File System Storage

- File Concept

- Access Methods

- Disk and Directory Structure

- File-System Mounting

- File Sharing

- 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

- Contiguous logical address space
- <u>Types:</u>
  - Data
    - numeric
    - character
    - binary
  - Program
- Contents defined by file's creator
  - Many types
    - Consider **text file, source file, executable file**

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
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure

File info Window on Mac OS X

File Operations

- File is an **abstract data type**
- **Create**
- **Write –** at **write pointer** location
- **Read –** at **read pointer** location
- **Reposition within file -** seek
- **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:

  - **Open-file table**: tracks 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

  - Disk location of the file: cache of data access information

  - Access rights: per-process access mode information

Open File Locking

- Provided by some operating systems and file systems
  - Similar to reader-writer locks
  - **Shared lock** similar to reader lock – several processes can acquire concurrently
  - **Exclusive lock** similar to writer lock
- Mediates access to a file
- Mandatory or advisory:
  - **Mandatory** – access is denied depending on locks held and requested
  - **Advisory** – processes can find status of locks and decide what to do

File Locking Example – Java API

```java
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
        FileLock sharedLock = null;
        FileLock exclusiveLock = null;
        try {

                RandomAccessFile raf = new RandomAccessFile("file.txt", "rw");
                // get the channel for the file
                FileChannel ch = raf.getChannel();
                // this locks the first half of the file - exclusive
                exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);
                /** Now modify the data . . . */
                // release the lock
                exclusiveLock.release();
```

```
        // this locks the second half of the file - shared
    sharedLock = ch.lock(raf.length()/2+1, raf.length(), SHARED);
            /** Now read the data . . . */
            // release the lock
            sharedLock.release();
    } catch (java.io.IOException ioe) {
            System.err.println(ioe);
    }finally {
            if (exclusiveLock != null)
            exclusiveLock.release();
            if (sharedLock != null)
            sharedLock.release();
    }
  }
}
```

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 com-pressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

File Structure

- None - sequence of words, bytes
- <u>Simple record structure</u>
  - Lines
  - Fixed length
  - Variable length
- <u>Complex Structures</u>
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- <u>Who decides:</u>
  - Operating system
  - Program

Sequential-access File
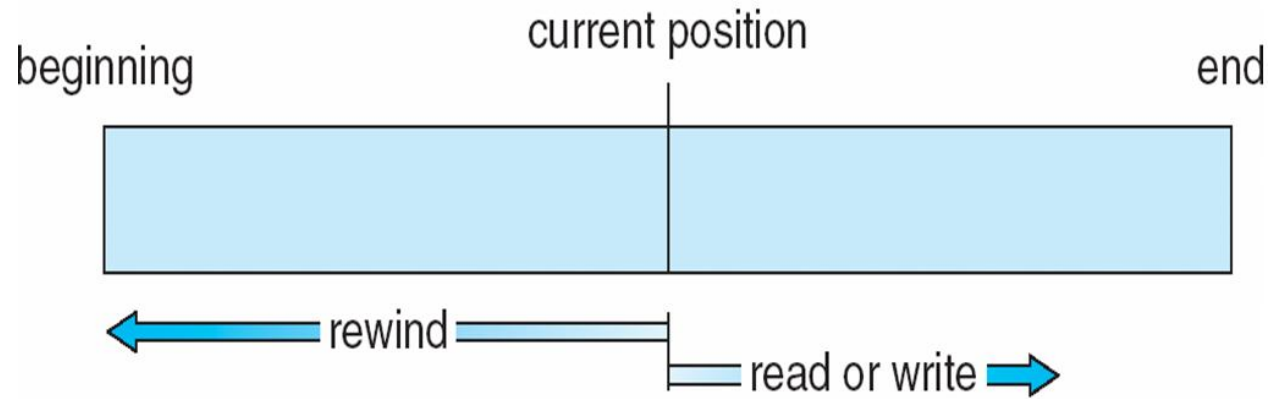
Access Methods

- **Sequential Access**

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

- **Direct Access –** file is fixed length logical records

  > `read n`
  > `write n`
  > `position to n`
  >   `read next`
  >   `write next`
  > `rewrite n`

  n = relative block number

- Relative block numbers allow OS to decide where file should be placed

Simulation of Sequential Access on Direct-access File

sequential access	implementation for direct access
reset	$cp = 0$;
read next	read cp; $cp = cp + 1$;
write next	write cp; $cp = cp + 1$;

Other Access Methods

- Can be built on top of base methods
- General involve creation of an index for the file
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item)
- If too large, index (in memory) of the index (on disk)
- IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS
- VMS operating system provides index and relative files as another example

Example of Index and Relative Files