File systems: basic management of information

- **file: sequence of bytes stored on a computer**
  - content is arbitrary; any structure is imposed by the creator of the file, not by the operating system
- **file system: software that provides hierarchical storage and organization of files, usually on a single computer**
  - part of the operating system

File Systems: managing stored information

- **logical structure: users and programs see a hierarchy of folders (aka directories) and files**
  - a folder contains references to folder and files
  - "root" folder ultimately leads to all others
  - a file is just a sequence of bytes
    - contents determined and interpreted by programs, not the operating system
  - a folder is a special file that contains names of other folders & files
    - plus other information like size, time of change, etc.
    - contents are completely controlled by the operating system

File Systems: managing stored information

- **physical structure: disk drives (aka hard drives) operate in tracks, sectors, etc.**
  - other storage devices have other physical properties
    - CD-ROM, DVD, external USB drives, network drive
- **the operating system converts between logical and physical views**
  - does whatever is necessary to maintain the file/folder illusion
  - hides physical details so that programs don't depend on them
  - presents a uniform interface to disparate physical media
- **the "file system" is the part of the operating system that does this conversion**

How the file system converts logical to physical

- **disk is physically organized into sectors, or <u>blocks</u> of bytes**
  - each sector is a fixed number of bytes, like 512 or 1024 or …)
  - reading and writing always happens in sector-sized blocks
- **each file occupies an integral number of blocks**
  - files **never** share a block
  - some space is wasted: a 1-byte file wastes all but 1 byte of the block
- **if a file is bigger than one block, it occupies several blocks**
  - the blocks are not necessarily adjacent on the disk
- **need a way to keep track of the blocks that make up the file**
- **this is usually done by a separate "file allocation table" that lists the blocks that make up each file**
  - this table is stored on disk too so it persists when machine is turned off
  - lots of ways to implement this

Converting logical to physical, continued

- **every block is part of some file, or reserved by operating system, or unused**
- **"file allocation table" keeps track of blocks**
  - by chaining/linking them together
    - first block of a file points to second, second points to third, etc.
    - last block doesn't point to a successor (because it doesn't have one)
  - or (much more common) by some kind of table or array
    - that keeps track of related blocks
- **also keeps track of unused blocks**
  - disk starts out with most blocks unused ("free")
    - some are reserved for file allocation table, etc.
  - as a file grows, blocks are removed from the unused list and attached to the list for the file:
    - to grow a file, remove a block from the list of unused blocks
    - and add it to the blocks for the file

Converting logical to physical: directories

- **a directory / folder <u>is a file</u>**
  - stored in the same file system
  - uses the same mechanisms
- **but it contains information about other files and directories**
- **the directory entry for a file tells where to find the blocks**
- **the directory entry also contains other info about the file**
  - name (e.g., midterm.doc)
  - size in bytes, date/time of changes, access permissions
  - whether it's an ordinary file or a directory
- **the file system maintains the info in a directory**
  - very important to keep directory info consistent
  - application programs can change it only indirectly / implicitly

Finding files; root directory

- **all files are ultimately accessible from the "root" directory/folder**
  - e.g., C: on Windows, / on Unix and Mac
- **to access the contents of a file named**
  C:\Program Files\Adobe\Acrobat 8.0\Acrobat\acrobat.exe
  - read the blocks of C:, look for an entry with name "Program Files"
  - read the blocks of the Program Files directory, look for "Adobe"
  - read the blocks of Adobe, look for "Acrobat 8.0"
  - read the blocks of Acrobat 8.0, look for "Acrobat"
  - read the blocks of Acrobat, look for "acrobat.exe"
  - read the blocks of acrobat.exe
- **all but the last of these are directories/folders**
- **the long name is often called the "path name"**
  - since it describes a path through the file system hierarchy

What happens when you say "Open"?

- **search for file in sequence of directories as given by components of its name**
  - report an error if any component can't be found
- **read blocks of the file as needed**
  - using the location information in the file allocation table to find the blocks
  - store (some of) them in RAM

What happens when you say "Save"?

- **make sure there's enough space (enough unused blocks)**
  - don't want to run out while copying from RAM to disk
- **create a temporary file with no bytes in it**
- **copy the bytes from RAM and/or existing file to temporary file:**

```
while (there are still bytes to be copied) {
    get a free block from the unused list
    copy bytes to it until it's full or there are no more bytes to copy
    link it in to the temporary file
}
```

- **update the directory entry to point to the new file**
- **move the previous blocks (of old version) to the unused list**
  - or to recycle bin / trash

What happens when you remove a file?

- **move the blocks of the file to the unused list**
- **set the directory entry so it doesn't refer to any block**
  - set it to zero, maybe
- **recycle bin**
  - recycle bin is just another directory
  - removing a file just puts the name, location info, etc., in that directory instead
- **"emptying the trash" moves blocks into unused list**
  - removes entry from Recycle / Trash directory
- **why "removing" a file isn't enough**
  - usually only changes a directory entry
  - often recoverable by simple guesses about directory entry contents
  - file contents are often still there even if directory entry is cleared

Electronic discovery

- **... electronic discovery as we know it will undergo some important changes. These changes are being driven by amendments to the Federal Rules of Civil Procedure (FRCP) that become effective on December 1, 2006.**
- **Rule 26(a) will broaden the definition of electronic items that may be subject to discovery from "documents" or "data compilations" to include all electronically stored information. Thus, whereas before parties might have been able to try to shield certain types of electronic information from discovery, conceivably the other side now can demand everything from standard Word documents and emails to voicemail messages, instant messages, blogs, backup tapes, and database files.**

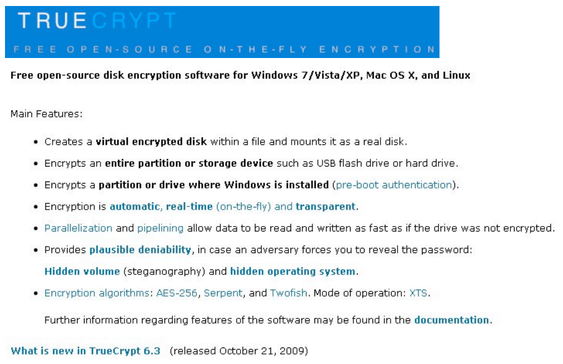
From http://technology.findlaw.com/articles/00006/010189.html

CITP talk by former judge Ron Hedges, 10/15/09:
http://citp.princeton.edu/events/lectures/ron-hedges/

Network file systems

- **software system for accessing remote files across networks**
- **user programs access files and folders as if they are on the local machine**
- **operating system converts these into requests to ship information to/from another machine across a network**
- **there has to be a program on the other end to respond to requests**
- **"mapping a network drive" or "mounting your H: drive" sets up the connections**
- **subsequent reads and writes go through the network instead of the local disk**

Encrypted file systems


TRUECRYPT

FREE OPEN-SOURCE ON-THE-FLY ENCRYPTION

Free open-source disk encryption software for Windows 7/Vista/XP, Mac OS X, and Linux

Main Features:

- Creates a **virtual encrypted disk** within a file and mounts it as a real disk.
- Encrypts an **entire partition or storage device** such as USB flash drive or hard drive.
- Encrypts a **partition or drive where Windows is installed** (pre-boot authentication).
- Encryption is **automatic**, **real-time** (on-the-fly) and **transparent**.
- Parallelization and pipelining allow data to be read and written as fast as if the drive was not encrypted.
- Provides **plausible deniability**, in case an adversary forces you to reveal the password:
  Hidden volume (steganography) and **hidden operating system**.
- Encryption algorithms: AES-256, Serpent, and Twofish. Mode of operation: XTS.

  Further information regarding features of the software may be found in the **documentation**.

What is new in TrueCrypt 6.3 (released October 21, 2009)