CMSC421: Principles of Operating Systems

Nilanjan Banerjee

Assistant Professor, University of Maryland
Baltimore County
nilanb@umbc.edu
http://www.csee.umbc.edu/~nilanb/teaching/421/

Acknowledgments: Some of the slides are adapted from Prof. Mark Corner and Prof. Emery Berger's OS course at Umass Amherst

Announcements

- Homework 3 is out (due nov 27th)
- Will discuss midterm at the end of class
- Will have a session on Project 2 on Wednesday (after talking about file systems)

File Systems (Lets start with the disk)

- Disk (hard drive) is a block device
  - You can read and write blocks from the hard drive
  - E.g. give me block number 50, or block number 100
  - Blocks are usually 1KB in size

- You can also create logical block sizes
  - E.g. using the command dd
  - Example of creating files without file system (demo?)

- You can write file systems for block devices (e.g., cdrom, harddrive, flash drives)
- Another type of devices is character devices?
  - Examples?
  - What are the major differences between char and block devices

File system structure and file manipulations

- File systems are made of directories
  - In linux the root directory is /

- All directories are children of some directory
  - Directories follow a tree structure

- Directories consist of files

- Files are associated with two things
  - Name of the file
  - Pointer to the data stored in the file

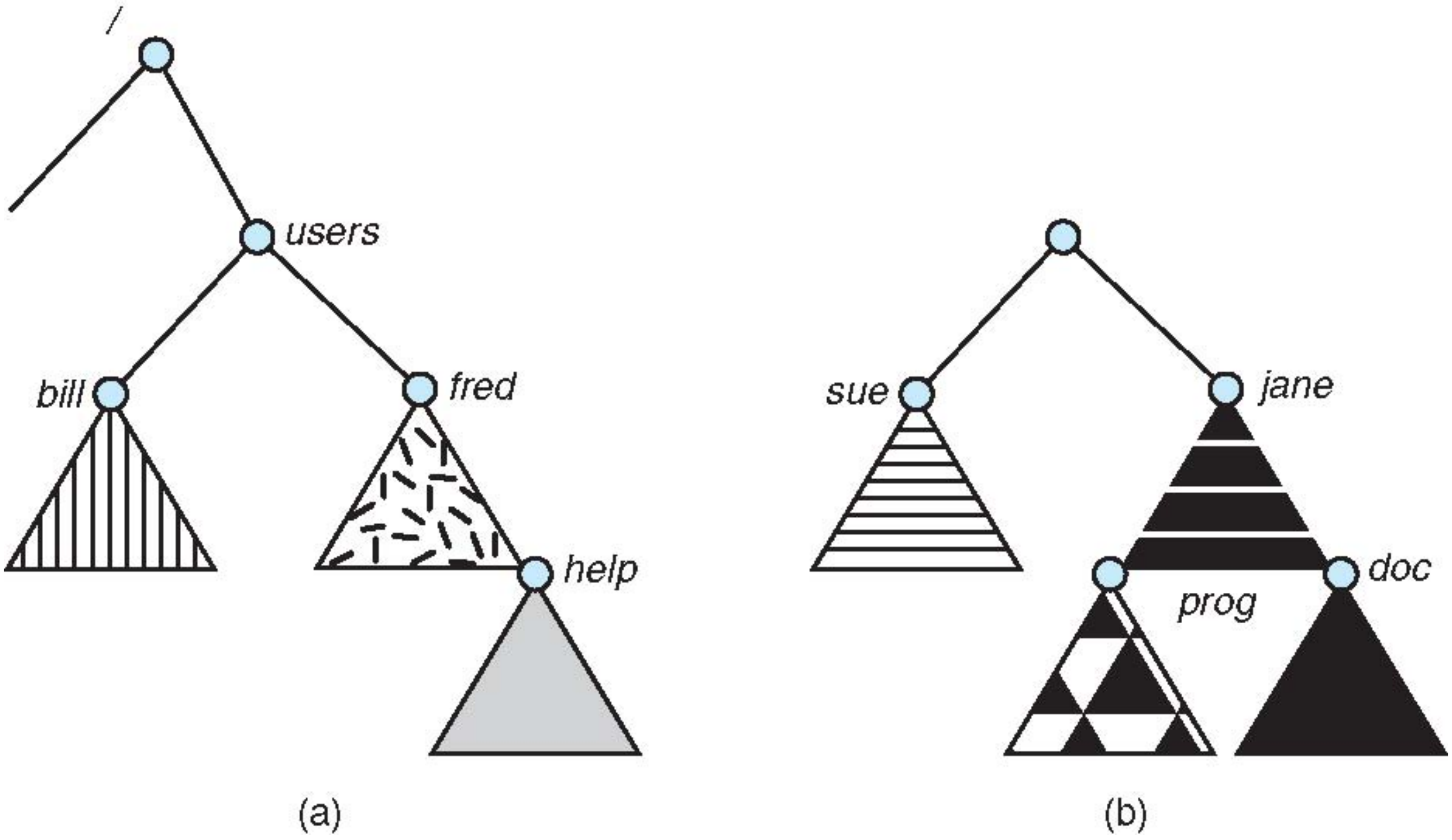
Concept of virtual file systems (primer)

- In linux you can use the concept of mounting to add a custom file system to your directory tree

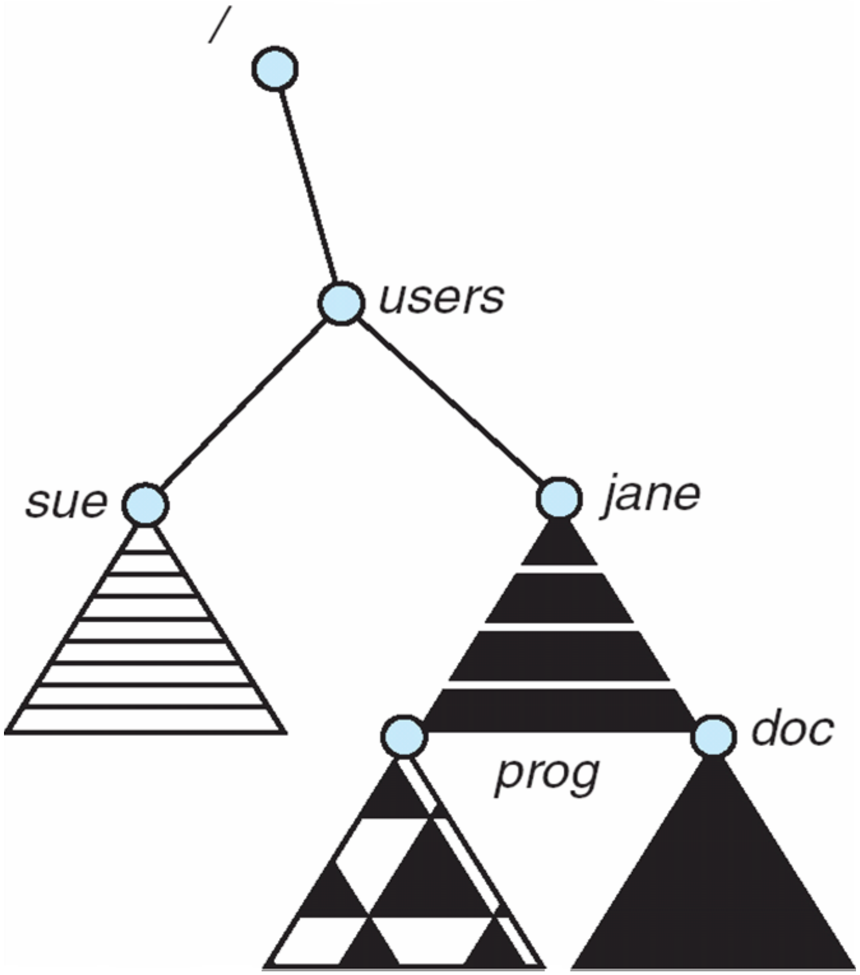
- You can also mount directories on remote machines onto your file system tree

- Lets take a look at a demo

Virtual FSs allow mount points



Mount Point



File Concept

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

File Structure

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures (pdf or doc format)
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program

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

File Operations

- File is an **abstract data type**
- **Create**
- **Write**
- **Read**
- **Delete**
- **Truncate**
- *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

Open Files

- Several pieces of data are needed to manage open 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
  - Access rights: per-process access mode information

Common file operations (you might be familiar with)

- Creating a file
- Open and reading a file
- Deleting a file
- Creating a soft link to a file
- Creating a hard link to a file
- Append a file
- Read last few bytes/characters of a file

File Protection

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

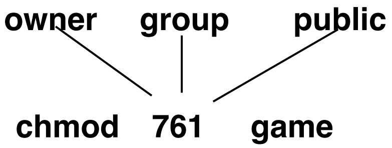
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

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

			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.



owner group public

chmod 761 game

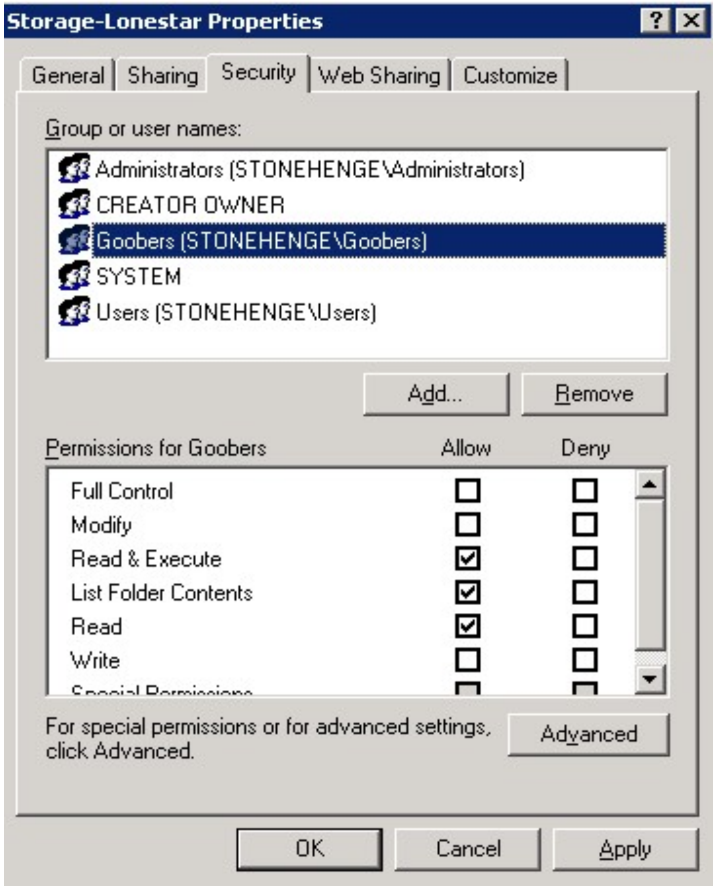
Access Control - chmod

- Can read bits via ls, set bits via chmod

```
elnux14> ls -l ack.scm
-rw-r-----  1 emery fac 197 Feb 25 15:19 ack.scm
elnux14> chmod -r ack.scm
elnux14> ls -l ack.scm
--w-------  1 emery fac 197 Feb 25 15:19 ack.scm
elnux14> cat ack.scm
cat: ack.scm: Permission denied
```

Access Control Lists (ACLs) in Windows

- ACLs are more expressive
  - Specify different rights per user or group
  - Opinion: one of the biggest UNIX problems



Storage-Lonestar Properties

Group or user names:

Administrators (STONEHENGE\Administrators)
CREATOR OWNER
Goobers (STONEHENGE\Goobers)
SYSTEM
Users (STONEHENGE\Users)

Permissions for Goobers	Allow	Deny
Full Control	□	□
Modify	□	□
Read & Execute	☑	□
List Folder Contents	☑	□
Read	☑	□
Write	□	□
Special Permissions	□	□

For special permissions or for advanced settings, click Advanced.

Access Methods

- **Sequential Access**

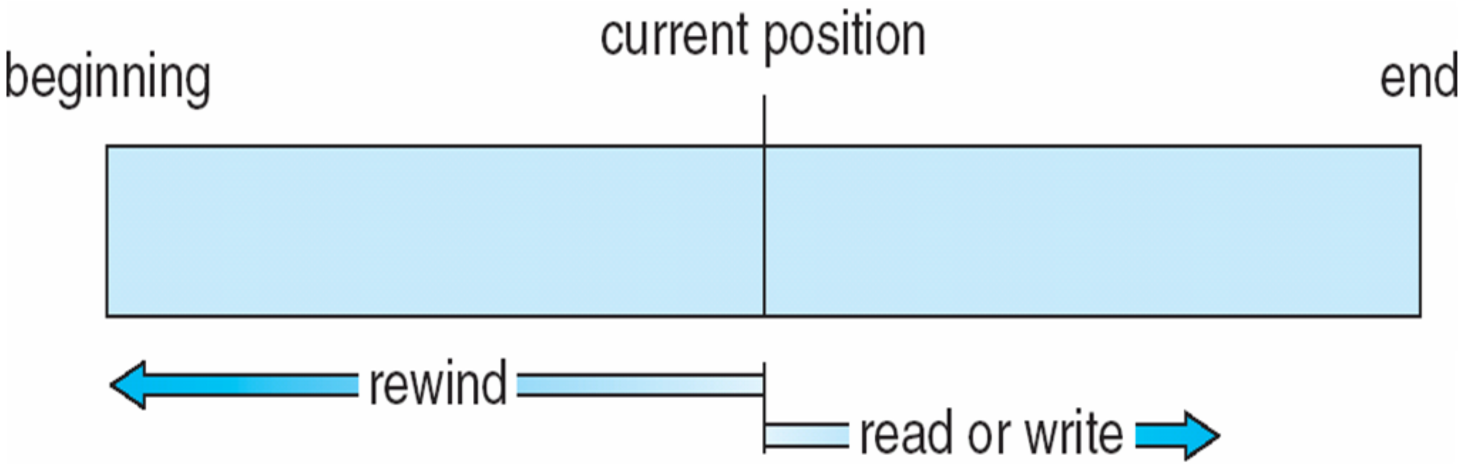
  read next
  write next
  reset
  no read after last write
  (rewrite)

- **Direct Access**

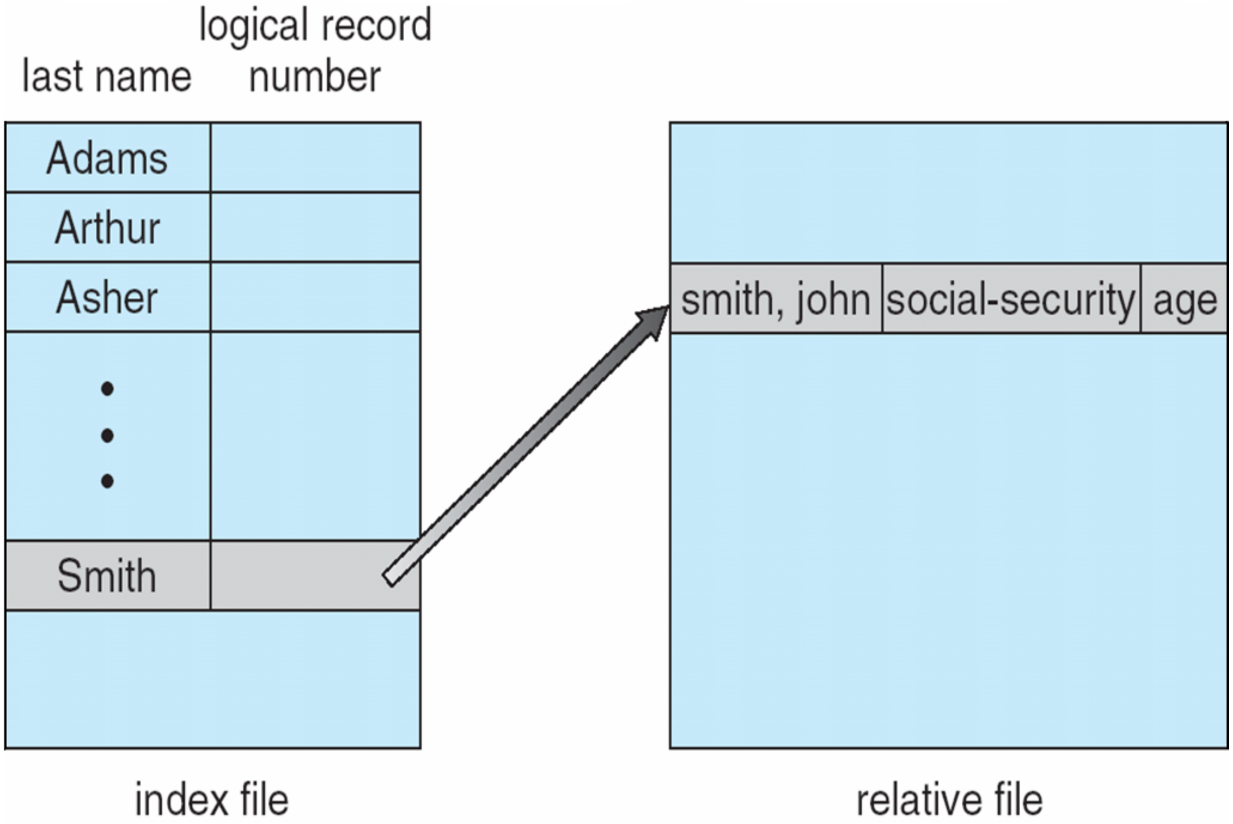
  read *n*
  write *n*
  position to *n*
  read next
  write next
  rewrite *n*

n = relative block number

Sequential-access File



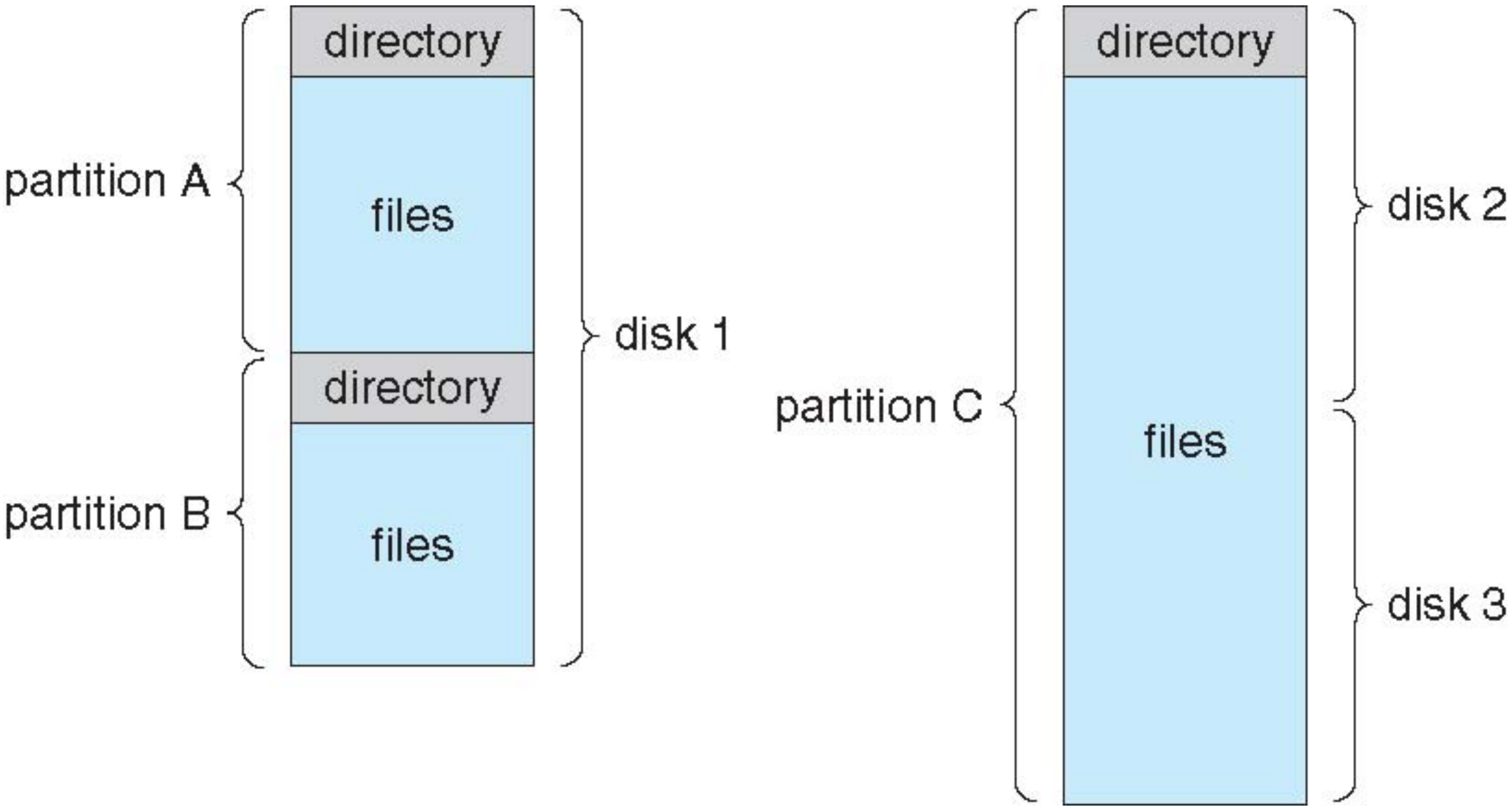
Example of Index and Relative Files



Directories

- Directory – just special file
  - Contains metadata, filenames
  - Store pointers to files
- Typically **hierarchical tree**
  - odd exposure of data structure to user

A Typical File-system Organization



Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Organize the Directory (Logically) to Obtain

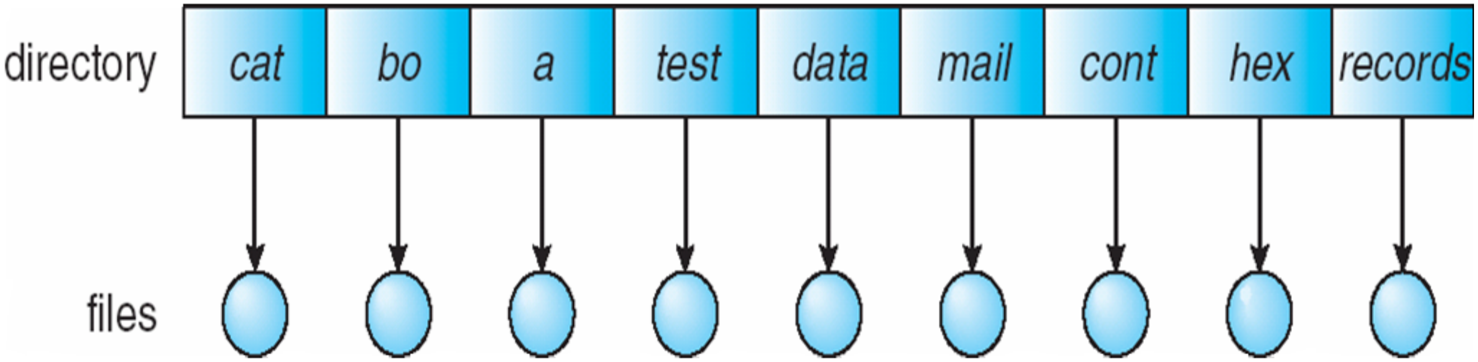
- Efficiency – locating a file quickly

- 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, …)

Single-Level Directory

- A single directory for all users

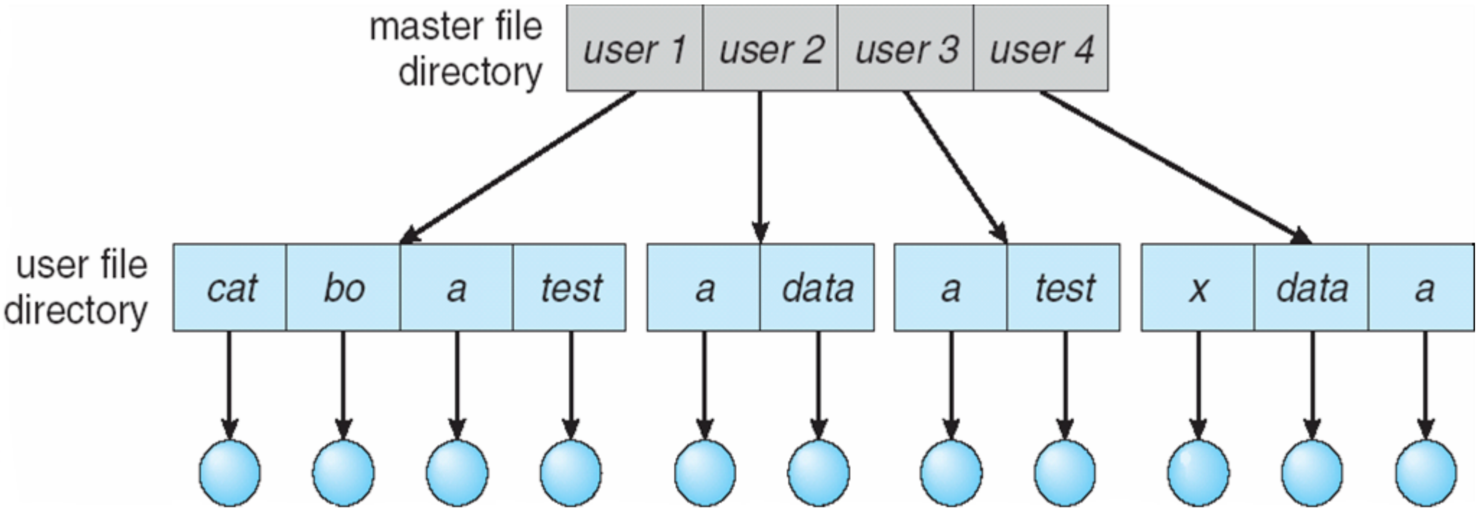


Naming problem

Grouping problem

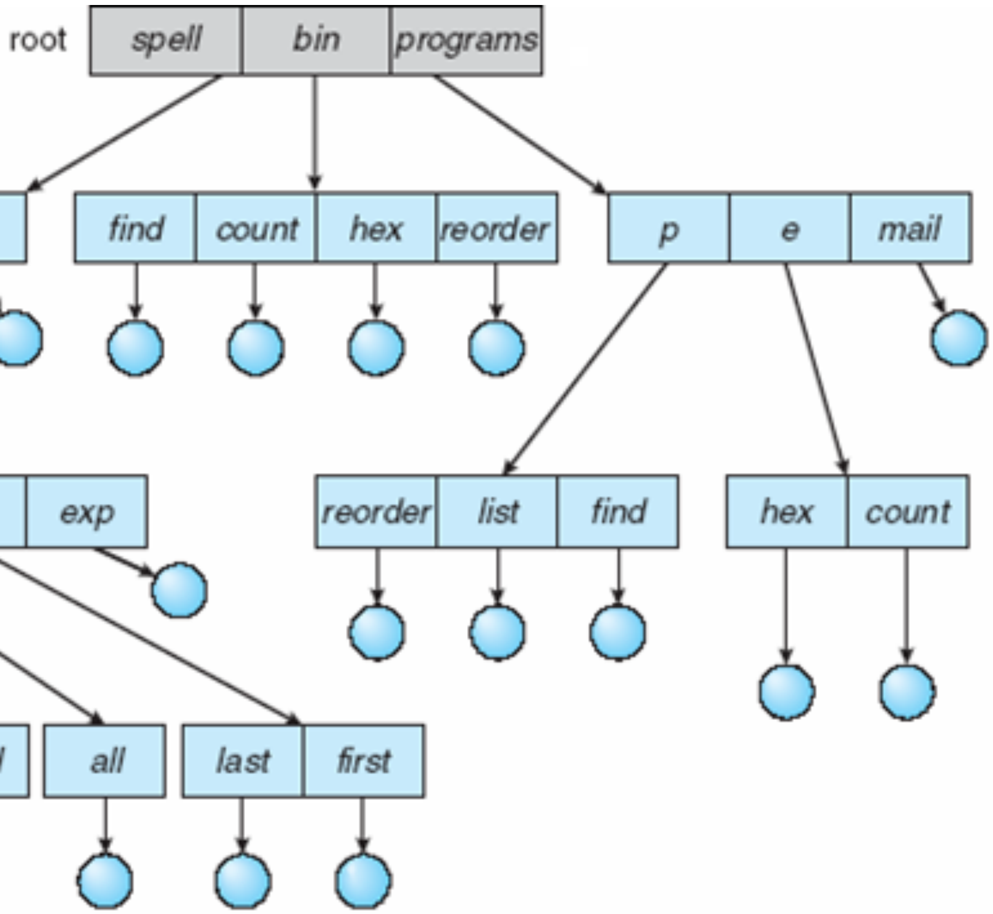
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories



Tree-Structured Directories (Cont.)

- Efficient searching

- Grouping Capability

- Current directory (working directory)
  - cd /spell/mail/prog

Tree-Structured Directories (Cont)

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

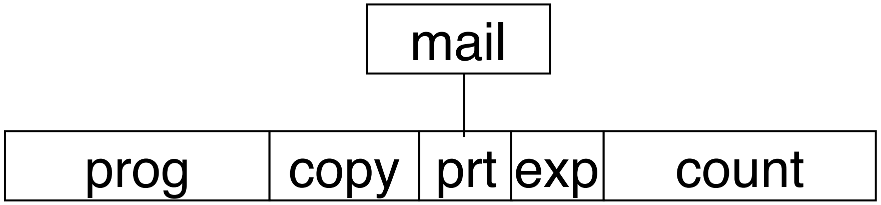
```
rm <file-name>
```

- Creating a new subdirectory is done in current directory

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

Example: if in current directory /mail

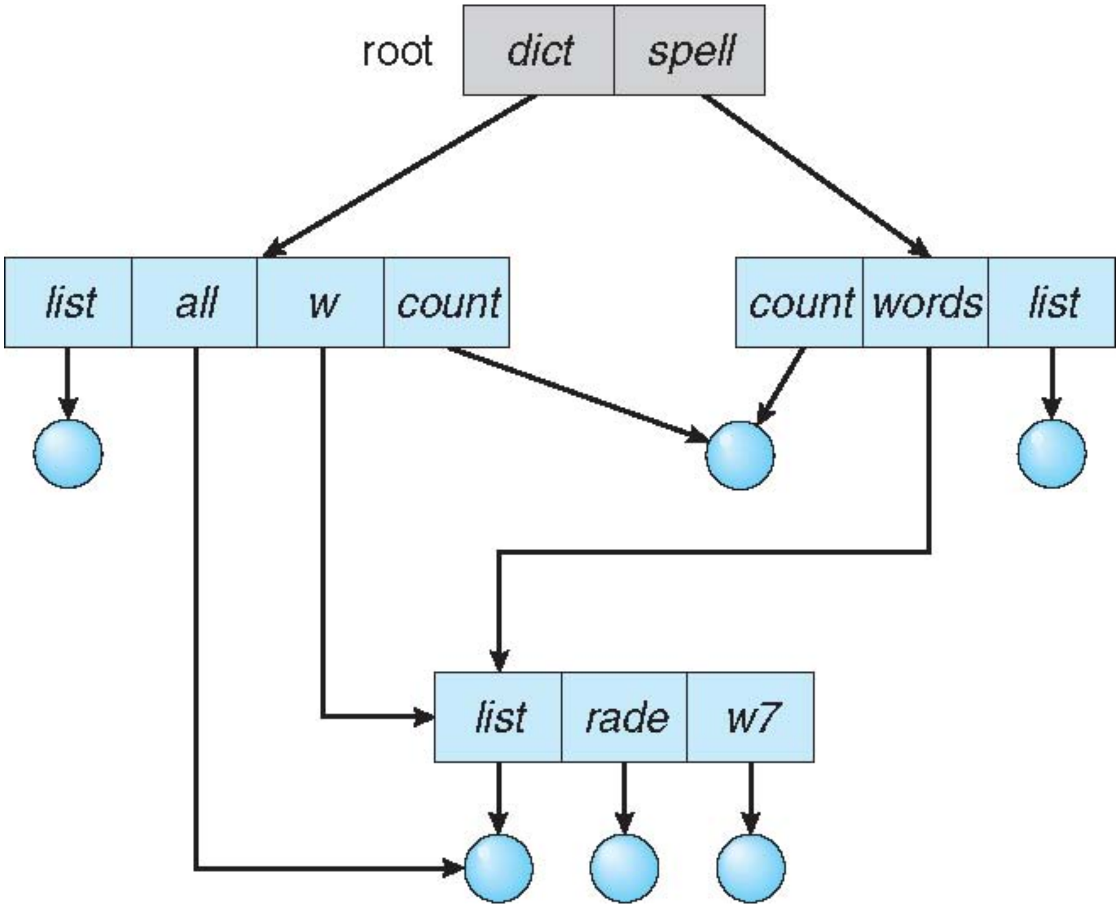
```
mkdir count
```



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

Acyclic-Graph Directories

- Have shared subdirectories and files (how do you accomplish this?)



Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)

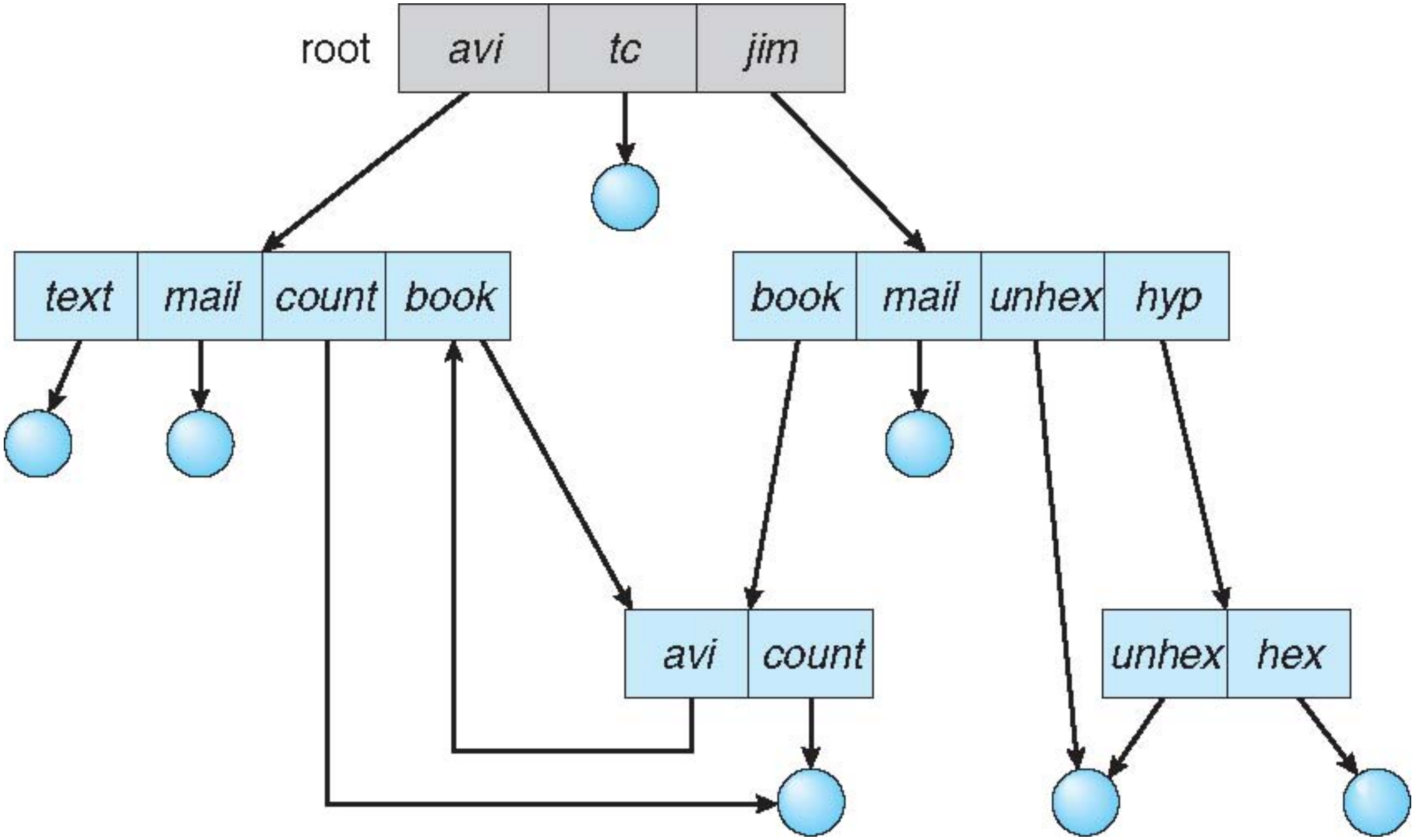
- If *dict* deletes *list* ⇒ dangling pointer

  Solutions:

  - Backpointers, so we can delete all pointers
    Variable size records a problem
  - Entry-hold-count solution

- New directory entry type
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

General Graph Directory



General Graph Directory (Cont.)

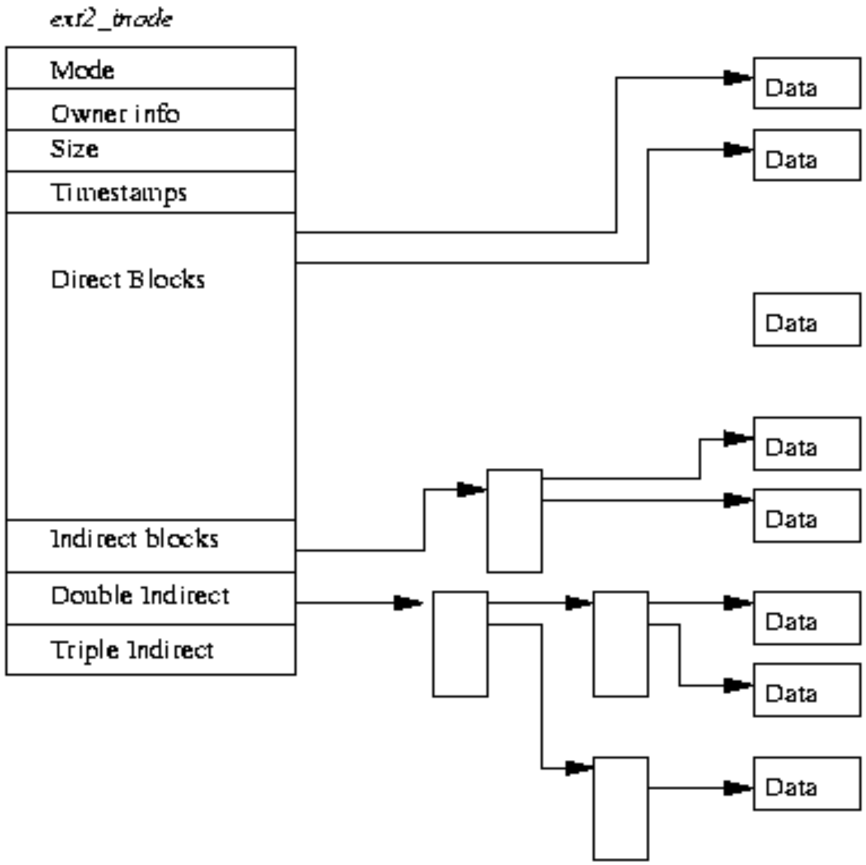
- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - Garbage collection
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

How are files organized: Blocks

- Storage organized as a **sequence of blocks**
  - Unit or reading and writing
  - Read, modify, write sequence
- File system tracks free and full blocks
  - typically stored in a bitmap

Inodes

- On disk data structure
  - Describes where all the bits of a file are



Directories

- Directory – just special file
  - Contains metadata, filenames
    - pointers to **inodes**
- Typically **hierarchical tree**
  - odd exposure of data structure to user

Lets chat about the midterm