Reprise: what an operating system does

- **manages CPU, schedules and coordinates running programs**
  - switches CPU among programs that are actually computing
  - suspends programs that are waiting for something (e.g., disk, network)
  - keeps individual programs from hogging resources
- **manages memory (RAM)**
  - loads programs in memory so they can run
  - swaps them to disk and back if there isn't enough RAM (virtual memory)
  - keeps separate programs from interfering with each other
  - and with the operating system itself (protection)
- **manages and coordinates input/output to devices**
  - disks, display, keyboard, mouse, network, ...
  - provides fairly uniform interface to disparate devices
- **manages files on disk (file system)**
  - provides hierarchy of folders/directories and files for storing information

How applications use the operating system

- **operating system provides its services as functions to be called from application programs**
  - "system calls" in Unix, Application Program Interface ("API") in Windows

    "what is the exact time?"
    "allocate more memory to me"
    "read N bytes from file F into memory location M"
    "write N bytes from memory location M into file F"
    "establish a network connection to www.princeton.edu"
    "write N bytes to the network connection"
    "I'm all done; get rid of me"
- **operating system provides interface for applications to use**
  - programs access machine capabilities only through this interface
  - different physical hardware can provide the same interface
  - programs can be moved to any system that provides the same interface
  - different operating systems can provide the same interface
  - one operating system can simulate the interface provided by another
- **operating system hides details of specific hardware**

Example of system-call level coding

- **C program to copy input to output ("copy" command)**
- **read, write, exit are system calls**

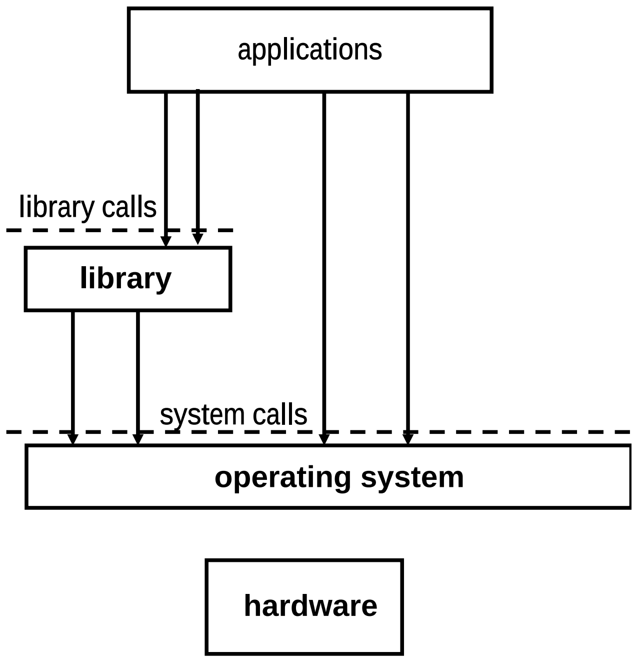
```
main() {
  char buf[8192];
  int n;
  while ((n = read(0, buf, sizeof(buf))) > 0)
    write(1, buf, n);
  exit(0);
}
```

Software is organized into "layers"

- **each layer presents an interface that higher layers can use**
  - defines a "platform" for putting more on top
  - insulates the higher layer from how the lower layer is implemented
  - often called "Application Program Interface" or API
- **operating system ("kernel")**
  - lowest software layer, on top of hardware
    (usually: virtual machine is on top of another program, e.g., an operating system)
  - presents its capabilities as system calls
- **libraries**
  - code to be used as building blocks in programs
  - present their capabilities as functions / APIs
- **applications**
  - e.g., browser, word processor, mailer, compiler, directory lister, ...
  - use libraries and system calls through APIs

Layering

- **an application generally calls multiple libraries**
  - might not make direct system calls
- **a library generally calls other libraries**
- **library and system call levels define interfaces (APIs)**
- **programmers may not know what is "library" and what is "system call"**



Where's the line between OS and applications?

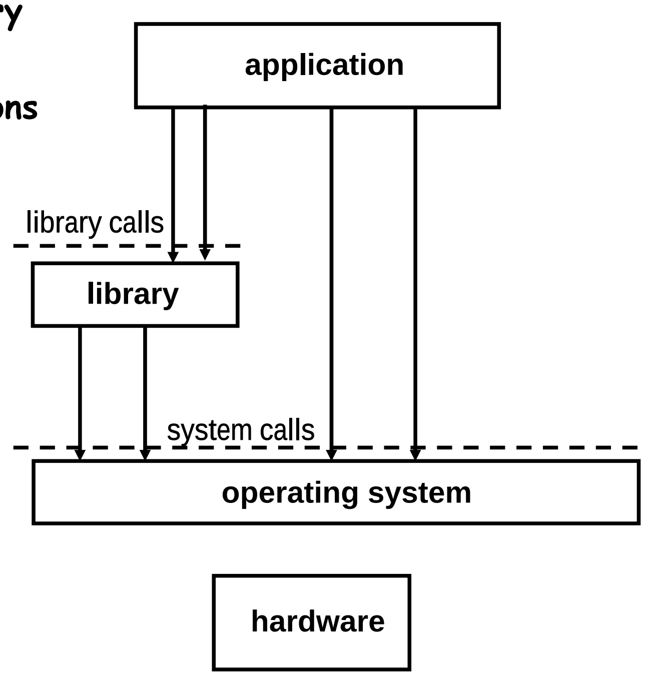
- **there are lots of ways to create layers, glue them together**
- **many choices of what to include in kernel or put in library**

- **"operating system" and "kernel" are not well defined**
  - "Windows" might mean everything (OS, applications, etc)
  - "Windows OS" usually means the part that controls the rest
  - "Linux" may mean "kernel" or may mean "kernel + applications"
  - dividing line is not always clear

- **"kernel"**
  - minimal part that runs regardless of what else the system is being used for or is doing
  - provides essential, central services
  - controls shared resources
  - protects information, enforces privacy and security
  - user programs can only use it through its defined interfaces
  - usually runs in hardware-supported protected mode

Interface issues

- **application/kernel boundary**
- **interface ownership**
- **independent implementations**
- **platforms**
- **middleware**
- **virtual machines**



Where's the line between applications and OS?

- **"operating system" and "kernel" are not well defined**
  - "Windows" might mean everything (OS, applications, etc)
  - "Windows OS" usually means the part that controls the rest
- **Dept of Justice v Microsoft was partly about this question**
  - is Internet Explorer part of the operating system?
  - will the system be damaged or restricted if IE is removed or replaced?
- **Microsoft said Yes, DoJ said No**
  - http://www.usdoj.gov/atr/cases/ms_index.htm

USA v. Microsoft Chronology (you are not expected to remember this)

- **7/94: Microsoft sued for Sherman Antitrust Act violations; judgment against MS**
- **10/97: Dept of Justice says MS is still doing it**
- **5/98: DoJ & 18 states sue MS**
- **10/98-2/99: trial with judge Thomas Jackson**
  - Prof Ed Felten one of government expert witnesses
  - showed it was quite possible to remove IE from Win95 without harm

- **11/99: "findings of fact": MS is a monopoly, used its power to stifle competition**
- **4/00: "findings of law": MS violated Sherman act**
- **6/00: Judge Jackson approves proposal to split MS into an "operating system company" and an "applications company"**
- **6/01: appeals court affirms conclusions, but overturns breakup remedy**
- **8/01: new judge Colleen Kollar-Kotelly to decide penalties, remedies**
- **9/01: DoJ backs off on major issues**
- **11/01: proposed settlement:**
  - MS can add anything to OS, can't keep OEMs from installing other software, can't retaliate, must charge uniform prices, must disclose technical information

- **2/02: 11 states pursue suit anyway**
  - Prof Andrew Appel an expert witness for states
- **11/02: Kollar-Kotelly's decision basically leaves settlement alone**
- **10/05: Kollar-Kotelly scolds Microsoft for delay, violations of settlement**
- **10/07: 7 states want agreement extended for 5 more years!**
- **8/13/09: latest joint status report on compliance**

What's an API?

Operating systems perform many functions, including allocating computer memory and controlling peripherals such as printers and keyboards. Operating systems also function as platforms for software applications. They do this by "exposing" — i.e., making available to software developers — routines or protocols that perform certain widely-used functions. These are known as Application Programming Interfaces, or "APIs."

Excerpted from Final Judgment

State of New York, et al v. Microsoft Corporation

US District Court, District of Columbia, Nov 1, 2002

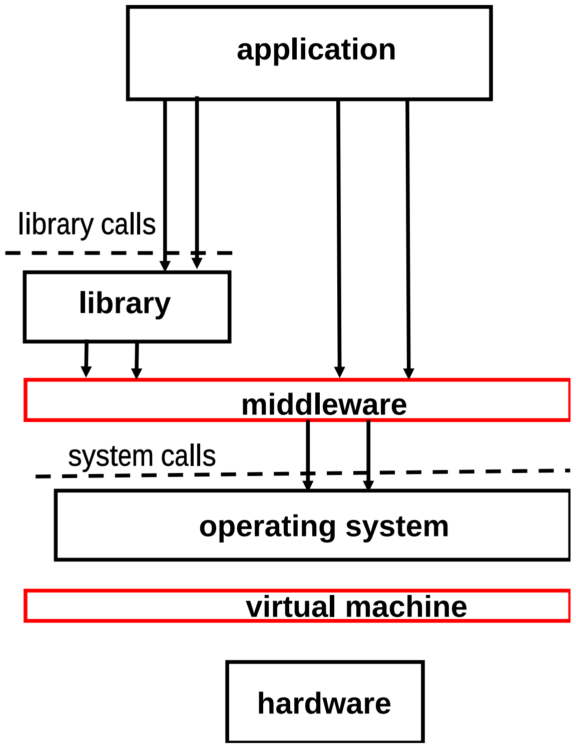
Platforms, middleware, virtual machines

- **platform: hardware or software on which applications can run**
- **middleware: uses OS interface but exposes its own APIs to developers, so applications using it can move to any OS where the middleware has been moved**

  (e.g., browser-based software)

- **virtual machine: software that mimics behavior of hardware so other software can run on it**

  (can be above the operating system too, as in VMWare)



Cloud computing

- **buzzword du jour?**
  - 'Cloud' has been a go-to metaphor for almost as long as the Internet has existed, conveying a sense that the Internet was intangible and bigger than the sum of its parts."

    (Wall Street Journal, 9/23/08)

- **software services delivered via the Internet**
  - Gmail, Yahoo mail
  - Facebook, Twitter, ...
  - Flickr
  - Google Docs
  - Windows Live
  - Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.