File systems and databases: managing information

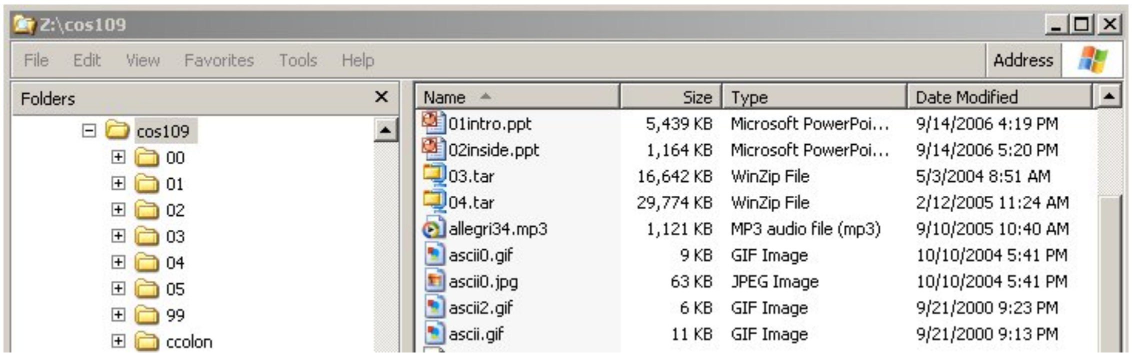
- **file: sequence of bytes stored on a computer**
  - content is arbitrary; any structure is imposed by the creator of the file, not by the operating system
- **file system: software that provides hierarchical storage and organization of files, usually on a single computer**
  - part of the operating system
- **database: integrated collection of logically related records**
  - data is organized and structured for efficient systematic access
- **database system: software that provides efficient access to information in a database**
  - not usually part of an operating system

File Systems: managing stored information

- **logical structure: users and programs see a hierarchy of folders (or 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
- **physical structure: disk drives operate in tracks, sectors, etc.**
  - other storage devices have other physical properties
- **the operating system converts between these two 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**

Disks

- **a place to store information when the power is turned off**
- **usually based on magnetic surfaces, rotating machinery**
- **logical / functional structure: folders (directories) and files**
  - your information: papers, mail, music, web page, …
  - programs and their data: Firefox, Word, iTunes, …
  - operating system(s): Windows, MacOS, Unix/Linux, ...
  - bookkeeping info: where things are physically



cos109
00
01
02
03
04
05
99
ccolon

Name	Size	Type	Date Modified
01intro.ppt	5,439 KB	Microsoft PowerPoi...	9/14/2006 4:19 PM
02inside.ppt	1,164 KB	Microsoft PowerPoi...	9/14/2006 5:20 PM
03.tar	16,642 KB	WinZip File	5/3/2004 8:51 AM
04.tar	29,774 KB	WinZip File	2/12/2005 11:24 AM
allegri34.mp3	1,121 KB	MP3 audio file (mp3)	9/10/2005 10:40 AM
ascii0.gif	9 KB	GIF Image	10/10/2004 5:41 PM
ascii0.jpg	63 KB	JPEG Image	10/10/2004 5:41 PM
ascii2.gif	6 KB	GIF Image	9/21/2000 9:23 PM
ascii.gif	11 KB	GIF Image	9/21/2000 9:13 PM

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

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**

Databases and database systems

- **informally, database is a large collection of information**
- **more formally, an organized collection of logically related records**
- **data items have fixed set of attributes**
  - name, address, phone number, gender, income, social security number, ...
- **each record has these attributes for a single person / instance**

- **database system supports**
  - very efficient search for records with specific properties
    all the women in 08540 with income > $100K
  - high volumes of traffic with concurrent access and update
    "ACID": atomic, consistent, isolated, durable
- **major examples**
  - Oracle (owns Peoplesoft)
  - MySQL (open source, now owned by Sun, in turn owned by Oracle...)
  - SQLite (open source, in devices like iPhone)