Chapter 10: File-System Interface

Prof. Li-Pin Chang

CS@NYCU

Chapter 10: File-System Interface

- File Concept
- Directory Structure
- File and directory operations
- File aliasing
- File-System Mounting
- File permission and protection

Objectives

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

File Concept

- A computer resource to write data to and read data from storage device
- A contiguous logical address space

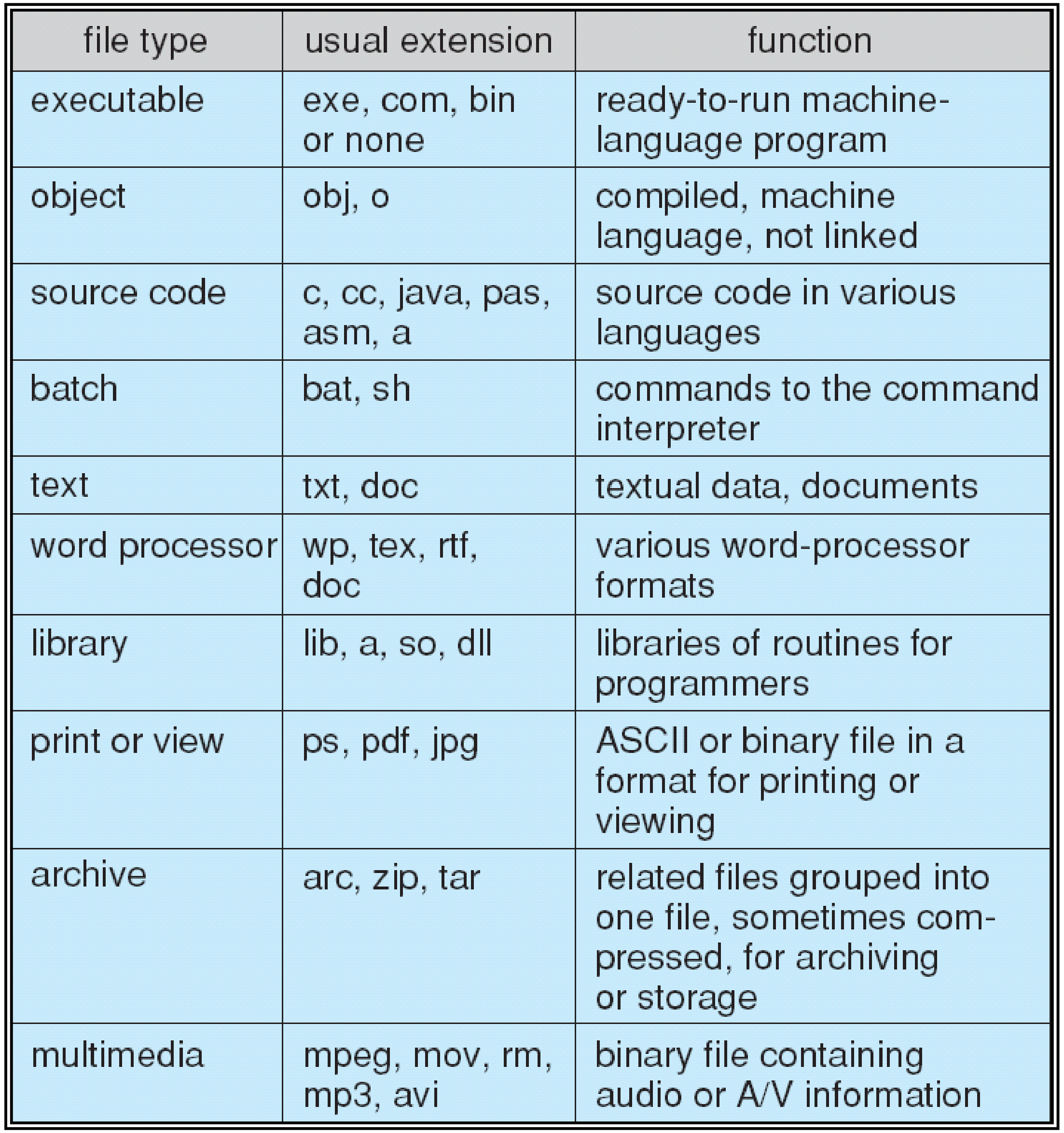
UNIX file types:

- Regular files
- Device files (device node)
- Directory files
- Links

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
  - Regular, directory, device, link (system functionality)
  - .c , .exe, .bat (user purpose)
- 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

File Types – Name, Extension



file type	usual extension	function
executable	exe, com, bin or none	ready-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, rtf, doc	various word-processor formats
library	lib, a, so, dll	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	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

File Operations

Function	**description**
fopen()	create a new file or open a existing file
fclose()	closes a file
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file
fread()	reads a number of bytes from a file
fwrite()	writes a number of bytes to a file
fseek()	set the position to desire point
link()	make a new name for a file
unlink()	decrement the reference count of a file (delete on ref=0)

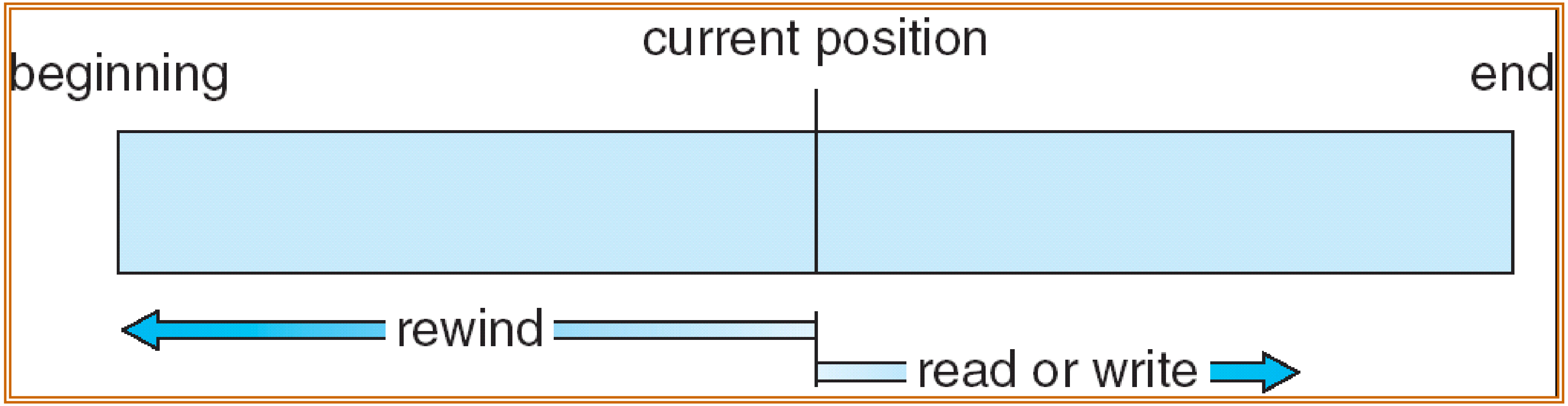
Why Opening/Closing Files

- Information required to manage opened files:
    - File pointer: pointer to last read/write location, per process that has the file open
    - File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it (e.g., removal of USB drives)
    - Disk location of the file
    - Access rights: per-process access mode information
- These are called "metadata", i.e., data of data
- The file system caches metadata when opening files for efficient operations; it also flushes modified metadata to disk when closing files

fopen(): Binary or Text?

- fopen(“abc.txt”,”r+t”);
- fopen(“xyz.mp3”,”rb”);
- Text mode
  - Translate Ctrl-Z (1A) into EOF
  - Translation between \r\n and \n for different OSes
    - UNIX: \n (0A) Windows \r\n (0D 0A)
  - Possibility of filtering out the MSB (only 7 LSBs used)
- Binary mode
  - Raw input

File Accessing Model

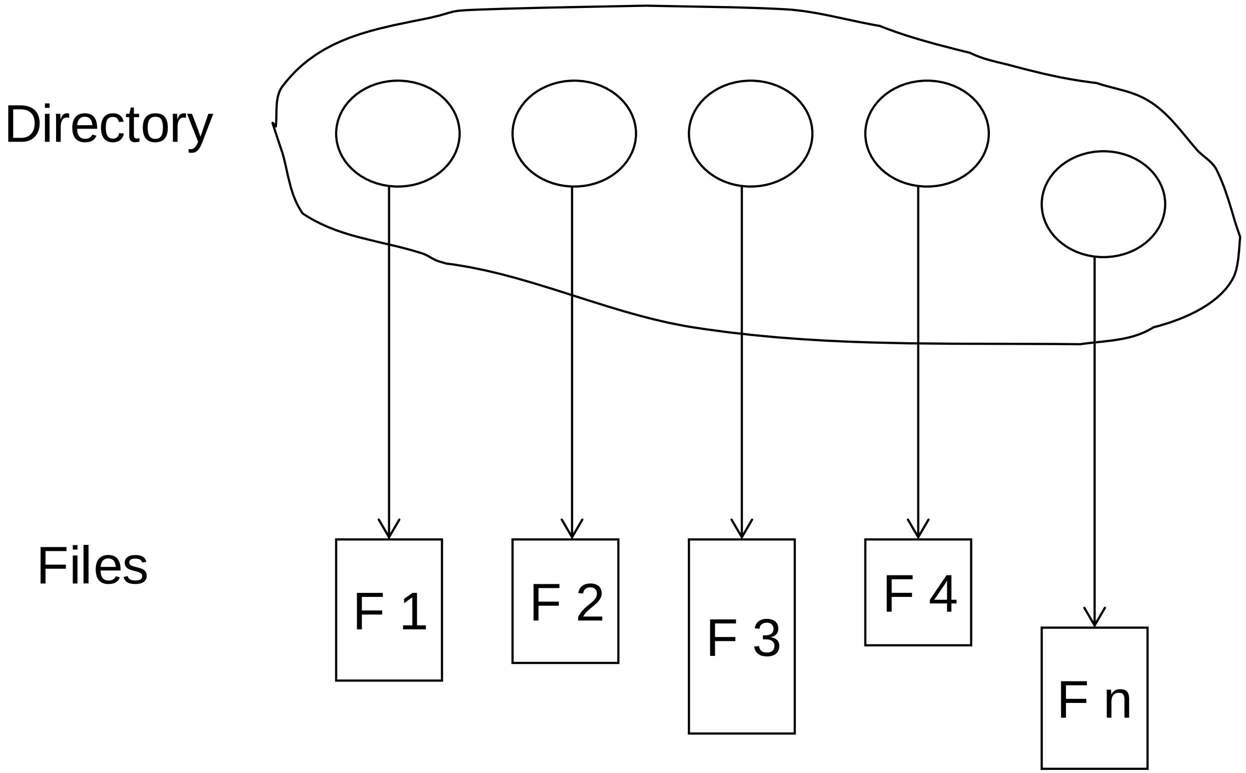


Device Node

- Commonly appear under the /dev directory
- Can be manually created using `mknode` command, with proper device major-minor #’s assigned
- Device drivers register themselves to the kernel using the device major-minor #’s
- open(), close(), read(), write() a device node will communicate with the device driver registered with the same major-minor #’s as that of the device node
- Example: `open() on /dev/sda` → `#M8m0`

Directory

- A collection of nodes containing information about all files

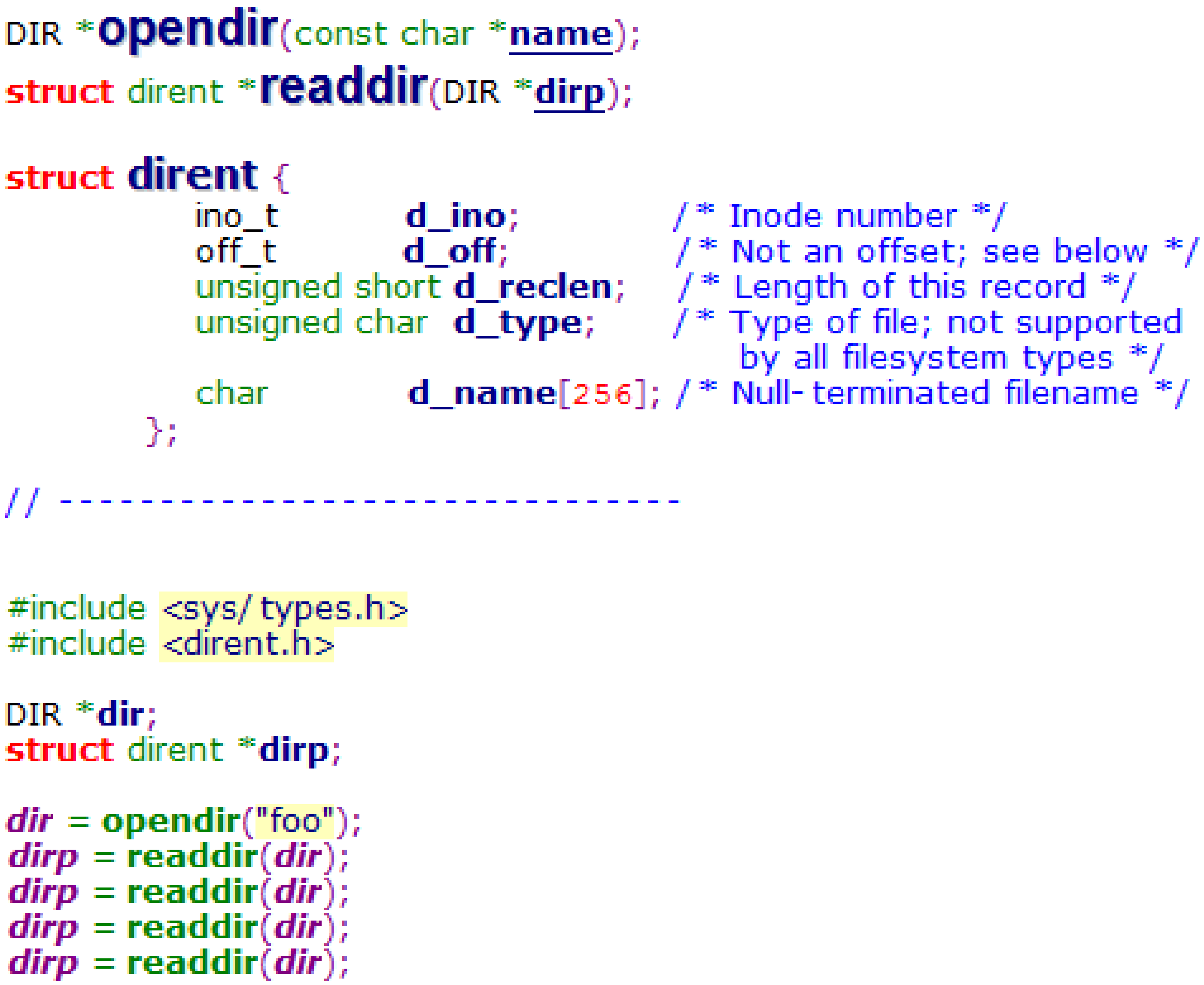


Directory itself is a file, too

Directory Operations

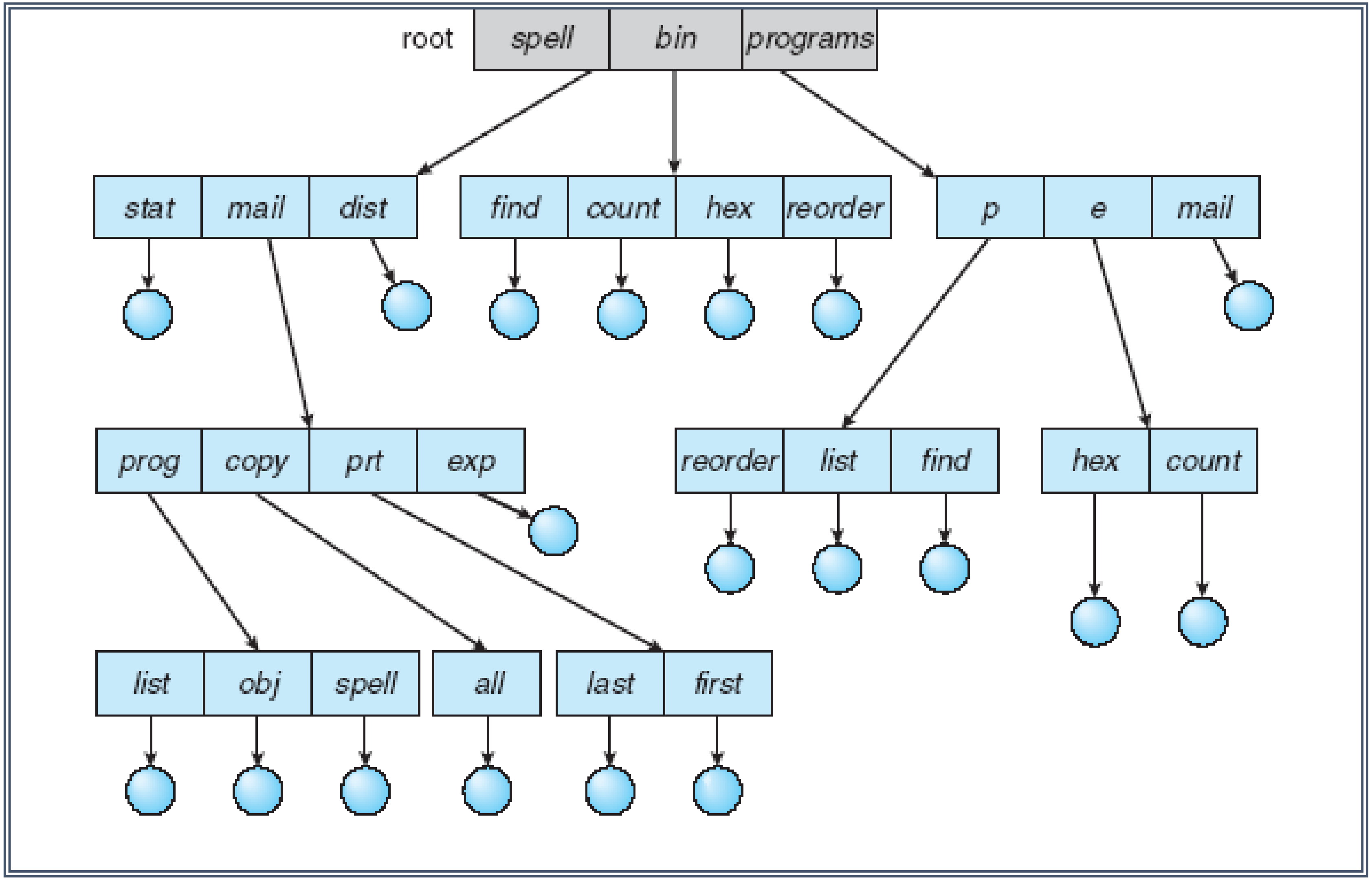
- Search for a file
- Create a file
- Delete a file
  - If the deleted file is a directory?
    - Recursively delete all its files and sub-directories?
  - If the deleted file is a regular file?
- Directory enumeration (listing)
- Rename a file

Open and read a directory



```
DIR *opendir(const char *name);
struct dirent *readdir(DIR *dirp);

struct dirent {
          ino_t          d_ino;            /* Inode number */
          off_t          d_off;            /* Not an offset; see below */
          unsigned short d_reclen;         /* Length of this record */
          unsigned char  d_type;           /* Type of file; not supported
                                              by all filesystem types */
          char           d_name[256]; /* Null- terminated filename */
       };

// ------------------------------


#include <sys/ types.h>
#include <dirent.h>

DIR *dir;
struct dirent *dirp;

dir = opendir("foo");
dirp = readdir(dir);
dirp = readdir(dir);
dirp = readdir(dir);
dirp = readdir(dir);
```

Tree-Structured Directories



Tree-Structured Directories (Cont)

- The *current working directory (CWD)* environment variable (per process)
  - “.” and “..”
- Absolute or relative path name
- Traverse the file system

```
char *getcwd(char *buf, size_t size);
int chdir(const char *path);
```

Tree-Structured Directories (Cont)

- Creating a new file is done in current directory
- Delete a file

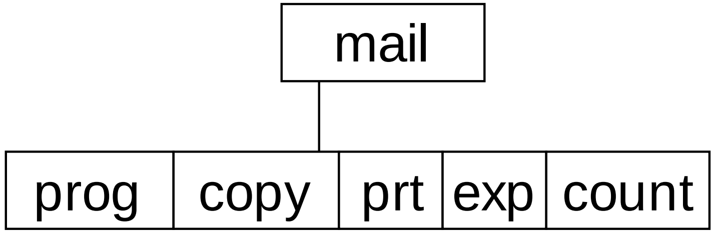
```
rm <file-name>
```

- Creating a new subdirectory is done in current directory

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

Example: if in current directory /mail

```
mkdir count
```



mail

prog copy prt exp count

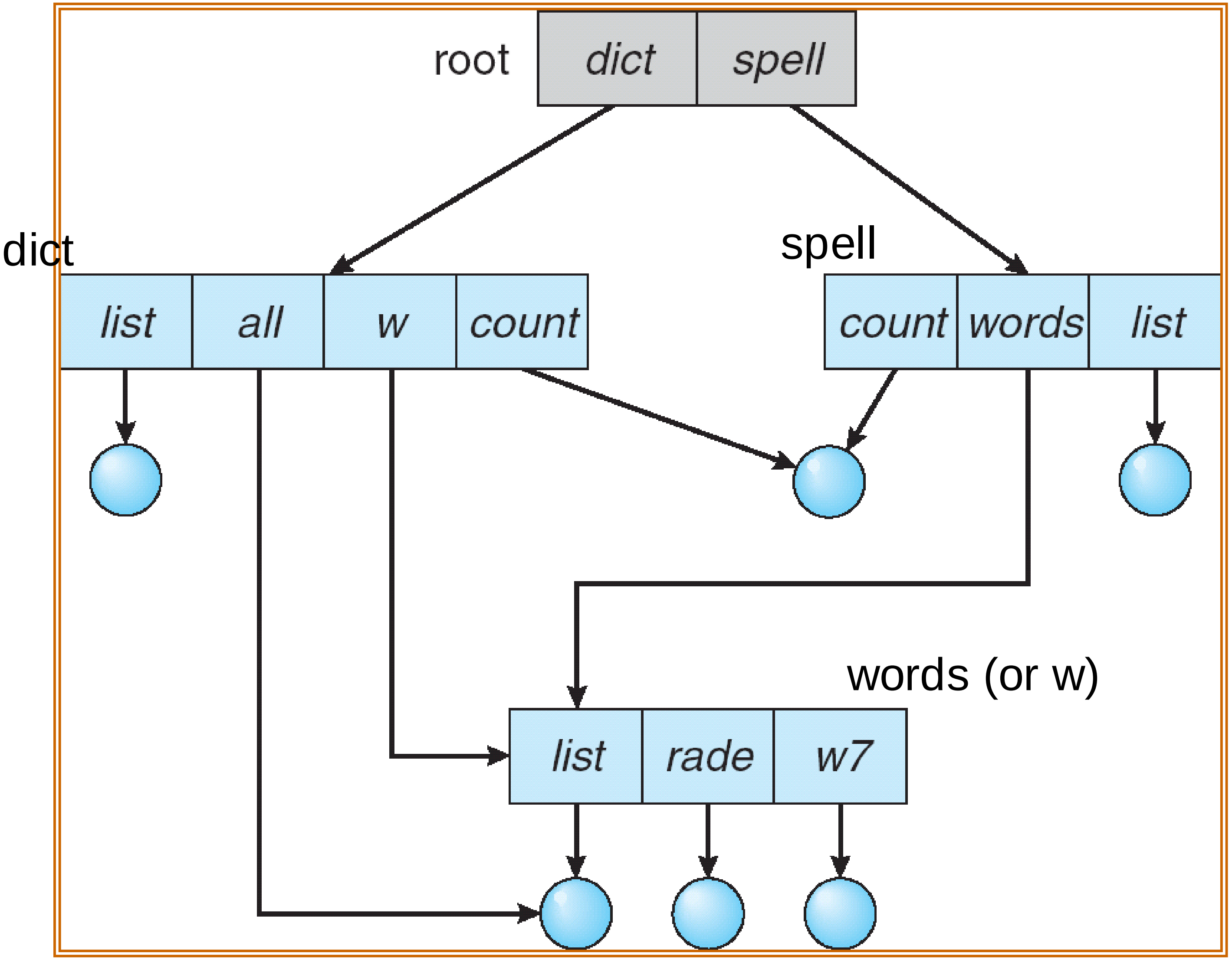
Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”
rm -r or del /s

File Aliasing (Link)

- A file may have two different names (alias)

- A file link
  - Another name of (pointer to) an existing file
  - Resolve the link – follow pointer to locate the file

Acyclic-Graph Directories

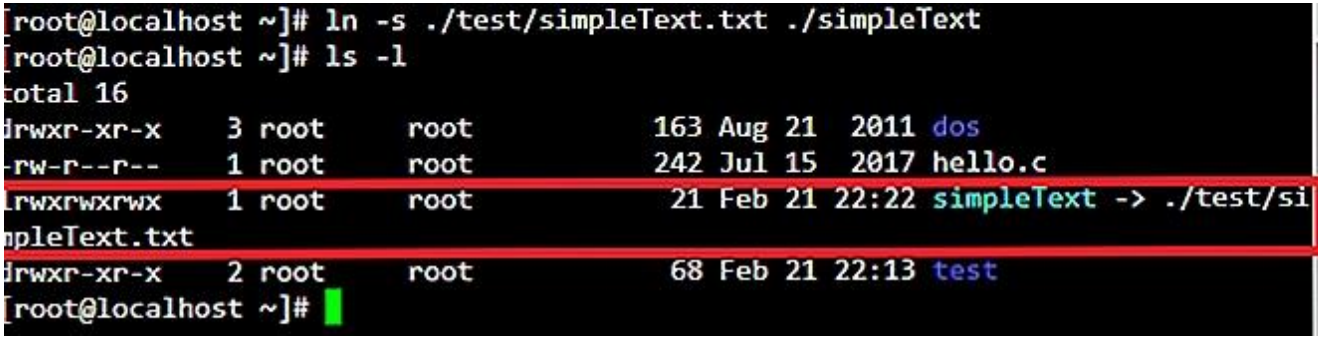


Softlinks

- Softlinks (symbolic link)
  - String substitution
  - Independent of file system
  - Appearing as a link file

- Usage
  - UNIX: ln -s [target] [link]
  - Windows (NTFS): junction.exe [link] [target]

https://tw .arip-photo.org/736330-how-to-list-symbolic-link-WLWBSA



```
[root@localhost ~]# ln -s ./test/simpleText.txt ./simpleText
[root@localhost ~]# ls -l
total 16
drwxr-xr-x    3 root     root          163 Aug 21  2011 dos
-rw-r--r--    1 root     root          242 Jul 15  2017 hello.c
lrwxrwxrwx    1 root     root           21 Feb 21 22:22 simpleText -> ./test/si
mpleText.txt
drwxr-xr-x    2 root     root           68 Feb 21 22:13 test
[root@localhost ~]#
```

Hardlinks

- Hardlinks
  - A link file that refers to the target file using file system internal location information
  - File-system-dependent
  - Nothing different from a regular file
  - The target file has a link count > 1; use unlink() to delete files
- Usage
  - UNIX: ln [target] [link]
  - Windows (NTFS): fsutil hardlink create [link] [target]

Problems with Aliasing

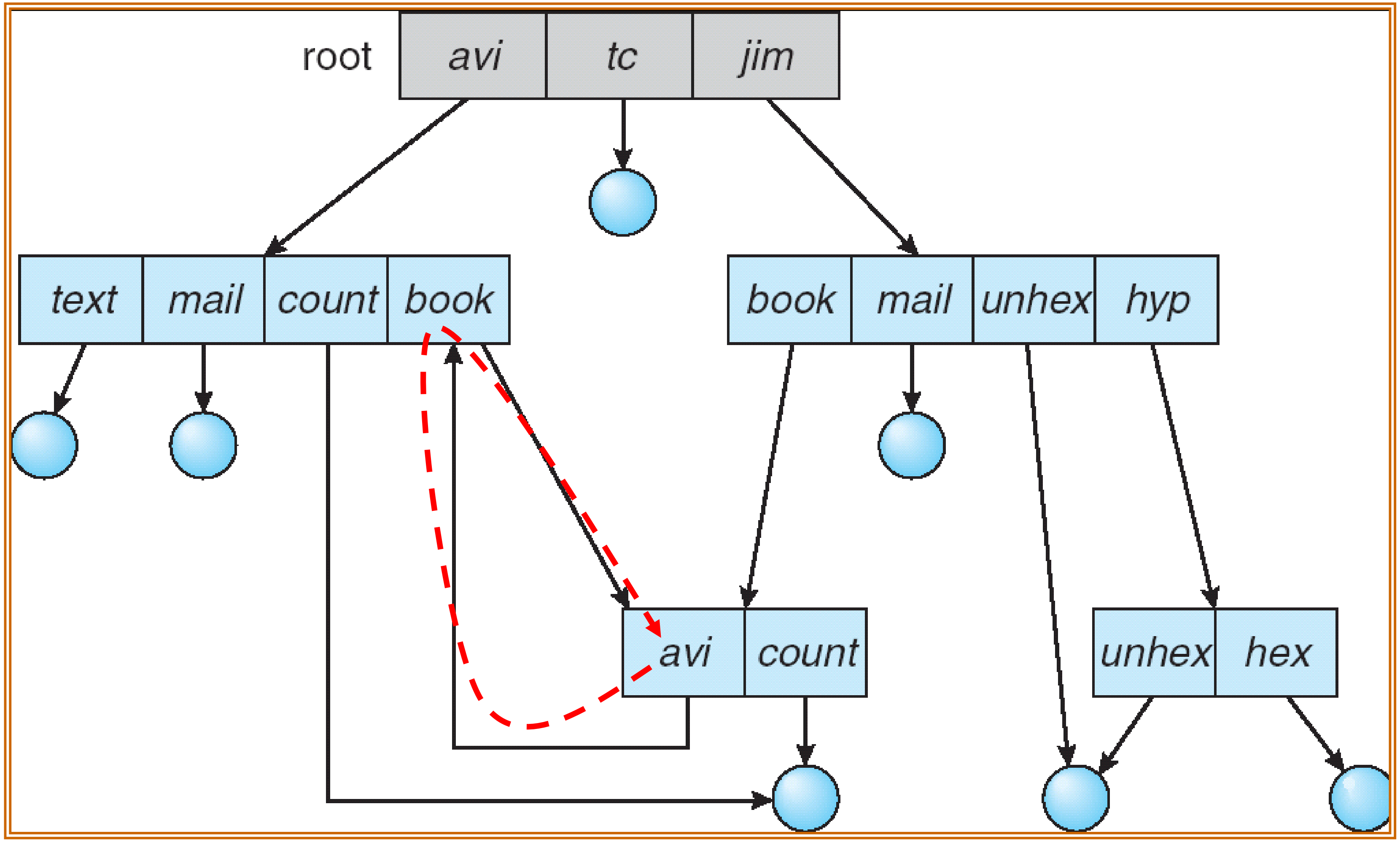
- Backup– Duplication problem
  - May duplicate files during backup
  - “cp -a” or “rsync” to preserve hard links as many as possible

Problems with Aliasing

- Loop – Endless file path
- Loops caused by hard links
  - Hard links to directories are forbidden in recent UNIX implementations
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK (less practical)
- Loops caused by soft links
  - Soft links to directories are still possible
  - Linux: Keep a time-to-live counter (e.g., 40)
  - Windows: Limiting the pathname length (~ 260 chars)

Modern UNIX implementations do not allow *hard links* to directories! (soft links are still possible)

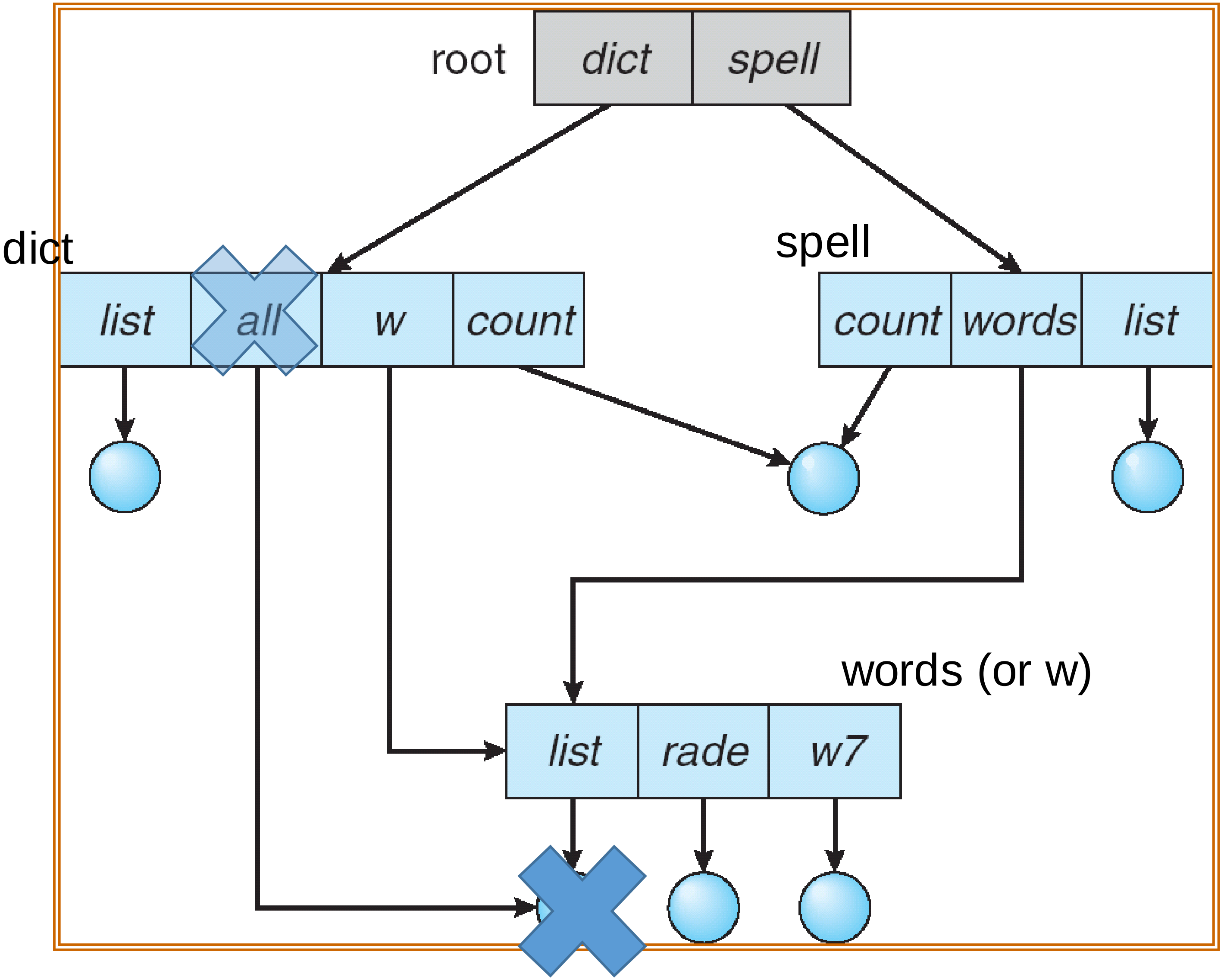
Loop in directories



Problems with Aliasing

- Deletion– Dangling pointer problem
  - Deleting "all" in dict makes the symbolic link "list" dangling
- Solutions:
  - Hard links require proper management of dangling pointers as referring to undefined storage address may expose security issues
  - ✓(hard link) Backpointers, so we can delete all pointers
  - ✓(hard link) Entry-hold-count solution (unlink() in UNIX)
  - Soft links are less problematic
  - ✓(sym link) Leave a symboic link dangling

Acyclic-Graph Directories



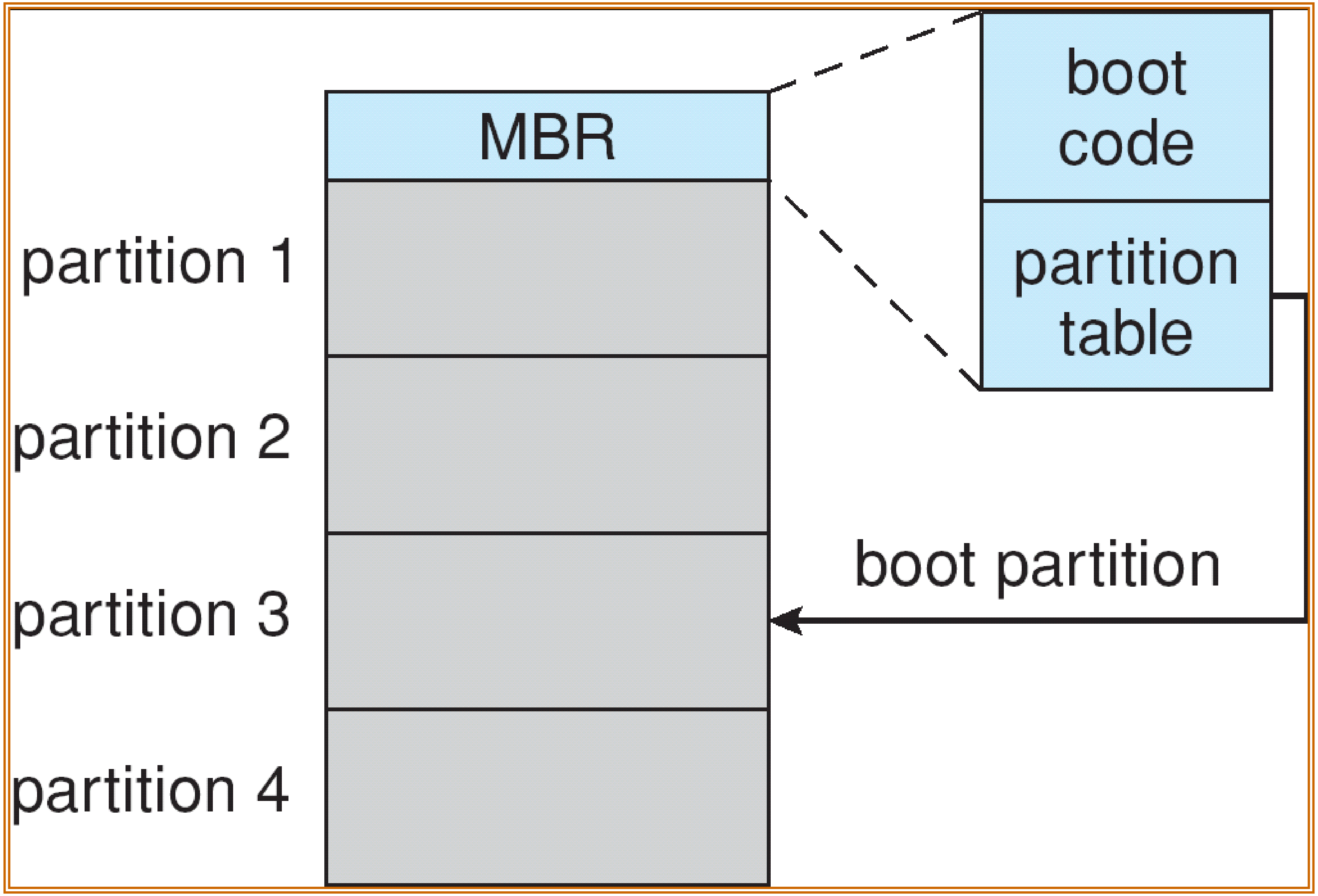
Problems with Aliasing

- Dangling pointers
- Softlink (symbolic link)
  - Simply leave the symbolic link dangling
  - /bin/ls → /sbin/ls
- Hardlink
  - link is established inside the file system
  - Keep a reference count
  - Creating hardlink to the file: +count
  - Removing a hardlink to the file: -count
  - When count==0: remove the file

Soft link vs. Hard link: Revisit

- Softlink
  - Can span over different file systems
  - Dangling pointer problem
- Hardlink
  - More efficient than soft links
  - Can not span over different file systems
  - Often confusing, cannot tell which file is the “original one”

Legacy MBR Partitions



Partitioning a Disk

- The very first step of preparing a hard drive
- Use fdisk or other GUI utilities
- Partitions can be formatted into different file systems or used as a swap device
- The bios loads the MBR, which in turn loads the next loader in the boot partition
    - An OS or a boot manager
- MBR partition tables are being replaced by GPT, which allows larger partition sizes and unlimited partitions

Formatting a Partition

- To use a disk to hold files, the operating system still needs to record its own data structures on the disk.
  - Logical formatting, high-level formatting or "making a file system".
  - Writing file system metadata
- Low-level formatting, or physical formatting — Dividing a disk into sectors that the disk controller can read and write.
  - Remapping bad tracks to spare tracks
  - Zoned-bit encoding

Mounting a File System

- A file system must be mounted before it can be accessed
- A unmounted file system is mounted at a mount point
- Mounting a file system
  - `mount -t ext4 /users /dev/hda1`
  - Specify the **file system type**
  - Find the file-system superblock in the partition device node
  - Specify the **mounting point** of the file-system naming space

Protection

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

- Types of access
  - Read
  - Write
  - Execute
  - Append (regards to disk space)
  - Delete
  - List

FTP

File Sharing – Multiple Users

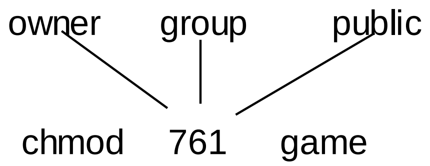
- User IDs identify users, allowing permissions and protections to be per-user
- Group IDs allow users to be in groups, permitting group access rights

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

			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



owner group public

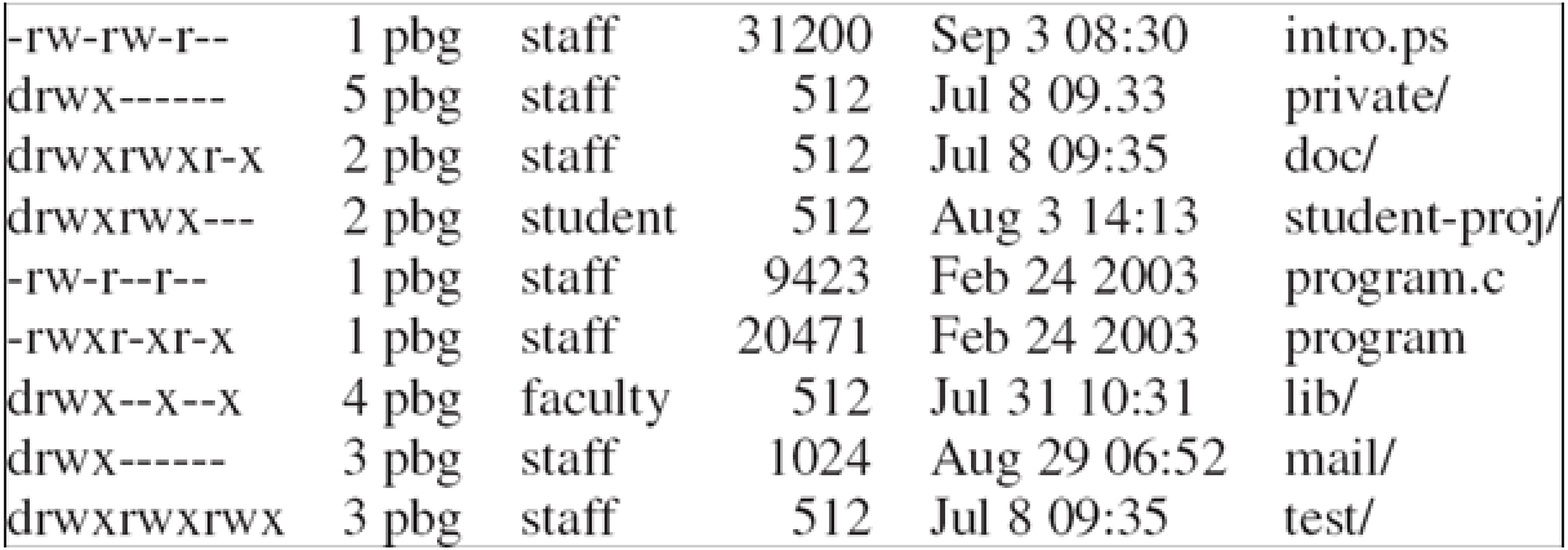
chmod 761 game

UNIX File Permission Management Utilities

- adduser: create a user
- mkgrp: create a group
- addgrp: add a user to a group
- chown: change the owner of a file
- chgrp: change the group of a file
- chmod: change file permissions

- Users are managed by /etc/password
- Groups are managed by /etc/group

A Sample UNIX Directory Listing



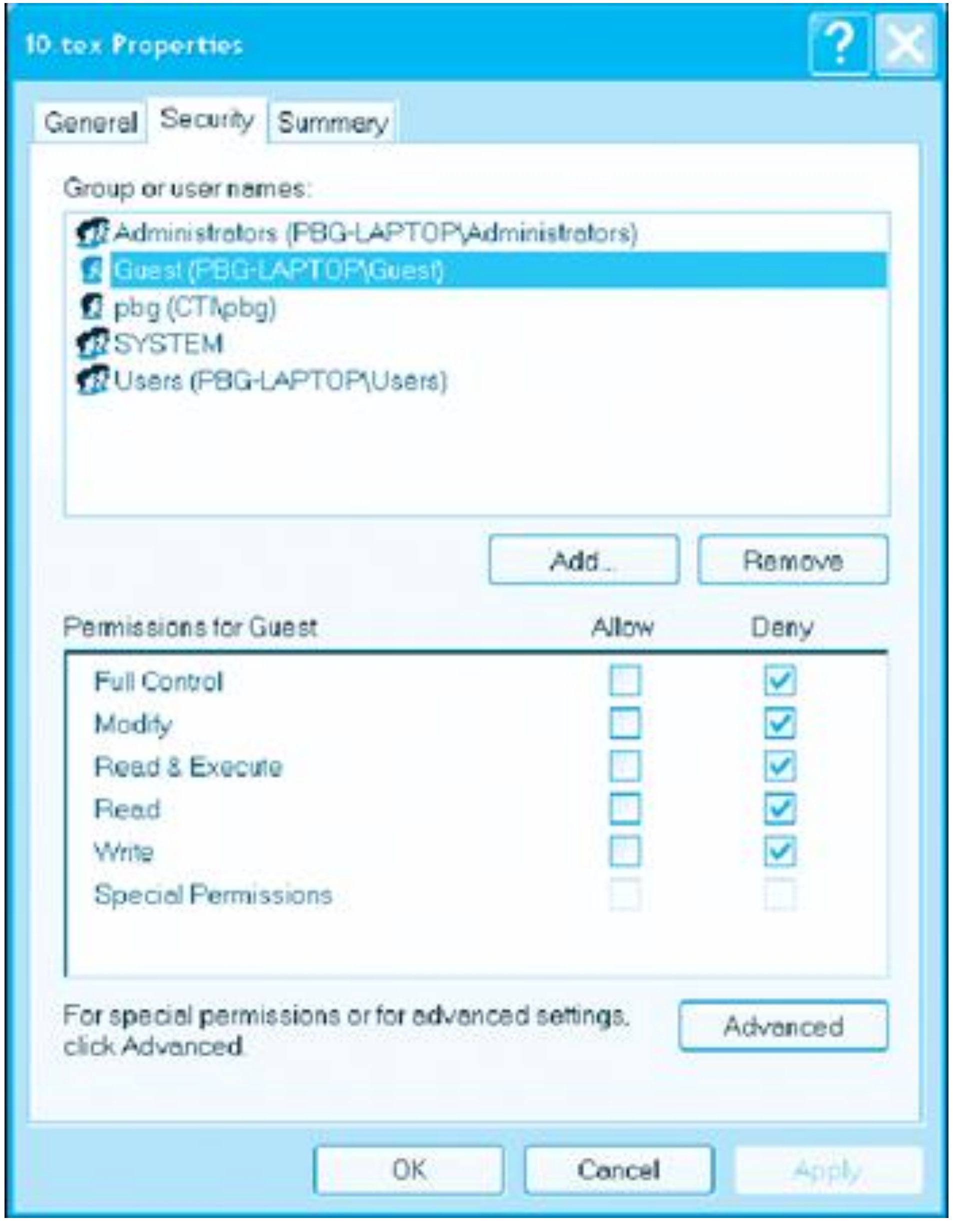
```
-rw-rw-r--    1 pbg   staff      31200   Sep 3 08:30    intro.ps
drwx------    5 pbg   staff        512   Jul 8 09.33    private/
drwxrwxr-x    2 pbg   staff        512   Jul 8 09:35    doc/
drwxrwx---    2 pbg   student      512   Aug 3 14:13    student-proj/
-rw-r--r--    1 pbg   staff       9423   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/
```

[Permission] [hard link count][Owner] [group] [filesize] [date] [filename]

- Regular file: link count >=1, file is deleted when link count =0
- A directory: link count is 2+n
  - 1 from its own directory entry + 1 from "." of itself
  - n from ".." of all its sub-directories

A directory with the permission "x" = the directory can be searched/entered

Windows XP Access-control List Management



Group or user names:

- Administrators (PBG-LAPTOP\Administrators)
- Guest (PBG-LAPTOP\Guest)
- pbg (CTI\pbg)
- SYSTEM
- Users (PBG-LAPTOP\Users)

Permissions for Guest	Allow	Deny
Full Control	□	☑
Modify	□	☑
Read & Execute	□	☑
Read	□	☑
Write	□	☑
Special Permissions	□	□

For special permissions or for advanced settings, click Advanced.

End of Chapter 10