1.10 Application Bases

IBM PC immediately spawned a huge software industry
- Independent software vendors (ISVs) market software packages to run under MS-DOS operating system.
- Operating system must present environment conducive to rapid and easy application development
  - Otherwise unlikely to be adopted widely

- Application base
  - Combination of hardware and operating system used to develop applications
  - Developers and users unwilling to abandon established application base
    - Increased financial cost and time spent relearning
    - Think about the issues of “backward compatibility”

1.11 Operating System Environments

- We focus on general purpose OS
  - Same concepts can be applied to other high-end or embedded systems
- Operating systems intended for high-end environments
  - Special design requirements and hardware support needs
    - Large main memory
    - Special-purpose hardware
    - Large numbers of processes
- Embedded systems
  - Characterized by small set of specialized resources
  - Provide functionality to devices such as cell phones and PDAs
  - Efficient resource management key to building successful operating system

1.11 Operating System Environments

- Real-time systems
  - Require that tasks be performed within particular (often short) time frame
    - Autopilot feature of an aircraft must constantly adjust speed, altitude and direction
    - Air traffic control, medical patient monitoring, power plant control
    - Such actions cannot wait indefinitely—and sometimes cannot wait at all
- Virtual machines (VMs)
  - Software abstraction of a computer
  - Often executes on top of native operating system
- Virtual machine operating system
  - Manages resources provided by virtual machine
- Applications of virtual machines
  - Allow multiple instances of an operating system to execute concurrently
  - Emulation
    - Software or hardware mimics functionality of hardware or software not present in system
  - Promote portability
1.11 Operating System Environments

Figure 1.2 Schematic of a virtual machine.

1.12 Operating System Components and Goals

- Computer systems have evolved:
  - Early systems contained no operating system,
  - Later gained multiprogramming and timesharing machines,
  - Personal computers and finally truly distributed systems,
  - Filled new roles as demand changed and grew.

1.12.1 Core Operating System Components

- User interaction with operating system:
  - Often, through special application called a shell that is usually text-based in UNIX or a GUI as in Windows.
  - Kernel:
    - Software that contains core components of operating system.

- Typical operating system components include:
  - Processor scheduler
  - Memory manager
  - I/O manager
  - Interprocess communication (IPC) manager
  - File system manager

1.12.2 Operating System Goals

- Users expect certain properties of operating systems:
  - Efficiency
  - Robustness
  - Scalability
  - Extensibility
  - Portability
  - Security
  - Protection
  - Interactivity
  - Usability

1.13 Operating System Architectures

- Today’s operating systems tend to be complex:
  - Provide many services
  - Support variety of hardware and software
  - Operating system architectures help manage this complexity:
    - Organize operating system components
    - Specify privilege with which each component executes.
1.13.1 Monolithic Architecture

- Monolithic operating system
  - Every component contained in kernel
  - Any component can directly communicate with any other
  - Tend to be highly efficient
  - A single-level extension of bare hardware
- Disadvantage
  - Difficulty determining source of subtle errors
  - Difficult to maintain (need recompilation and reinstallation of the complete system)
  - Eg: MS-DOS

1.13.2 Layered Architecture

- Layered approach to operating systems
  - Tries to improve on monolithic kernel designs
  - Groups components that perform similar functions into layers
  - Each layer communicates only with layers immediately above and below it
  - Processes’ requests might pass through many layers before completion
  - System throughput can be less than monolithic kernels
  - Additional methods must be invoked to pass data and control

Structural Organization

- Layered
  - OS functions are split into two main layers
  - OS Kernel
    - The most fundamental functions necessary to manage the system resources
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions
    - A large number of functions for one level

Structural Organization (cont.)

- System library (system programs):
  - Provide a greatly expanded suite of functionalities to meet application demands
  - Shell, compiler, etc.
  - Eg: UNIX

Structural Organization of OSs
1.13.2 Layered Architecture

System Calls

- The interface between a process and the OS
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)
- Three general methods are used to pass parameters between a running program and the operating system.
  - Parameters in registers.
  - Parameters in a table in memory and the table address in a register \( \rightarrow \) LINUX, unlimited
  - Stack (push, pop) \( \rightarrow \) unlimited
- Organization of Kernel (BSD)
  - Machine independent 80% (100% C)
  - Machine dependent 20% (90% C, 10% Assembly)
  - Only 2% of kernel is in assembly

System Calls

- The interface between a process and the OS
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)
- Three general methods are used to pass parameters between a running program and the operating system.
  - Parameters in registers.
  - Parameters in a table in memory and the table address in a register \( \rightarrow \) LINUX, unlimited
  - Stack (push, pop) \( \rightarrow \) unlimited
- Organization of Kernel (BSD)
  - Machine independent 80% (100% C)
  - Machine dependent 20% (90% C, 10% Assembly)
  - Only 2% of kernel is in assembly

MS-DOS System Structure

- MS-DOS (Microsoft Disk Operating System)
  - Written to provide the most functionality in the least space (limited hardware capability)
  - Not divided into modules (monolithic)
  - Its interfaces and levels of functionality are not well separated (not modular)

UNIX System Structure

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application program</td>
<td></td>
</tr>
<tr>
<td>Resident system program</td>
<td></td>
</tr>
<tr>
<td>MS-DOS device drivers</td>
<td></td>
</tr>
<tr>
<td>ROM BIOS device drivers</td>
<td></td>
</tr>
</tbody>
</table>
### Layered Approach

- Modularization of a system
- 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.
  - Similar to OSI 7 layer for networking

### OS/2 Layer Structure

#### Problems
- Efficiency
  - (ex) I/O operation in user program → I/O layer for system call → CPU scheduling layer → Hardware
  - System call takes longer than a nonlayered system
- Windows NT (layered)
  - Lower performance to Windows 95
  - NT 4.0 → removing layers and integrating them more closely

### Microkernel System Structure

#### As the UNIX OS expanded, the kernel became larger and difficult to manage
- In the mid 1980’s, researchers at CMU developed an OS called Mach that modularized the kernel using the microkernel approach
- Removes all nonessential components from the kernel and implementing them as system- and user-level programs
- 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

#### 1.13.3 Microkernel Architecture

- Microkernel operating system architecture
  - Provides only small number of services
  - Attempt to keep kernel small and scalable
  - High degree of modularity
    - Extensible, portable and scalable
    - Increased level of intermodule communication
    - Can degrade system performance

#### Examples
- Digital UNIX, MacOS X Server OS
- Windows NT
  - Hybrid structure: layered + microkernel
  - Designed to run various applications, Win32 (native Windows applications), OS/2, and POSIX (Portable Operating System Interface)
  - Provides a server that runs in user space for each application type
  - Kernel coordinates the message passing between client applications and application servers
1.13.4 Networked and Distributed Operating Systems

- Network operating system
  - Runs on one computer
  - Allows its processes to access resources on remote computers

- Distributed operating system
  - The most significant recent trend in OS
  - Single operating system
  - Manages resources on more than one computer system
  - Issues: consistent timeline (logical clocks, time stamps)
  - MIT’s Chord OS and Amoeba OS by Vrije Univ.
  - Goals include:
    - Transparent performance
    - Scalability
    - Fault tolerance
    - Consistency

1.13.3 Microkernel Architecture

Figure 1.5 Microkernel operating system architecture.