# Reprise: what an operating system does

- **manages CPUs, 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 services to be accessed by application programs**
  - Unix "system calls", Windows Application Programming Interface ("API")
    "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 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

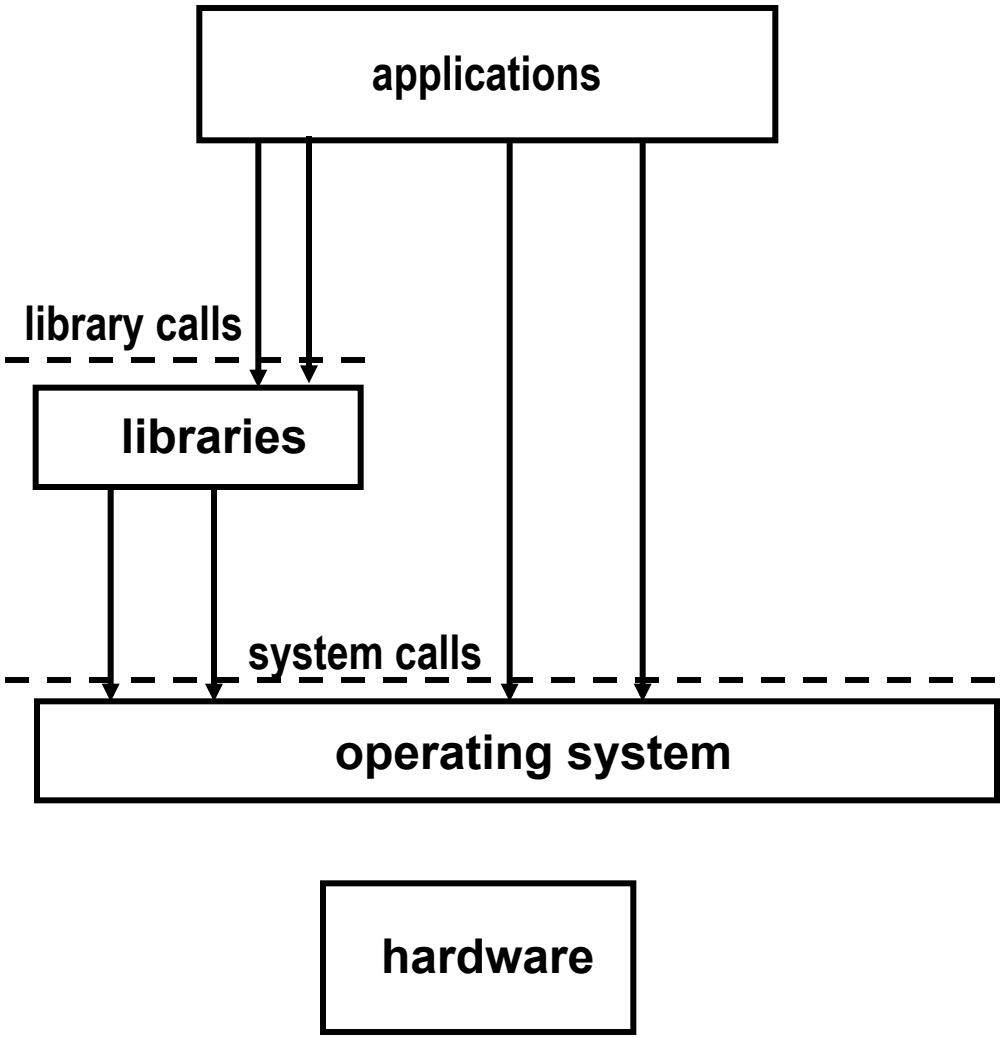
- **operating system ("kernel")**
  - lowest software layer, on top of hardware
    (usually: virtual machine is on top of another program, e.g., an operating system)
  - presents its capabilities as system calls

- **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

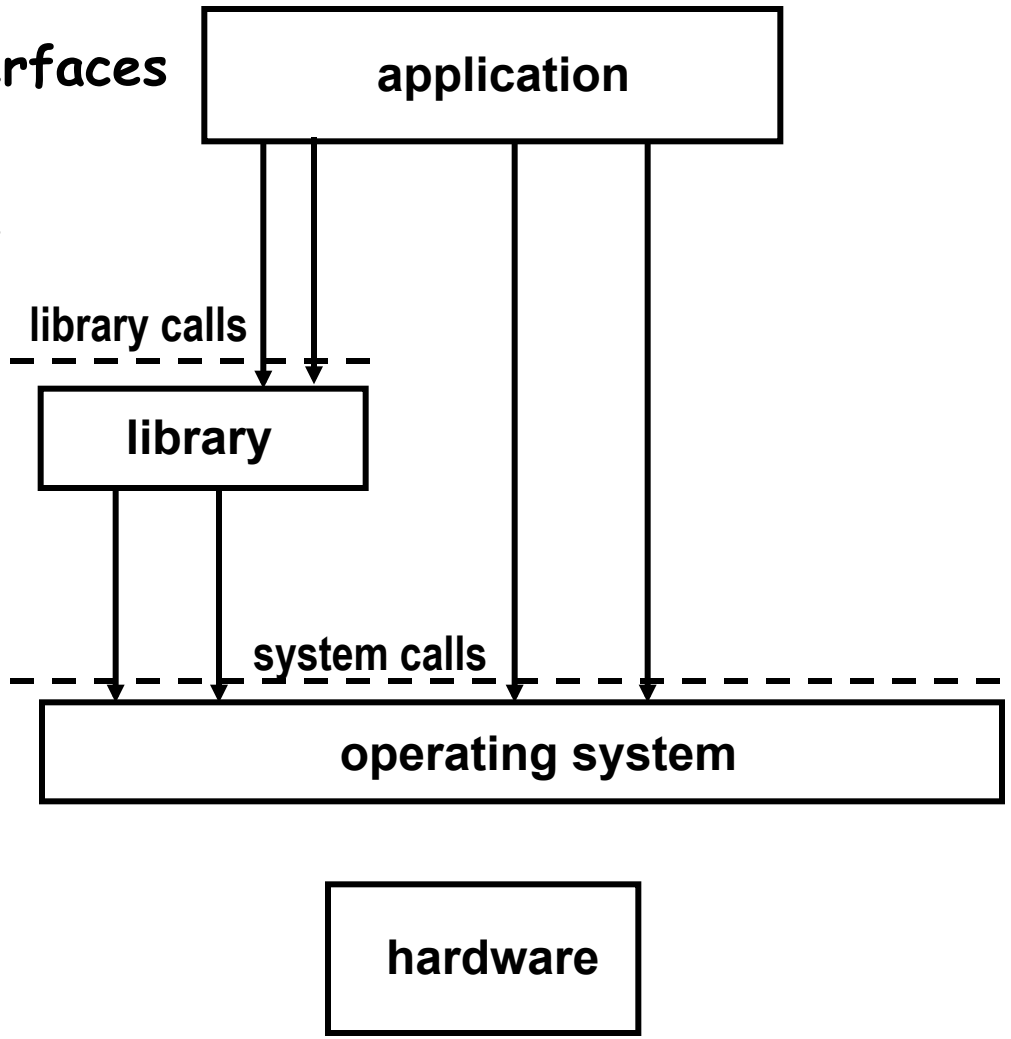
# Layering

- **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"**



# Interface issues

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



# Where's the line between OS and applications?

- **there are lots of ways to create layers and 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

# Microsoft antitrust case (1994-2011)

- **"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

- **what is operating system and what is application?**

- **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

# 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

# API fragment

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

## Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

### Syntax

```
alert("sometext");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
```

# Sample Java API (tiny excerpt)

**Java™ 2 Platform Standard Ed. 5.0**

All Classes

Packages
java.applet
java.awt
java.awt.color

*XPathExpression*
XPathExpressionExceptio
XPathFactory
XPathFactoryConfiguratio
*XPathFunction*
XPathFunctionException
*XPathFunctionResolver*
*XPathVariableResolver*
ZipEntry
ZipException
ZipFile
ZipInputStream
ZipOutputStream
ZoneView
_BindingIteratorImplBase
_BindingIteratorStub
_DynAnyFactoryStub
_DynAnyStub
_DynArrayStub
_DynEnumStub
_DynFixedStub
_DynSequenceStub
_DynStructStub

## Constructor Summary

**ZipFile**(File file)
Opens a ZIP file for reading given the specified File object.

**ZipFile**(File file, int mode)
Opens a new ZipFile to read from the specified File object in the specified mode.

**ZipFile**(String name)
Opens a zip file for reading.

## Method Summary

| | |
|---|---|
| void | **close**()<br>Closes the ZIP file. |
| Enumeration<? extends ZipEntry> | **entries**()<br>Returns an enumeration of the ZIP file entries. |
| protected void | **finalize**()<br>Ensures that the close method of this ZIP file is called when there are no more references to it. |
| ZipEntry | **getEntry**(String name)<br>Returns the zip file entry for the specified name, or null if not found. |
| InputStream | **getInputStream**(ZipEntry entry)<br>Returns an input stream for reading the contents of the specified zip file entry. |
| String | **getName**()<br>Returns the path name of the ZIP file. |
| int | **size**()<br>Returns the number of entries in the ZIP file. |

# Independent implementations of an interface

- **who owns an interface?**
- **can interfaces be owned?**

- **company A sells something (hardware or software)**
- **company A publishes (widely) the API for programming it**
  - with the intent that third parties will develop applications for the thing
  - and thus make it more attractive so company A will sell more

- **company B uses A's interface definition to make a cheaper version of the thing that works the same**
  - so all the third-party applications will run on B's cheaper version
  - thus cutting into A's market

- **company A sues company B**

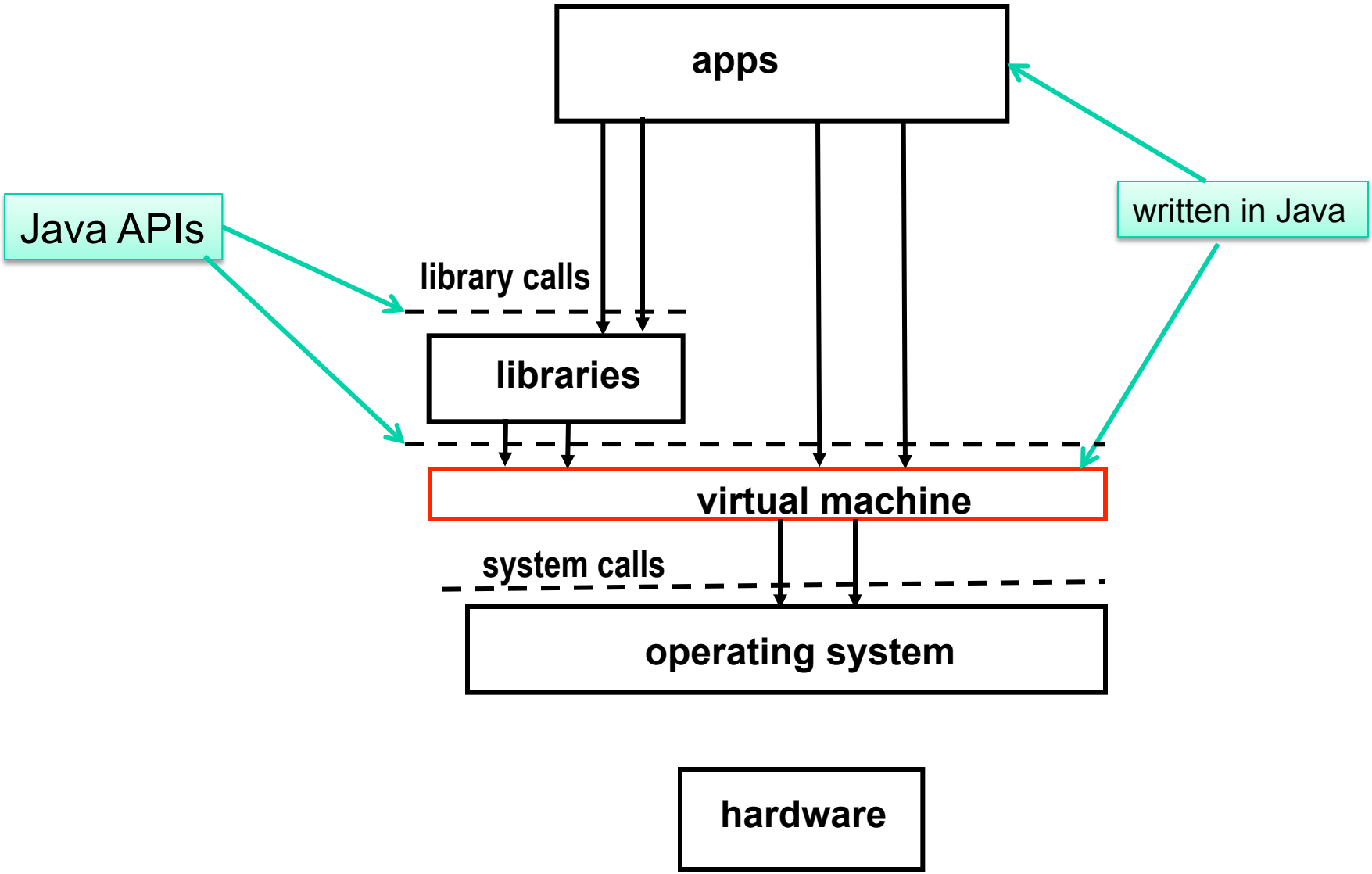
- **who should win?**

# Oracle v Google (May 2010)

- **partly patent, partly copyright**
- **patent part thrown out**
- **remaining copyright part mostly about Java APIs**

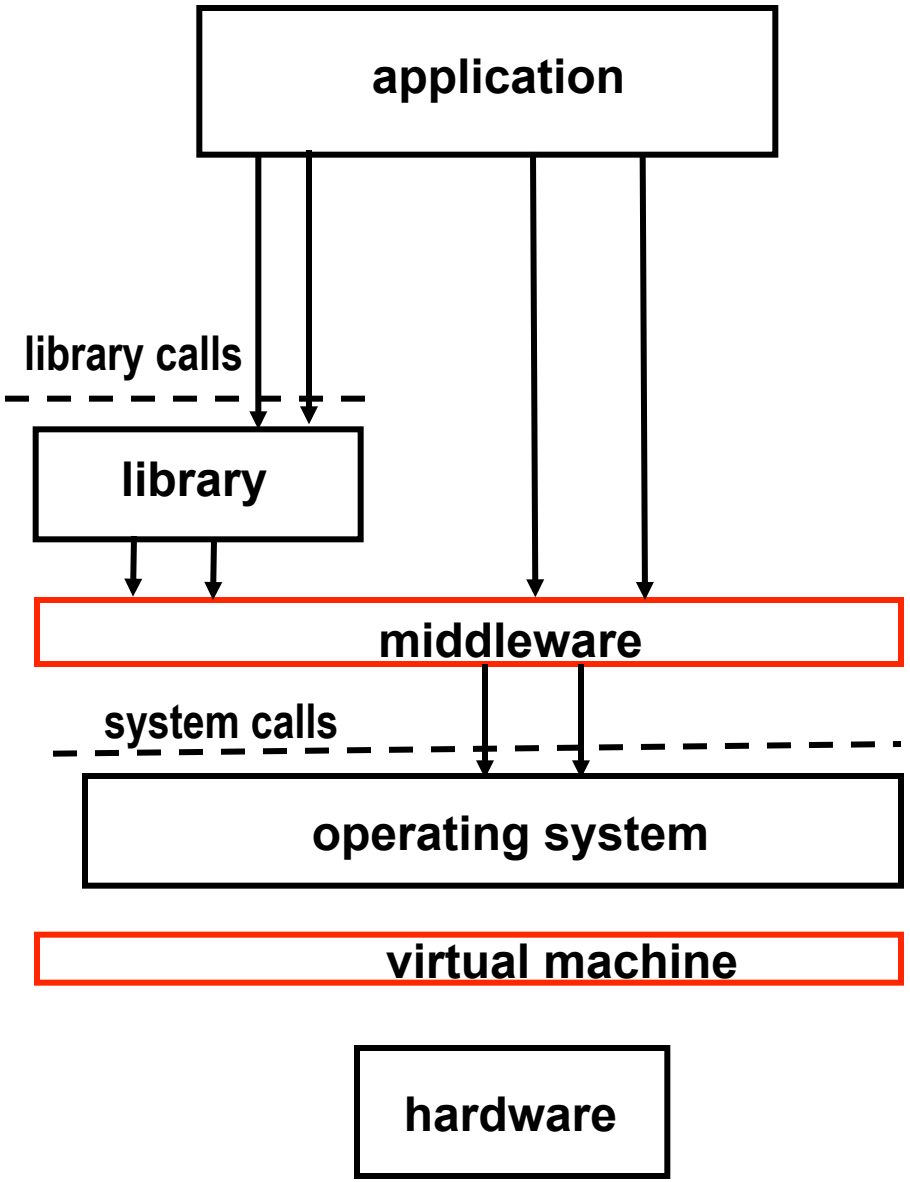
- **did Google violate Oracle's copyright by re-using the APIs verbatim?**
- **(re-implementing the code behind them was not at issue)**

# Android phone organization



# 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)

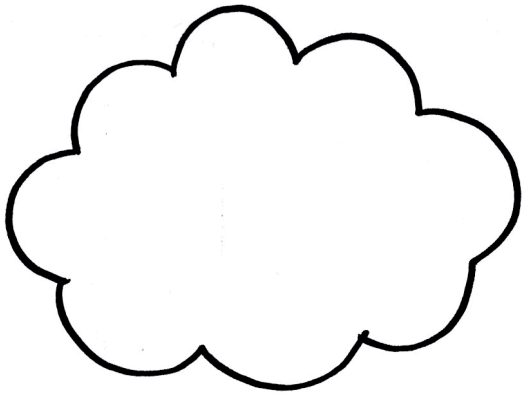


# Cloud computing

- **'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, Office 360
  - Amazon Web Services (AWS)



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