# Lecture 11: Interfaces

- **In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans and combinations of these.**
  - (Wikipedia, the source of all truth)

- **there has to be agreement about what information is exchanged and how**

- **lots of technical issues**

- **surprisingly, some important legal issues**

# Reprise: what an operating system does

- **manages CPU(s), 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, buses, 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 services to be accessed by application programs**
  - Unix "system calls", Windows Application Programming Interface ("API")
    "what is the exact time?"
    "allocate M more bytes of RAM to me"
    "read N bytes from file F into memory starting at location M"
    "write N bytes from memory locations starting at M into file F"
    "set up 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 an 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 Programming Interface" or API

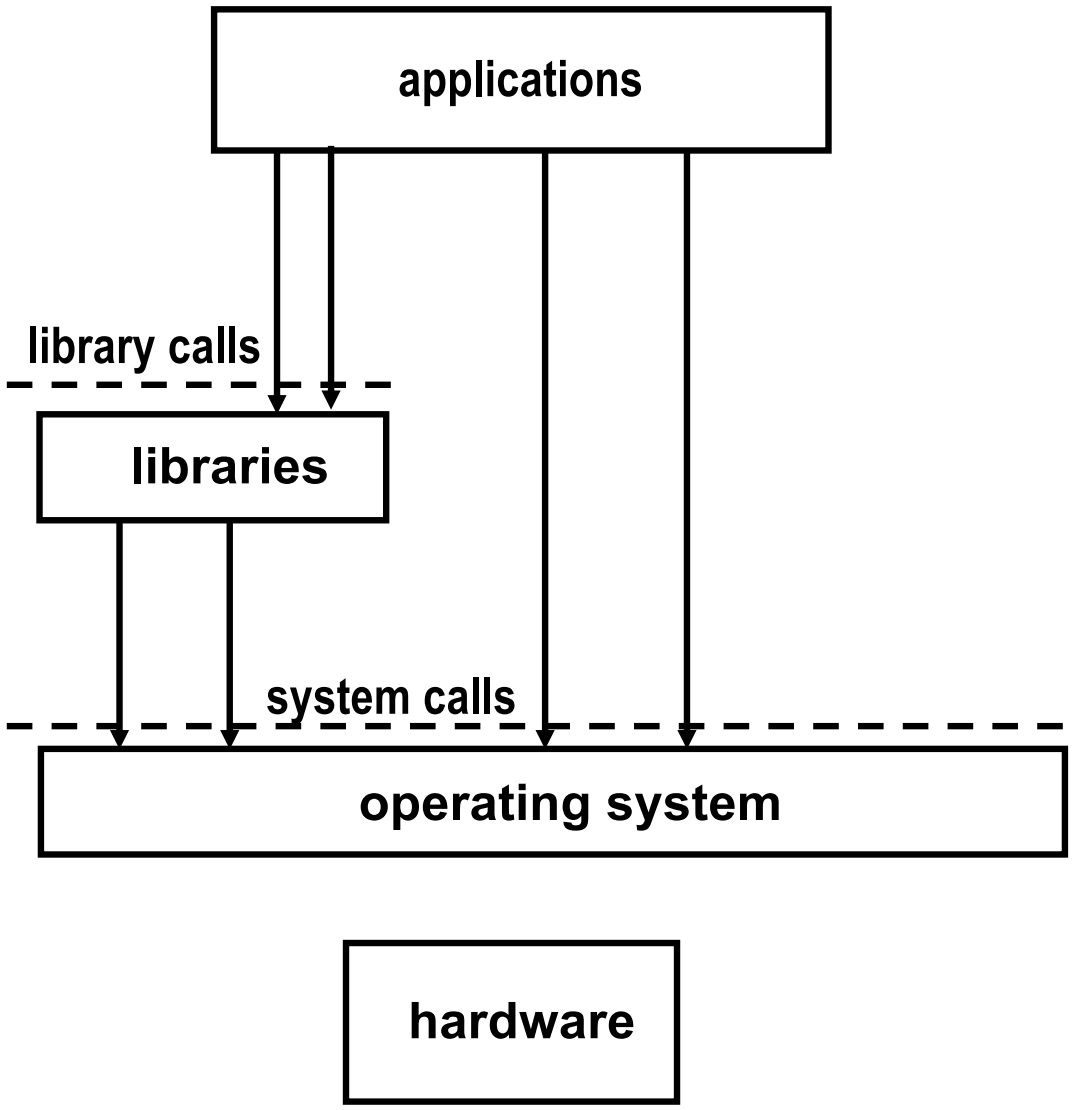
- **operating system ("kernel")**
  - lowest software layer, on top of hardware
    (usually: virtual machine is on top of another program, e.g., an operating system)
  - presents its capabilities as system calls

- **libraries**
  - code to be used as building blocks in programs
  - present their capabilities as 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"**



# 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

# Sample Python API

**input**([*prompt*])

If the *prompt* argument is present, it is written to standard output without a trailing newline. The function then reads a line from input, converts it to a string (stripping a trailing newline), and returns that. When EOF is read, `EOFError` is raised. Example:

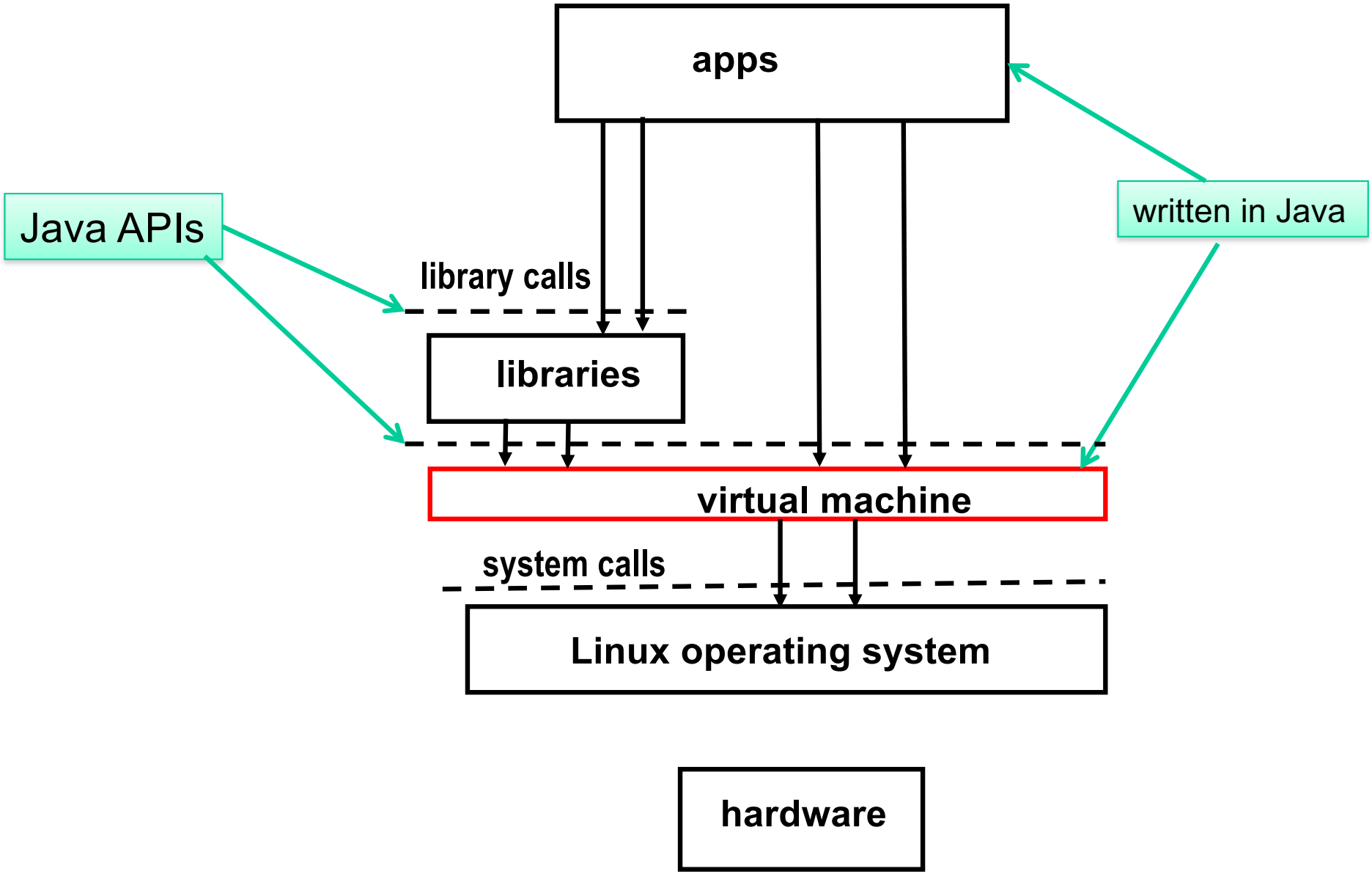
```
>>> s = input('--> ')
--> Monty Python's Flying Circus
>>> s
"Monty Python's Flying Circus"
```

If the `readline` module was loaded, then `input()` will use it to provide elaborate line editing and history features.

Raises an auditing event `builtins.input` with argument `prompt` before reading input

Raises an auditing event `builtins.input/result` with the result after successfully reading input.

# Android phone organization



# Oracle v Google (from the decision in May, 2012)

**The Java language, like C and C++, is a human-readable language. Code written in a human-readable language — "source code" — is not readable by computer hardware.**

**Only "object code," which is not human-readable, can be used by computers. Most object code is in a binary language, meaning it consists entirely of 0s and 1s. Thus, a computer program has to be converted, that is, compiled, from source code into object code before it can run, or "execute". In the Java system, source code is first converted into "bytecode," an intermediate form, before it is then converted into binary machine code by the Java virtual machine.**

# SUPREME COURT OF THE UNITED STATES

Syllabus

GOOGLE LLC *v.* ORACLE AMERICA, INC.

CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

No. 18–956. Argued October 7, 2020—Decided April 5, 2021

(e) The fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world. Applying the principles of the Court's precedents and Congress' codification of the fair use doctrine to the distinct copyrighted work here, the Court concludes that Google's copying of the API to reimplement a user interface, taking only what was needed to allow users to put their accrued talents to work in a new and transformative program, constituted a fair use of that material as a matter of law. In reaching this result, the Court does not overturn or modify its earlier cases involving fair use. Pp. 35–36.

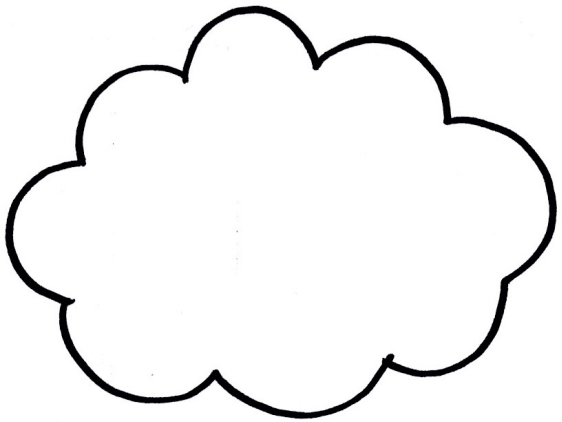
886 F. 3d 1179, reversed and remanded.

BREYER, J., delivered the opinion of the Court, in which ROBERTS, C. J., and SOTOMAYOR, KAGAN, GORSUCH, and KAVANAUGH, JJ., joined. THOMAS, J., filed a dissenting opinion, in which ALITO, J., joined. BARRETT, J., took no part in the consideration or decision of the case.

# Cloud computing APIs

- **'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, ...
  - Facebook, Twitter, Instagram, …
  - Google Docs, calendar,
  - Windows Live, Office 360
  - Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform

- **most cloud services have an API for access by programs**