File systems, databases, cloud storage

- **file: a sequence of bytes stored on a computer**
  - content is arbitrary (just bytes); 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 (or nearby)**
  - a significant part of the operating system

- **database: an integrated collection of logically related records**
  - data is organized and structured for efficient systematic access
  - may be distributed across lots of machines & geographically dispersed
- **database system: software that provides efficient access to information in a database**
  - not usually part of the operating system

- **cloud storage: the same thing, but on someone else's computers**

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 (e.g., SSD) 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**

How the file system converts logical to physical

- **disk is physically organized into sectors, or blocks 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 (conceptually only) 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 itself, 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 is a file**
  - 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**

  IT DOES NOT CONTAIN THE DATA ITSELF

- **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

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 / trash**
  - recycle bin or trash 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

- **the file system doesn't have to be local**
  - the data could be on some other computer

- **need software 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**

Cloud storage

- **the file system doesn't have to be local**
  - the data could be on some other computer

- **need software 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**
  - connecting to Google Drive or Dropbox or iCloud or ... sets up the connections

- **subsequent reads and writes go through the network instead of the local disk**

Cloud computing

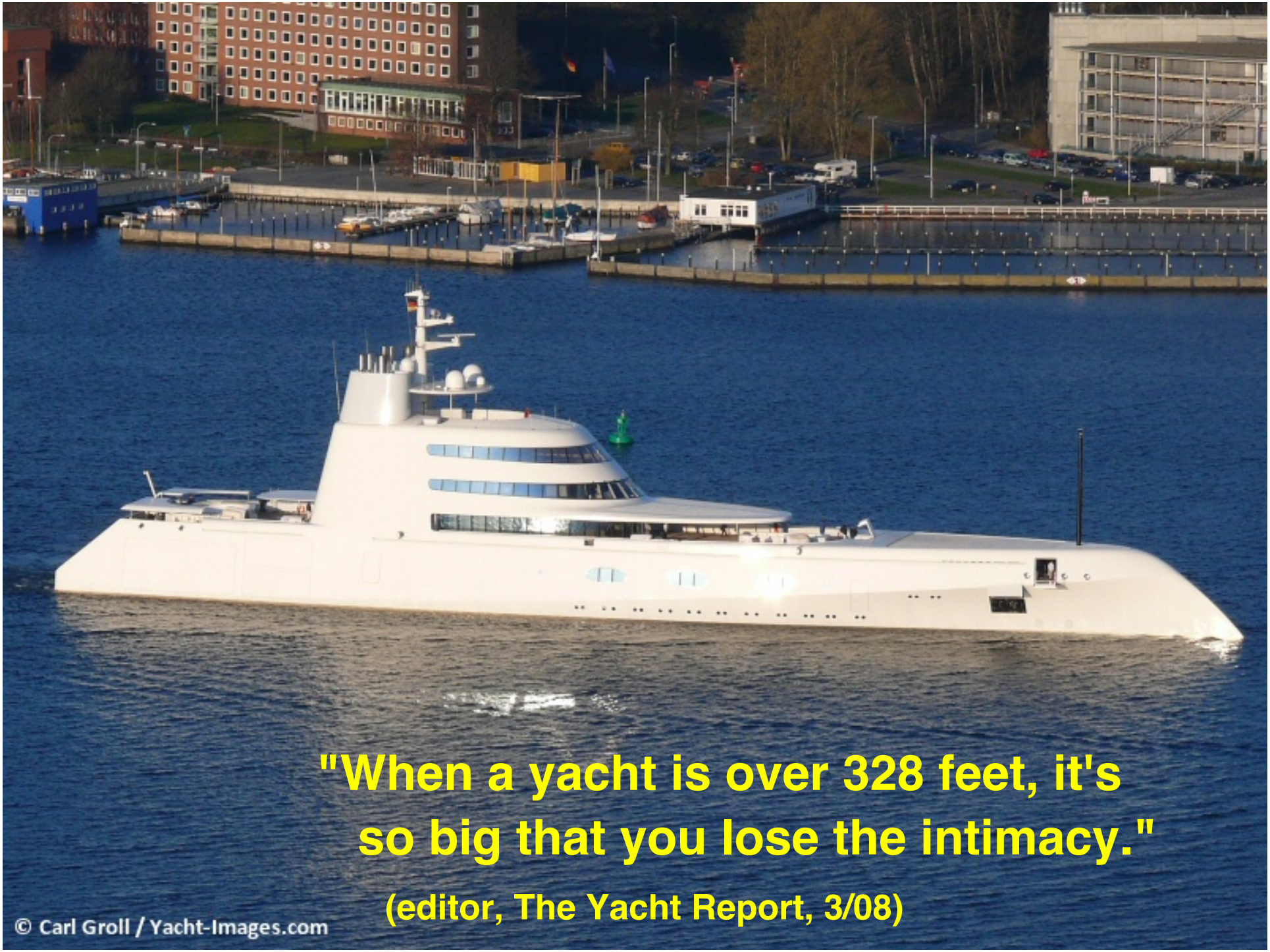
- **put data on computers that are somewhere else**
  - not on your laptop
  - access it via the Internet

- **do (most of) the actual computing on computers that are somewhere else**
  - not on your laptop
  - but owned by someone else
  - use the Internet to connect to the programs and the data

- **Amazon Web Services, Google Cloud Platform, Microsoft Azure, ...**
  - can rent processors, operating systems, data storage, ...
  - scales easily, easier to administer,
- **relies on virtual machines**
  - gives users the appearance of having their own hardware and systems

Browser as operating system

- **a browser provides many of the services that an operating system does**
  - can use "the cloud" for storage and computation
  - programs mostly run in cloud; browser is an interface
  - email, social networks, games, Google docs (and similar), ...
- **how about a *computer* that only runs a browser?**
  - Chromebook: runs Chrome OS (Linux-based operating system)
  - applications and data are in the cloud, not on computer itself
  - very little local storage and local apps


"When a yacht is over 328 feet, it's so big that you lose the intimacy."
(editor, The Yacht Report, 3/08)
© Carl Groll / Yacht-Images.com