CSCI315 – Operating Systems Design

Department of Computer Science

Bucknell University

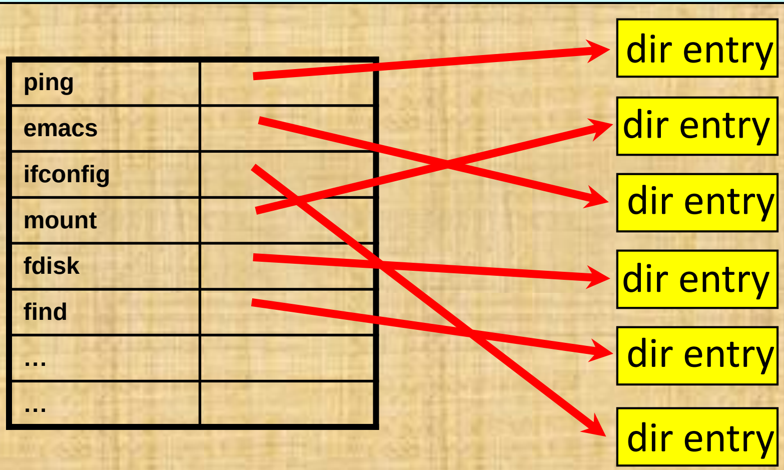
File Meta Data, Directories

Ch 13.3-13.4

This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough, and other instructors.
Xiannong Meng, Fall 2021.

Directory Structure

Directory: a symbol table that maps file names into directory entries. Each directory entry contains meta-data of the file such as owner name, date, protection. (Or, directory is a file about files.)



ping	
emacs	
ifconfig	
mount	
fdisk	
find	
…	
…	

Both the directory structure and the files reside on disk. Backups of these two structures are kept on back-up storage.

Linux Directory Entry

One directory entry, a directory consists of a number of these entries.

```
struct dirent {
   ino_t          d_ino;       /* inode number */
   off_t          d_off;       /* not an offset; see NOTES */
   unsigned short d_reclen;    /* length of this record */
   unsigned char  d_type;      /* type of file; not supported
                                  by all file system types */
   char           d_name[256]; /* filename */
};
```

man 3 readdir

Operations on Directories

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

Example of Directory Listing

```
dirp = opendir(dname);
if (dirp != NULL) { // it is a directory
   printf("directory : %s\n",dname);
   for (dp = readdir(dirp);
         NULL != dp;
         dp = readdir(dirp)) {
     printf("%s\n", dp->d_name);
    }
closedir (dirp);
```

For the complete program, see http://www.eg.bucknell.edu/~cs315/F2021/meng/code/files/list_dir.c

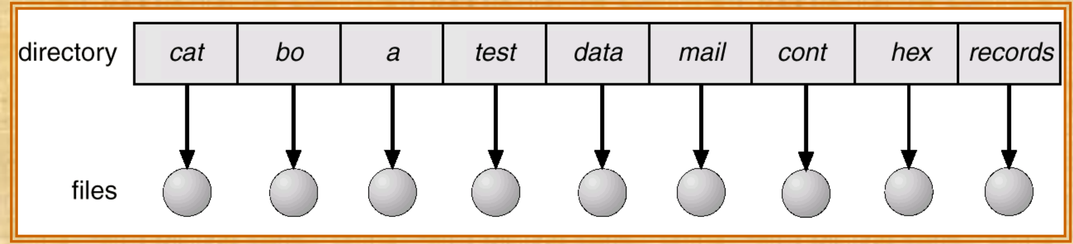
```
[xmeng@linuxremote1 files]$ gcc list_dir.c
[xmeng@linuxremote1 files]$ ./a.out ../
directory : ../
.
..
thread
sync
process
deadlock
scheduling
memory
files
[xmeng@linuxremote1 files]$ ./a.out ./
directory : ./
.
..
file-test.c
a.out
file-test.c~
list_dir.c
hello.txt
list_dir.c~
[xmeng@linuxremote1 files]$
```

Goals of Directory Logical Organization

- **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.

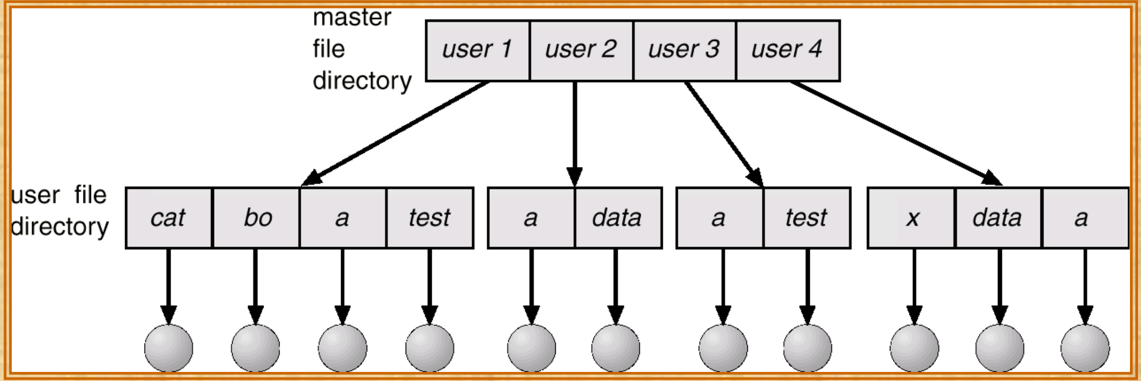


Drawbacks:

Naming problem
Grouping problem

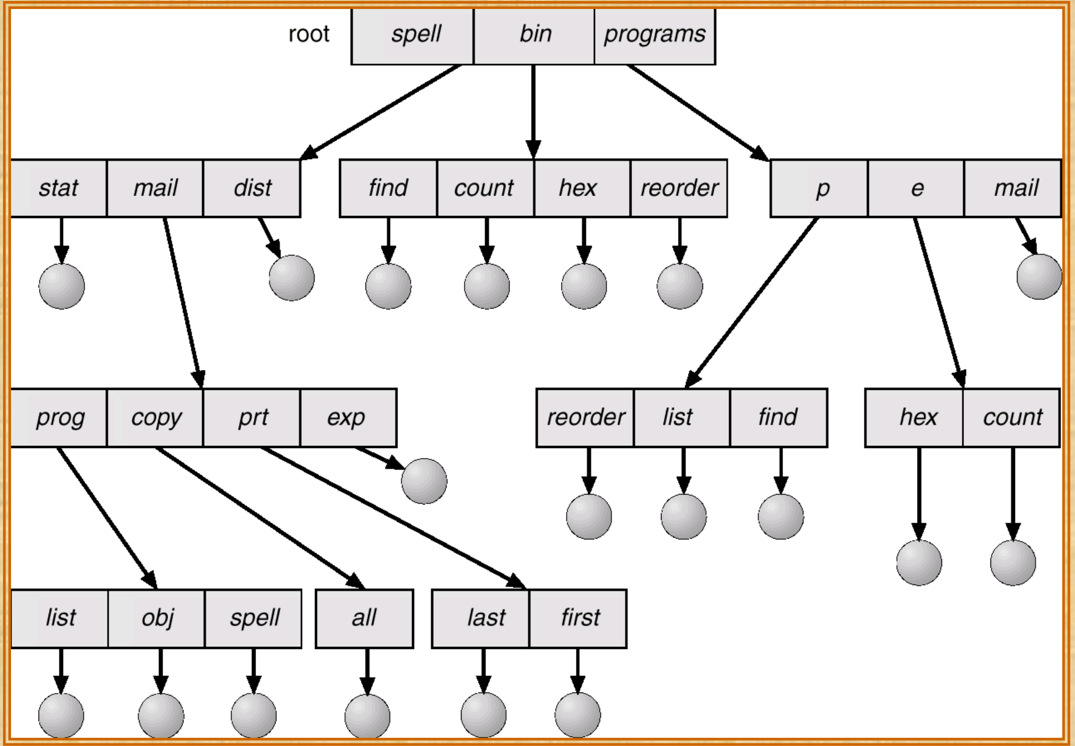
Two-Level Directory

A 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,
  - **type** list.

Tree-Structured Directories (Cont.)

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

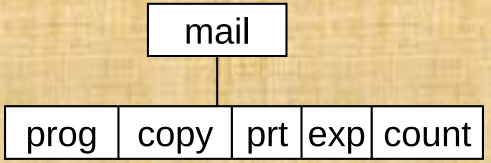
  rm <file-name>
- Creating a new subdirectory is done in current directory.

  mkdir <dir-name>

  Example: if in current directory **/mail**

  mkdir count

  will add **count** as a subdirectory under **mail**



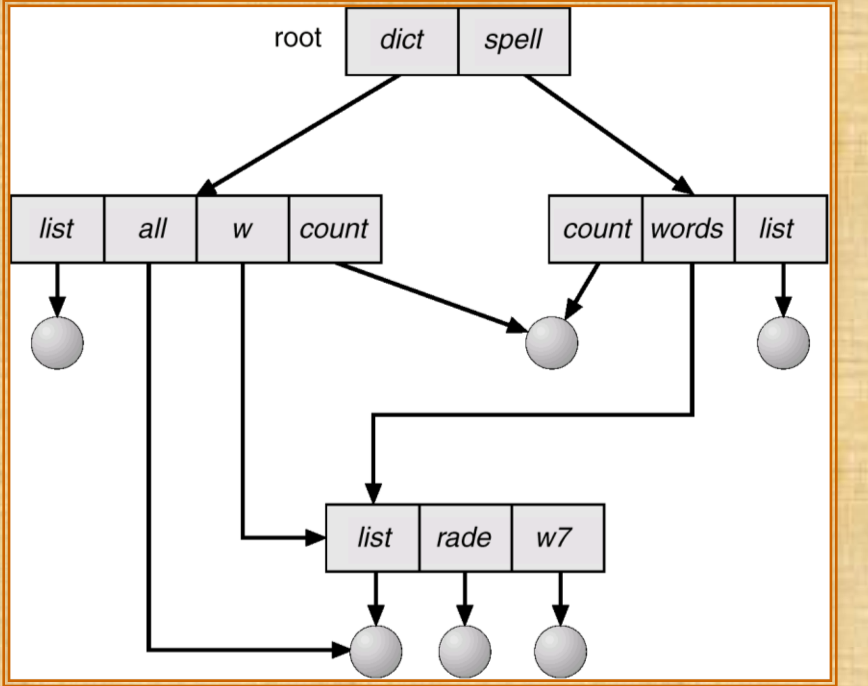
mail

prog copy prt exp count

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

Acyclic-Graph Directories

Have shared subdirectories and files.



Acyclic-Graph Directories (Cont.)

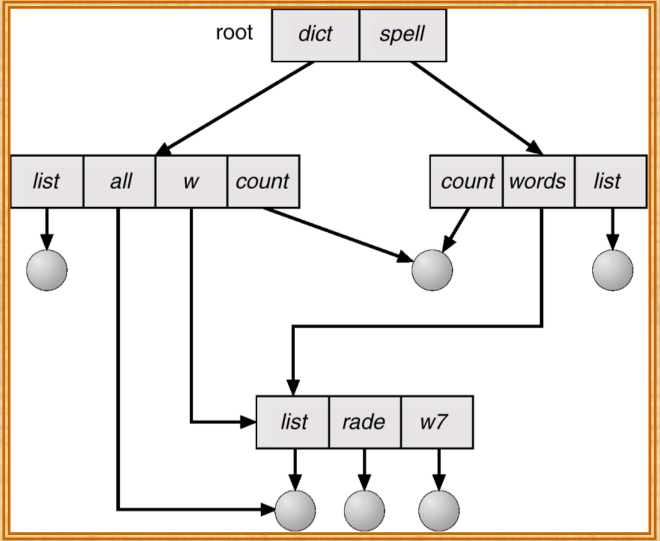
- Two different names (aliasing).
- If *dict* deletes *list* ⇒ dangling pointer.

  Solutions:

  - Backpointers, so we can delete all pointers. Variable size records a problem.
  - Backpointers using a daisy chain organization.
  - Entry-hold-count solution.

Acyclic-Graph Directories

Have shared subdirectories and files.

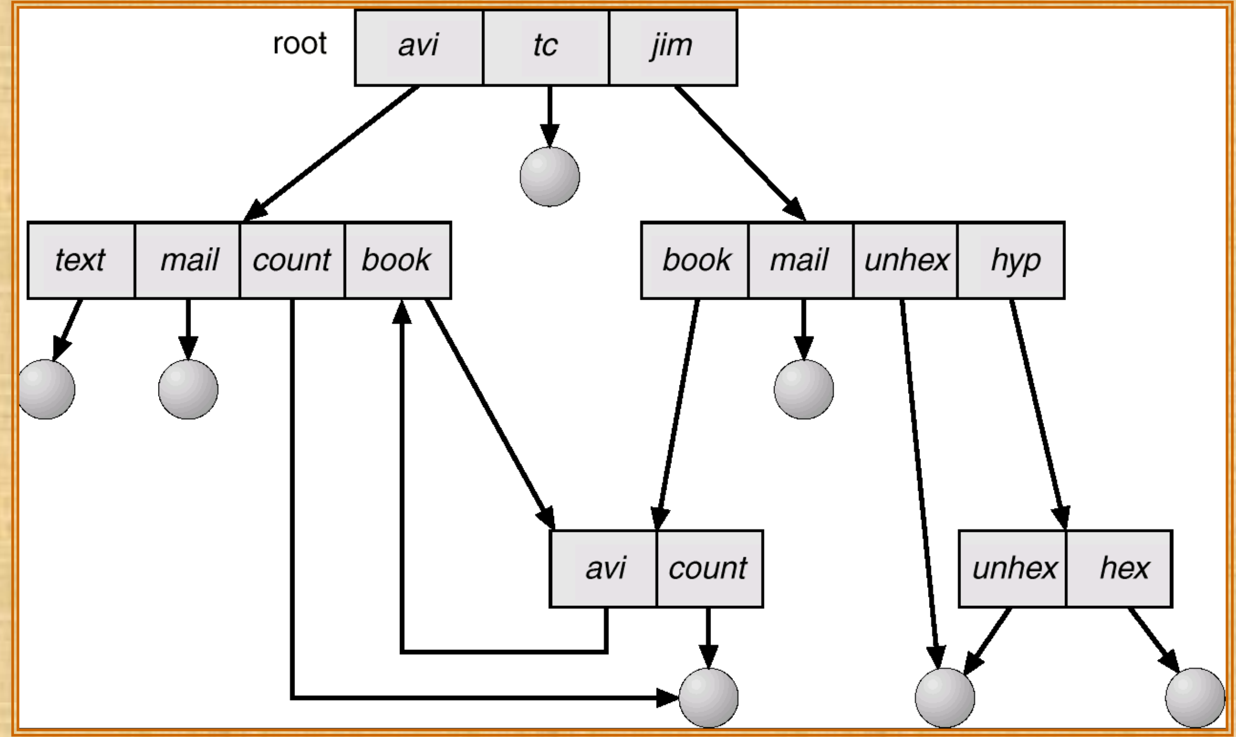


links: { soft (symbolic)
hard

Unix: **ln** (read man page);

need to keep a reference count on each file or directory.

General Graph Directory

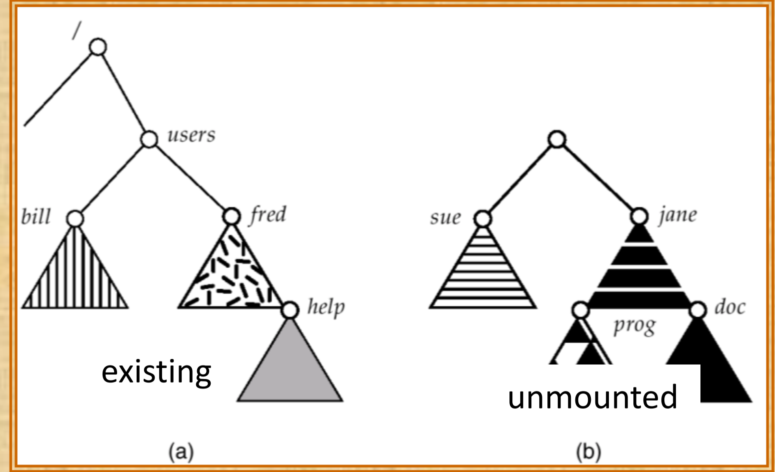


General Graph Directory (Cont.)

- **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.

File System Mounting

- A file system (partition) must be **mounted** before it can be accessed. Mounting allows one to attach the file system on one device to the file system on another device.
- A unmounted file system needs to be attached to a **mount point** before it can be accessed.



In our Linux systems, the **/home** directory is mounted on all Linux computers, including the ones in the labs.

Try the command **pwd** on any Linux computer, you'd see the same files and directories.

File Sharing

- 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. Our Linux systems use a variation of it.

Protection

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

Discretionary Access Control (DAC)

- **Types of access:**
  - Read,
  - Write,
  - Execute,
  - Append,
  - Delete,
  - List.

Protection

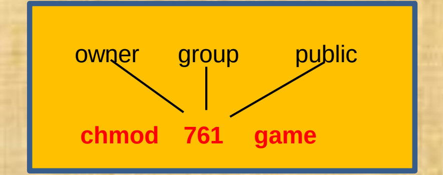
- **Mandatory Access Control (MAC):**
  - **System policy:** files tied to access levels = (public, restricted, confidential, classified, top-secret).
  - Process also has access level: can read from and write to all files at same level, can only read from files below, can only write to files above.
- **Role-Based Access Control (RBAC):**
  - **System policy:** defines **“roles”** (generalization of the Unix idea of groups).
  - Roles are associated with access rules to sets of files and devices.
  - A process can change roles (in a pre-defined set of possibilities) during execution.

Access Lists and Groups

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

```
                                   RWX
a) owner access          7 ⇒ 1 1 1
                                   RW
b) group access          6 ⇒  1 1 0
                                      X
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

Associate a group with a file: `chgrp G game`

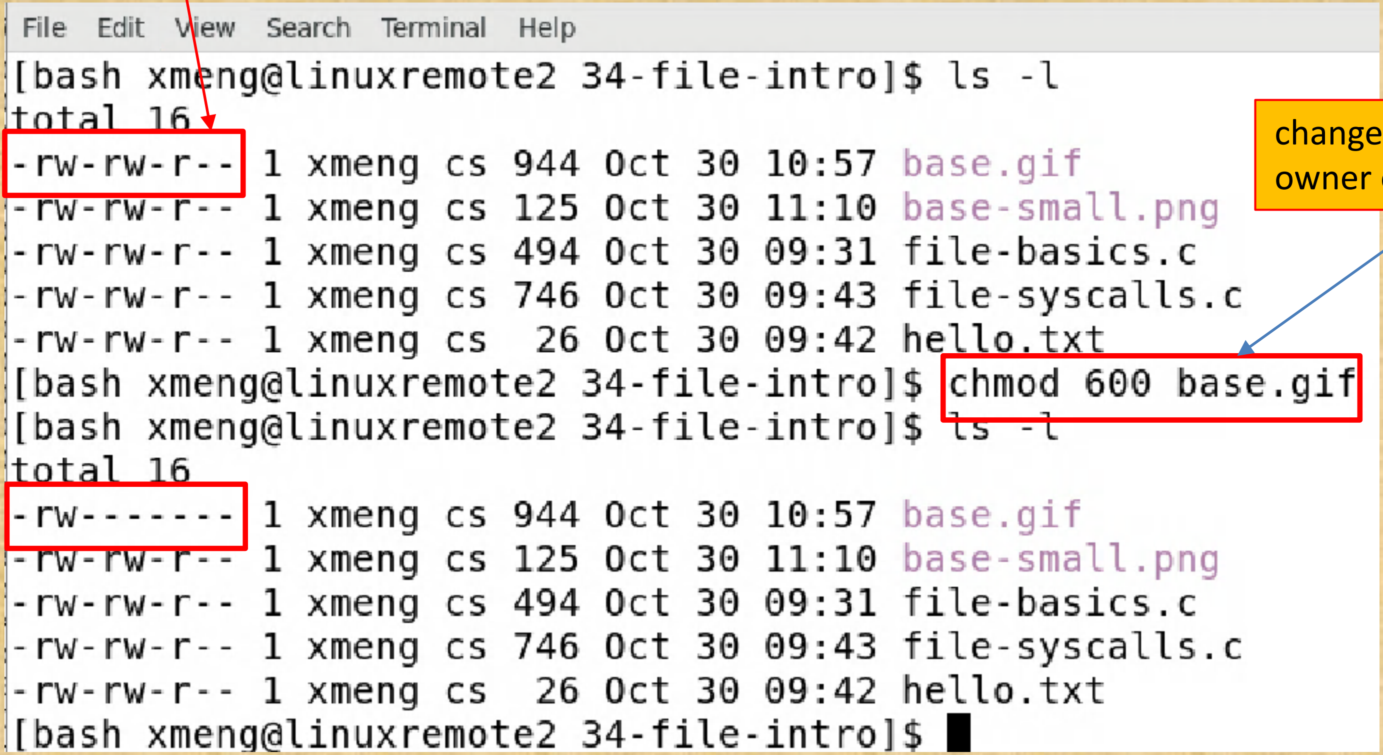
File Protection Example

protection bits 664 or ugo rw,rw,r

u: owner
g: group
o: world

6: 110
4: 100

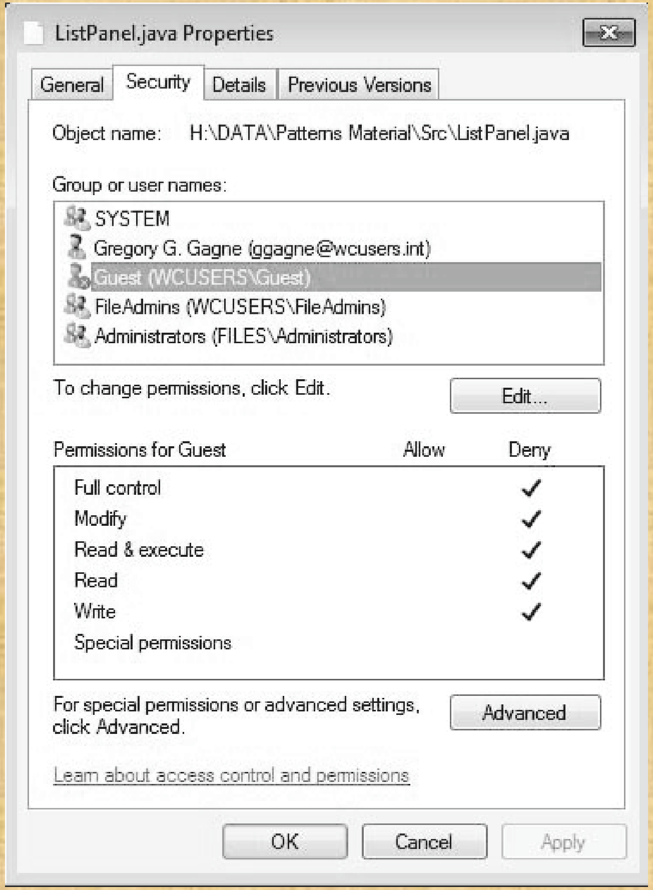
600



```
[bash xmeng@linuxremote2 34-file-intro]$ ls -l
total 16
-rw-rw-r-- 1 xmeng cs 944 Oct 30 10:57 base.gif
-rw-rw-r-- 1 xmeng cs 125 Oct 30 11:10 base-small.png
-rw-rw-r-- 1 xmeng cs 494 Oct 30 09:31 file-basics.c
-rw-rw-r-- 1 xmeng cs 746 Oct 30 09:43 file-syscalls.c
-rw-rw-r-- 1 xmeng cs  26 Oct 30 09:42 hello.txt
[bash xmeng@linuxremote2 34-file-intro]$ chmod 600 base.gif
[bash xmeng@linuxremote2 34-file-intro]$ ls -l
total 16
-rw------- 1 xmeng cs 944 Oct 30 10:57 base.gif
-rw-rw-r-- 1 xmeng cs 125 Oct 30 11:10 base-small.png
-rw-rw-r-- 1 xmeng cs 494 Oct 30 09:31 file-basics.c
-rw-rw-r-- 1 xmeng cs 746 Oct 30 09:43 file-syscalls.c
-rw-rw-r-- 1 xmeng cs  26 Oct 30 09:42 hello.txt
[bash xmeng@linuxremote2 34-file-intro]$ 
```

changed so only owner can read/write

Windows 7 Access-Control List Management



Object name: H:\DATA\Patterns Material\Src\ListPanel.java

Group or user names:

- SYSTEM
- Gregory G. Gagne (ggagne@wcusers.int)
- Guest (WCUSERS\Guest)
- FileAdmins (WCUSERS\FileAdmins)
- Administrators (FILES\Administrators)

To change permissions, click Edit.

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

For special permissions or advanced settings, click Advanced.