4.6 FILE SHARING

Multiple Users

• On a multi-user system, more information needs to be stored for each file:

  • The owner (user) who owns the file, and who can control its access.

  • The group of other user IDs that may have some special access to the file.

  • What access rights are given to the owner (User), the Group, and to the rest of the world.

• Some systems have more complicated access control, allowing or denying specific accesses to specifically named users or groups.

Remote File Systems

- The advent of the Internet introduces issues for accessing files stored on remote computers
- The original method was ftp, allowing individual files to be transported across systems as needed. Ftp can be either account and password controlled, or **anonymous**, not requiring any user name or password.
- Various forms of **distributed file systems** allow remote file systems to be mounted onto a local directory structure, and accessed using normal file access commands.
- The WWW has made it easy once again to access files on remote systems without mounting their file systems, generally using (anonymous) ftp as the underlying file transport mechanism.

The Client-Server Model

- When one computer system remotely mounts a filesystem that is physically located on another system, the system which physically owns the files acts as a **server**, and the system which mounts them is the **client.**
- User IDs and group IDs must be consistent across both systems for the system to work properly. (I.e. this is most applicable across multiple computers managed by the same organization, shared by a common group of users.)
- The same computer can be both a client and a server. (E.g. crosslinked file systems.)
- There are a number of security concerns involved in this model:
- Servers commonly restrict mount permission to certain trusted systems only. Spoofing (a computer pretending to be a different computer) is a potential security risk.

- Servers may restrict remote access to read-only.
- Servers restrict which filesystems may be remotely mounted. Generally the information within those subsystems is limited, relatively public, and protected by frequent backups.
- The NFS (Network File System) is a classic example of such a system.

Distributed Information Systems

- The **Domain Name System, DNS,** provides for a unique naming system across all of the Internet.
- Domain names are maintained by the **Network Information System, NIS**, which unfortunately has several security issues. NIS+ is a more secure version, but has not yet gained the same widespread acceptance as NIS.
- Microsoft's **Common Internet File System, CIFS**, establishes a **network login** for each user on a networked system with shared file access. Older Windows systems used **domains**, and newer systems (XP, 2000), use **active directories.** User names must match across the network for this system to be valid.
- A newer approach is the **Lightweight Directory-Access Protocol, LDAP,** which provides a **secure single sign-on** for all users to access all resources on a network. This is a secure system which is gaining in popularity, and which has the maintenance advantage of combining authorization information in one central location.

Failure Modes

- When a local disk file is unavailable, the result is generally known immediately, and is generally non-recoverable. The only reasonable response is for the response to fail.
- However when a remote file is unavailable, there are many possible reasons, and whether or not it is unrecoverable is not readily apparent. Hence most remote access systems allow for blocking or delayed response, in the hopes that the remote system (or the network) will come back up eventually.

Consistency Semantics

Consistency Semantics deals with the consistency between the views of sharedfiles on a networked system. When one user changes the file, when do otherusers see the changes?

UNIX Semantics

- The UNIX file system uses the following semantics:
- Writes to an open file are immediately visible to any other user who has the file open.

- One implementation uses a shared location pointer, which is adjusted for all sharing users.

Session Semantics

- The Andrew File System, AFS uses the following semantics:
- Writes to an open file are not immediately visible to other users.
- When a file is closed, any changes made become available only to users who open the file at a later time.

Immutable-Shared-Files Semantics

Under this system, when a file is declared as **shared** by its creator, it becomes immutable and the name cannot be re-used for any other resource. Hence it becomes read-only, and shared access is simple.

4.7 PROTECTION

- Files must be kept safe for reliability (against accidental damage), and protection (against
- Deliberate malicious access.) The former is usually managed with backup copies.
- One simple protection scheme is to remove all access to a file.
- However this makes the file unusable, so some sort of controlled access must be arranged.

Types of Access

The following low-level operations are often controlled:

- Read - View the contents of the file
- Write - Change the contents of the file.
- Execute - Load the file onto the CPU and follow the instructions contained therein.
- Append - Add to the end of an existing file.
- Delete - Remove a file from the system.
- List -View the name and other attributes of files on the system.
- Higher-level operations, such as copy, can generally be performed through combinations of the above.

Access Control

- One approach is to have complicated **Access Control Lists, ACL,** which specify exactly what access is allowed or denied for specific users or groups.
- The AFS uses this system for distributed access.
- Control is very finely adjustable, but may be complicated, particularly when the specific users involved are unknown. (AFS allows some wild cards, so for example all users on a certain remote system may be trusted, or a given username may be trusted when accessing from any remote system.)
- UNIX uses a set of 9 access control bits, in three groups of three. These correspond to R, W, and X permissions for each of the Owner, Group, and Others. The RWX bits control the following privileges for ordinary files and directories:

bit	Files	Directories
R	Read (view) file contents.	Read directory contents. Required to get a listing of the directory.
W	Write (change) file contents.	Change directory contents. Required to create or delete files.
X	Execute file contents as a program.	Access detailed directory information. Required to get a long listing, or to access any specific file in the directory. Note that if a user has X but not R permissions on a directory, they can still access specific files, but only if they already know the name of the file they are trying to access.

- In addition there are some special bits that can also be applied:
- The set user ID (SUID) bit and/or the set group ID (SGID) bits applied to executable files temporarily change the identity of whoever runs the program to match that of the owner / group of the executable program. Setting of these two bits is usually restricted to root, and must be done with caution, as it introduces a potential security leak.
- The sticky bit on a directory modifies write permission, allowing users to only delete files for which they are the owner. This allows everyone to create files in /tmp, for example, but to only delete files which they have created, and not anyone else's.
- The SUID, SGID, and sticky bits are indicated with an S, S, and T in the positions for execute permission for the user, group, and others, respectively. If the letter is lower

case, (s, s, t), then the corresponding execute permission is not also given. If it is upper case, (S, S, T), then the corresponding execute permission IS given.

Other Protection Approaches and Issues

- Some systems can apply passwords, either to individual files, or to specific subdirectories,
- or to the entire system. There is a trade-off between the number of passwords that must be maintained and the amount of information that is vulnerable to a lost or forgotten password.
- Older systems which did not originally have multi-user file access permissions must now be **retrofitted** if they are to share files on a network.
- Access to a file requires access to all the files along its path as well. In a cyclic directory structure, users may have different access to the same file accessed through different paths.
- Sometimes just the knowledge of the existence of a file of a certain name is a security (or privacy) concern. Hence the distinction between the R and X bits on UNIX directories.

4.8 FILE SYSTEM IMPLEMENTATION

FILE-SYSTEM STRUCTURE

Hard disks have two important properties that make them suitable for secondary storage of files in file systems:

(1) Blocks of data can be rewritten in place, and

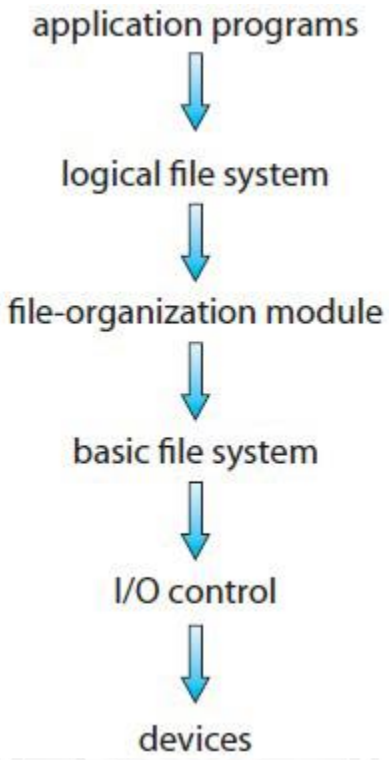
(2) They are direct access, allowing any block of data to be accessed with only (relatively) minor movements of the disk heads and rotational latency.

- Disks are usually accessed in physical blocks, rather than a byte at a time. Block sizes may range from 512 bytes to 4K or larger.
- File systems organize storage on disk drives, and can be viewed as a layered design:
- At the lowest layer are the physical devices, consisting of the magnetic media, motors & controls, and the electronics connected to them and controlling them. Modern disk put more and more of the electronic controls directly on the disk driveitself, leaving relatively little work for the disk controller card to perform.

I/O Control consists of **device drivers**, special software programs

(often written in assembly) which communicate with the devices by reading and writing special codes directly to and from memory addresses corresponding to the controller card's registers. Each controller card (device) on a system has a different set of addresses (registers, a.k.a. **ports**) that it listens to, and a unique set of command codes and results codes that it understands.

- The **basic file system** level works directly with the device drivers in terms of retrieving and storing raw blocks of data, without any consideration for what is in each block. Depending on the system, blocks may be referred to with a single block number, (e.g. block # 234234), or with head-sector-cylinder combinations.
- The **file organization module** knows about files and their logical blocks, and how they map to physical blocks on the disk. In addition to translating from logical to physical blocks, the file organization module also maintains the list of free blocks, and allocates free blocks to files as needed.
- The **logical file system** deals with all of the meta data associated with a file (UID,GID, mode, dates, etc), i.e. everything about the file except the data itself. This level manages the directory structure and the mapping of file names to **file control blocks, FCBs**, which contain all of the meta data as well as block number information for finding the data on the disk.
- The layered approach to file systems means that much of the code can be used uniformly fora wide variety of different file systems, and only certain layers need to be file system specific. Common file systems in use include the UNIX file system, UFS, the Berkeley Fast File System, FFS, Windows systems FAT, FAT32, NTFS, CD-ROM systems ISO 9660, and for Linux the extended file systems ext2 and ext3 (among 40 others supported.)


application programs
logical file system
file-organization module
basic file system
I/O control
devices