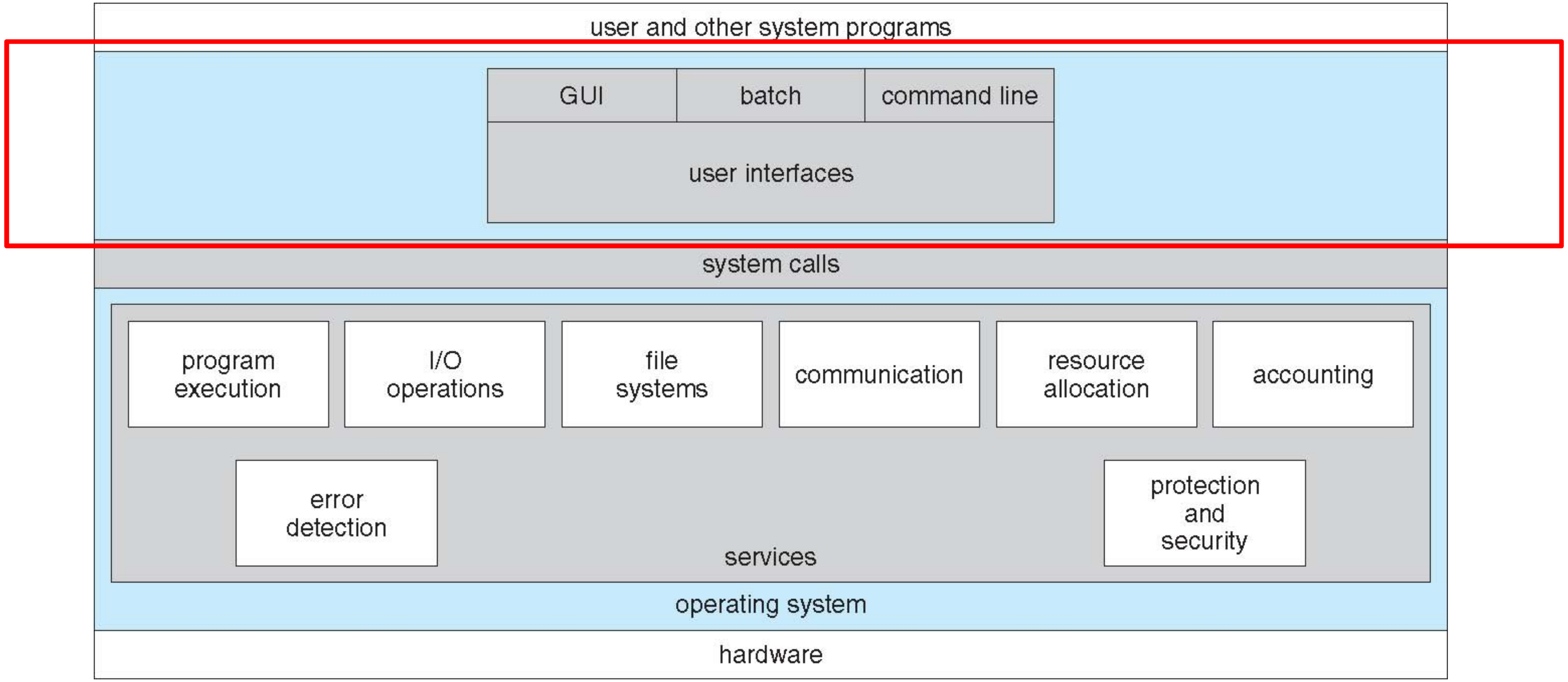
Operating-System Structures

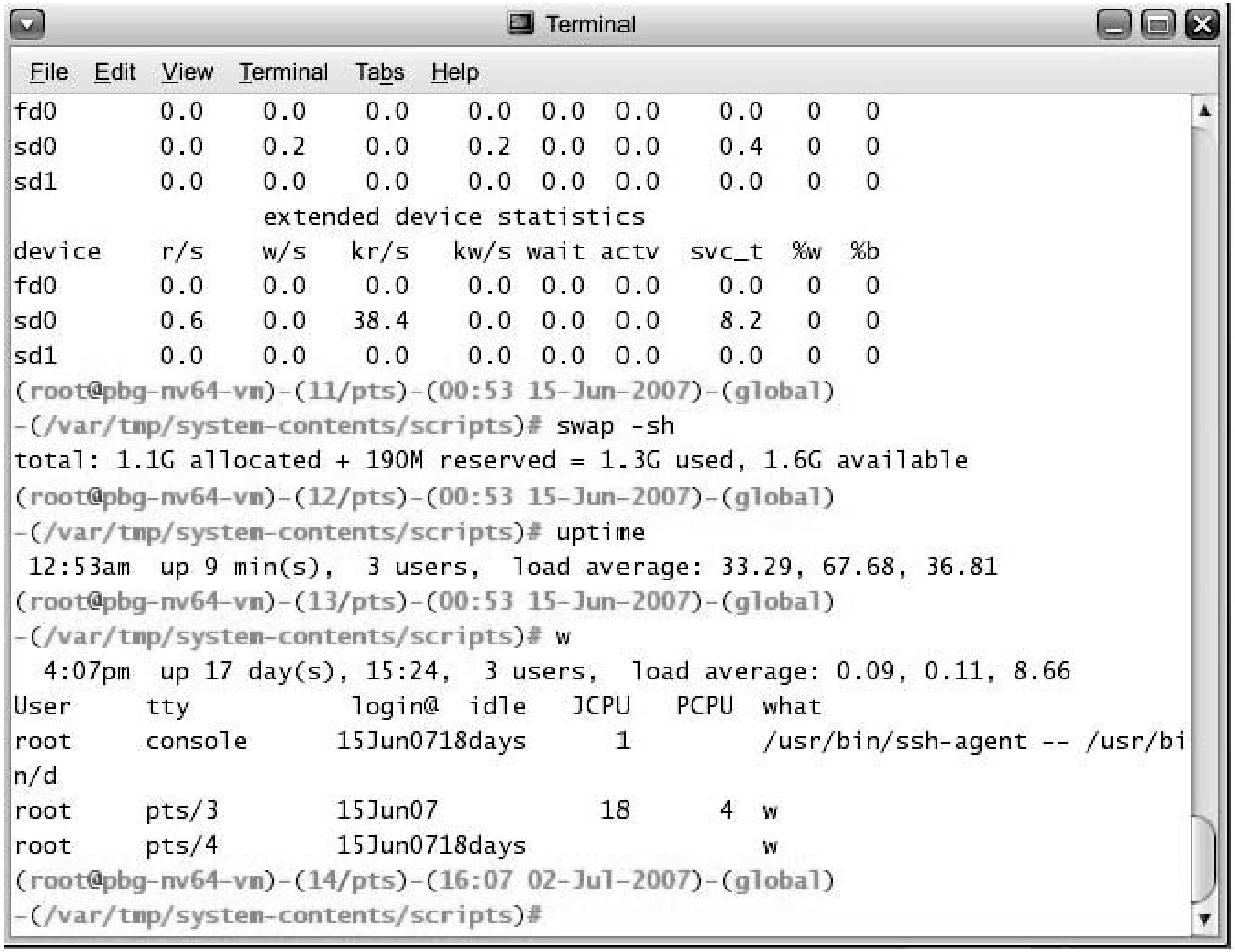
Fan Wu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Spring 2020

Operating System Services Structure



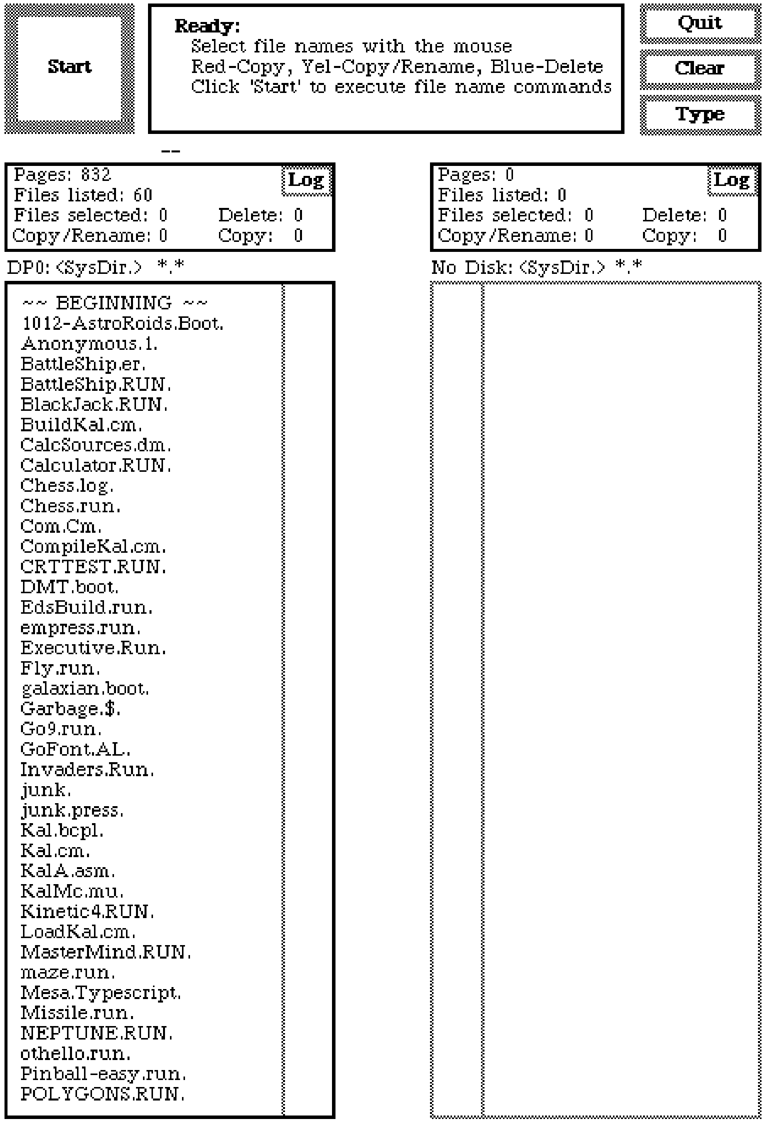
Operating systems provide an environment for execution of programs and services to programs and users

Bourne Shell Command Interpreter



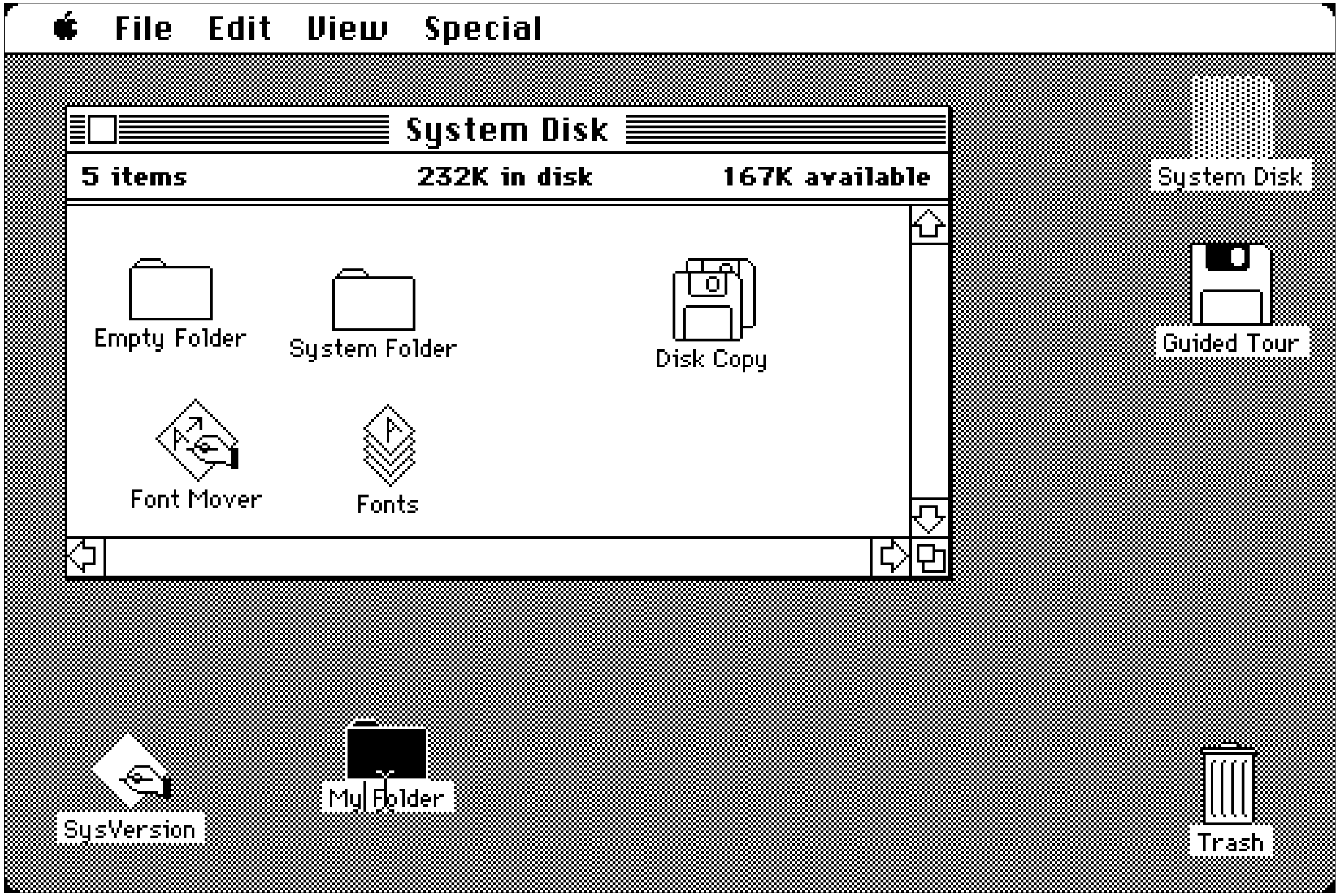
```
Terminal
File  Edit  View  Terminal  Tabs  Help
fd0         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0    0
sd0         0.0     0.2     0.0     0.2  0.0  0.0     0.4   0    0
sd1         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0    0
                    extended device statistics
device      r/s     w/s     kr/s    kw/s wait actv  svc_t  %w   %b
fd0         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0    0
sd0         0.6     0.0    38.4     0.0  0.0  0.0     8.2   0    0
sd1         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0    0
(root@pbg-nv64-vm)-(11/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vm)-(12/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# uptime
 12:53am  up 9 min(s),  3 users,  load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vm)-(13/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# w
  4:07pm  up 17 day(s), 15:24,  3 users,  load average: 0.09, 0.11, 8.66
User     tty           login@  idle   JCPU   PCPU  what
root     console      15Jun0718days      1         /usr/bin/ssh-agent -- /usr/bi
n/d
root     pts/3        15Jun07           18      4  w
root     pts/4        15Jun0718days                w
(root@pbg-nv64-vm)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

First GUI (1973)

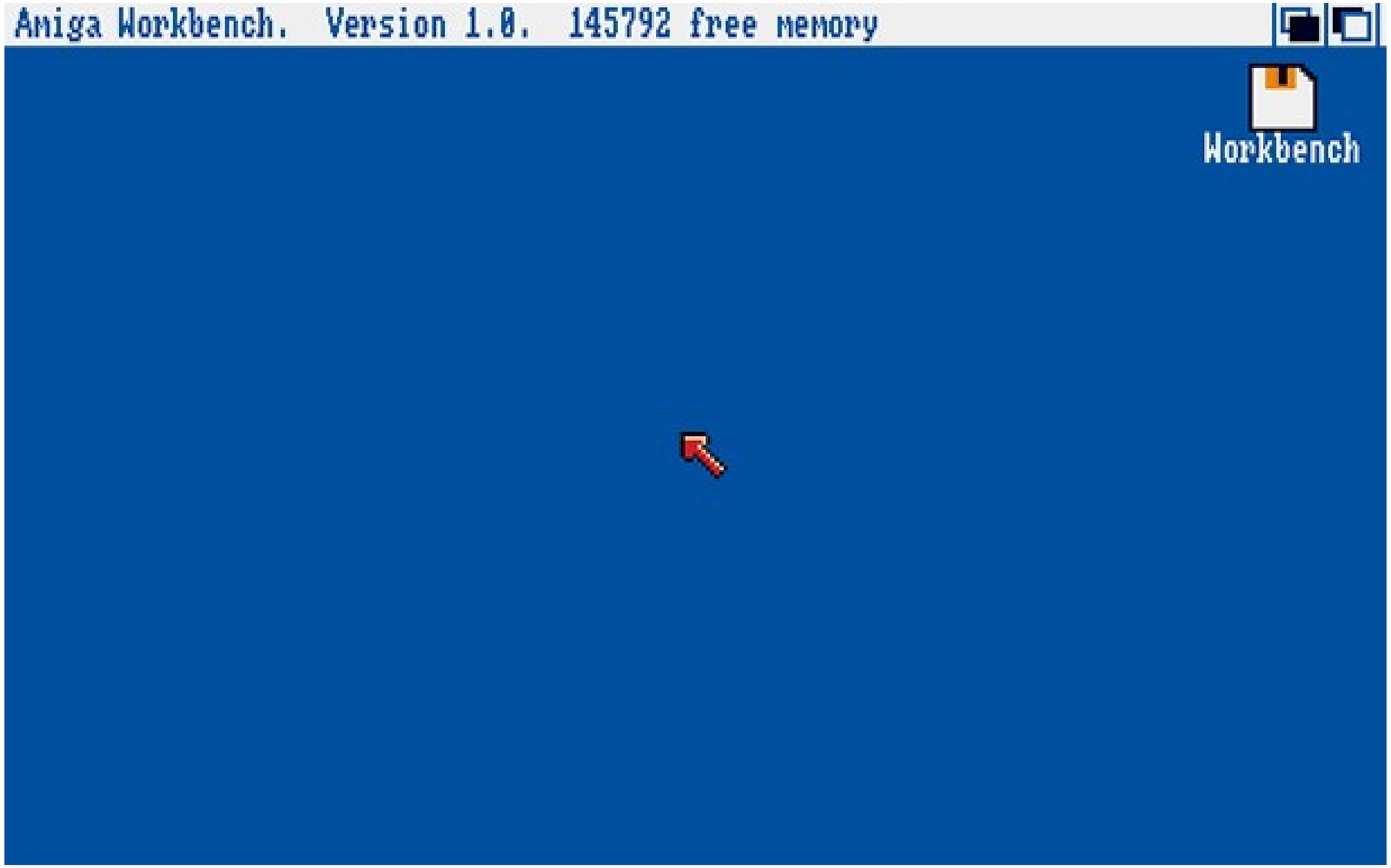


The first appeared on the Xerox Alto computer in 1973.

Mac OS System 1.0 (1984)

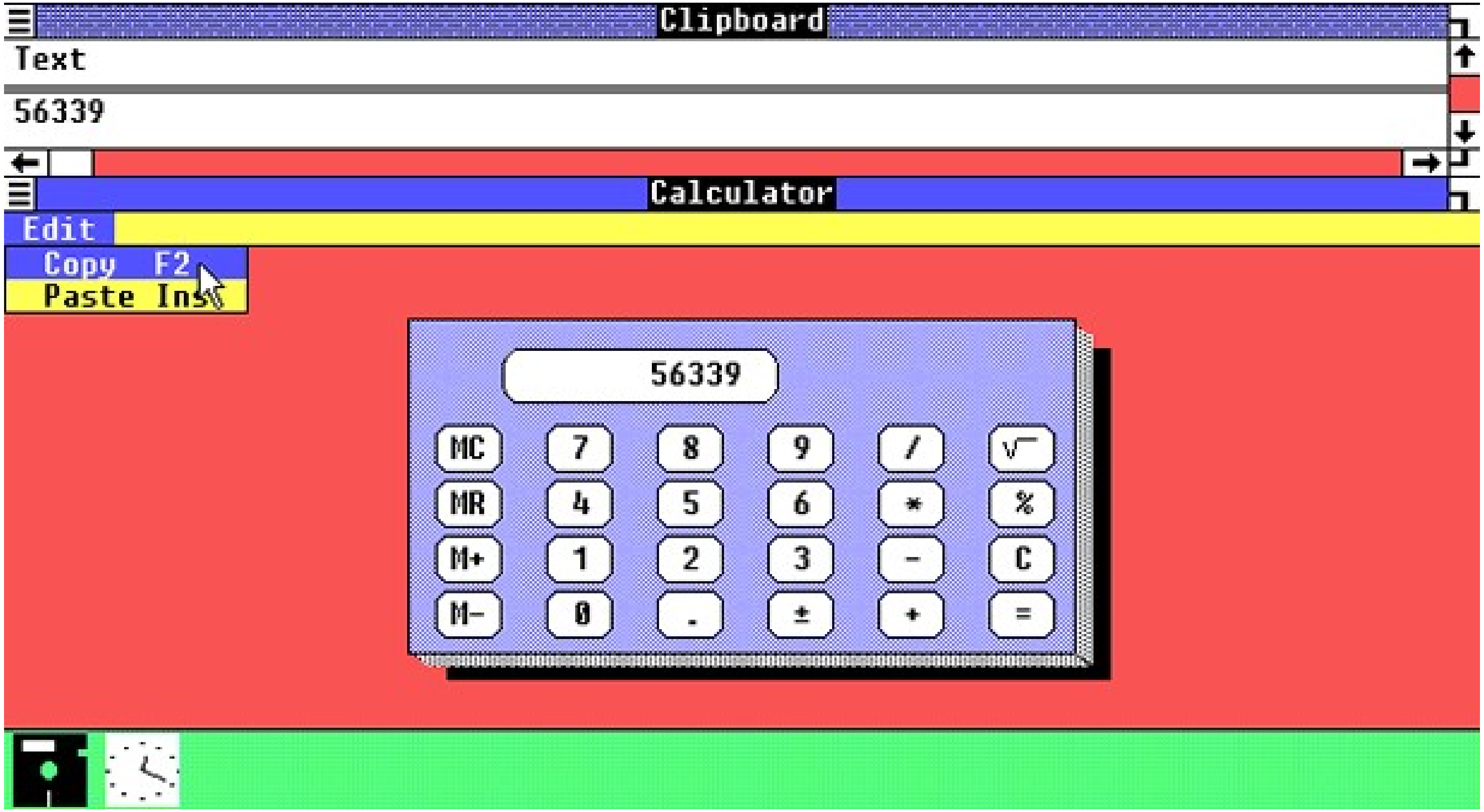


Amiga Workbench 1.0 (1985)

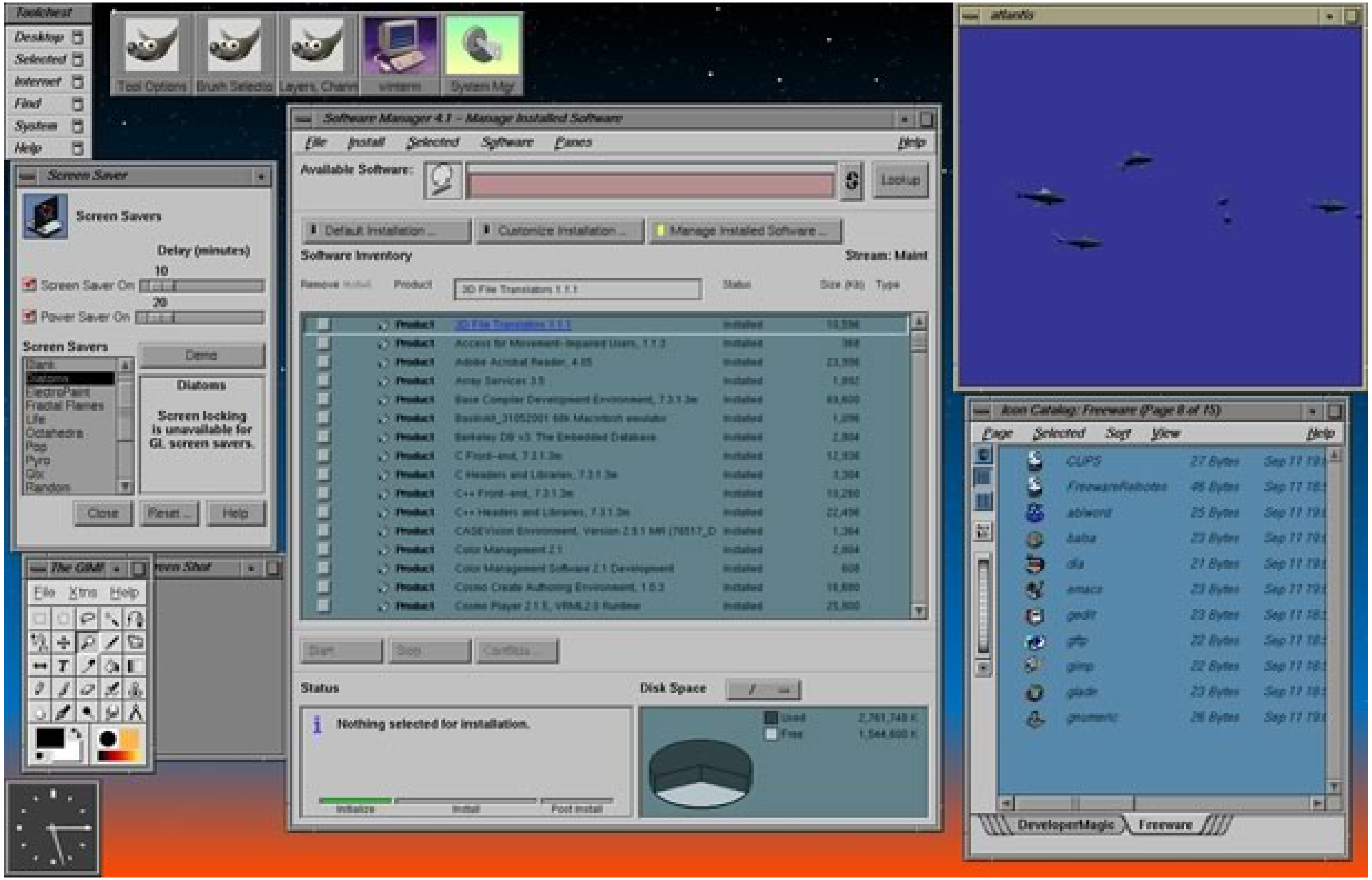


The first GUI with color graphics.

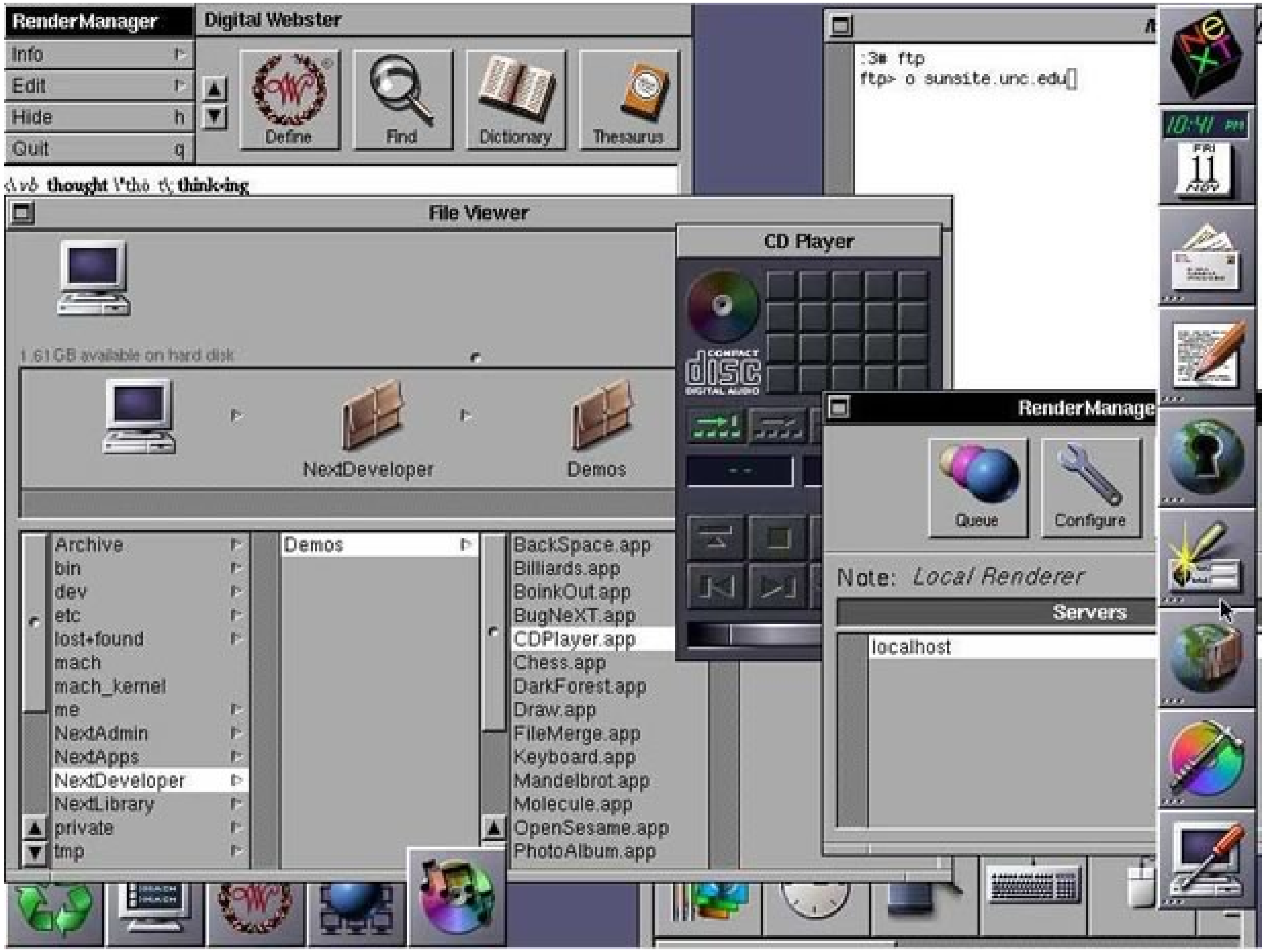
Windows 1.0x (1985)



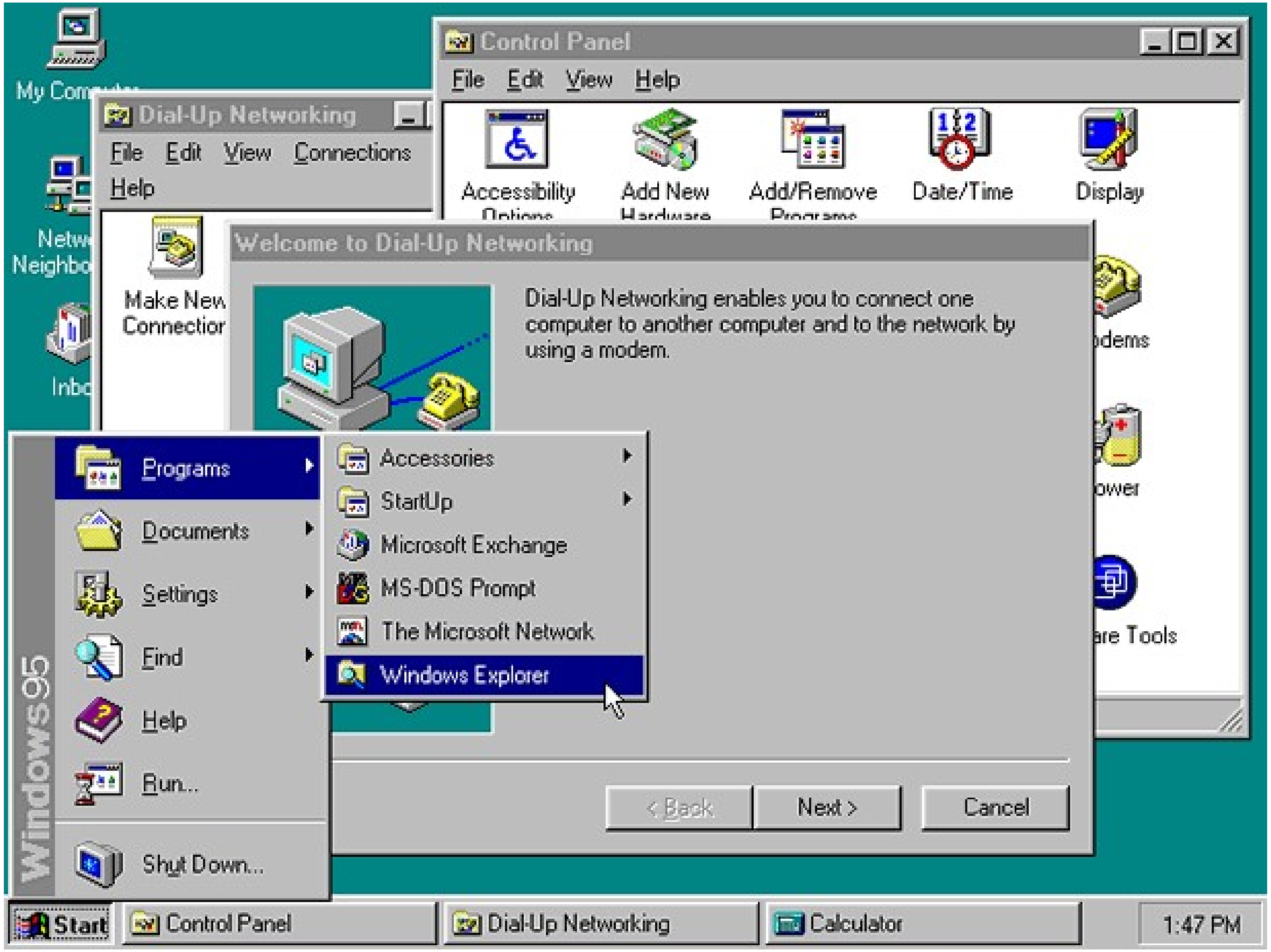
IRIX 3 (released in 1986, first release 1984)



NeXTSTEP / OPENSTEP 1.0 (1989)



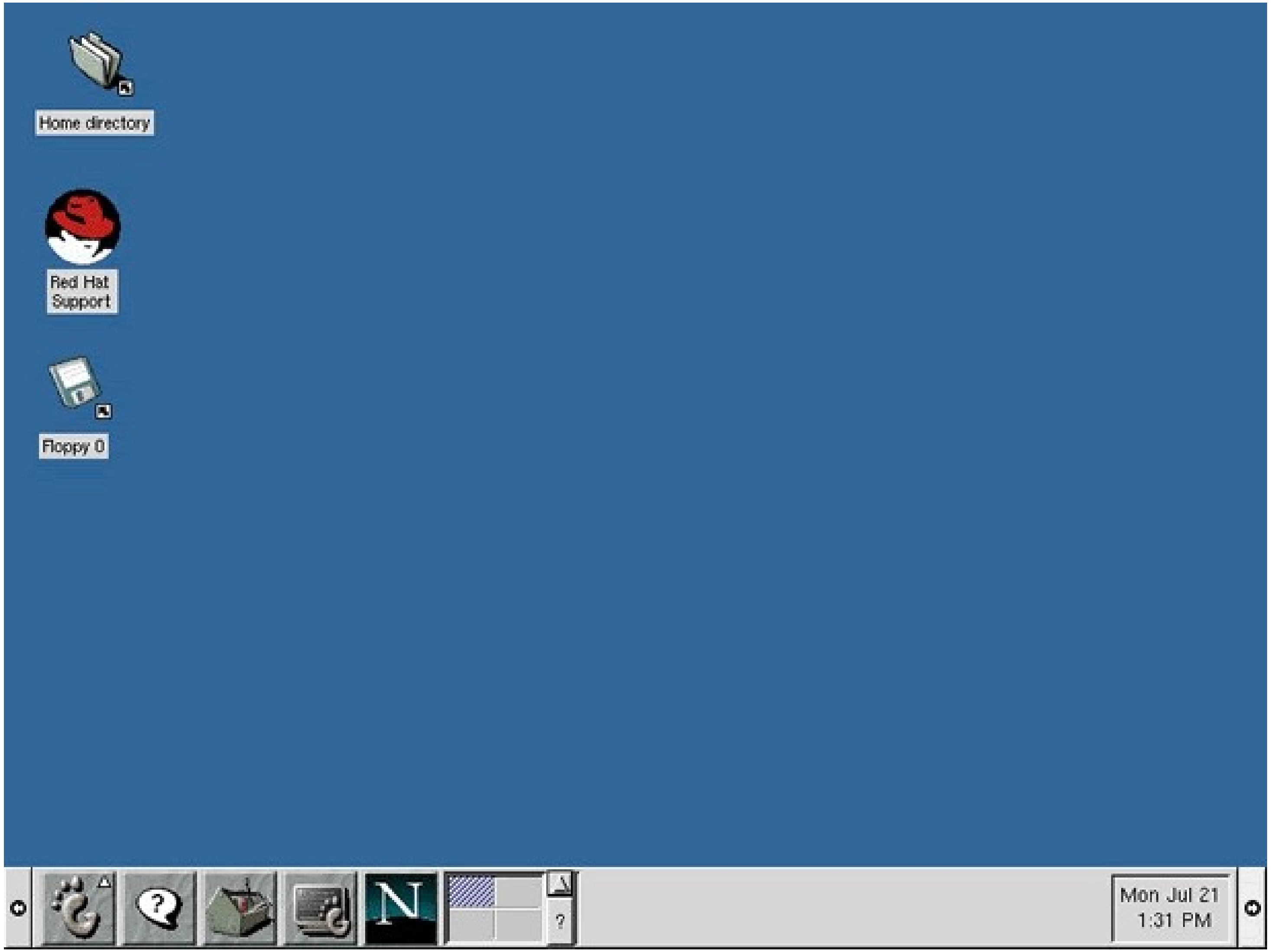
Windows 95 (1995)



KDE 1.0 (1998)

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

GNOME 1.0 (1999)

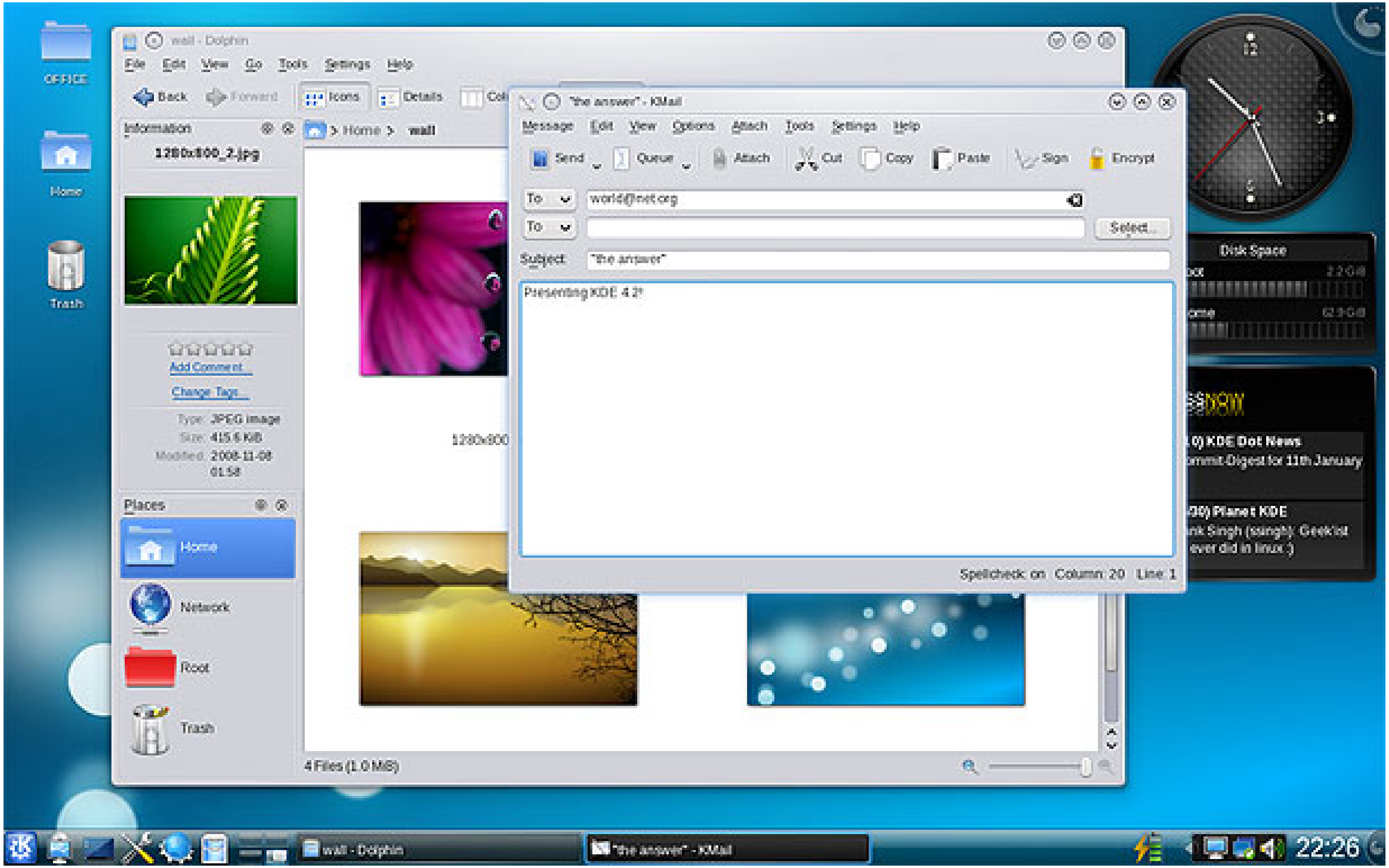


Windows XP (released in 2001)

Windows Vista (released in 2007)

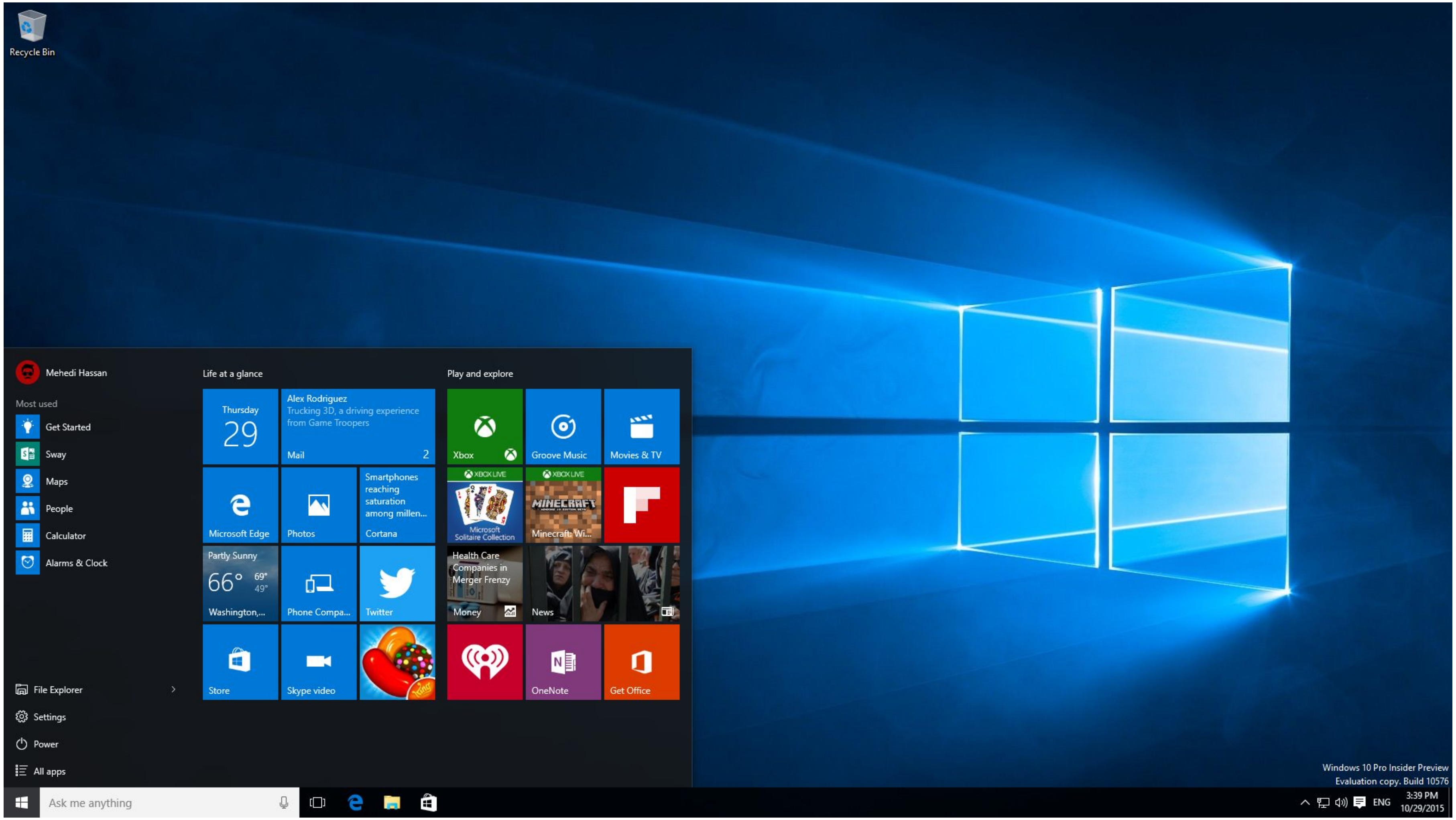
Mac OS X Leopard (released in 2007)

KDE (v4.0 Jan. 2009, v4.2 Mar. 2009)

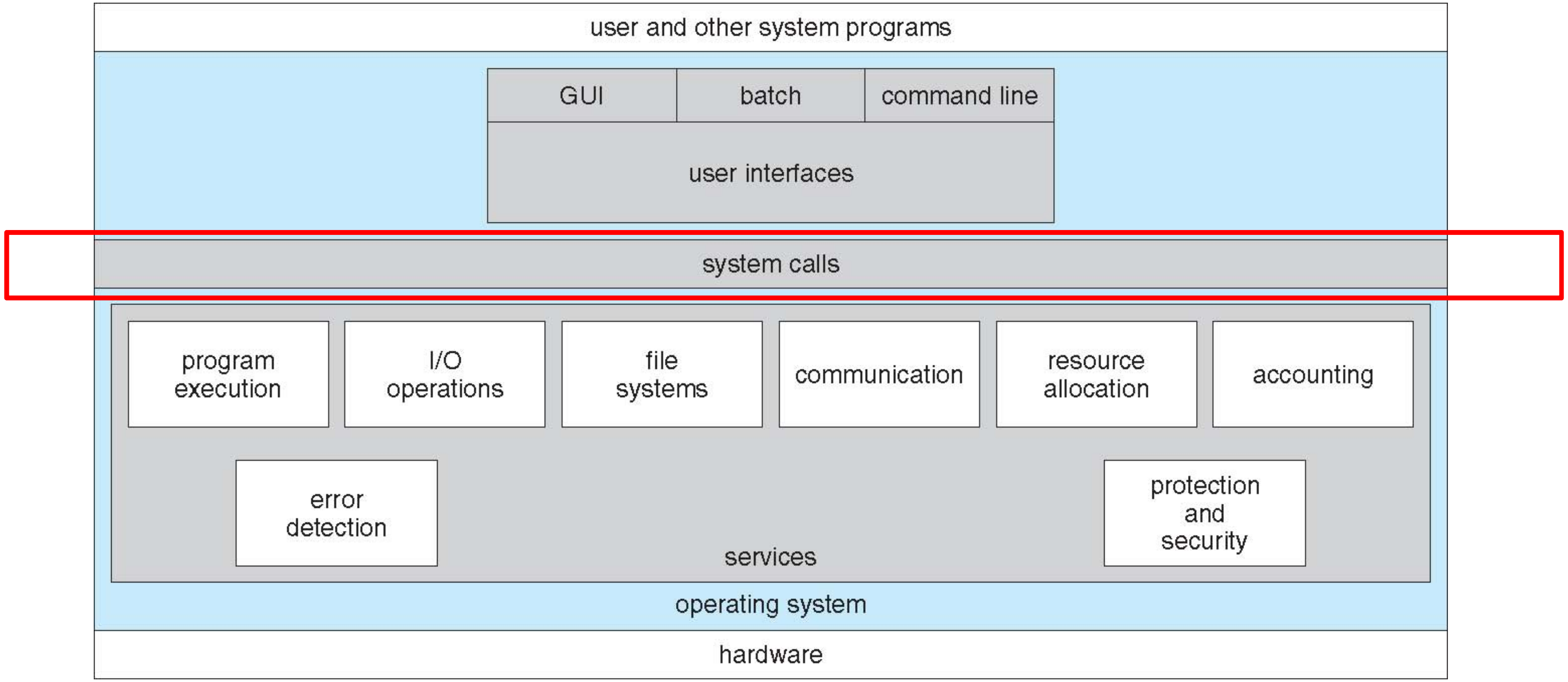


上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Windows 10 (July 2015)

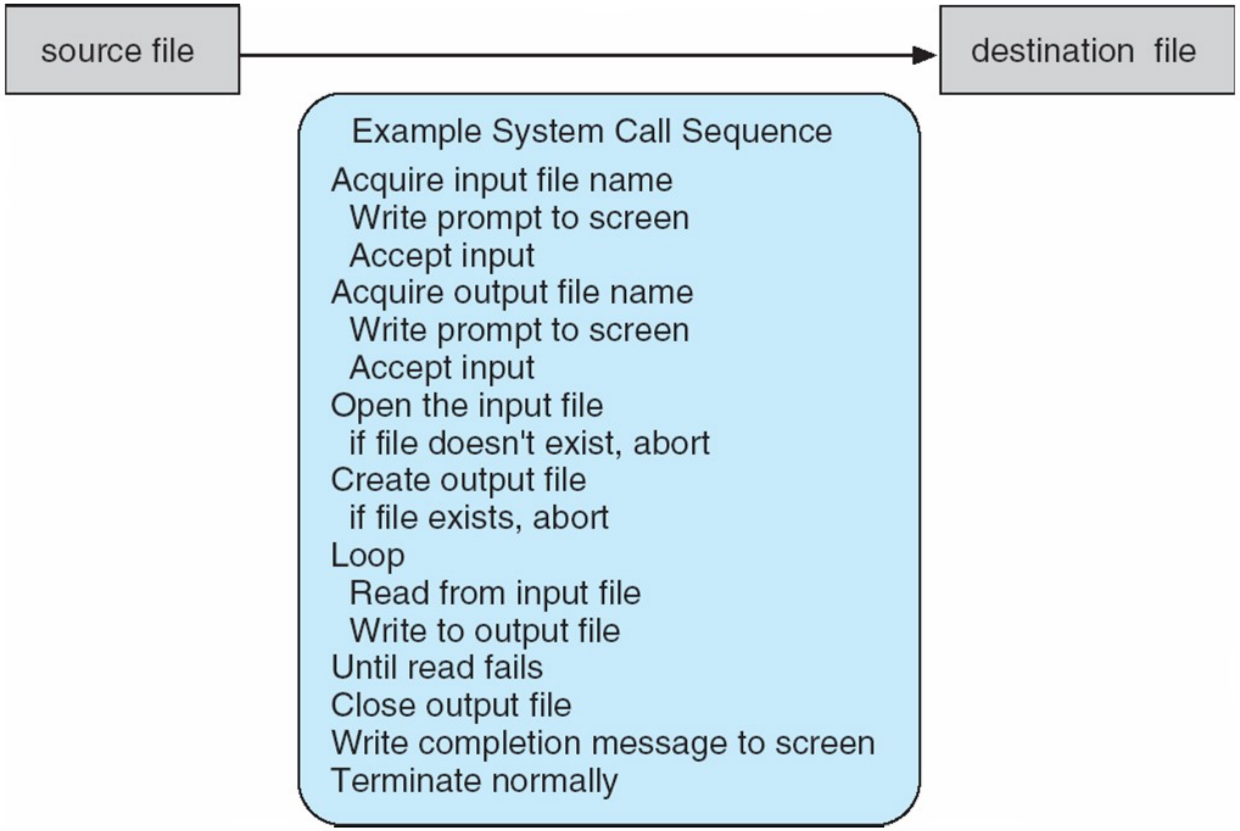


A View of Operating System Services

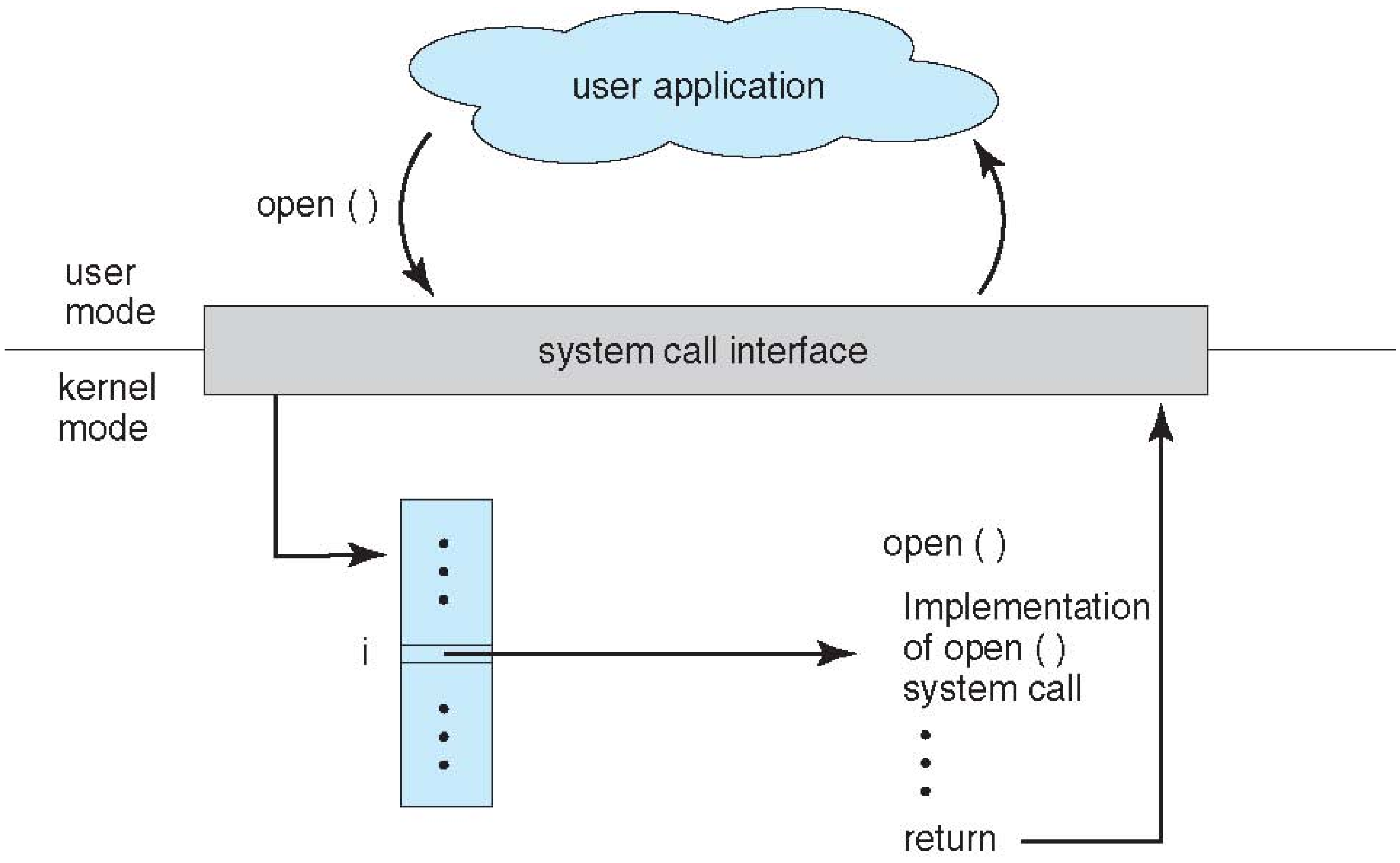


System Call

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Example: System call sequence to copy the contents of one file to another file



System Call – OS Relationship



API

- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
- Three most common APIs
  - Win32 API for Windows
  - POSIX API for POSIX-based systems (UNIX, Linux, and Mac OS X)
  - Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?
  - Program portability
  - System calls are often more detailed and difficult to work with than the API

Example of Standard API

- Consider the ReadFile() function in the
- Win32 API—a function for reading from a file

return value

```
BOOL   ReadFile   (HANDLE        file,
                   LPVOID        buffer,
                   DWORD         bytes To Read,
                   LPDWORD       bytes Read,
                   LPOVERLAPPED  ovl);
```

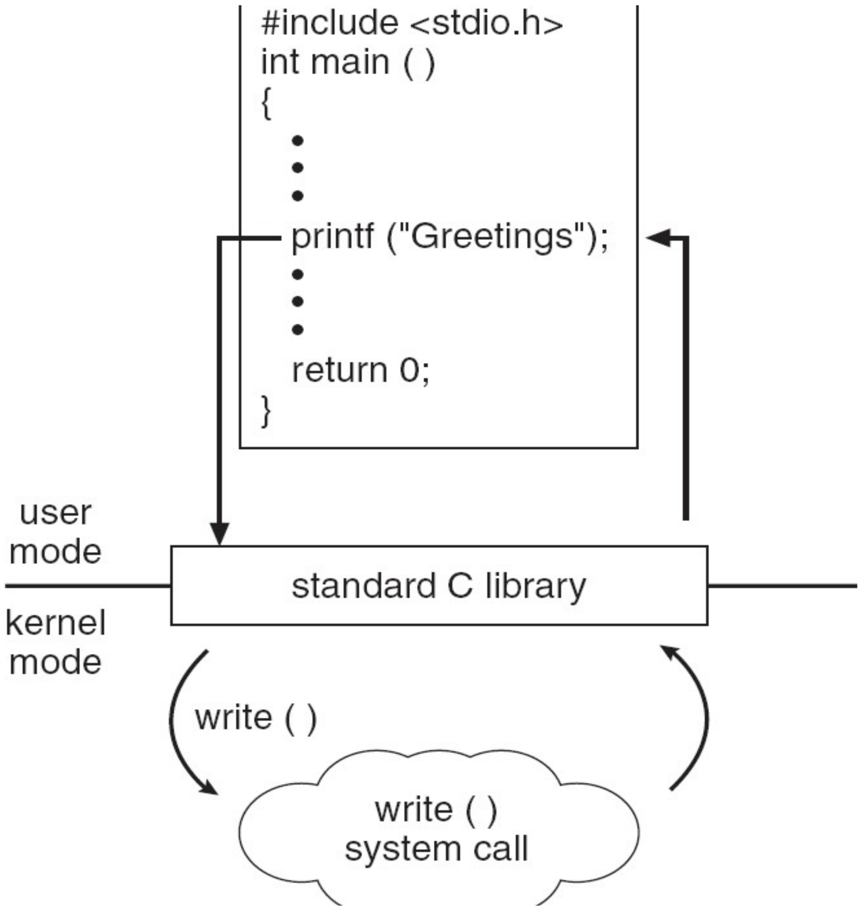
function name

parameters

- A description of the parameters passed to ReadFile()
  - HANDLE file—the file to be read
  - LPVOID buffer—a buffer where the data will be read into and written from
  - DWORD bytesToRead—the number of bytes to be read into the buffer
  - LPDWORD bytesRead—the number of bytes read during the last read
  - LPOVERLAPPED ovl—indicates if overlapped I/O is being used

Standard C Library Example

- C program invoking printf() library call, which calls write() system call

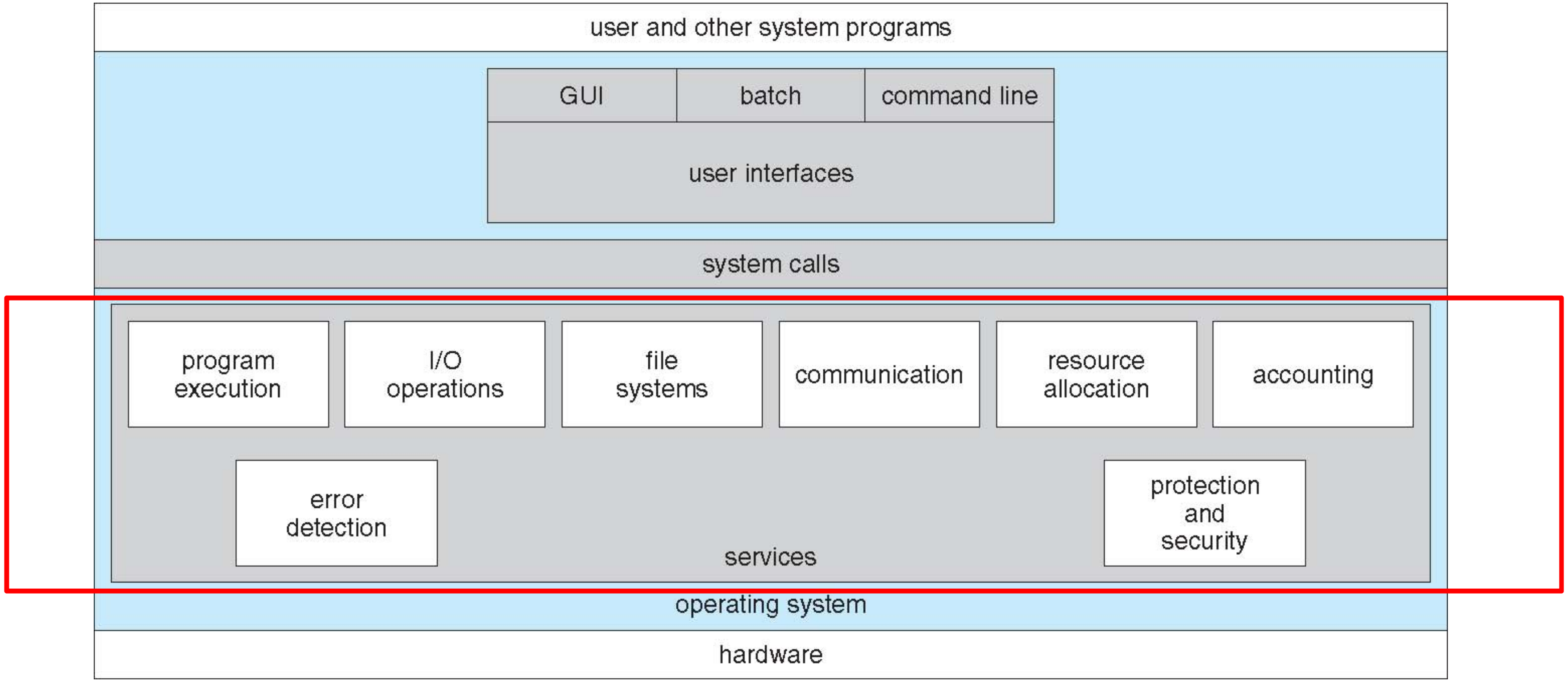


上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

A View of Operating System Services



Operating System Services

- Operating-system services:
  - **User interface** - Almost all operating systems have a user interface (UI).
    - **Graphics User Interface (GUI)**, **Command-Line (CLI)**, **Batch**
  - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device
  - **File-system manipulation** - Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

Operating System Services (Cont.)

- **Communications** – Processes may exchange information, on the same computer or between computers over a network
  - Communications may be via shared memory or through message passing (packets moved by the OS)
- **Error detection** – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Operating System Services (Cont.)

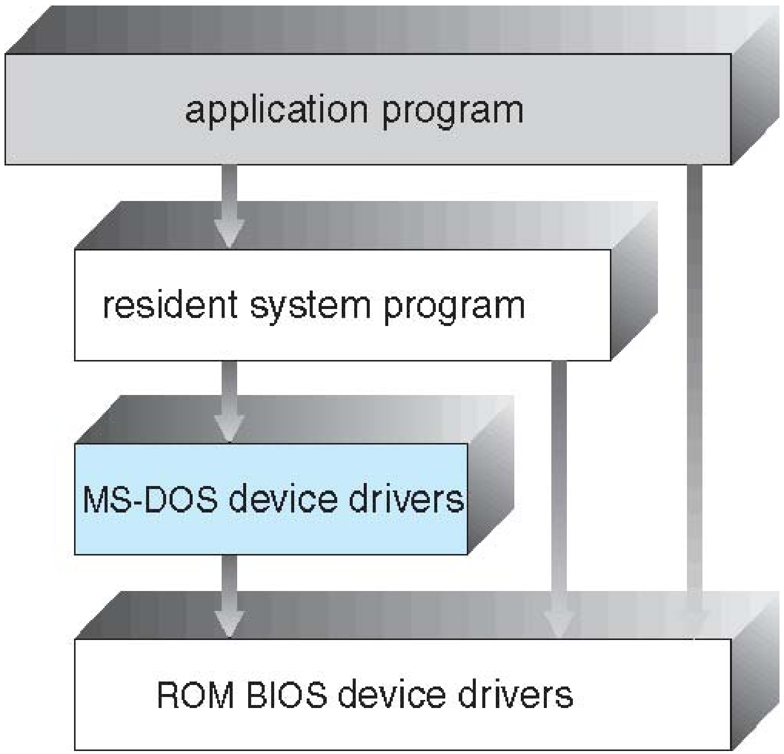
- **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
- **Accounting -** To keep track of which users use how much and what kinds of computer resources
- **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
  - **Protection** involves ensuring that all access to system resources is controlled
  - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

Operating-System Structure

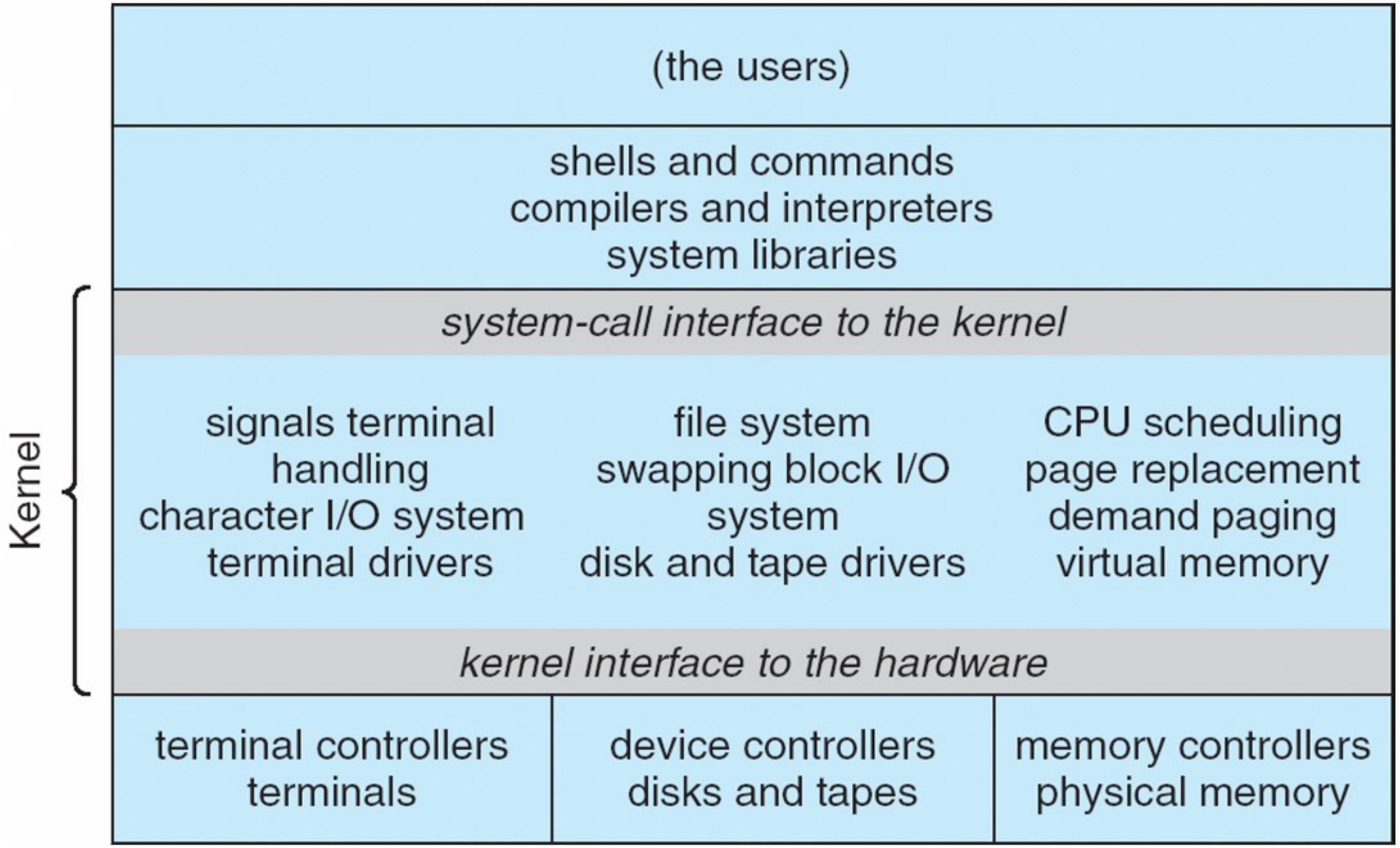
Structure of Components and Interconnections

Simple Structure

- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

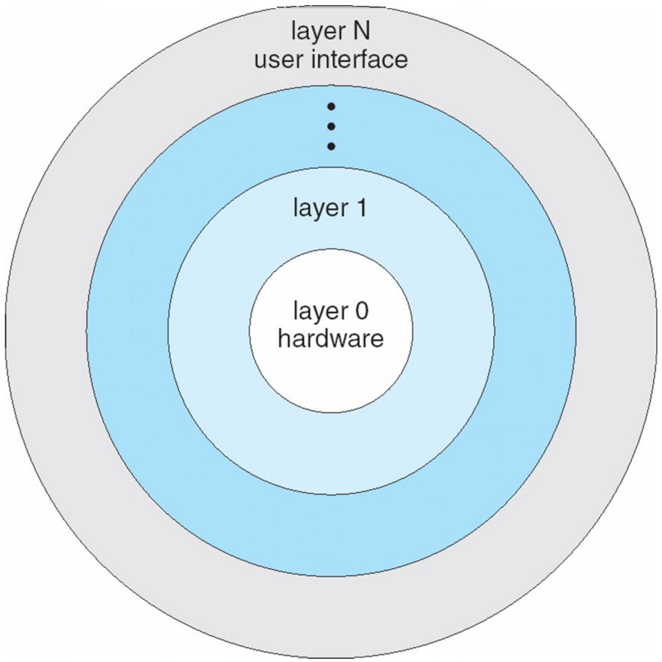


Traditional UNIX System Structure



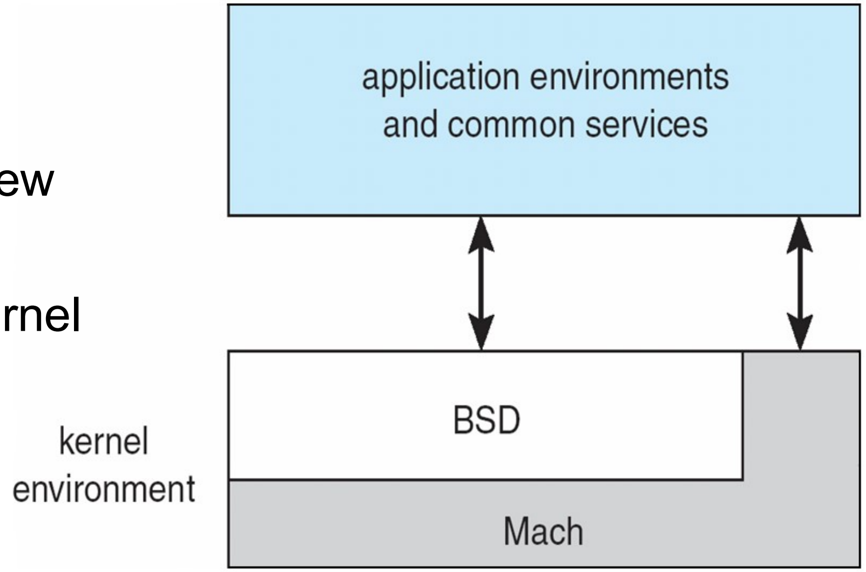
Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers
- The main advantage of the layered approach is simplicity of construction and debugging



Microkernel System Structure

- Moves as much from the kernel into "*user*" space
- Communication takes place between user modules using message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication

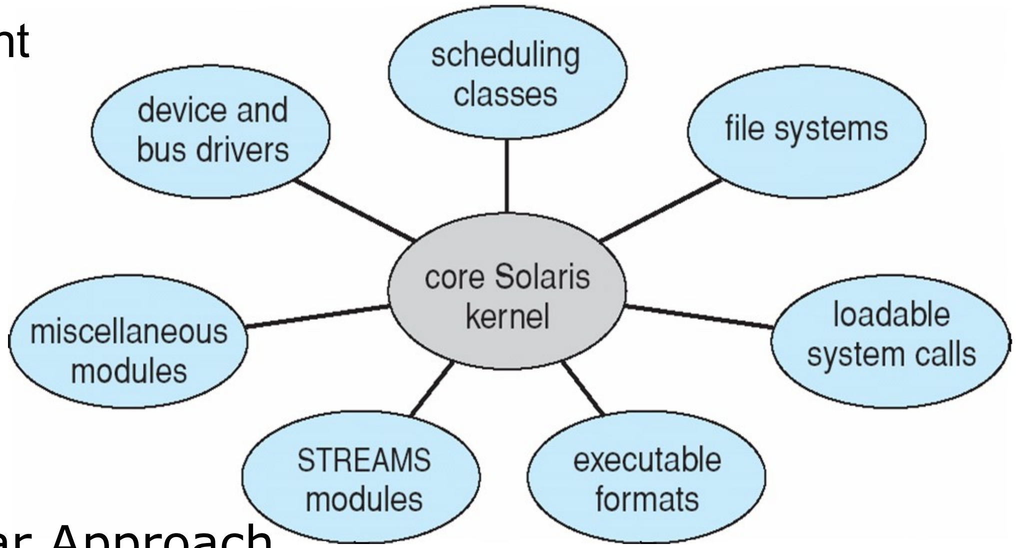


Mac OS X Structure

Modules

- Most modern operating systems implement kernel modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel

- Overall, similar to layers but with more flexibility
- Like microkernel but more efficient

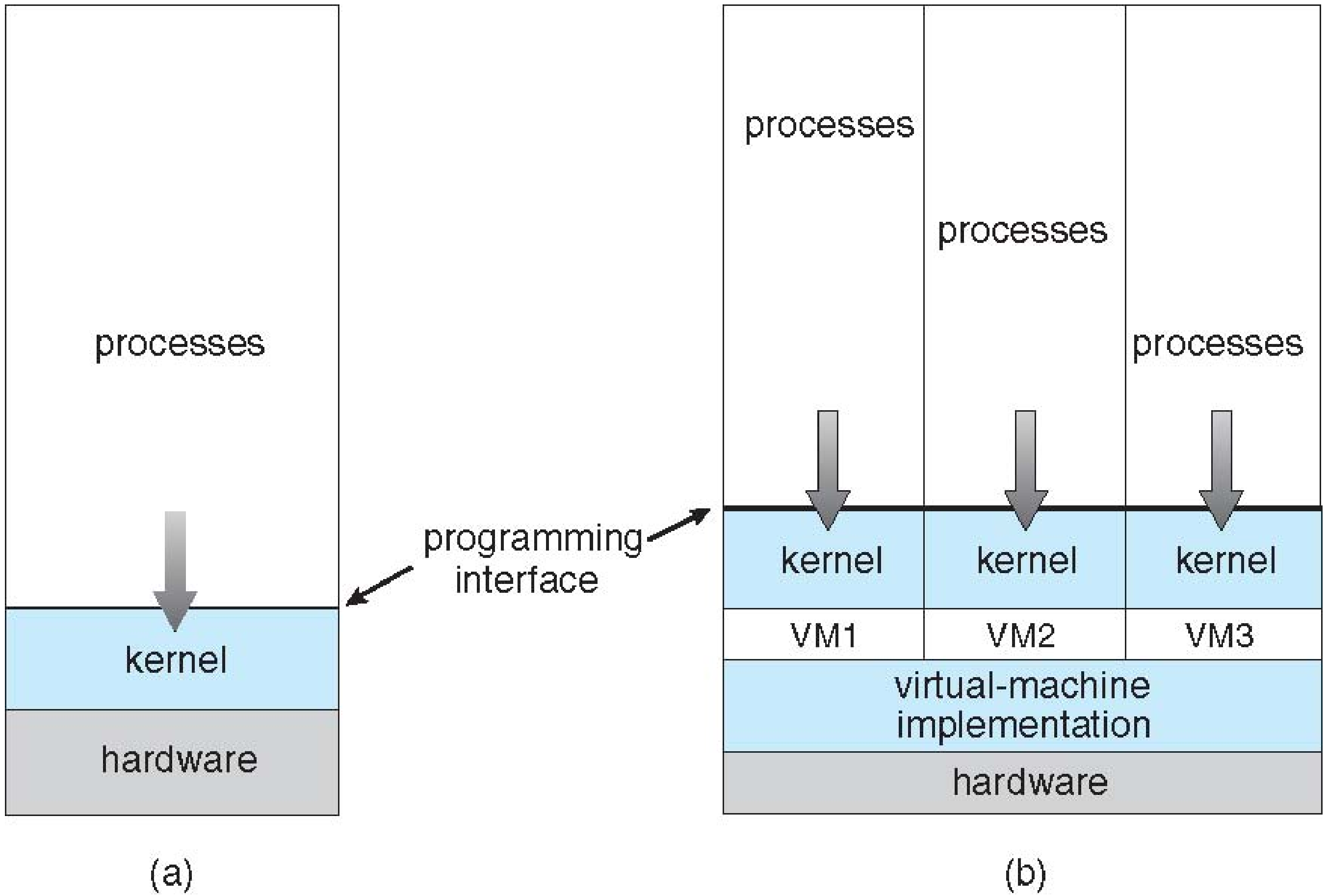


Solaris Modular Approach

Virtual Machines

- A **virtual machine** takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface *identical* to the underlying bare hardware.
- The operating system **host** creates the illusion that a process has its own processor and (virtual) memory.
- Each **guest** is provided with a (virtual) copy of underlying computer.

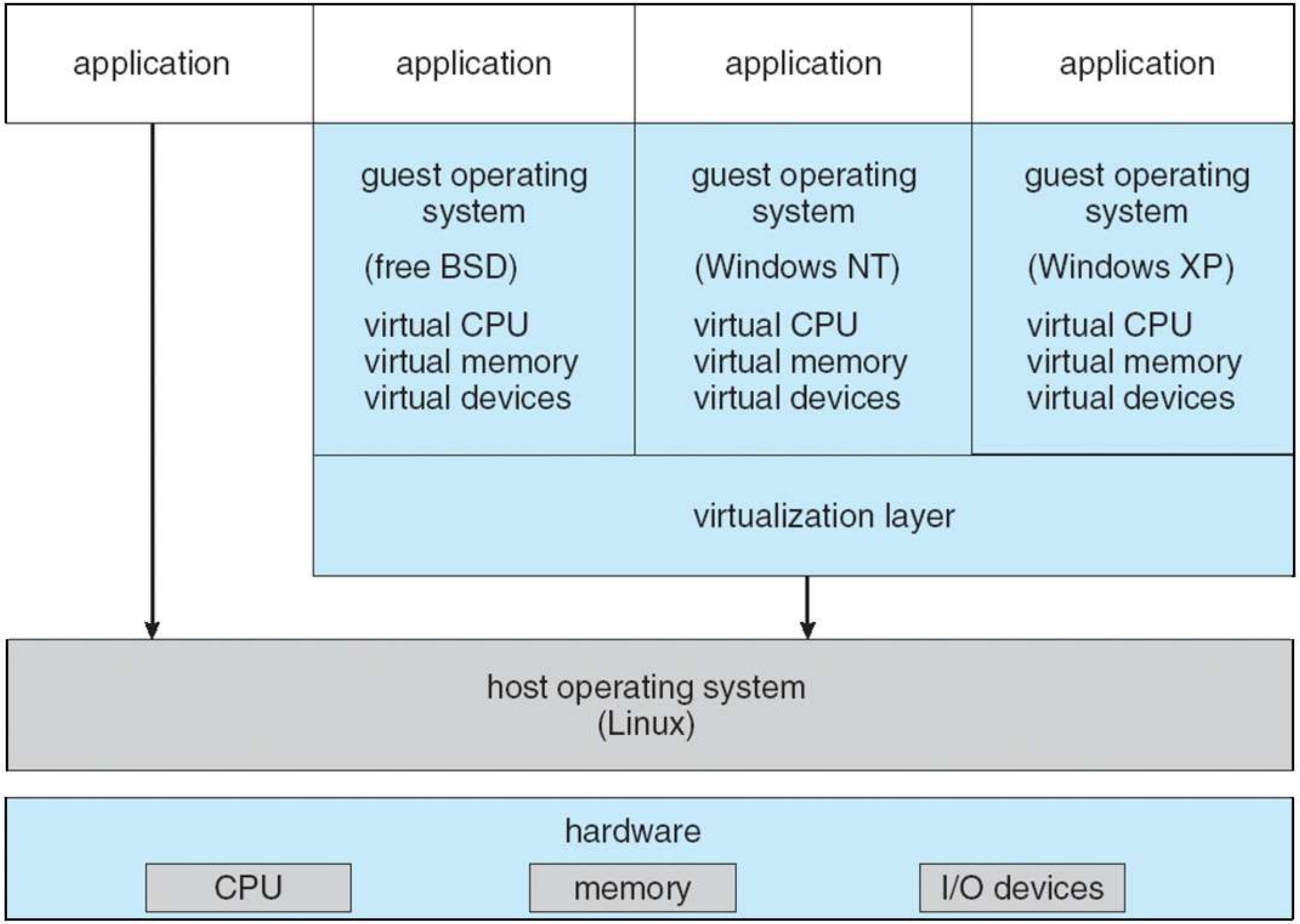
Virtual Machines (Cont.)



(a) Nonvirtual machine

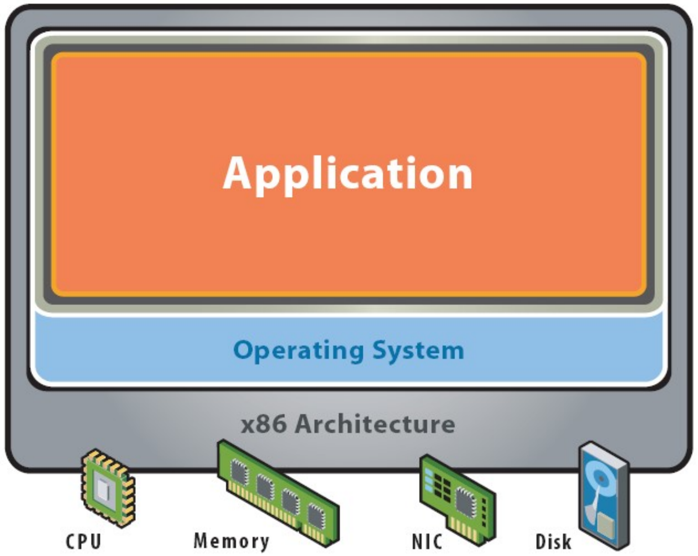
(b) Virtual machine

vmware Architecture



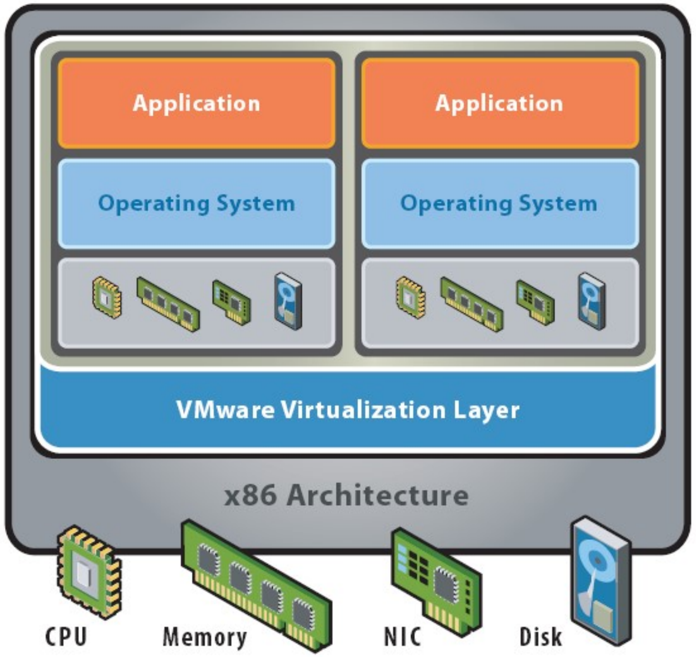
Benefits of Virtualization

Before Virtualization



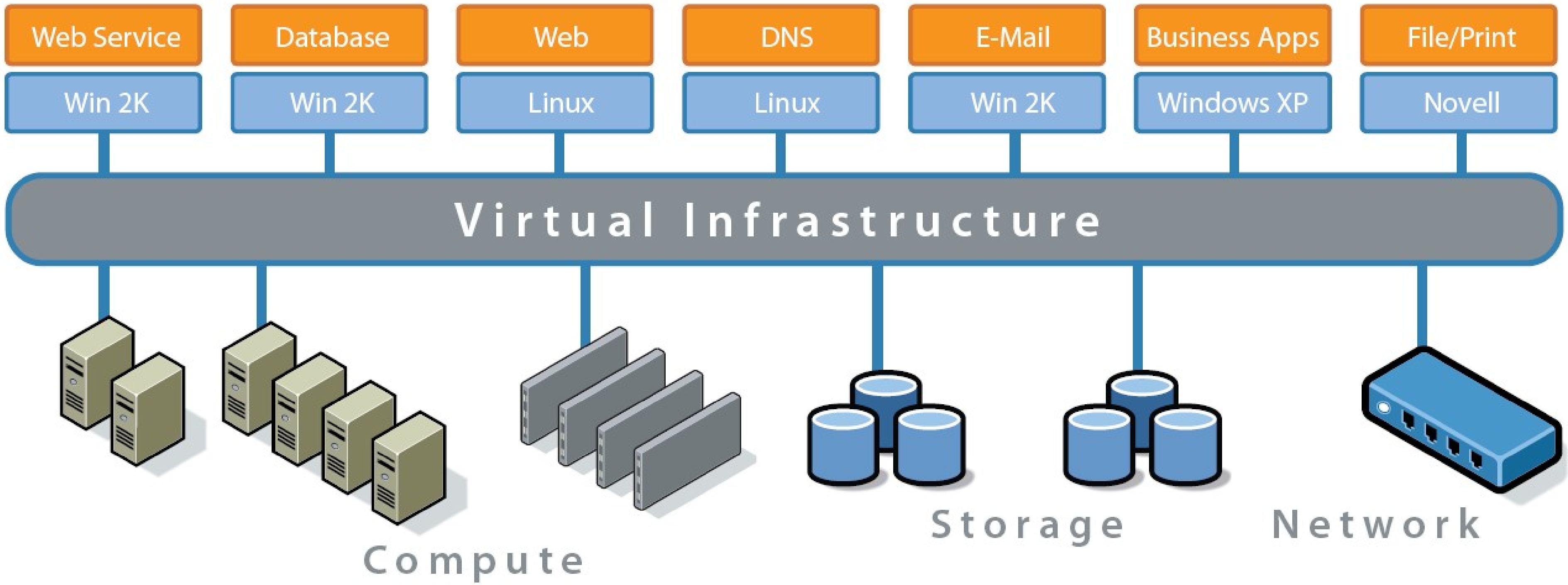
- Single OS image per machine
- Software and hardware tightly coupled
- Underutilized resources
- Inflexible and costly infrastructure

After Virtualization

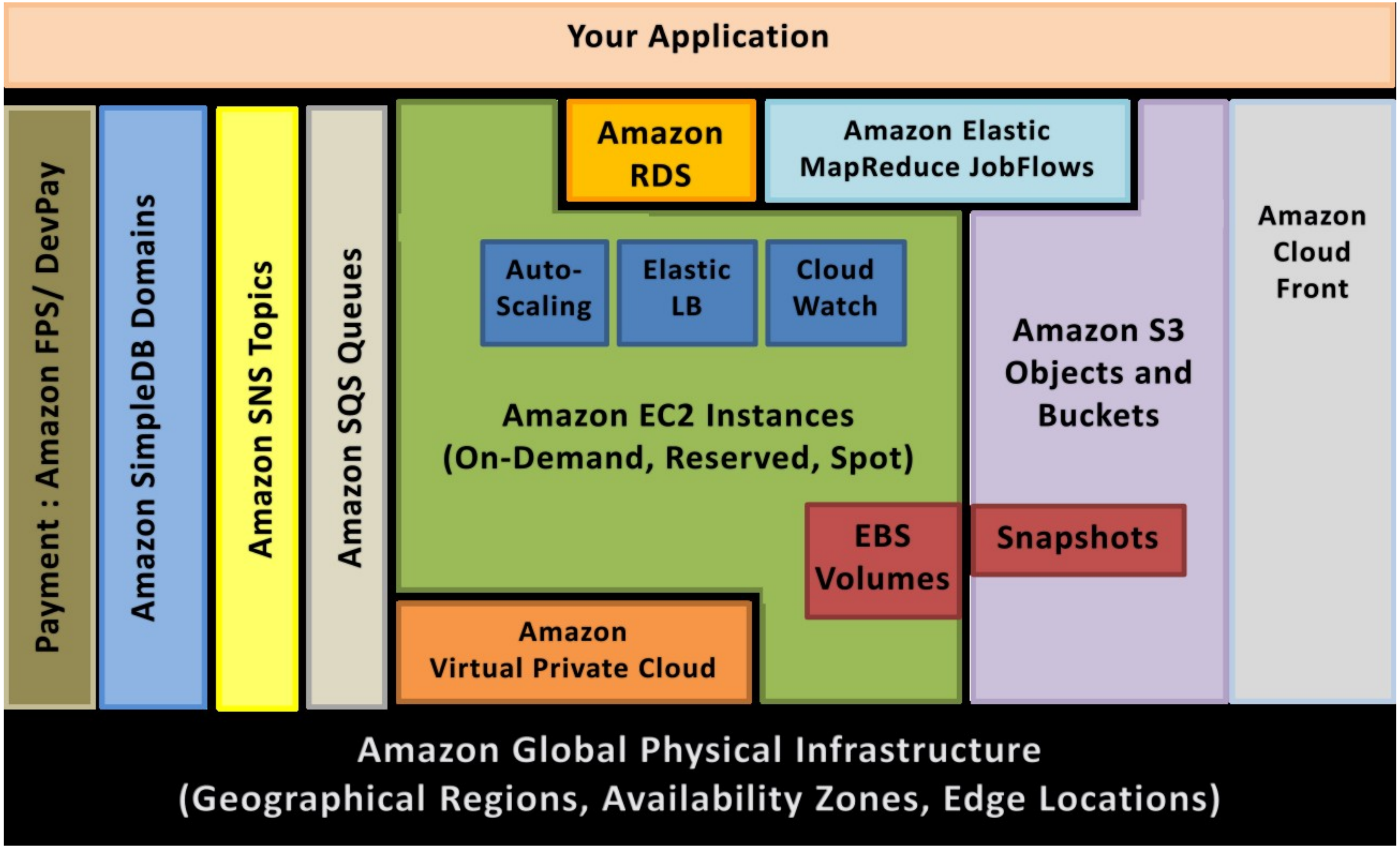


- Multiple OSs on a single machine
- Hardware-independence of operating system and applications
- Better utilization of resources
- Encapsulating OS and application into virtual machines

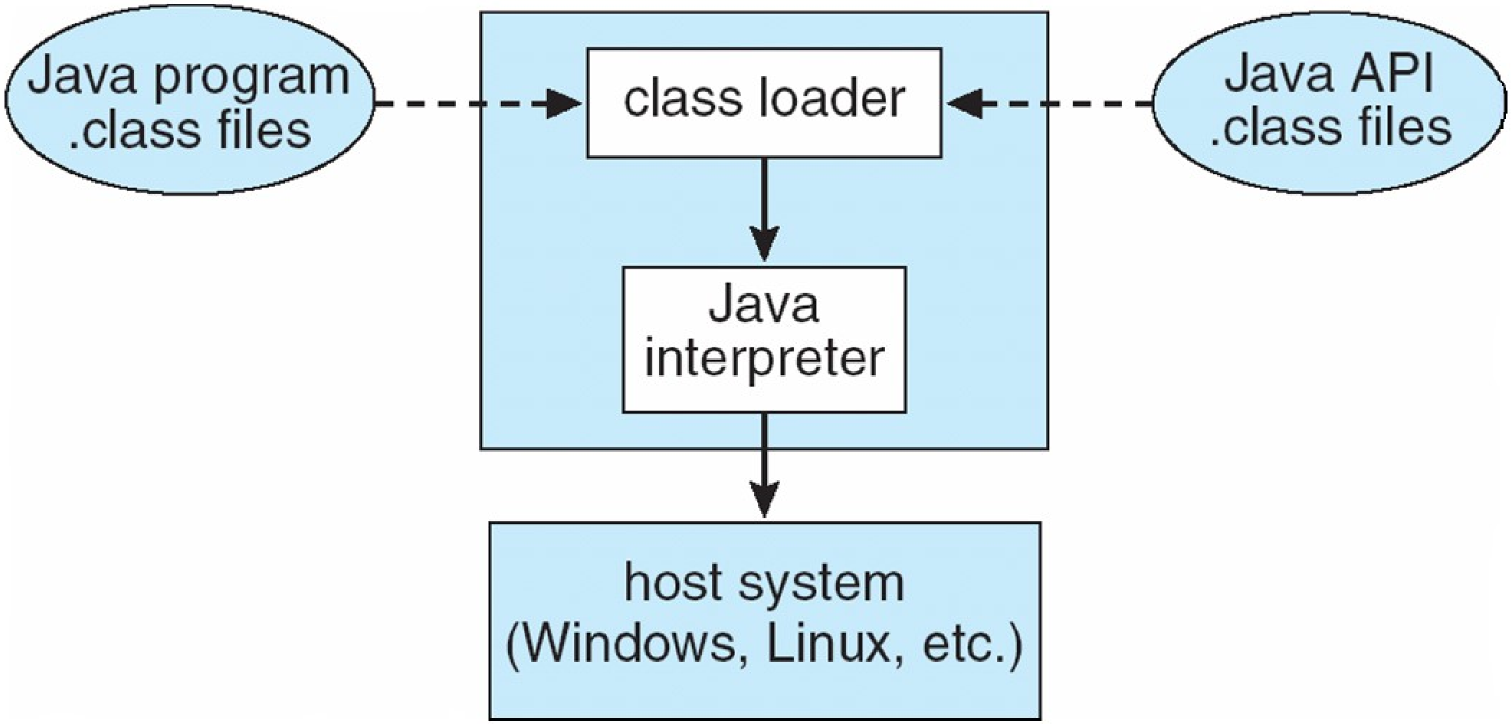
Virtual Infrastructure for Data Center



Amazon Elastic Compute Cloud (EC2)



The Java Virtual Machine



Homework

- Reading
  - Chapter 2: Operating-System Structures