0117401: Operating System
计算机原理与设计
Chapter 11: File system interface(文件系统接口)

陈香兰
xlanchen@ustc.edu.cn
http://staff.ustc.edu.cn/~xlanchen

Computer Application Laboratory, CS, USTC @ Hefei
Embedded System Laboratory, CS, USTC @ Suzhou

May 20, 2016

温馨提示：

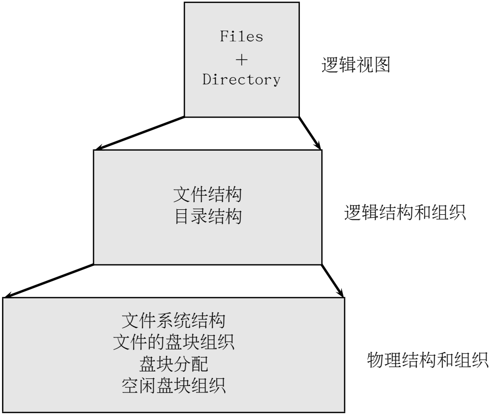
为了您和他人的工作学习，
请在课堂上关机或静音。

不要在课堂上接打电话。

提纲

1. File Concept
2. Access Methods (访问方式)
3. Directory Structure (目录结构)
4. File System Mounting (文件系统挂载)
5. File sharing (文件共享)
6. Protection
7. 小结和作业

File System


Files
+
Directory
逻辑视图
文件结构
目录结构
逻辑结构和组织
文件系统结构
文件的盘块组织
盘块分配
空闲盘块组织
物理结构和组织

Chapter Ojbectives

- 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

Outline

File Concept

- OS provides a uniform logical view of infomation storage despite the various storage media (nonvolatile).
- A file is a logical storage unit.
  - A file is a named collection of related information that is recorded on secondary storage.
  - Types:
    - Data: numeric; character; binary
    - Program
  - In general, a file is a sequence of bits, bytes, lines, or records.
    - The meaning is defined by the file's creator and user.
  - A file has a certain defined structure, which depends on its type.
    - Example: text files, source files, object files, executable files
  - Contiguous logical address space

File Concept

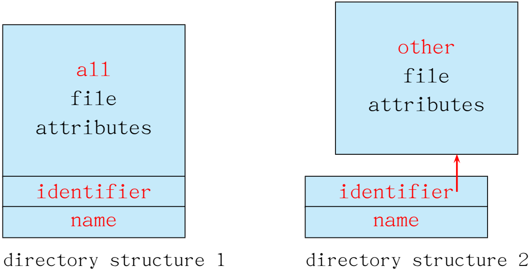
- File concept
  1. File attributes
  2. File operations
  3. File types
  4. File structures
  5. Internal file structure

1. File Attributes（文件属性）

- A file's attributes vary from one OS to another but typically consist of these:
  - Name -- The only information kept in **human-readable** form
    - A name is usually a string of characters, such as ``example.c''
    - Uppercase vs. lowercase: care or not care
  - Identifier -- Unique tag, usually a number, identifies file within FS
    - The **non-human-readable** name for the file
  - Type -- Needed for systems that support different types
  - Location -- A pointer to **file location on device**
  - Size -- Current file size; may also include MAX size
  - Protection -- Access-control（访问控制）information: who can do reading, writing, executing
  - Time, date, and user identification -- Data for protection, security, and usage monitoring

1. File Attributes (文件属性)

- Information about files are kept in the **directory** structure, which is also maintained on the secondary storage



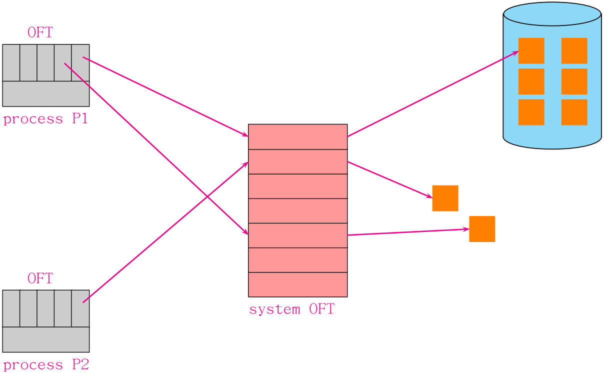
- Typically, a directory entry only **consists of the file's name and its unique identifier.**
  The identifier in turn locates the other file attributes.

2. File Operations (文件操作)

- File is an abstract data type. OS provides the 6 **basic** system calls
  1. **Create** : allocate space + create an directory entry
  2. **Write** : write pointer
  3. **Read** : read pointer
  4. **Reposition within file** : also known as **seek**
  5. **Delete** : release space + erase the directory entry
  6. **Truncate** : file len=0; release space; all other attributes remain unchanged
- others:
  - For **file** : append, rename
  - For file **attribute**: chown, chmod, ...
  - For **directory** & **directory entries**:
    - 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

2. File Operations (文件操作)

- **Open Files & Open-File Table**
  - **Open-file table**, OFT: a small table containing information about all open files
  - Several processes may open the same file at the same time ⇒**2-levels**: a **per-process table** & a **system-wide table** with process-independent information



2. File Operations (文件操作)

- Open Files & Open-File Table
  - Several pieces of data are needed to manage open files:
    - **File pointer**: pointer to last read/write location, process-dependent
    - **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
    - **Disk location of the file**: the information needed to locate the file on disk, always is kept in memory
    - **Access rights**: per-process access mode information

2. File Operations (文件操作)

- **Open file locking**: Provided by some OSes and FSes
  - allow one process to lock a file and prevent other processes from gaining access to it
  - functionality is similar to reader-writer locks
  - OS- or FS-dependent

  1. **Mandatory**: for example, Windows OSes, or
     - access is denied depending on locks held and requested;
     - OS ensures locking integrity
  2. **Advisory**: for example, UNIX
     - processes can find status of locks and decide what to do
     - up to software developers

3. 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
work 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, sometimes compressed, for archiving/storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

4. File Structure

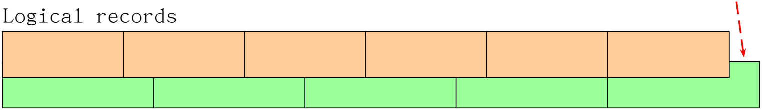
- Sometimes, file types can indicate the internal structure of file
- File structures(文件结构)(逻辑上)
  - None - sequence of words, bytes
  - Simple record structure
    - Lines
    - Fixed length;
    - Variable length
  - Complex Structures
    - Formatted document
    - Relocatable load file
- Can simulate last two with first method

4. File Structure

- System-supported file structures
  - Most modern OSes support a minimal number of file structures directly
    - Example: UNIX sees every file as a sequence of 8-bit bytes
  - Benefits:
    - Applications have more flexibility
    - Simplifies the OS

5. Internal file structure

- How to locate an offset within a file?
  - Logical file (record) (vary in length)
    → Physical block (fixed size)
- Solution: Packing -- packing a number of logical records into physical blocks.
  - Pack & unpack: convert between logical records and physical blocks
  - Internal fragmentation will occur



Logical records

Physical blocks

Outline

2 Access Methods (访问方式)

Access Methods（访问方式）

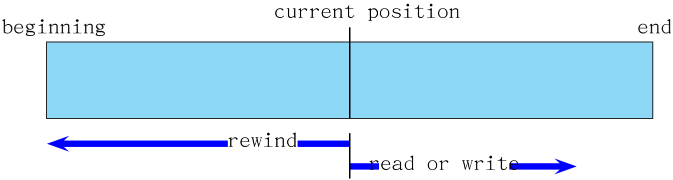
- Files store information. When it is used, this information must be accessed and read into computer memory
- On a logical perspective of users, access a file of records
    1. Sequential Access（顺序访问方式）
    2. Direct Access（直接访问方式）
    3. Indexed Access（索引访问方式）

1. Sequential Access (顺序访问方式)

- Sequential Access (顺序访问方式): the simplest access method.

  Information in the file is processed in order, one record after the other.
  - This is a most common access mode.

    For example: editors, compilers
  - A tape model of file
- File operations & the effect on file pointer
  - read/write next
  - reset
  - rewind/forward n



2. Direct Access（直接访问方式）

- Direct Access（直接访问方式）
  Information in the file is processed in no particular order.
  - File is made up of a numbered sequence of fixed-length logical records
    - A disk model of a file, allow random access, immediate access
      For example: databases, or an ailine-reservation system
  - Can move quickly to any record location by supplying a relative record number (n)
    - Read n & Write n,
      File pointer $= L * n$, $0 \leq n \leq N$, where N is the last record number, L is the fixed length of each record.
    - = Position n & read/write next, for example:

```
seek(20);          // move to rec. 20
seek(-1);           // move to rec. 19
read();
```

2. Direct Access (直接访问方式)

- Simulation of sequential access on a direct-access file

sequential access	implementation for direct access
reset	cp=0;

read next	read cp;
	cp=cp+1;

write next	write cp;
	cp=cp+1;

- How can we get n?
  If the record is with variable length, then ?

3. Indexed Access (索引访问方式)

- To improves search time and reduce I/O
  1. **Make an index file** for the file, which contains pointers to various records
  2. **Search the index file first**,
  3. and then use the pointer to access the file directly and to find the desired record.

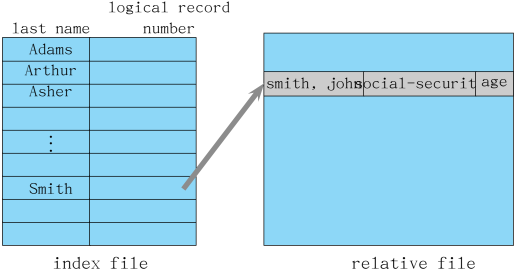


Figure: Example of index and relative files

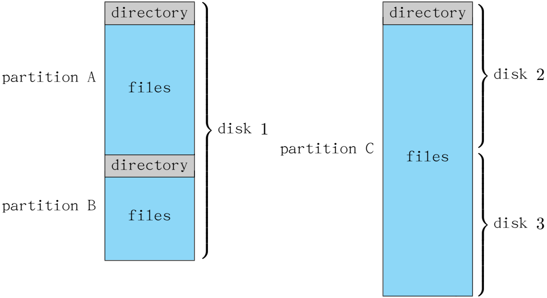
- With **large files**, the index file itself may become too large to be kept in memory ⇒ **Multi-level index table**

Outline

3 Directory Structure (目录结构)

A Typical File-system Organization

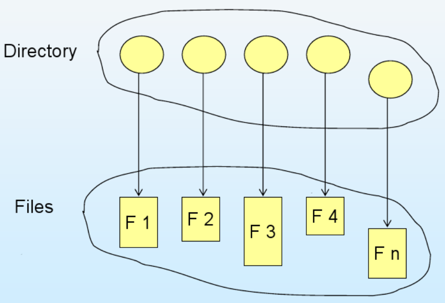
- Partition (mini-disks, volumes)
    1. One disk
    2. Part of a disk: provide separate logical spaces on one disk
    3. N disks: group several disks into a single logical space



- Partition = files + directories
    - Directory: holds file information (name, location, size, type, ...) for all files in that partition

Directory Overview

- **Directory**:
  A collection of nodes containing information about all files



- Directory + files: **all reside on disk**
- Backups of these two structures are kept on tapes

Directory Overview

- Information in a directory entry

  - File attributes
    - Name
    - Type
    - Address
    - Current length
    - Maximum length
    - Date last accessed (for archival)
    - Date last updated (for dump)
    - Owner ID (who pays)
    - Protection information

In DOS

- Directory entry = FCB (file control block)
- 32 bytes each
- May cost many I/O operations to search for an entry

In UNIX

- Inode: Store most of file attributes
- Directory entry = file name + a pointer to the inode
- 16 bytes each

Directory Overview

- Operations performed on directory

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

  - Search in the table for an entry
  - Insert an entry
  - Delete an entry
  - Modify an entry
  - ...

Directory Overview

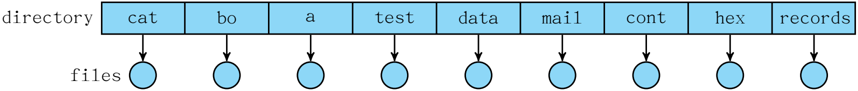
- Organize the directory (logically) to obtain
  1. Efficiency — locating a file quickly
  2. Naming — convenient to users
     - Two users can have same name for different files
     - The same file can have several different names
  3. Grouping — human convention
     - logical grouping of files by properties, (e.g., all Java programs, all games, ···)

Directory Structures (目录结构)

1. Single-level directory (单层目录)
2. Two-level directory (双层目录)
3. Tree-structured directory (树型结构目录)
4. Acyclic-graph directory (无环图目录)
5. General-graph directory (通用图目录)

1. Single-Level Directory (单层目录)

- A single directory for all users

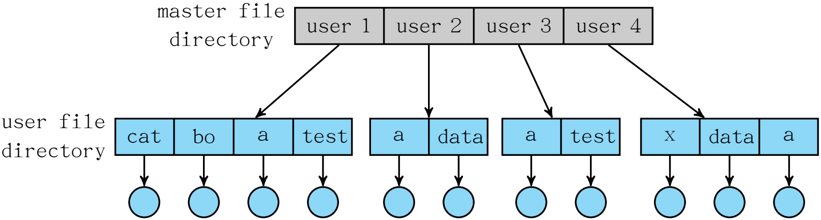


- Easy to support and understand.
- But if there are large numbers of files and/or users ...
  - Very low searching speed, O(N)
  - Naming problem
    - Small naming space & Name collision
    - MS-DOS: 11 bytes for filename
    - UNIX: 256 bytes
  - protection VS sharing;
  - grouping problem

2. Two-Level Directory (双层目录)

- **Two-Level Directory**: Separate directory for each user

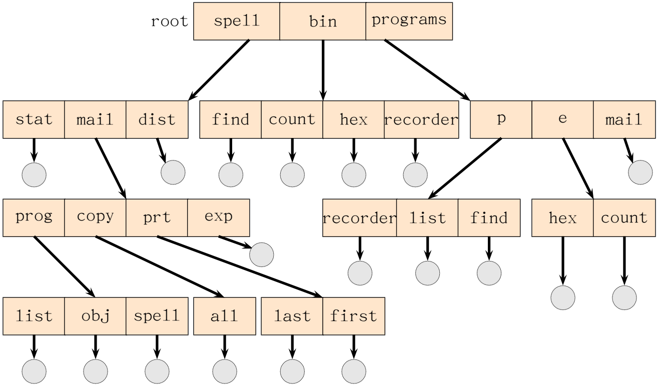
1. **User File Directory, UFD**
   - Each entry owns information for a user's file
2. **Master file directory, MFD**
   - Each entry contains:
     (1) User name,
     (2) A pointer to his UFD



2. Two-Level Directory (双层目录)

- Can have the same file name for different user
- Efficient searching
- No grouping capability
- Easy management
    - Add/delete a user
- Security VS. Sharing
    - MFD, system administrator
    - UFD, isolated from other users
    - Directory tree (seen as an inverted tree) & path name
    - How to share? E.g. system-wide files (dara, program, ...)
        - copy for each user?
        - searching path
- A UFD may be very large, then ...

3. Tree-Structured Directories（树型结构目录）



- Root directory（根目录）& directory（目录）& subdirectory（子目录）

3. Tree-Structured Directories (树型结构目录)

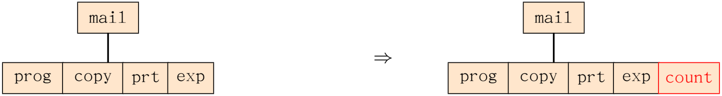
- **Regular file** VS. **subdirectory**
  - Treat a subdirectory like another file
  - Use a special bit in the directory entry to distinguish a file (0) from a subdirectory (1)
- **Current directory** (当前目录) (working/searching directory)
  - Creating a new file is done in current directory.
  - Initial current directory
- **Absolute** vs. **relative** path names (绝对/相对路径名)

  /spell/words/rade

  ../spell/words/rade

3. Tree-Structured Directories（树型结构目录）

- Operations
  - **Change** current directory: cd /spell/mail/prog
  - **Delete** a file: rm <file-name>
  - **List** a dictory: ls
  - **create** a new directory: mkdir <dir-name>
    - Example: if in current directory /mail
      mkdir count



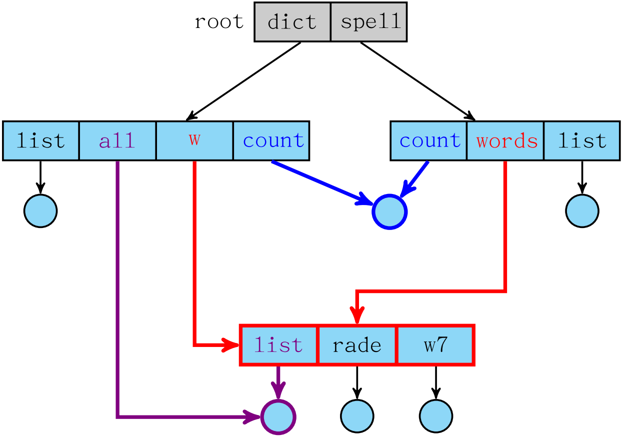
- **Delete** a directory
  - MS-DOS (only empty directory) VS. UNIX (optional)
- ...

3. Tree-Structured Directories（树型结构目录）

- Efficient searching
- Grouping Capability
- The tree structure
  prohibits the sharing of files and directories.

4. Acyclic-Graph Directories (无环图目录)

- Acyclic-Graph Directories
  - Have **shared** subdirectories and files, with no cycles
  - The same file or directory may be in two different directories, having two different names (**aliasing**)



4. Acyclic-Graph Directories (无环图目录)

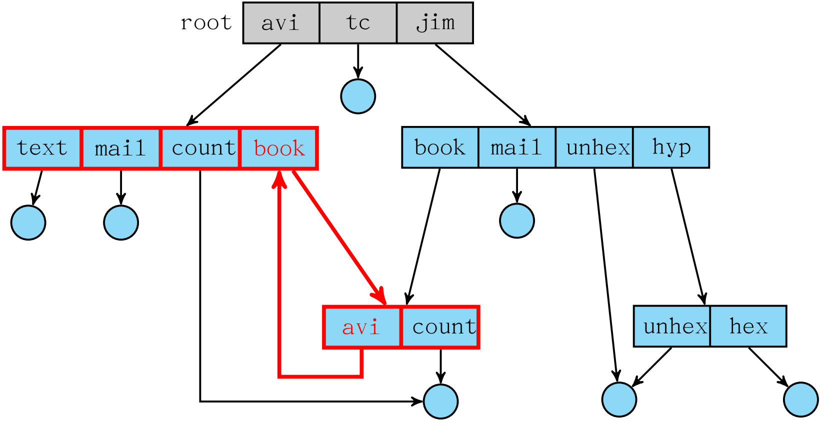
- Implementation
  1. Symbolic links (符号链接)
     - A special new directory entry (link)
     - The **content** of such file is the path name of the real file/ directory
     - How to traverse a directory contains symbolic links?
  2. Duplicates directory entries
     - Hard to maintain consistency

4. Acyclic-Graph Directories (无环图目录)

- Traversing problem
    - Different names, actual only one file
    - traverse more than once
- Deleting problem
    - If direct deletes list ⇒ dangling pointer
    - or preserve the file until all reference to it are deleted
    - Solutions:
        - File-reference list
        - Reference count: hard link (硬链接) in UNIX
- How to ensure there are no cycles?

5. General Graph Directory (通用图目录)

- If we allow cycles existed in directory



5. General Graph Directory (通用图目录)

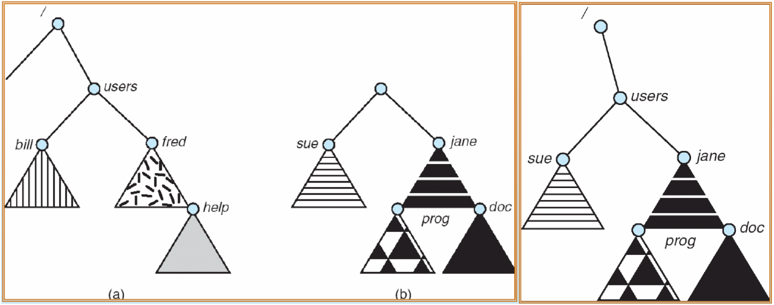
- The traversing problem and deleting problem still exists, even more complicatedly
  - Infinite loop
    - limit the access number of a directory while for a search
  - Garbage & garbage collection
- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

Outline

4 File System Mounting (文件系统挂载)

File System Mounting（文件系统挂载）

- A file system must be mounted before it can be accessed
- A unmounted file system is mounted at a **mount point（挂载点）**



(a) Existing.　(b) Unmounted Partition　(c) if using /users as **Mount Point**

Outline

5 File sharing （文件共享）

File sharing（文件共享）

- Sharing of files on multi-user systems is **desirable**
- Sharing may be done through a protection scheme
- On distributed systems, files may be shared across a network
- Network File System (NFS) is a common distributed file-sharing method

File sharing（文件共享）

1. Multiple Users share files
   - Multiple users⇒the issues of file sharing, file naming, file protection become preeminent
   - The system must control the sharing
     - allow by default, OR
     - require a user to specifically grant access to the file
   - More file and directory attributes are needed
     - Owner:User IDs identify users, allowing permissions and protections to be per-user
     - Group:Group IDs allow users to be in groups, permitting group access rights

File sharing (文件共享)

2. Remote File Systems
   - Uses **networking** to allow file system access between systems
     1. Manually via programs like FTP
     2. Automatically, seamlessly using distributed file systems
     3. 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
     - Example:
       NFS is standard UNIX client-server file sharing protocol
       CIFS is standard Windows protocol
     - Standard OS 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

File sharing (文件共享)

3. **Failure Modes**
   - Remote file systems add **new failure modes**, due to network failure, server failure
   - **Recovery** from failure can involve state information about status of each remote request
   - Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security

File sharing（文件共享）

4 Consistency Semantics

- Consistency semantics specify how multiple users are to access a shared file simultaneously
  - Similar to process synchronization algorithms
    Tend to be less complex due to disk I/O and network latency (for remote file systems
  - Andrew File System (AFS) implemented complex remote file sharing semantics
  - Unix file system (UFS) implements:
    Writes to an open file visible immediately to other users of the same open file
    Sharing file pointer to allow multiple users to read and write concurrently
  - AFS has session semantics
    Writes only visible to sessions starting after the file is closed

Outline

6 Protection

Protection

- Reliability (可靠性)
  - Guarding against physical damage
  - File systems can be damaged by
    - Hardware problems, power surges or failures, head crashed, dirt, temperature extremes, or Vandalism
  - Generally provided by duplicate copies of files (disk→tape, ...)
- Protection (保护，安全性)
  - Guarding against improper access

Protection in multi-user system

- The need to protect files is a direct result of the ability to access files (of other users).
  1. Complete protection with prohibiting access
  2. Free access with no protection
  3. Controlled access. √
- Controlled access: limiting the types of file access that can be made
  - Types of access: Read/Write/Execute/Append/Delete/ List
  - Higher-level functions may also be controlled: rename/ copy/edit/...
- File owner/creator should be able to control:
  - what can be done? by whom?
- Many protection mechanisms have been proposed.

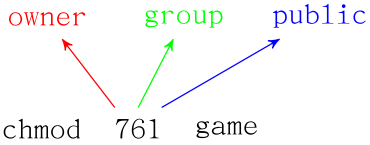
Access control （访问控制）

- **The most common approach** to the protection problem: ID-dependent access
  - Make access dependent on the ID of the user
- **The most general scheme** to implement ID-dependent access: Access control list (访问控制列表, ACL)
  - Associate with each file and directory an access list.
    - Access list specifies for each listed (allowed) user name and the types of (allowed) access allowed.
    - Stored in each directory entry
  - **Length problem**
    Solution: Three classes of users

			R	W	X
a) owner access	7	⇒	1	1	1
			R	W	X
b) group access	6	⇒	1	1	0
			R	W	X
c) public access	1	⇒	0	0	1

Access control (访问控制)

- About group:
  - 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.



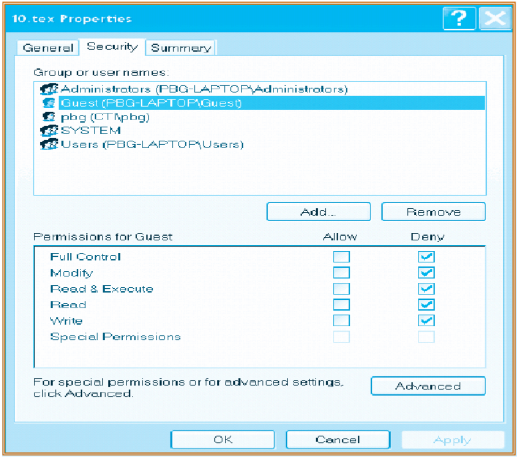
owner group public

chmod 761 game

  - Attach a group to a file

```
chgrp G game
```

Windows XP Access-control List Management



10.tex Properties

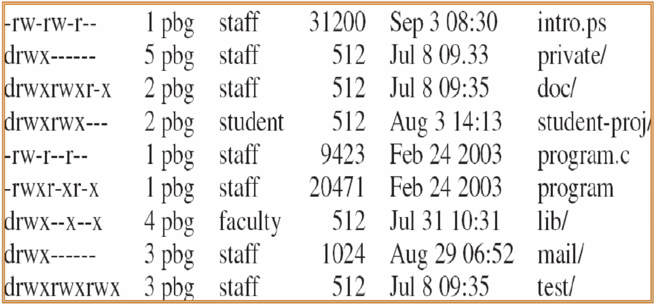
Group or user names:

- 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.

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/
```

Outline

7 小结和作业

小结

1. File Concept
2. Access Methods（访问方式）
3. Directory Structure（目录结构）
4. File System Mounting（文件系统挂载）
5. File sharing（文件共享）
6. Protection
7. 小结和作业

谢谢！