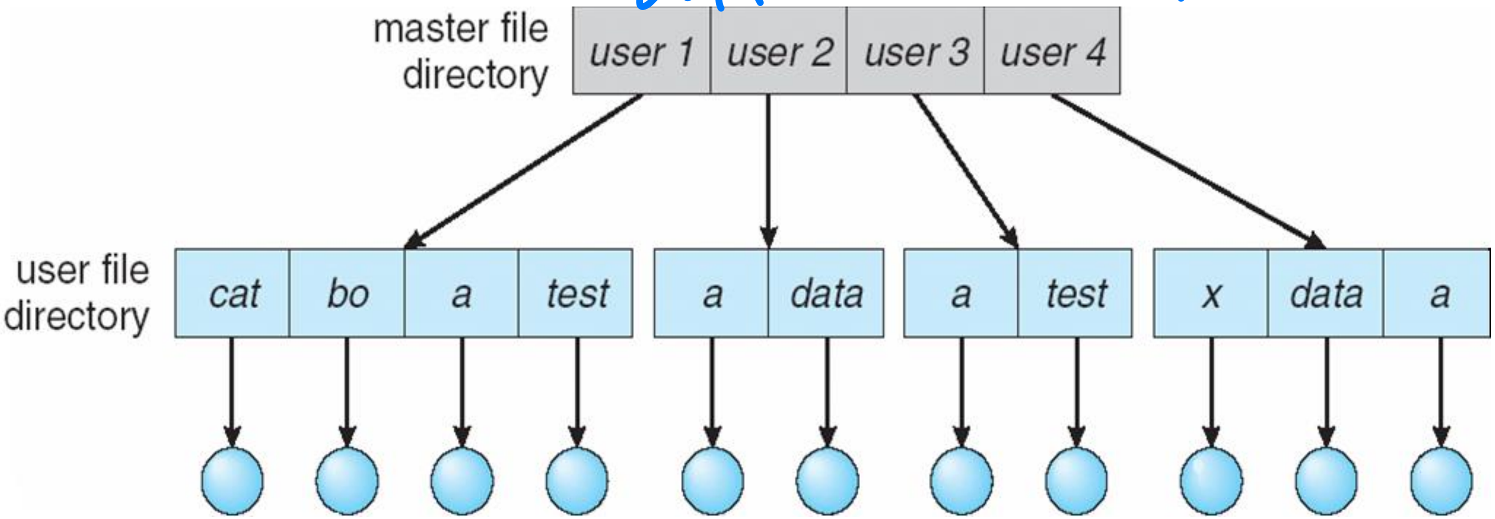
Naming problem 不可兩有 cat!

Grouping problem 不可group，例如 group animals!

Two-Level Directory

- Separate directory for each user

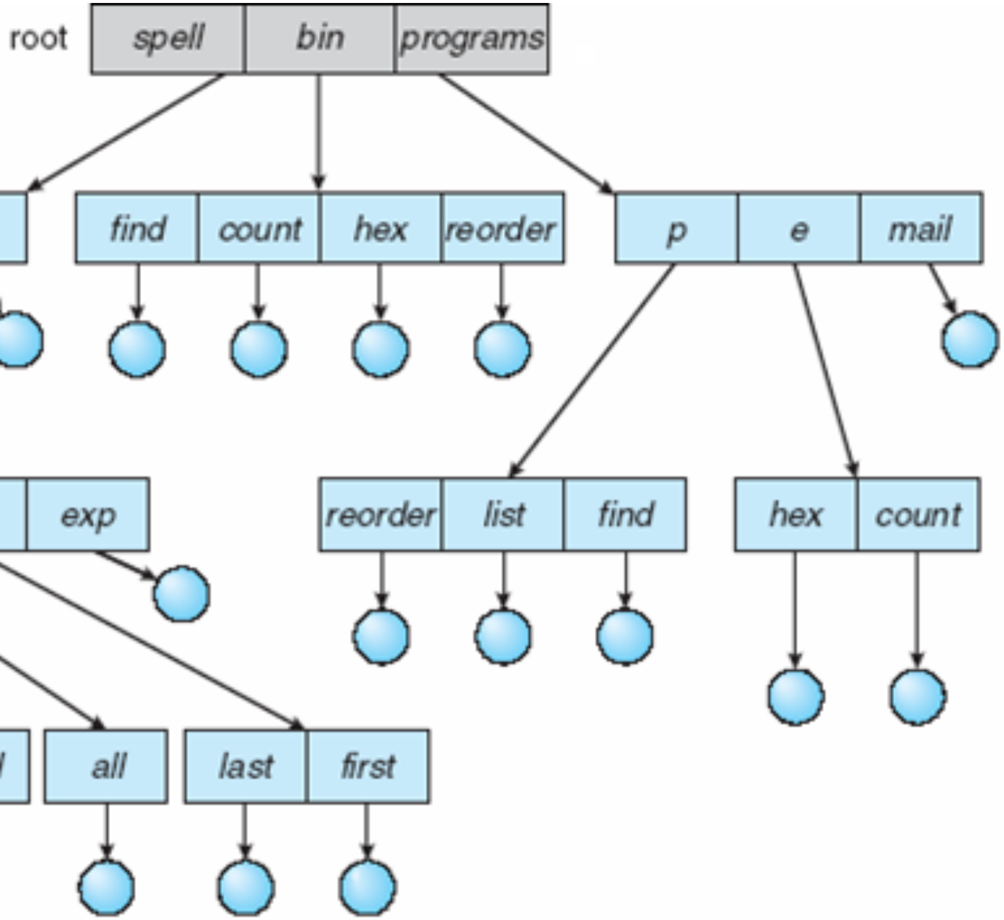
different files!



- Path name – need a pathname to identify a file/dir, e.g., /user1/cat
- Can have the same file name under different users (paths)
- More efficient searching than single-level directory
- No grouping capability

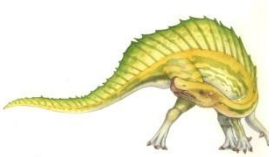
more levels

Tree-Structured Directories



Tree-Structured Directories (Cont.)

- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - `cd /spell/mail/prog` ✓ change direction!
  - `type list`



Tree-Structured Directories (Cont)

- Absolute or relative path name
- Creating a new file is done in the current directory
- Delete a file in the current directory

```
rm <file-name>
```

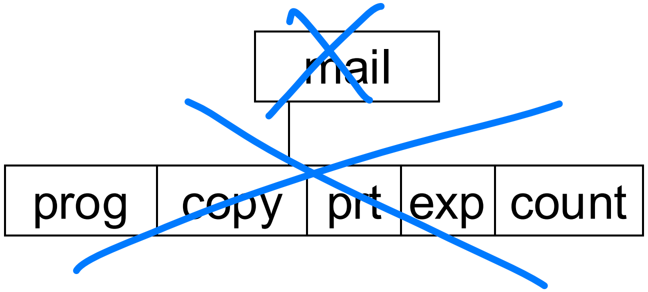
- Creating a new subdirectory is done in current directory

```
mkdir <dir-name>
```

Example: if in current directory `/mail`

```
mkdir count
```

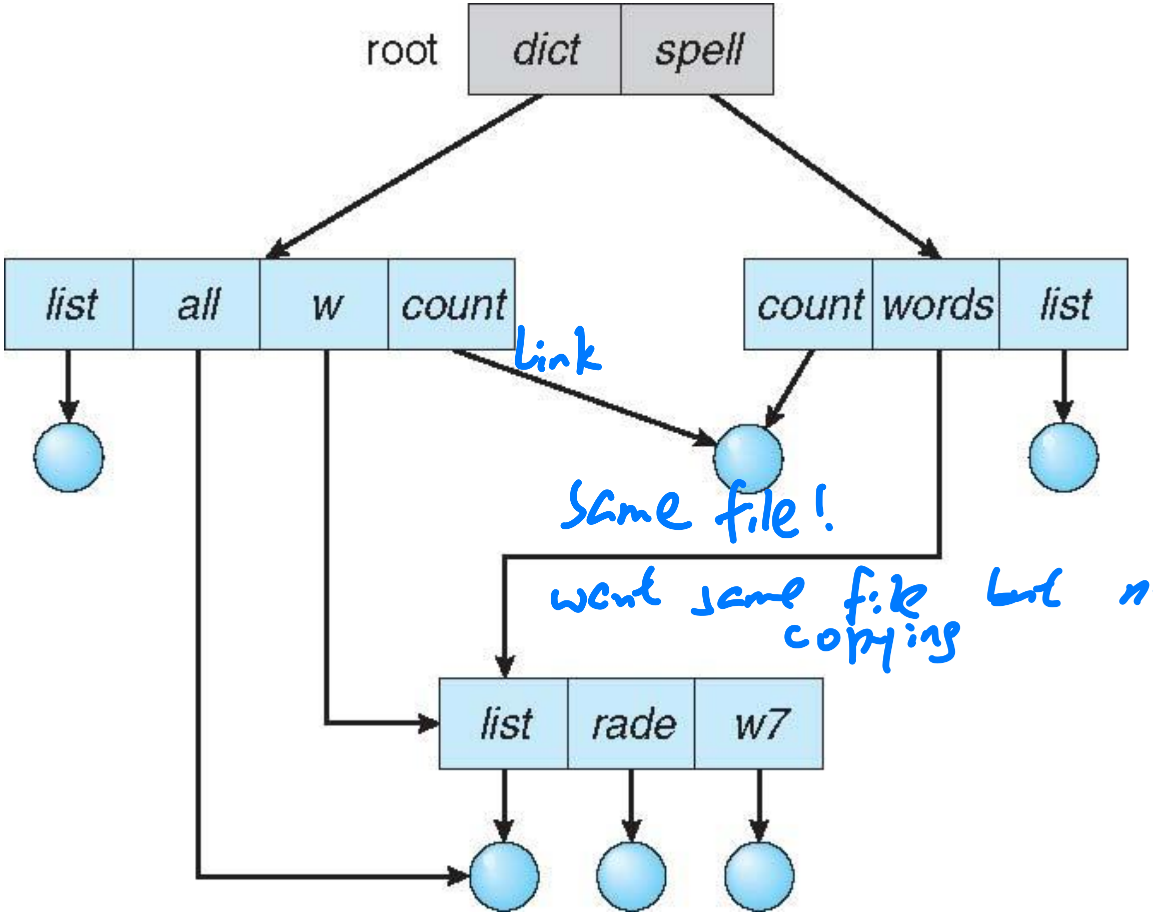
↓ delete mail ⇒ delete everything under mail node!



Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Acyclic-Graph Directories

- Have shared subdirectories and files – more flexible and complex



link

same file!

want same file but not copying

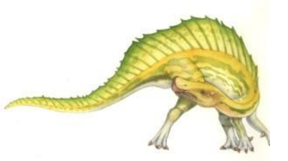
Acyclic-Graph Directories (Cont.)

- New directory entry type
  - Link – another name (pointer) to an existing file
  - Resolve the link – follow pointer to locate the file
- Two different (path) names (aliasing)
  - Ensure not traversing shared structures more than once

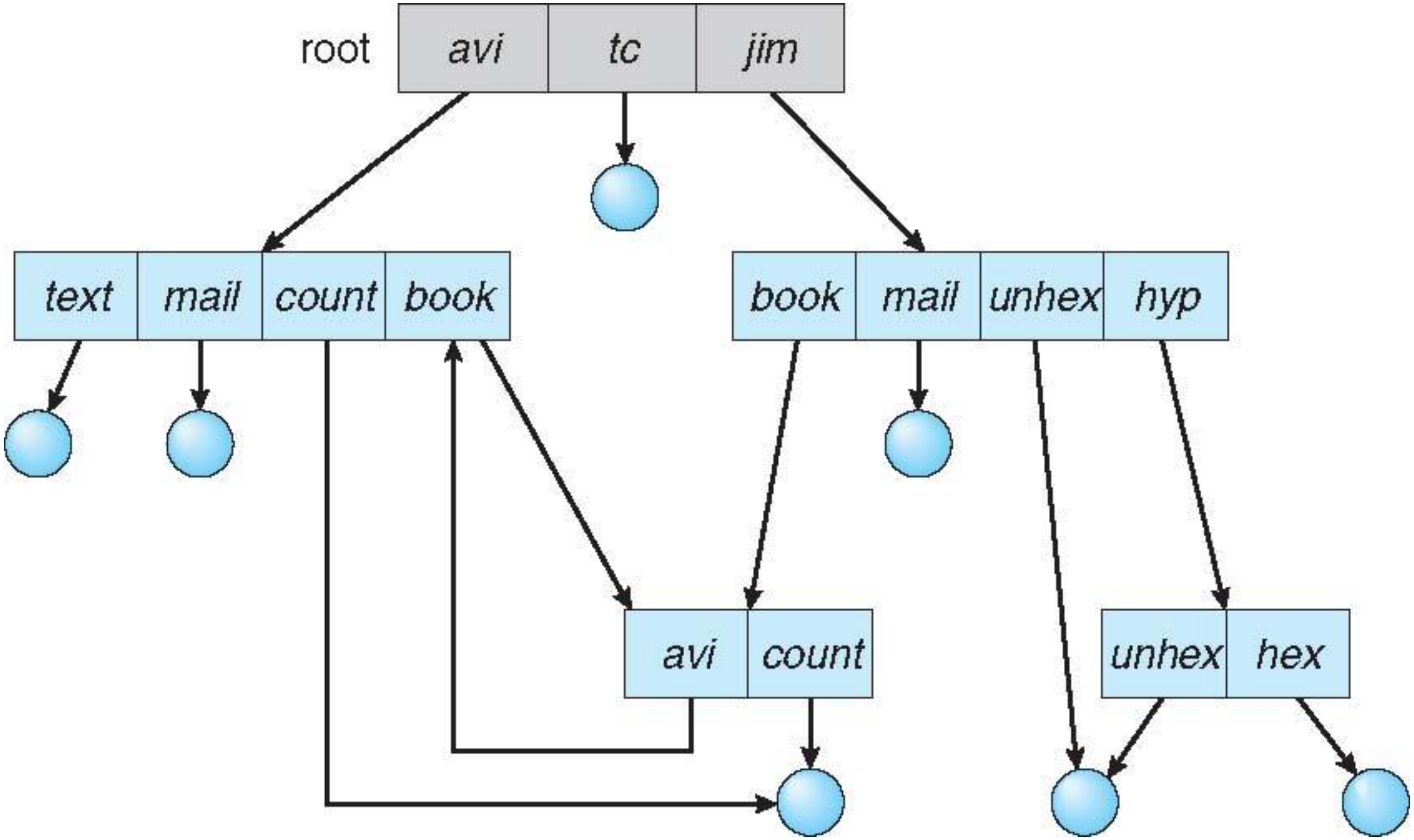
[illegible] dead lock!

- Deletion might lead to that dangling pointers that point to empty files or even wrong files
- There is also difficulty ensuring there is no cycles in a graph – complexity associated with it

Extra overhead!



General Graph Directory

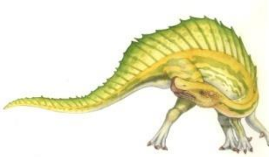


General Graph Directory (Cont.)

- How do we guarantee no cycles?
  - Allow only links to file not subdirectories – sometime not convenient ✓
  - Every time a new link is added use a cycle detection algorithm to determine whether there is a cycle or not – time consuming

high overhead!!!

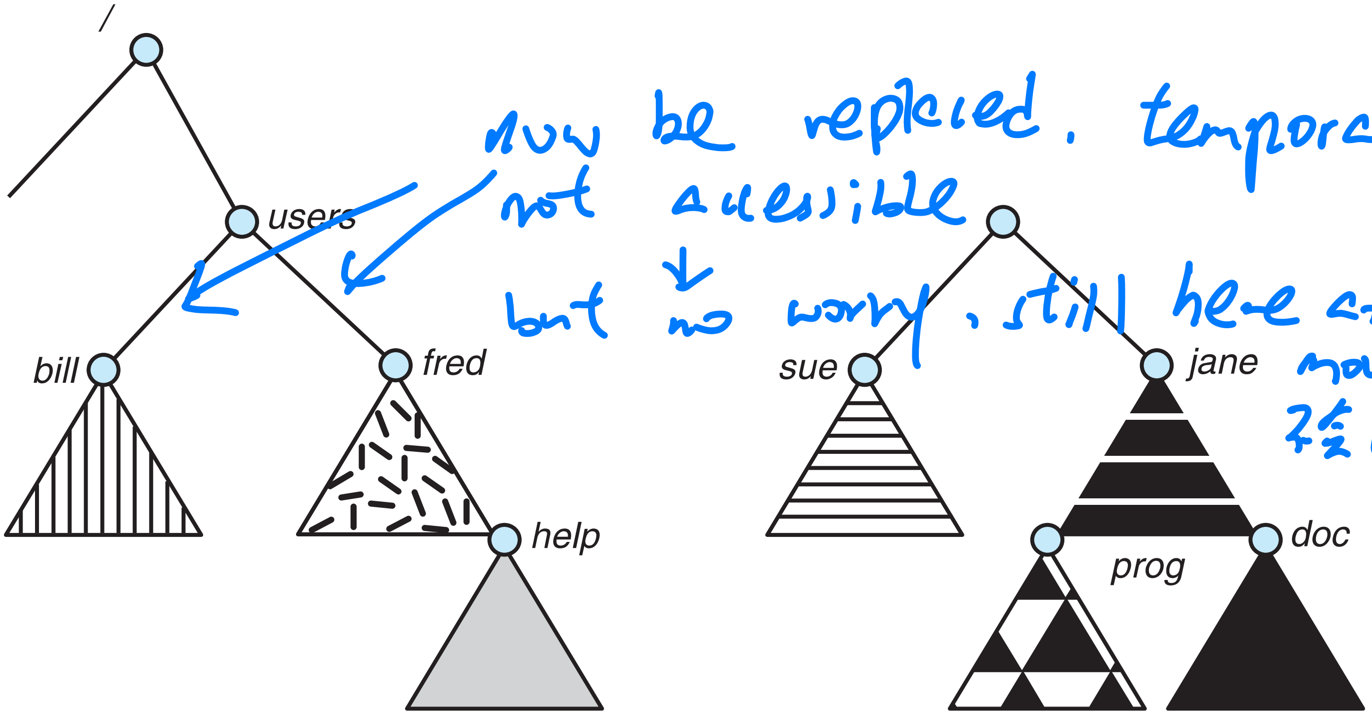
tradeoffs!



File System Mounting

- A file system must be mounted before it can be accessed – just like a file must be opened before it is used

- A unmounted file system (i.e., Fig. (b)), to be mounted at a mount point



(a)

(b)

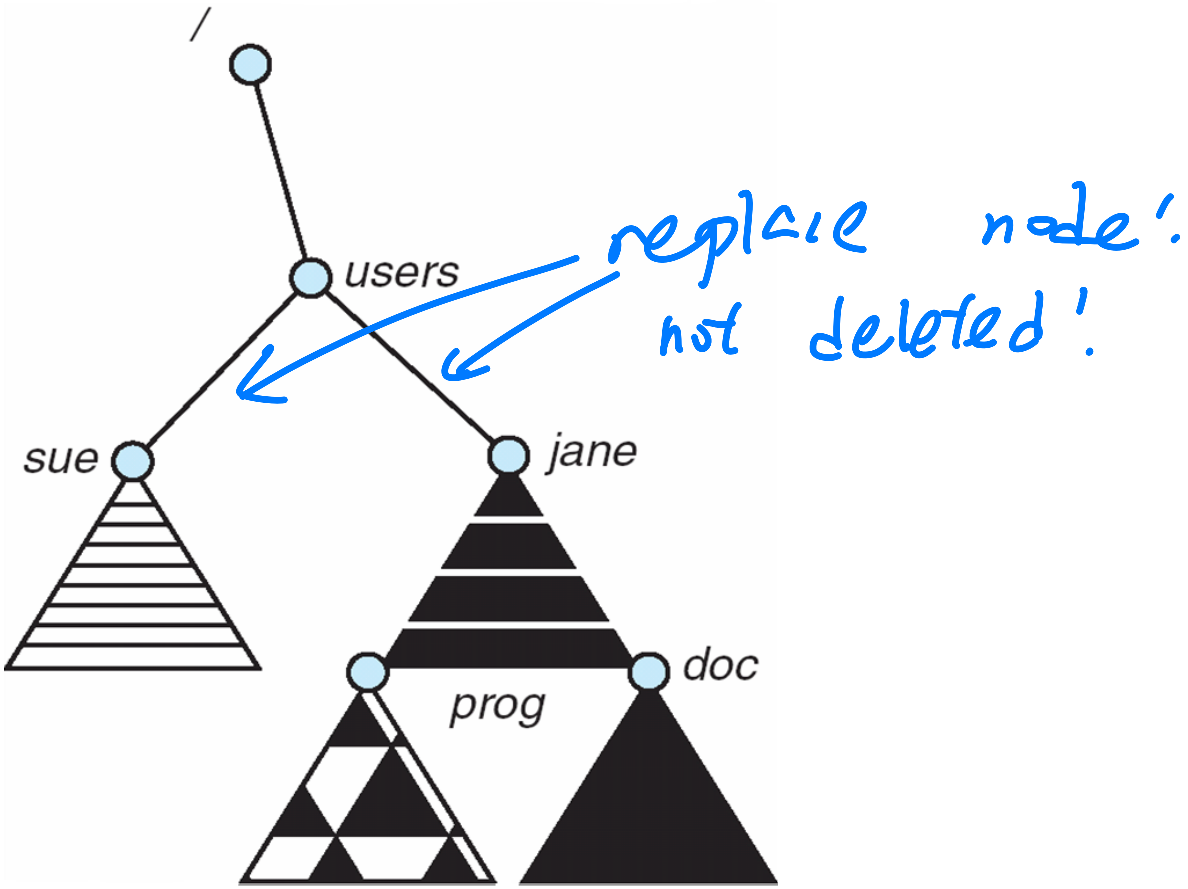
now be replaced. temporary not accessible

↓

but no worry, still here after mounting, 不会 lost!

Mount Point

- Volume is mounted at /users



File Sharing

3 students SP

- Sharing of files in multi-user systems is desirable
- Sharing may be done through a protection scheme

not must be protected

- In distributed systems, files may be shared across a network
  - Network File System (NFS) is a common distributed file-sharing method

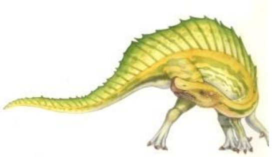
- With a multi-user system

add permission

  - User IDs identify users, allowing permissions and protections to be per-user

owner and groups

  - Group IDs allow users to be in groups, permitting group access rights
  - Owner of a file / directory
  - Group of a file / directory

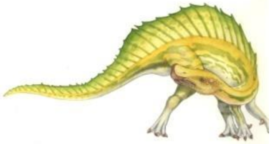


Protection

- File owner/creator of the file should be able to control:
  - what can be done
  - by whom

control

- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List



Access Lists and Groups

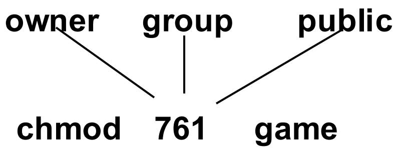
- Mode of access: read, write, execute } integer
- Three classes of users on Unix / Linux

read write execute (3 bits)

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

which one else can do what

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



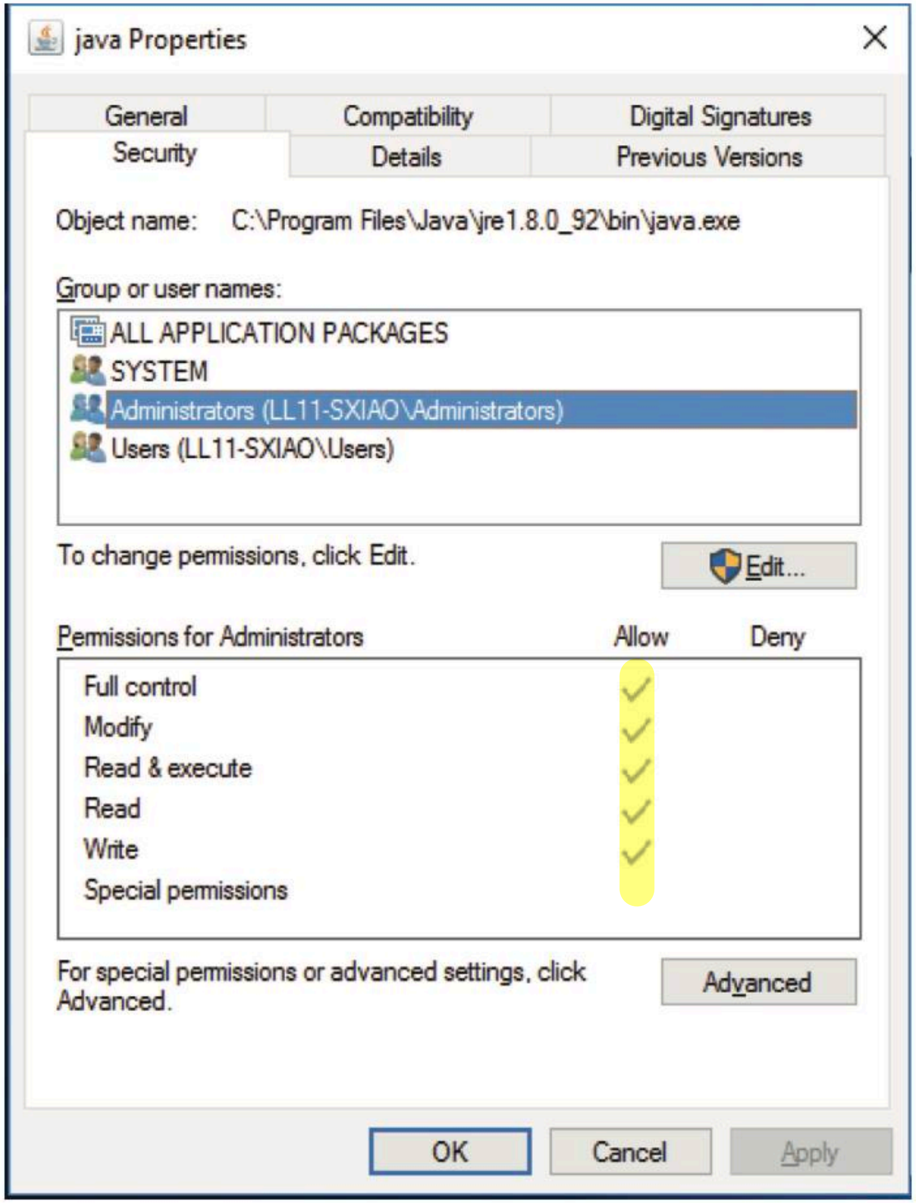
```
owner   group   public
chmod   761   game
```

Attach a group to a file

```
chgrp      G      game
```

↳ can have different groups

Windows 10 Access-Control List Management



java Properties

Object name: C:\Program Files\Java\jre1.8.0_92\bin\java.exe

Group or user names:

- ALL APPLICATION PACKAGES
- SYSTEM
- Administrators (LL11-SXIAO\Administrators)
- Users (LL11-SXIAO\Users)

To change permissions, click Edit.

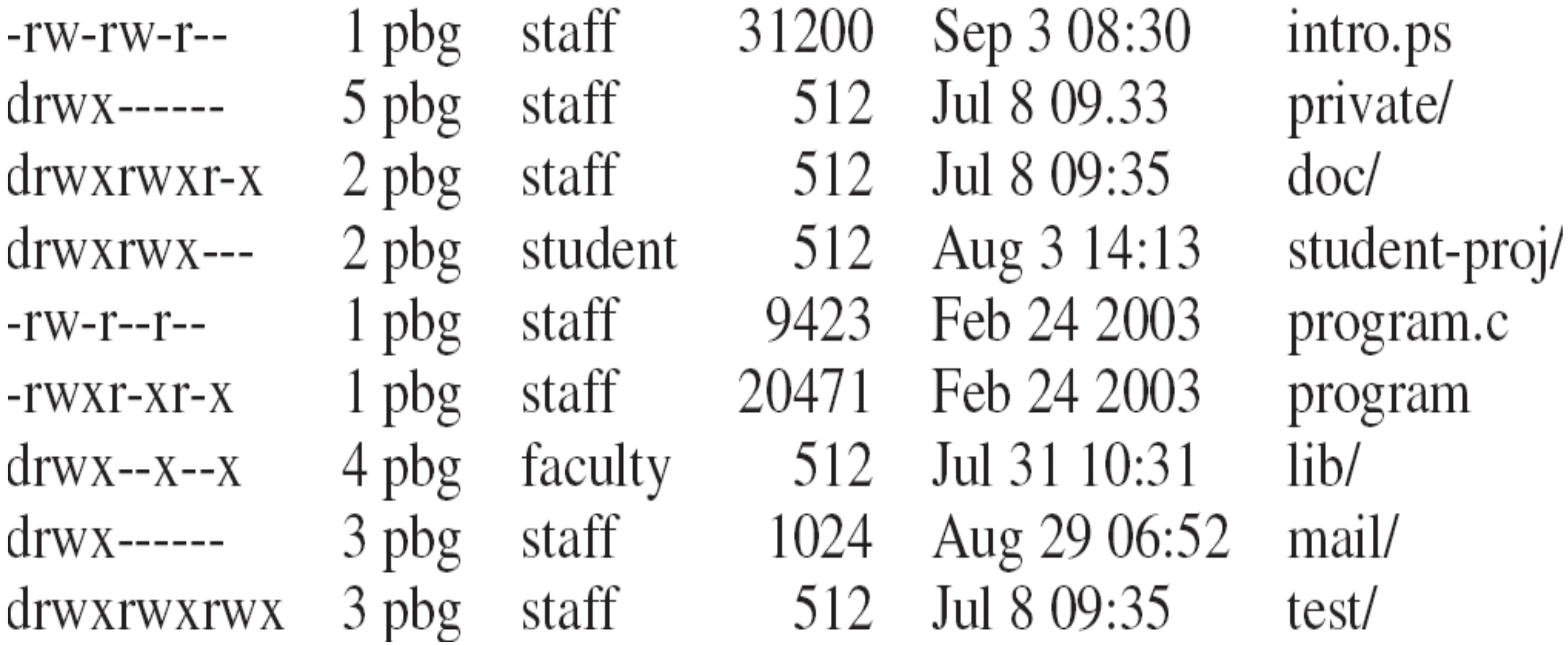
Permissions for Administrators	Allow	Deny
Full control	✓	
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced.

Almost like Linux, but Linux only here read write excute!

A Sample UNIX Directory Listing

d=directory
- =folder



```
-rw-rw-r--    1 pbg   staff     31200   Sep 3 08:30    intro.ps
drwx------    5 pbg   staff       512   Jul 8 09.33    private/
drwxrwxr-x    2 pbg   staff       512   Jul 8 09:35    doc/
drwxrwx---    2 pbg   student     512   Aug 3 14:13    student-proj/
-rw-r--r--    1 pbg   staff      9423   Feb 24 2003    program.c
-rwxr-xr-x    1 pbg   staff     20471   Feb 24 2003    program
drwx--x--x    4 pbg   faculty     512   Jul 31 10:31   lib/
drwx------    3 pbg   staff      1024   Aug 29 06:52   mail/
drwxrwxrwx    3 pbg   staff       512   Jul 8 09:35    test/
```

End of Chapter 13