Introduction to Computer Science Lesson 6

BSc in Computer Science
University of New York, Tirana

Assoc. Prof. Marenglen Biba

Operating Systems

- Today
  - The History of Operating Systems
  - Operating System Architecture
  - Coordinating the Machine’s Activities

- Next lesson
  - Handling Competition Among Processes
  - Security

What is an Operating System?

- An operating system is the software that controls the overall operation of a computer.

- It provides the means by which a user can store and retrieve files, provides the interface by which a user can request the execution of programs, and provides the environment necessary to execute the programs requested.

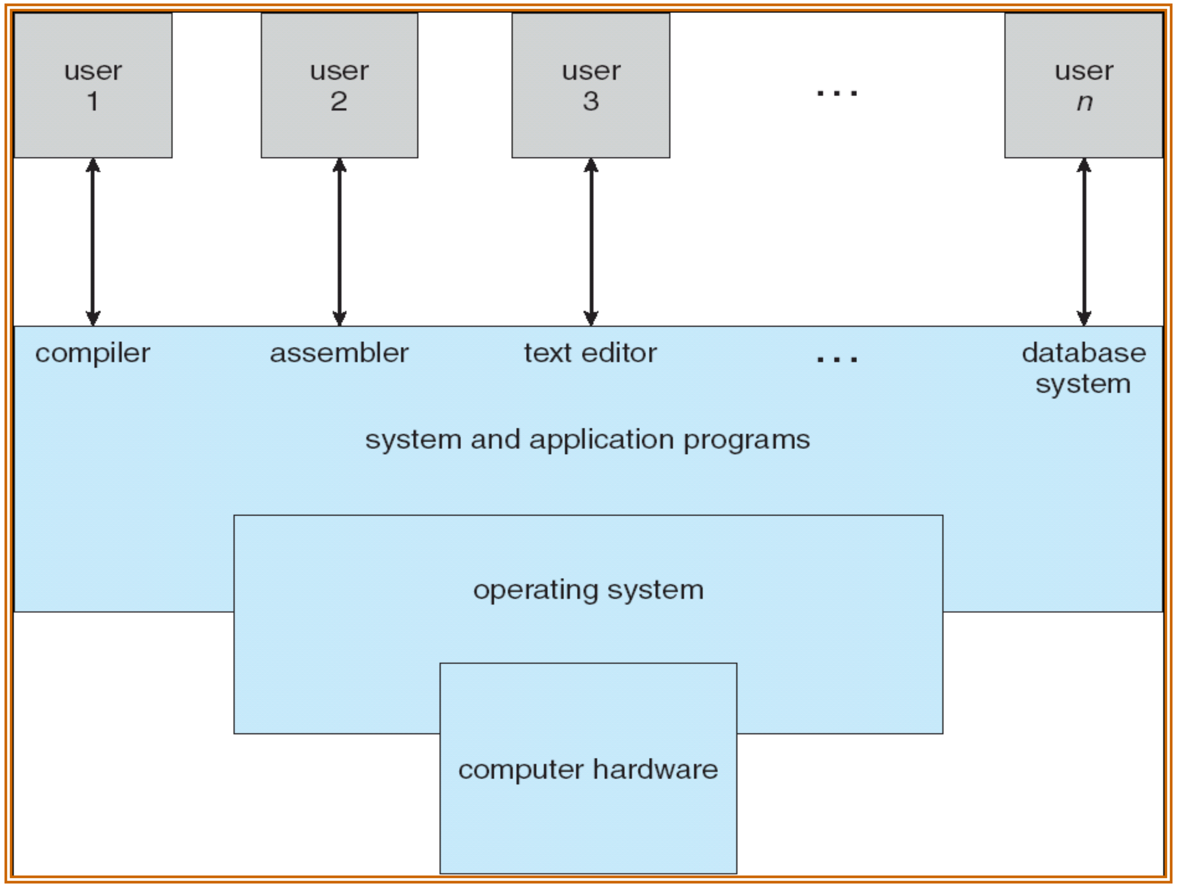
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.

Functions of Operating Systems

- Oversee operation of computer
- Store and retrieve files
- Schedule programs for execution
- Coordinate the execution of programs

Where is an OS positioned?



Example of Operating Systems

- Perhaps the best known example of an operating system is **Windows**, which is provided in numerous versions by Microsoft and widely used in the PC arena.
- Another well-established example is **UNIX**, which is a popular choice for larger computer systems as well as PCs.
- In fact, UNIX is the core of Mac OS, which is the operating system provided by Apple for its range of Mac machines.
- Still another example found on both large and small machines is **Linux**, which was originally developed non commercially by computer enthusiasts and is now available through many commercial sources, including IBM.

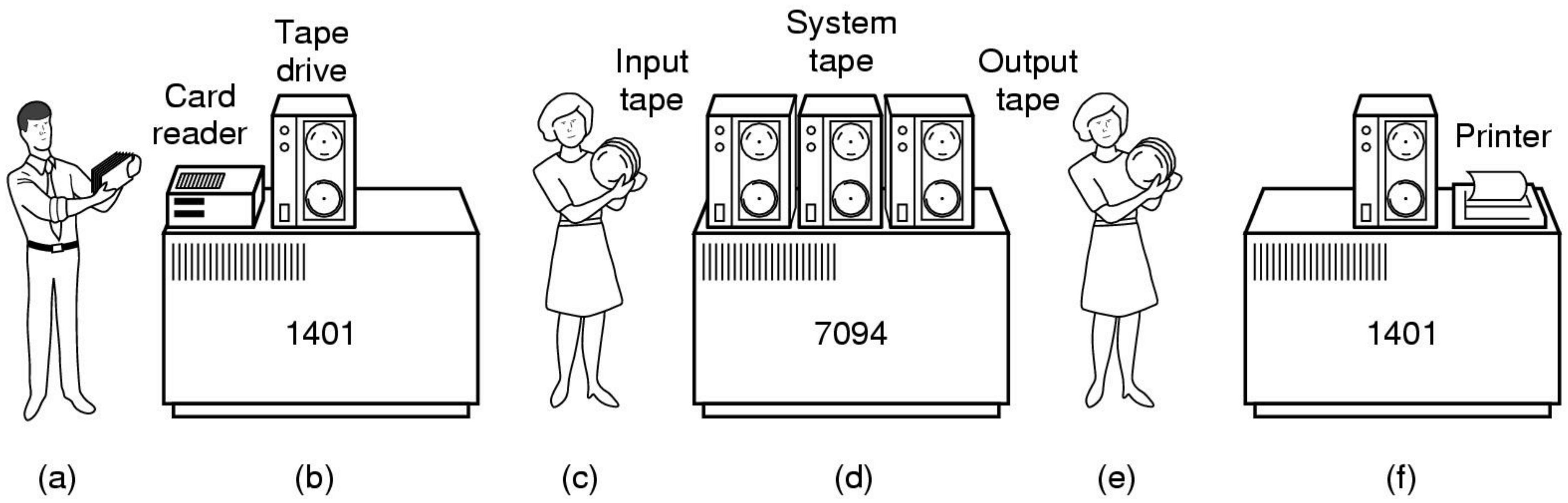
History of Operating Systems: machines

- First generation 1945 - 1955
  - vacuum tubes, plug boards
- Second generation 1955 - 1965
  - transistors, batch systems
- Third generation 1965 – 1980
  - ICs and multiprogramming
- Fourth generation 1980 – present
  - personal computers

First generation 1945 - 1955

- The computers of the 1940s and 1950s were not very flexible or efficient. Machines occupied entire rooms.
- Program execution required significant preparation of equipment in terms of mounting magnetic tapes, placing punched cards in card readers, setting switches, and so on.
- The execution of each program, called a **job**, was handled as an isolated activity- the machine was prepared for executing the program, the program was executed, and then all the tapes, punched cards, etc. had to be retrieved before the next program preparation could begin.
- When several users needed to share a machine, sign-up sheets were provided so that users could reserve the machine for blocks of time.
- During the time period allocated to a user, the machine was totally under that user's control – no one else could work.

History of Operating Systems

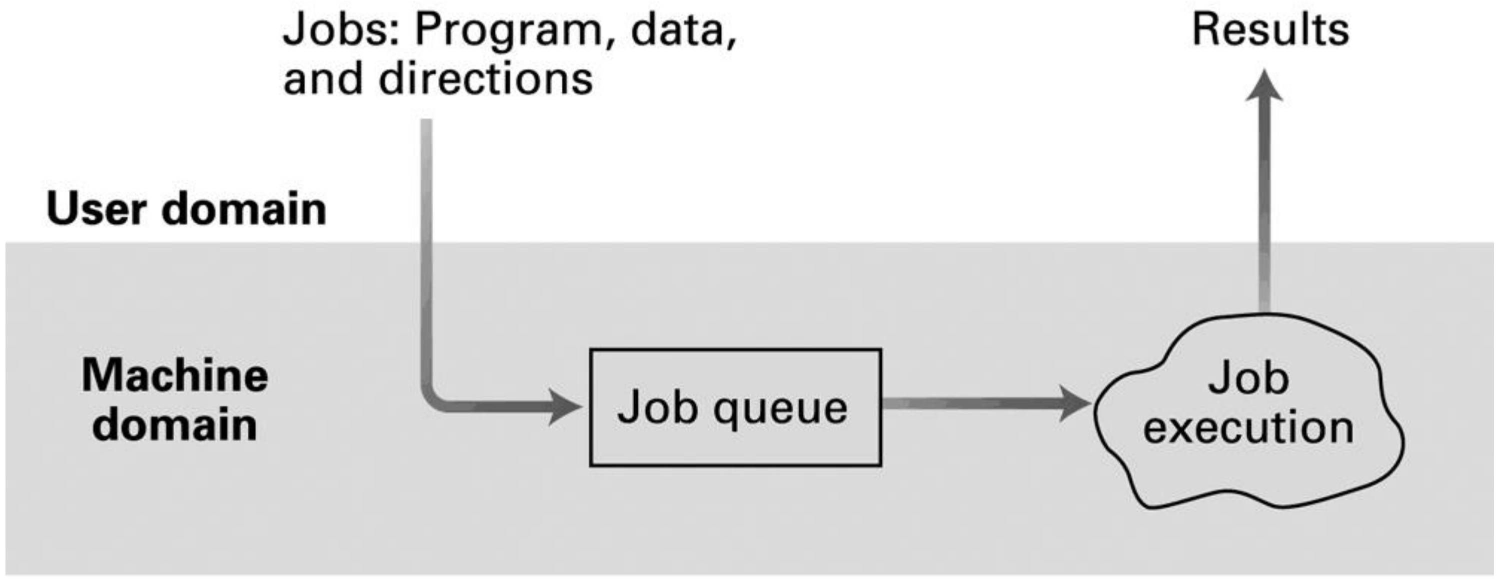


Early batch system
- bring cards to 1401
- read cards to tape
- put tape on 7094 which does computing
- put tape on 1401 which prints output

Batch Processing

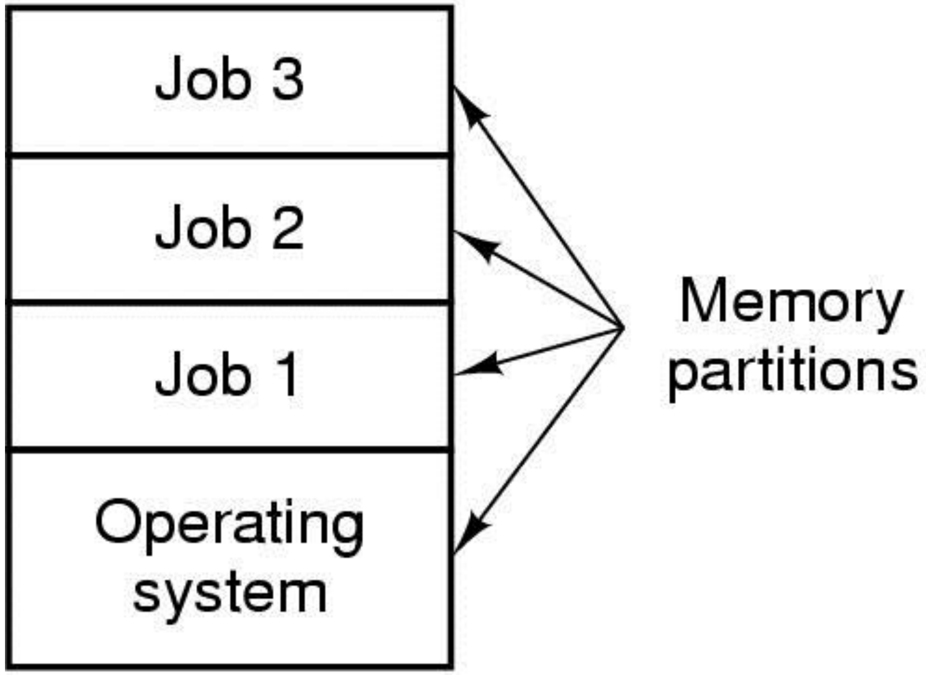
- One early development was the separation of users and equipment, which eliminated the physical transition of people in and out of the computer room.
- For this purpose a computer operator was hired to operate the machine. Anyone wanting a program run was required to submit it, along with any required data and special directions about the program's requirements, to the operator and return later for the results.
- The operator, in turn, loaded these materials into the machine's mass storage where a program called the operating system could read and execute them one at a time.
- This was the beginning of batch processing - the execution of jobs by collecting them in a single batch, then executing them without further interaction with the user.

Figure 3.1 **Batch processing**



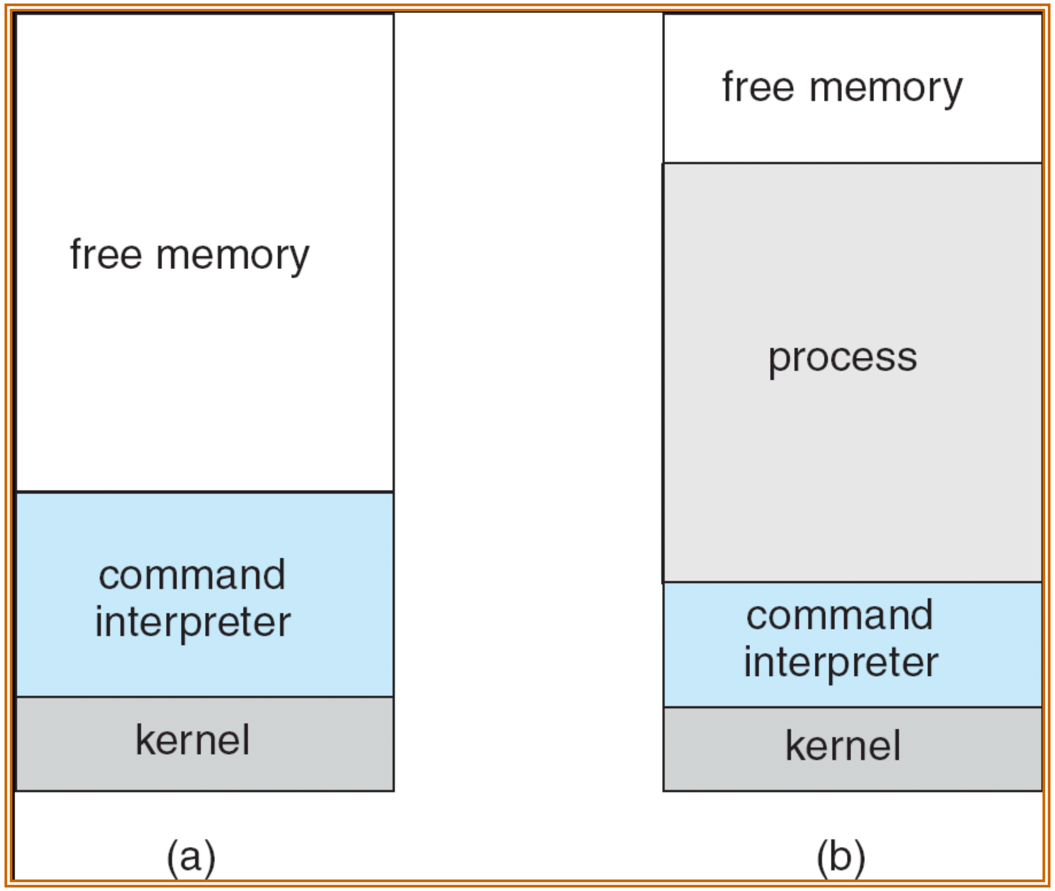
- In batch processing systems, the jobs residing in mass storage wait for execution in a job queue.
- A queue is a storage organization in which objects (in this case, jobs) are ordered in first-in, first-out (abbreviated FIFO) fashion.

1965-1980 - Multiprogramming



- Multiprogramming system
  – three jobs in memory

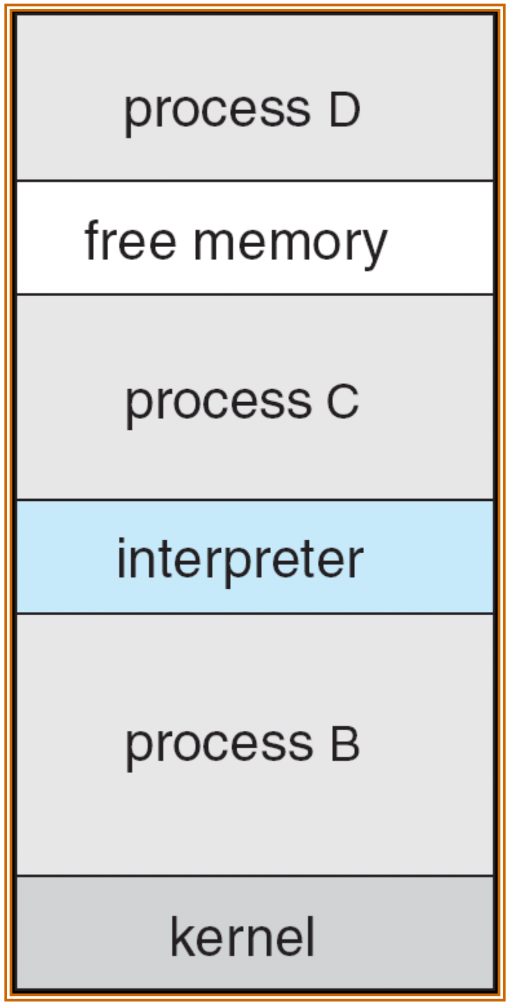
MS-DOS execution: single-tasking system



Does not create a new process for every program running but loads the program overwriting most of memory

(a) At system startup (b) running a program

FreeBSD Running Multiple Programs: multi-tasting system



Once the program starts, it can run in background and another program can be started again.

1980 – present

- Apple Macintosh
  - GUI: Graphical User Interface
- Microsoft Windows: 90s
  - Initially run over DOS
  - Not really a different OS
- Windows 95
  - Underlying DOS: only for booting and running old DOS programs.
  - Windows 98
  - Both W95 and Win98 retain large portions of 16-bit assembly language.
- Windows NT (New Technology)
  - Full 32-bit system
  - Would kill off DOS: Win NT 4.0
  - Win NT 4.0 was renamed to Windows 2000.

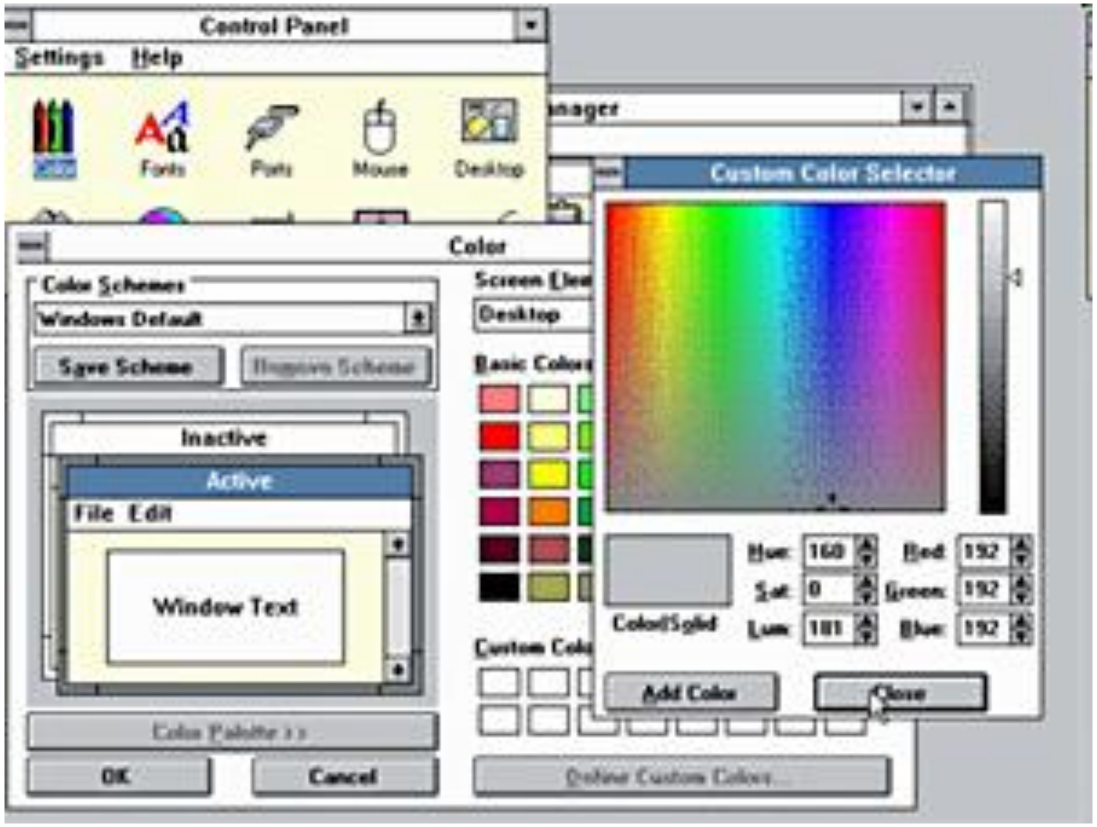
Fourth generation 1980 – present

- UNIX
  - Best for workstations, high-end computers, network servers
  - Popular on machines with high-performance RISC chips
  - Linux is also going strong on Pentium machines
- X Windows
  - Graphical User Interface for UNIX developed at M.I.T.
- Distributed Operating Systems
- Network Operating Systems

Windows

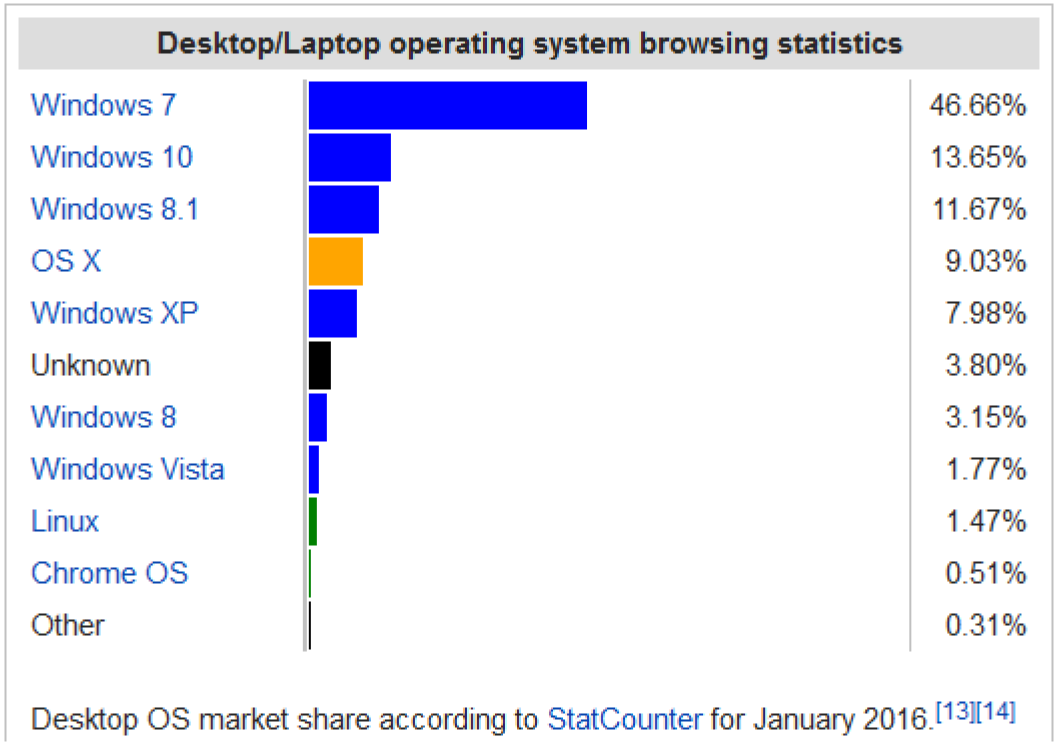
- Microsoft Windows is a series of software operating systems and graphical user interfaces produced by Microsoft.
- Microsoft first introduced an operating environment named Windows in November 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs).
- Microsoft Windows came to dominate the world's personal computer market, overtaking Mac OS, which had been introduced previously.
- The most recent client version of Windows is Windows 7; the most recent server version is Windows Server 2008 R2.

In the 90s: Windows 3.1



Windows 3.1 required a minimum of MS-DOS 3.1, a 386 processor with 2MB of RAM, and 6MB of hard disk space.

Market share: desktop OS

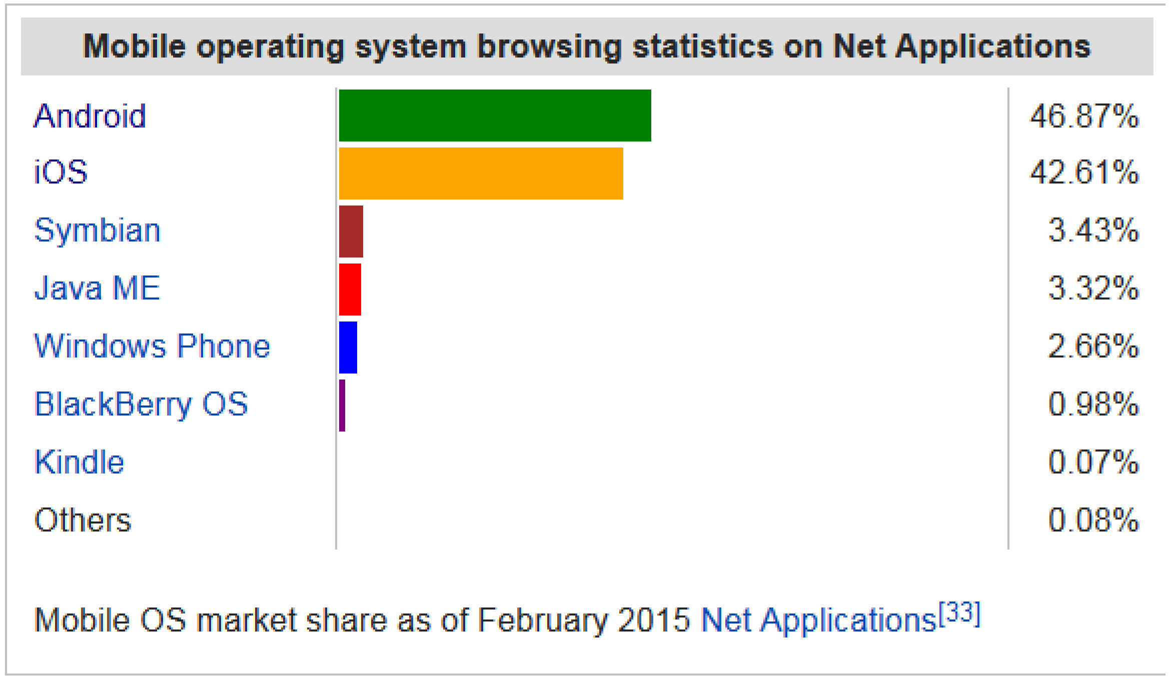


Desktop/Laptop operating system browsing statistics

Windows 7	46.66%
Windows 10	13.65%
Windows 8.1	11.67%
OS X	9.03%
Windows XP	7.98%
Unknown	3.80%
Windows 8	3.15%
Windows Vista	1.77%
Linux	1.47%
Chrome OS	0.51%
Other	0.31%

Desktop OS market share according to StatCounter for January 2016.[13][14]

Market share: mobile OS


Mobile operating system browsing statistics on Net Applications

Android	46.87%
iOS	42.61%
Symbian	3.43%
Java ME	3.32%
Windows Phone	2.66%
BlackBerry OS	0.98%
Kindle	0.07%
Others	0.08%

Mobile OS market share as of February 2015 Net Applications[33]

Linux

- For the computer enthusiast who wants to experiment with the internal components of an operating system, there is **Linux**.
- Linux is an operating system originally designed by Linus Torvalds while a student at the University of Helsinki.
- It is a nonproprietary product and available, along with its source code and documentation, without charge.
- Because it is freely available in source code form, it has become popular among computer hobbyists, students of operating systems, and programmers in general.
- Moreover, Linux is recognized as one of the most reliable operating systems available today. For this reason, several companies now package and market versions of Linux in an easily useable form, and these products are now challenging the long-established commercial operating systems on the market.
- You can learn more about **Linux** from the website at
  - http://www.linux.org.

UNIX

- Unix (officially trademarked as UNIX, sometimes also written as Unix with small caps) is a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, and Joe Ossanna.
- The Unix operating system was first developed in assembly language, but by 1973 had been almost entirely recoded in C, greatly facilitating its further development and porting to other hardware.
- As of 2007, the owner of the trademark is The Open Group, an industry standards consortium.
- Only systems fully compliant with and certified according to the Single UNIX Specification are qualified to use the trademark; others are called "Unix system-like" or "Unix-like".

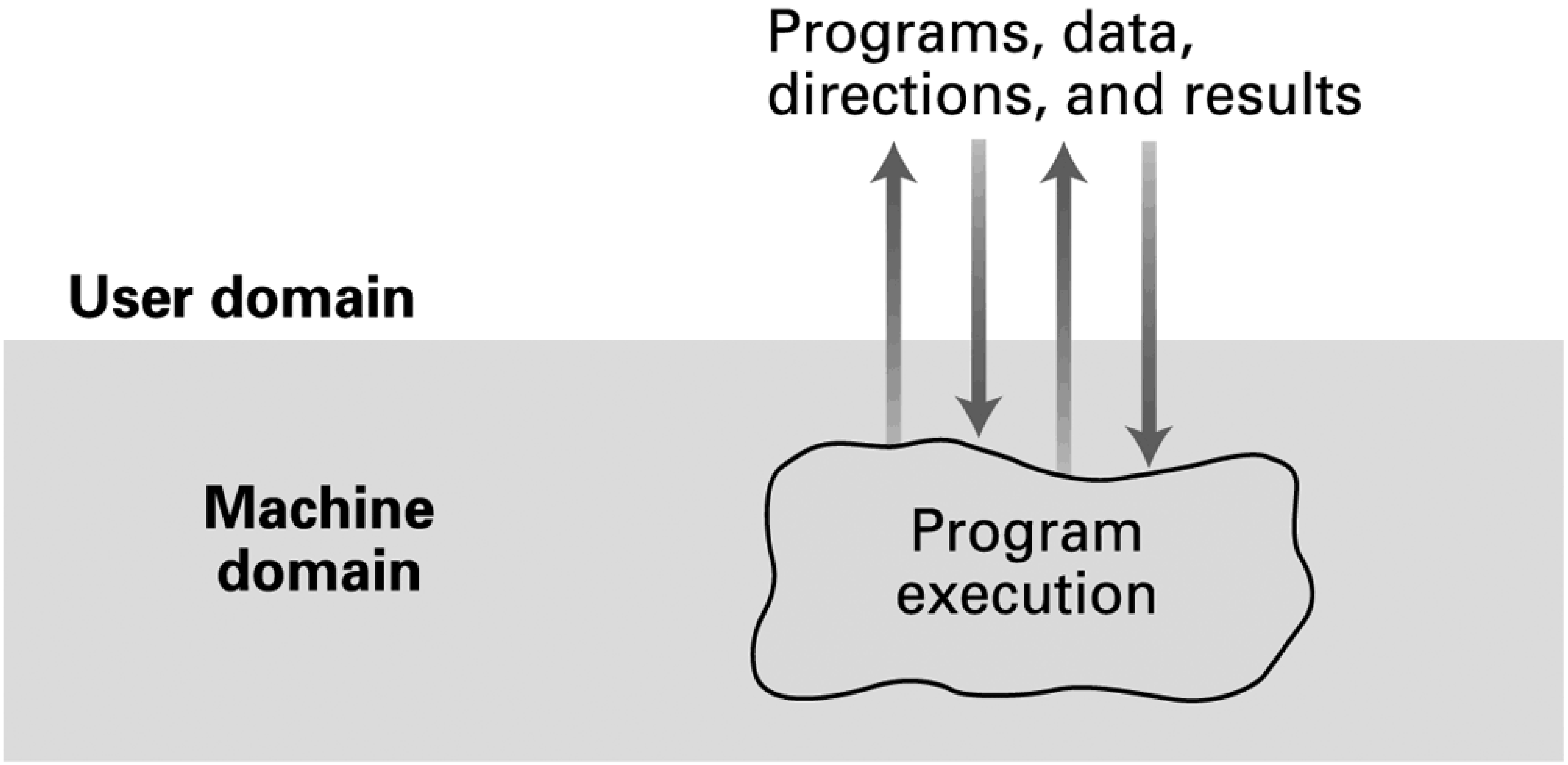
Evolution of Shared Computing

- Batch processing
- Interactive processing
  - Requires real-time processing
- Time-sharing/Multitasking
  - Implemented by Multiprogramming
- Multiprocessor machines

Drawbacks of batch processing

- A major drawback to using a computer operator as an intermediary between a computer and its users is that the users have **no interaction** with their jobs once they are submitted to the operator.
- This approach is acceptable for some applications, such as payroll processing, in which the data and all processing decisions are established in advance.
- However, it is not acceptable when the user must **interact** with a program during its execution.
- Examples include:
  - reservation systems in which reservations and cancellations must be reported as they occur;
  - word processing systems in which documents are developed in a dynamic write and rewrite manner;
  - computer games in which interaction with the machine is the central feature of the game.

Figure 3.2 **Interactive processing**



Real-time processing

- Paramount to successful interactive processing is that the actions of the computer be sufficiently fast to coordinate with the needs of the user rather than forcing the user to conform to the machine's timetable.
- Providing computer services in such a timely manner became known as **real-time processing**, and the actions performed were said to occur in **real-time**.
  - That is, to say that a computer performs a task in real time means that the computer performs the task quickly enough that it is able to keep up with activities in its external (real-world) environment.

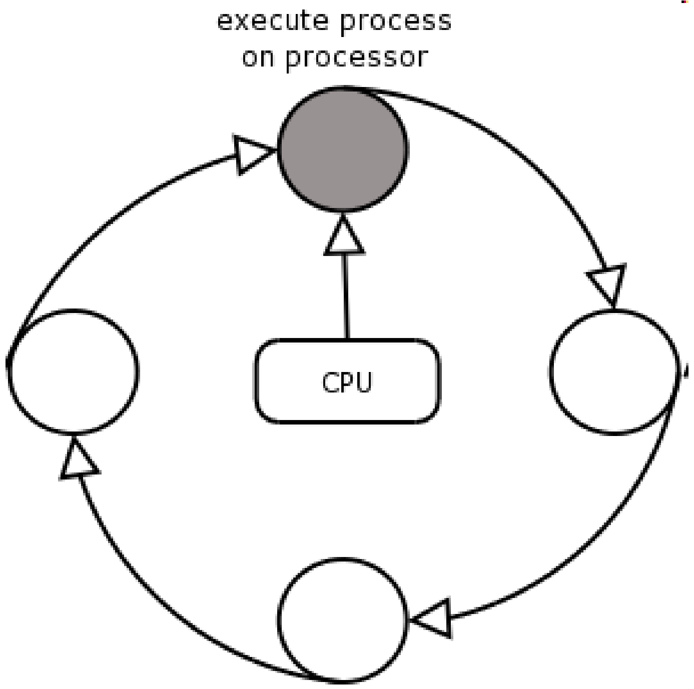
Real-time processing

- If interactive systems had been required to serve only one user at a time, real-time processing would have been no problem.
- But computers in the 1960s and 1970s were expensive, so each machine had to serve more than one user.
- In turn, it was common for several users, working at remote terminals, to seek interactive service from a machine at the same time, and real-time considerations presented obstacles.
- If the operating system insisted on executing only one job at a time, only one user would receive satisfactory real-time service.

Time-sharing

- A solution to real-time services was to design the operating system so that it rotated the various jobs in and out of execution by a strategy called **time-sharing:**
  - **which is the technique of dividing time into intervals and then restricting the execution of a job to only one interval at a time.**
- At the end of each interval, the current job is temporarily set aside and another is allowed to execute during the next interval.
- By rapidly shuffling the jobs back and forth in this manner, the illusion of several jobs executing simultaneously is created.
- Depending on the types of jobs being executed, early time-sharing systems were able to provide acceptable real-time processing to as many as 30 users simultaneously.
- Today, time-sharing is used in single-user as well as multiuser systems, although in the former it is usually called **multitasking**, in reference to the illusion of more than one task being performed simultaneously.

Time-sharing



Workstations

- With the development of multiuser, time-sharing operating systems, a typical computer installation was configured as a large central computer connected to numerous workstations.
- From these workstations, users could communicate directly with the computer from outside the computer room rather than submitting requests to a computer operator.
- Commonly used programs were stored in the machine's mass storage devices and operating systems were designed to execute these programs as requested from the workstations.
- In turn, the role of a computer operator as an intermediary between the users and the computer began to fade.

System Administrator

- Today, the existence of a computer operator has essentially disappeared, especially in the arena of personal computers where the computer user assumes all of the responsibilities of computer operation.
- Even most large computer installations run essentially unattended.
- Indeed, the job of computer operator has given way to that of a **system administrator** who manages the computer system. His duties are:
  - obtaining and overseeing the installation of new equipment and software,
  - enforcing local regulations such as the issuing of new accounts
  - establishing mass storage space limits for the various users
  - coordinating efforts to resolve problems that arise in the system

- Rather than operating the machines in a hands-on manner.

Load Balancing

- The evolution of operating systems continues.
- The development of multi-processor machines has led to operating systems that perform multitasking by assigning different tasks to different processors rather than by sharing the time of a single processor.
- These operating systems must wrestle with such problems as load **balancing**
  - dynamically allocating tasks to the various processors so that all processors are used efficiently
- as well as **scaling**
  - breaking tasks into a number of subtasks compatible with the number of processors available.

Distributed systems

- Moreover, the advent of computer networks in which numerous machines are connected over great distances has led to the necessity for software systems to coordinate the network's activities.

- Thus the field of networking is in many ways an extension of the subject of operating systems - the goal being to develop a single network-wide operating system rather than a network of individual operating systems.

Embedded systems

- Still another direction of research in operating systems focuses on devices that are dedicated to specific tasks such as medical devices, vehicle electronics, home appliances, cell phones, or other hand-held computers.
- The computer systems found in these devices are known as **embedded systems.**
- Embedded operating systems are often expected to conserve battery power, meet demanding real-time deadlines, or operate continuously with little or no human oversight.
- Successes in this endeavor are marked by systems such as:
  - VxWORKS, developed by Wind River Systems and used in the Mars Exploration Rovers named Spirit and Opportunity
  - Windows CE (also known as Pocket PC) developed by Microsoft
  - Palm OS developed by PalmSource, Inc., especially for use in hand-held devices.

What's in a Smartphone?

- As cell phones have become more powerful, it has become possible for them to offer services well beyond simply processing voice calls.
- A typical smartphone can now be used to text message, browse the Web, provide directions, view multimedia content — in short, it can be used to provide many of the same services as a traditional PC.
- As such, smartphones require full-fledged operating systems, not only to manage the limited resources of the smartphone hardware, but also to provide features that support the rapidly expanding collection of smartphone application software.
- The battle for dominance in the smartphone operating system market place promises to be fierce and will likely be settled on the basis of which system can provide the most imaginative features at the best price.
- Competitors in the smartphone operating system arena include Apple's iPhone OS, BlackBerry OS, Microsoft's Windows Phone and Google's Android.

Chapter 3: Operating Systems

- The History of Operating Systems
- Operating System Architecture
- Coordinating the Machine's Activities
- Handling Competition Among Processes
- Security

Types of Software

- Application software
- System software

Application software

- Application software consists of the programs for performing tasks particular to the machine's utilization.
- A machine used to maintain the inventory for a manufacturing company will contain different application software from that found on a machine used by an electrical engineer.
- Examples of application software include spreadsheets, database systems, desktop publishing systems, accounting systems, program development software, and games.

System Software

- In contrast to application software, system software performs those tasks that are common to computer systems in general.

- In a sense, the system software provides the infrastructure that the application software requires, in much the same manner as a nation's infrastructure (government, roads, utilities, financial institutions, etc.) provides the foundation on which its citizens rely for their individual lifestyles.

System Programs

- System programs provide a convenient environment for program development and execution. These can be divided into:
  - File manipulation
  - Status information
  - File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Application programs

- Most users’ view of the operation system is defined by system programs

System Programs

- File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- Status information
  - Some ask the system for info - date, time, amount of available memory, disk space, number of users
  - Others provide detailed performance, logging, and debugging information
  - Typically, these programs format and print the output to the terminal or other output devices
  - Some systems implement a registry - used to store and retrieve configuration information

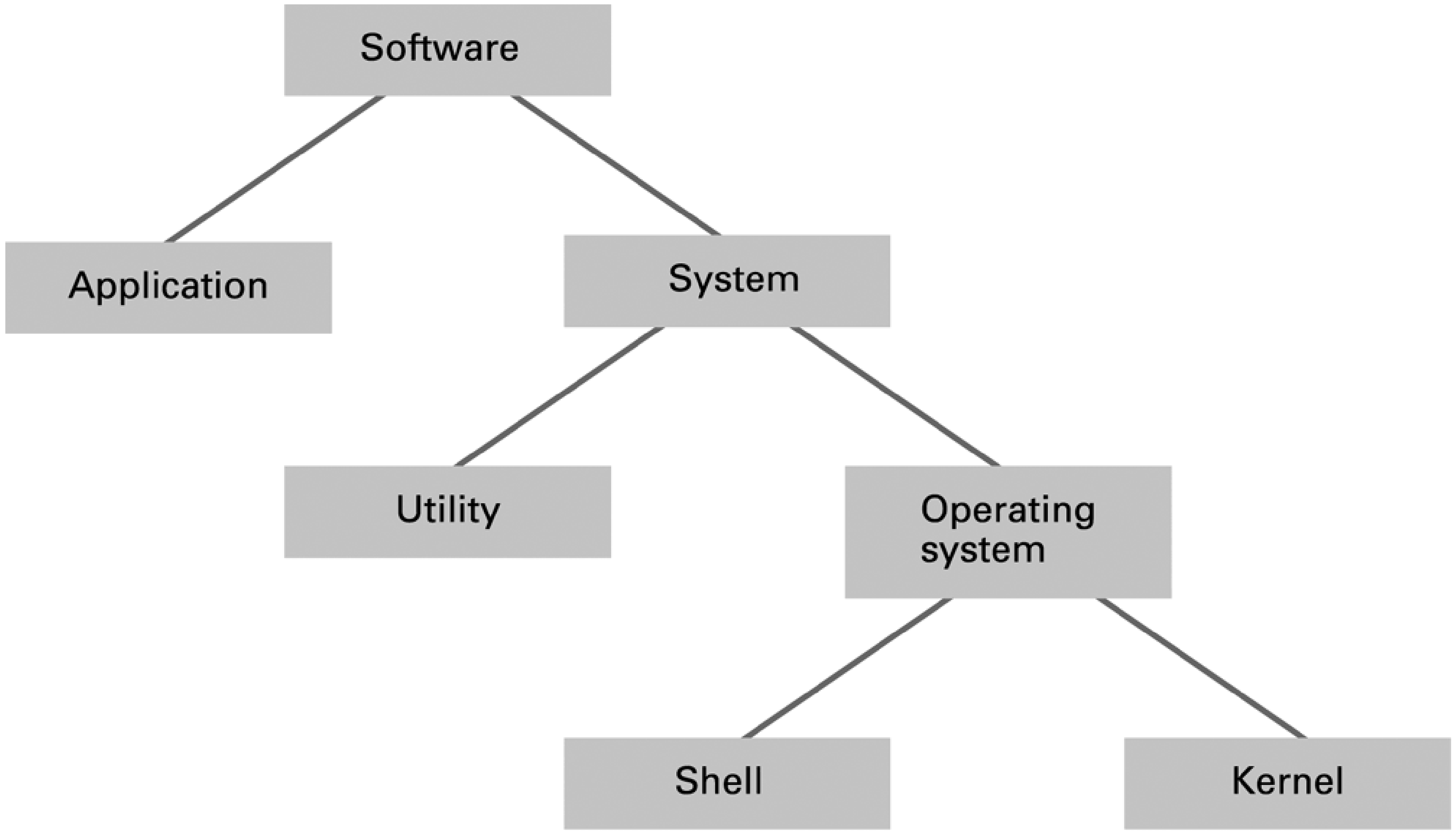
System Programs (cont'd)

- File modification
  - Text editors to create and modify files
  - Special commands to search contents of files or perform transformations of the text
- Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided
- Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

Utility software

- Within the class of system software are two categories: one is the operating system itself and the other consists of software units collectively known as utility software.
- The majority of an installation's utility software consists of programs for performing activities that are fundamental to computer installations but not included in the operating system.
- In a sense, utility software consists of software units that extend (or perhaps customize) the capabilities of the operating system.
- For example, the ability to format a magnetic disk or to copy a file from a magnetic disk to a CD is often not implemented within the operating system itself but instead is provided by means of a utility program.
- Other instances of utility software include software to compress and decompress data, software for playing multimedia presentations, and software for handling network communication.

Figure 3.3 **Software classification**



Utility software and competition

- Implementing certain activities as utility software, allows system software to be customized to the needs of a particular installation more easily than if they were included in the operating system.
- Unfortunately, the distinction between application software and utility software can be vague. From our point of view, the difference is whether the package is part of the computer's software infrastructure.
- Thus a new application may evolve to the status of a utility if it becomes a fundamental tool. When still a research project, software for communicating over the Internet was considered application software; today such tools are fundamental to most PC usage and would therefore be classified as utility software.
- The distinction between utility software and the operating system is equally vague. In particular, anti-trust lawsuits in the United States and Europe have been founded on questions regarding whether units such as browsers and media players are components of Microsoft's operating systems or utilities that Microsoft has included merely to squash competition.

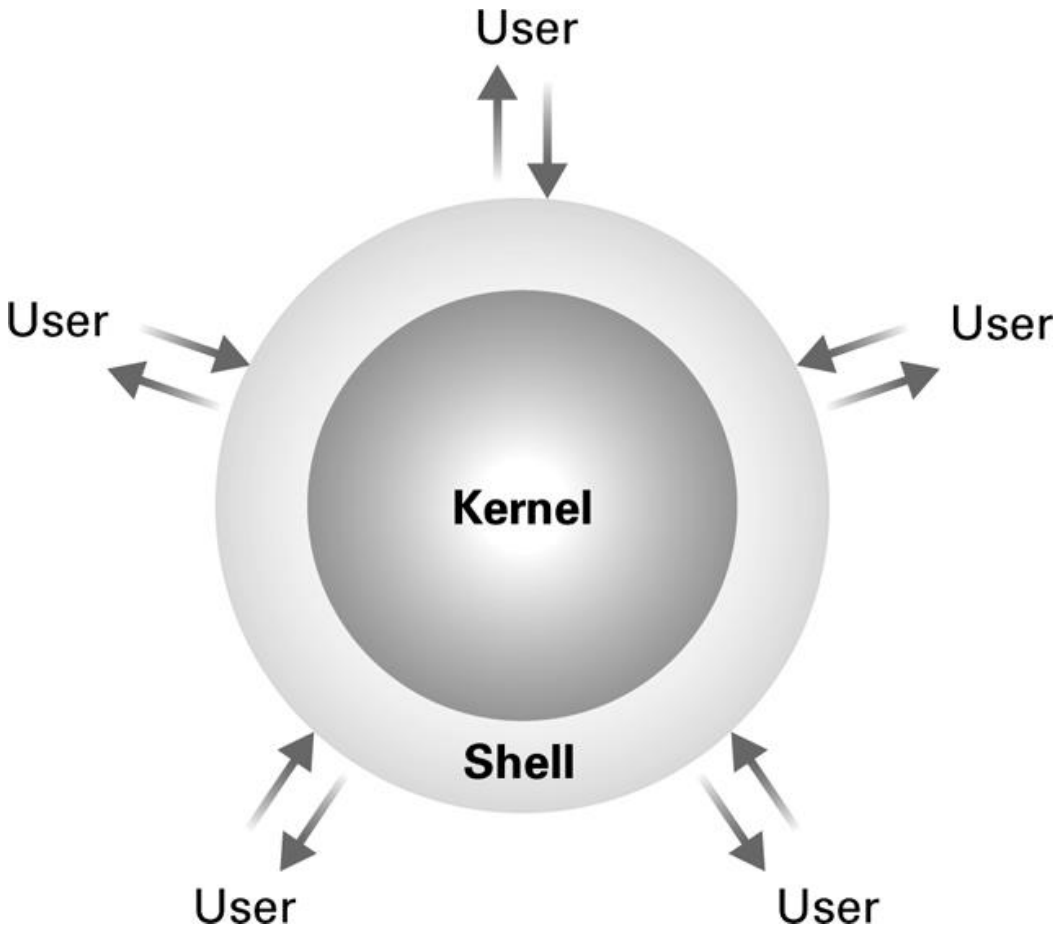
Operating System Components

- **Shell:** Communicates with users
  - Text based
  - Graphical user interface (GUI)
- **Kernel:** Performs basic required functions
  - File manager
  - Device drivers
  - Memory manager
  - Scheduler and dispatcher

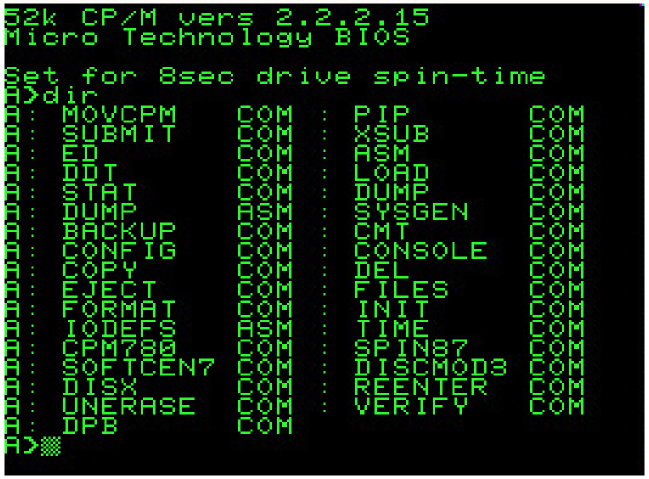
The Shell

- Although an operating system's shell plays an important role in establishing a machine's functionality, this shell is merely an **interface** between a user and the real heart of the operating system.
- This distinction between the shell and the internal parts of the operating system is emphasized by the fact that some operating systems allow a user to select among different shells to obtain the most compatible interface for that particular user.
- Users of the UNIX operating system, for example, can select among a variety of shells including the Bourne shell, the C shell, and the Korn shell.
- Moreover, early versions of Microsoft Windows were constructed by essentially replacing the text-based shell that was currently used with the operating system called MS-DOS with a GUI shell- the underlying operating system remained MS-DOS.

Figure 3.4 **The shell as an interface between users and the operating system**

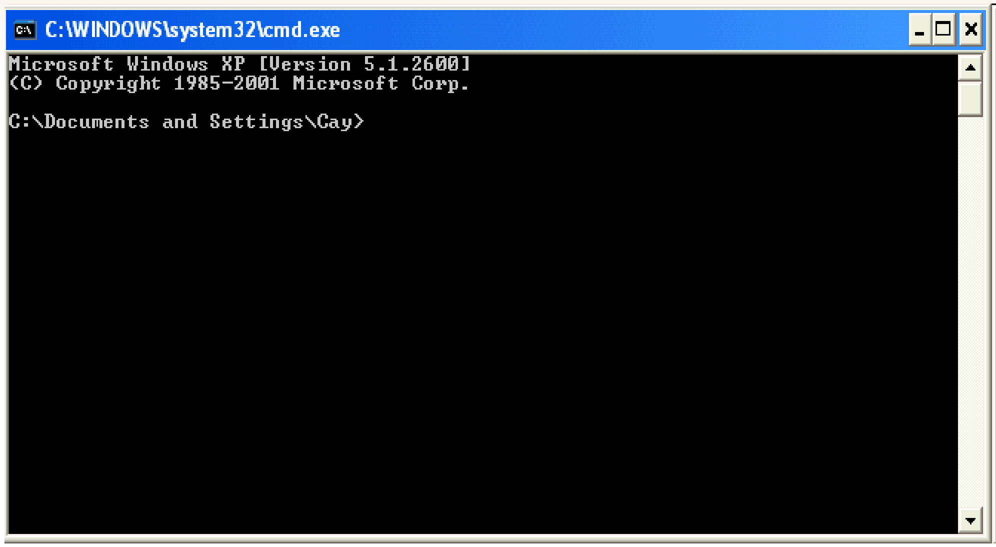


Shells



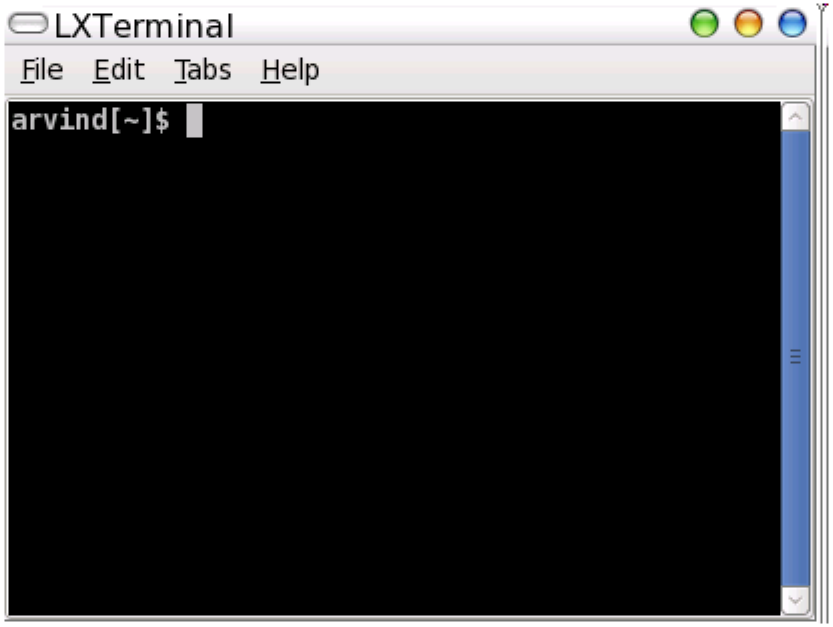
```
52k CP/M vers 2.2.2.15
Micro Technology BIOS

Set for 8sec drive spin-time
A>dir
A: MOVCPM   COM : PIP      COM
A: SUBMIT   COM : XSUB     COM
A: ED       COM : ASM      COM
A: DDT      COM : LOAD     COM
A: STAT     COM : DUMP     COM
A: DUMP     ASM : SYSGEN   COM
A: BACKUP   COM : CMT      COM
A: CONFIG   COM : CONSOLE  COM
A: COPY     COM : DEL      COM
A: EJECT    COM : FILES    COM
A: FORMAT   COM : INIT     COM
A: IODEFS   ASM : TIME     COM
A: CPM780   COM : SPIN87   COM
A: SOFTCEN7 COM : DISCMOD3 COM
A: DISX     COM : REENTER  COM
A: UNERASE  COM : VERIFY   COM
A: DPB      COM
A>
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Cay>
```



```
arvind[~]$
```

Window Manager

- An important component within today's GUI shells is the window manager, which allocates blocks of space on the screen, called windows, and keeps track of which application is associated with each window.
- When an application wants to display something on the screen, it notifies the window manager, and the window manager places the desired image in the window assigned to the application.
- In turn, when a mouse button is clicked, it is the window manager that computes the mouse's location on the screen and notifies the appropriate application of the mouse action.

The kernel

- In contrast to an operating system's shell, the internal part of an operating system is called the **kernel**.

- An operating system's kernel contains those software components that perform the very basic functions required by the computer installation.

File Manager

- One unit of the kernel is the file manager**,** whose job is to coordinate the use of the machine's mass storage facilities.
- More precisely, the file manager maintains records of all the files stored in mass storage, including:
  - where each file is located,
  - which users are allowed to access the various files
  - which portions of mass storage are available for new files or extensions to existing files.
- These records are kept on the individual storage medium containing the related files so that each time the medium is placed on-line, the file manager can retrieve them and thus know what is stored on that particular medium.

File Manager

- **Directory** (or **Folder**): A user-created bundle of files and other directories (subdirectories)
- **Directory Path:** A sequence of directories within directories

Device Drivers

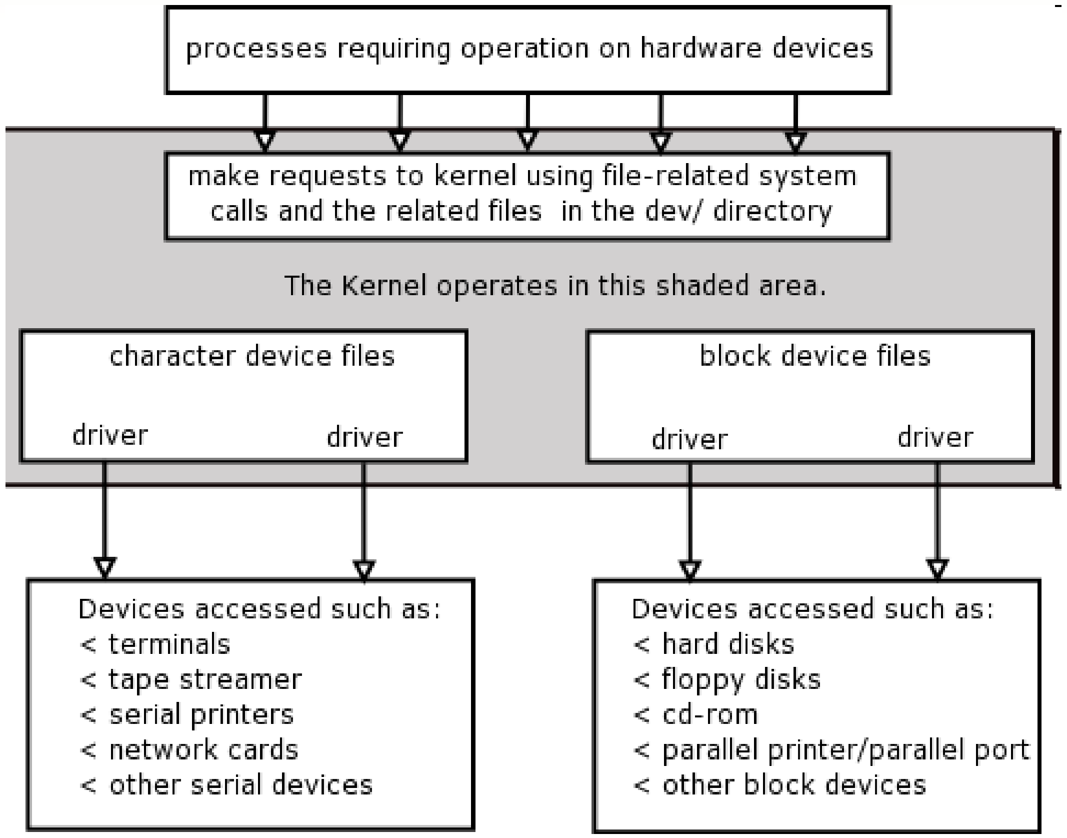
- Another component of the kernel consists of a collection of device drivers, which are the software units that communicate with the controllers (or at times, directly with peripheral devices) to carry out operations on the peripheral devices attached to the machine.

- Each device driver is uniquely designed for its particular type of device (such as a printer, disk drive, or monitor) and translates generic requests into the more technical steps required by the device assigned to that driver.

Device Drivers

- For example, a device driver for a printer contains the software for reading and decoding that particular printer's status word as well as all the other handshaking details.
- Thus, other software components do not have to deal with those technicalities in order to print a file.
- Instead, the other components can merely rely on the device driver software to print the file, and let the device driver take care of the details.
- In this manner, the design of the other software units can be independent of the unique characteristics of particular devices.
- The result is a generic operating system that can be customized for particular peripheral devices by merely installing the appropriate device drivers.

Device drivers


processes requiring operation on hardware devices
make requests to kernel using file-related system calls and the related files in the dev/ directory
The Kernel operates in this shaded area.
character device files
driver
driver
block device files
driver
driver
Devices accessed such as:
< terminals
< tape streamer
< serial printers
< network cards
< other serial devices
Devices accessed such as:
< hard disks
< floppy disks
< cd-rom
< parallel printer/parallel port
< other block devices

Accessing devices involves the kernel interacting with the I/O devices via a device driver.

Device drivers can be linked into the kernel image and then are available as needed.

Or they can be added into the kernel as a separate module and without rebooting the kernel.

Memory Manager

- Another component of an operating system's kernel is the memory manager**,** which is charged with the task of coordinating the machine's use of main memory.
  – Such duties are minimal in an environment in which a computer is asked to perform only one task at a time.
  – In these cases, the program for performing the current task is placed at a predetermined location in main memory, executed, and then replaced by the program for performing the next task.
- However, in multiuser or multitasking environments in which the computer is asked to address many needs at the same time, the duties of the memory manager are extensive.

Memory Manager

- In multitasking environments, many programs and blocks of data must reside in main memory concurrently.
- Thus, the memory manager must find and assign memory space for these needs and ensure that the actions of each program are restricted to the program's allotted space.
- Moreover, as the needs of different activities come and go, the memory manager must keep track of those memory areas no longer occupied.

Getting it Started (Bootstrapping)

- We have seen that an operating system provides the software infrastructure required by other software units, but we have not considered how the operating system itself gets started.

- This is accomplished through a procedure known as boot strapping (often shortened to booting) that is performed by a computer each time it is turned on.

- It is this procedure that transfers the operating system from mass storage (where it is permanently stored) into main memory (which is essentially empty when the machine is first turned on).

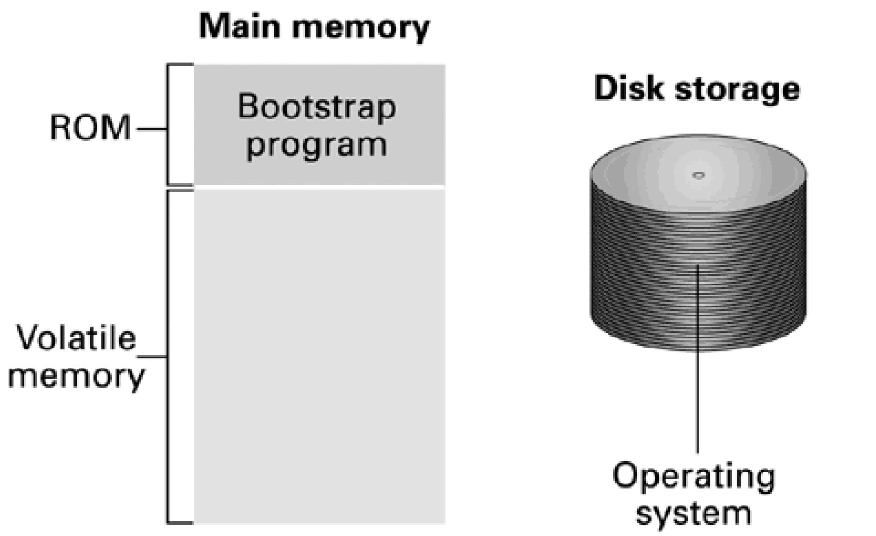
Booting

- A CPU is designed so that its program counter starts with a particular predetermined address each time the CPU is turned on.
- It is at this location that the CPU expects to find the beginning of the program to be executed.
- Conceptually, then, all that is needed is to store the operating system at this location.
- However, for both economic and efficiency reasons, a computer's main memory is typically constructed from volatile technologies - meaning that the memory loses the data stored in it when the computer is turned off.
- Thus, we need a means of replenishing main memory each time the computer is restarted.

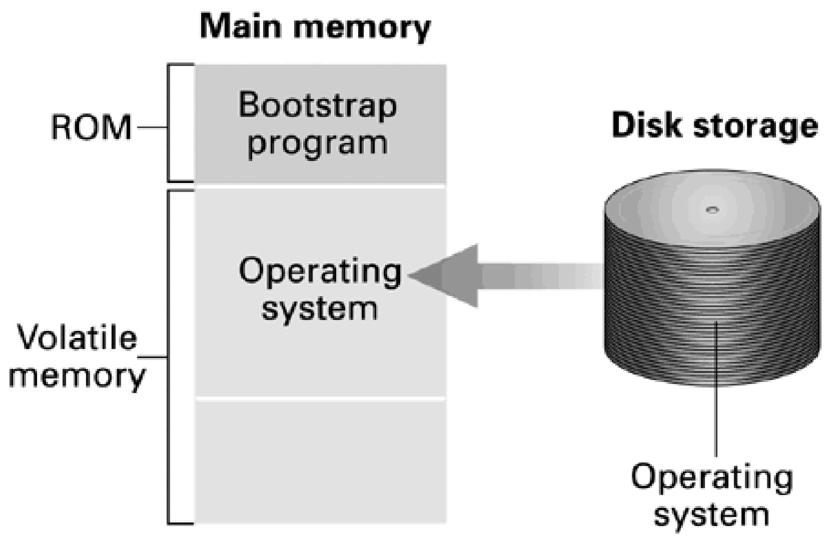
Read-Only Memory and bootstrap

- A small portion of a computer's main memory where the CPU expects to find its initial program is constructed from special nonvolatile memory cells.
  - Such memory is known as read-only memory (ROM) since its contents can be read but not altered.
  - More precisely, most ROM in today's PCs is constructed with flash memory technology (which means that it is not strictly ROM since it can be altered under special circumstances).
- The program stored in ROM is called the bootstrap.
  - This, then, is the program that is executed automatically when the machine is turned on.
  - Its task is to direct the CPU to transfer the operating system from a predetermined location in mass storage (typically a magnetic disk) into the volatile area of main memory.

Figure 3.5 **The booting process**

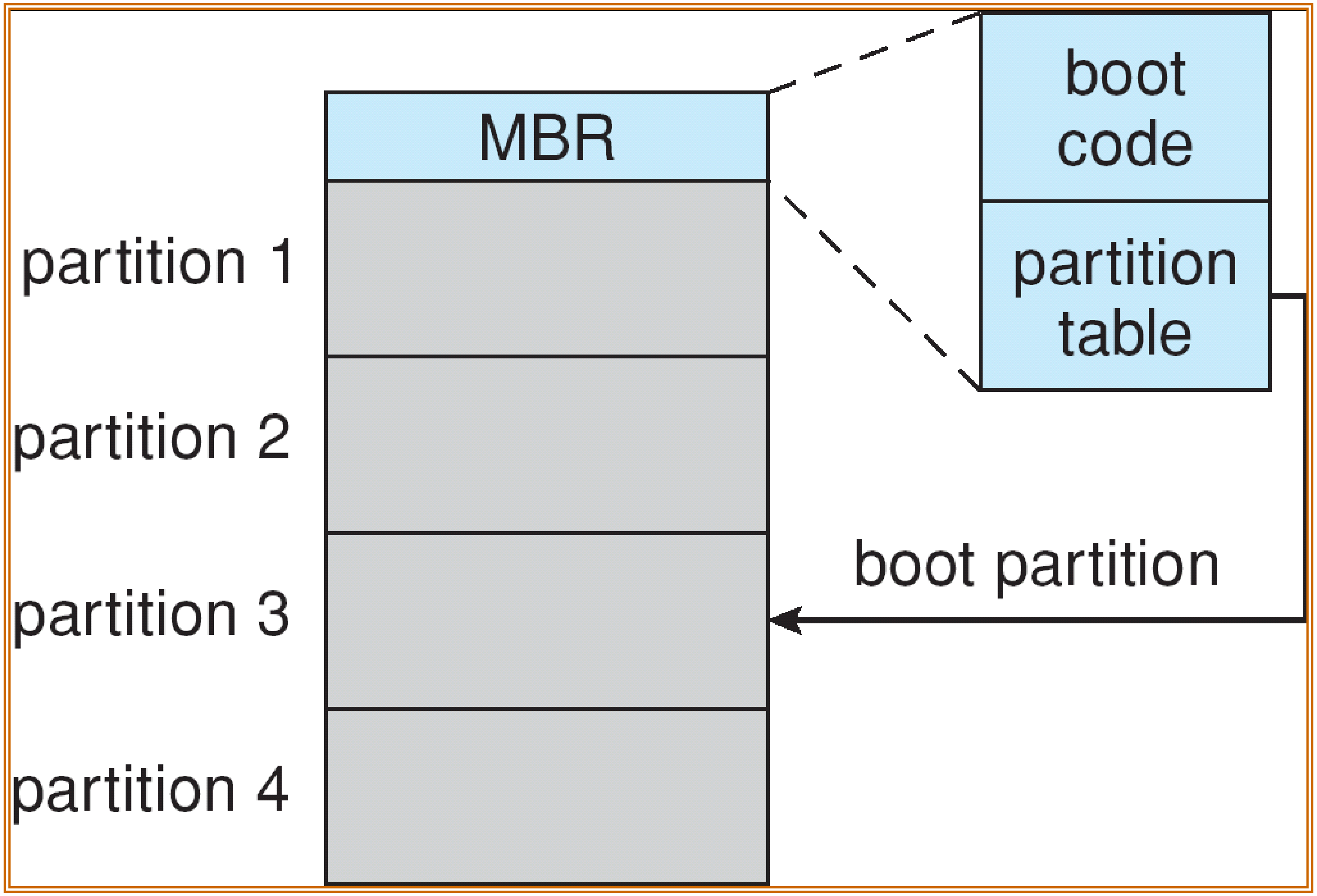


Step 1: Machine starts by executing the bootstrap program already in memory. Operating system is stored in mass storage.



Step 2: Bootstrap program directs the transfer of the operating system into main memory and then transfers control to it.

Booting from a Disk in Windows 2000



Modern boot loaders

- Modern boot loaders can copy an operating system into main memory from a variety of locations.
- For example, in embedded systems, such as smartphones, the operating system is copied from special flash (nonvolatile) memory;
- In the case of small workstations at large companies or universities, the operating system may be copied from a distant machine over a network.
- Once the operating system has been placed in main memory, the boot loader directs the CPU to execute a jump instruction to that area of memory.
- At this point, the operating system takes over and begins controlling the machine’s activities.
- The overall process of executing the boot loader and thus starting the operating system is called booting the computer.

Booting from mass storage: why?

- You may ask why desktop computers are not provided with enough ROM to hold the entire operating system so that booting from mass storage would not be necessary.
- While this is feasible for embedded systems with small operating systems, devoting large blocks of main memory in general-purpose computers to nonvolatile storage is not efficient with today's technology.
- Moreover, computer operating systems undergo frequent updates in order to maintain security and keep abreast of new and improved device drivers for the latest hardware.
- While it is possible to update operating systems and boot loaders stored in ROM, (often called a firmware update) the technological limits make mass storage the most common choice for more traditional computer systems.

Firmware and BIOS

- In addition to the boot loader, a PC’s ROM contains a collection of software routines for performing fundamental input/output activities such as receiving information from the keyboard, displaying messages on the computer screen, and reading data from mass storage.
- Being stored in nonvolatile memory such as FlashROM, this software is not immutably etched into the silicon of the machine — the hardware — but is also not as readily changeable as the rest of the programs in mass storage — the software.
- The term firmware was coined to describe this middle ground.
  - Firmware routines can be used by the boot loader to perform I/O activities before the operating system becomes functional.
  - For example, they are used to communicate with the computer user before the boot process actually begins and to report errors during booting.
- Widely used firmware systems include the BIOS (Basic Input/Output System) long used in “PCs”, the newer EFI (Extensible Firmware Interface), Sun’s Open Firmware (now a product of Oracle), and the CFE (Common Firmware Environment) used in many embedded devices.

Turn key systems

- Most special-purpose computers, such as those in household appliances, have all of their software permanently stored in their main memories where it is readily available each time the device is turned on.
- Such systems are known as turn key systems since they are ready to function with the flip of a switch or the turn of a key.
- With the rapid advances that are being made in memory technology, it may soon be that many of the steps in the booting process will become obsolete, and that general purpose computers will approach turn key status.

Operating Systems

- The History of Operating Systems
- Operating System Architecture
- Coordinating the Machine's Activities
- Handling Competition Among Processes
- Security

Process vs Programs

- The program itself is not a process
  - A program becomes a process only when the executable is loaded into main memory
  - A program is a passive entity whine the process in an active entity with a program counter assigned
- Two processes may be associated to the same program
  - But they are considered two separate execution sequences
  - If you invoke many copies of Firefox each of these is a separate process (or thread)

The concept of a process

- One of the most fundamental concepts of modern operating systems is the distinction between a program and the activity of executing a program.
- The former is a static set of directions, whereas the latter is a dynamic activity whose properties change as time progresses.
- This activity is known as a process. Associated with a process is the current status of the activity, called the process state.
- This state includes the current position in the program being executed (the value of the program counter) as well as the values in the other CPU registers and the associated memory cells.
  - Roughly speaking, the process state is a snapshot of the machine at a particular time.
  - At different times during the execution of a program (at different times in a process) different snapshots (different process states) will be observed.

Managing processes

- In a typical time-sharing computer installation, many processes are normally competing for the computer's resources.
- It is the task of the operating system to manage these processes so that:
  - each process has the resources (peripheral devices, space in main memory, access to files, and access to a CPU) that it needs
  - independent processes do not interfere with one another
  - processes that need to exchange information are able to do so.

Process Administration: Scheduler

- The tasks associated with coordinating the execution of processes are handled by the scheduler and dispatcher within the operating system's kernel.
- The scheduler:
  - maintains a record of the processes present in the computer system
  - introduces new processes to this pool
  - removes completed processes from the pool
- Thus when a user requests the execution of an application, it is the scheduler that adds the execution of that application to the pool of current processes.

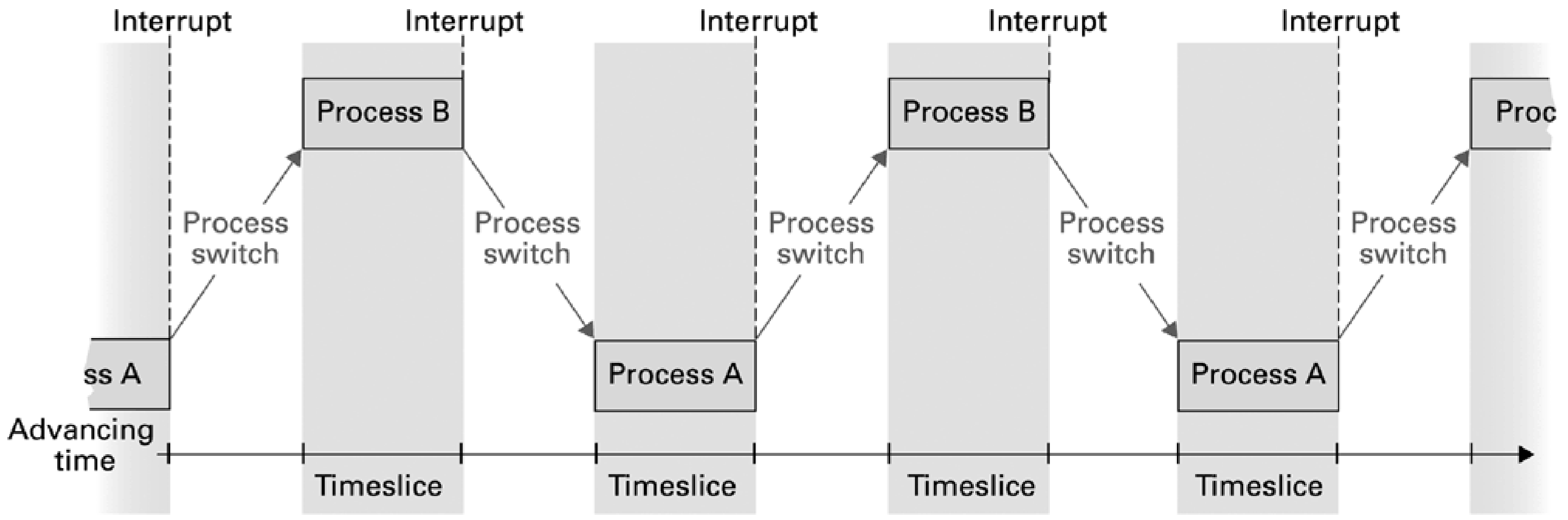
Process Administration: Dispatcher

- The dispatcher is the component of the kernel that ensures that the scheduled processes are actually executed.
- In a time-sharing system this task is accomplished by time-sharing; that is, dividing time into short segments, each called a time slice (typically no more than 50 milliseconds), and then switching the CPU's attention among the processes as each is allowed to execute for one time slice.
- The procedure of changing from one process to another is called a process switch or a context switch.

Interrupts

- The use of interrupts for terminating time slices is only one of many applications of a computer's interrupt system.
- There are many situations in which an interrupt signal is generated, each with its own interrupt routine.
- Indeed, interrupts provide an important tool for coordinating a computer's actions with its environment.
  - For example, both clicking a mouse and pressing a key on the keyboard generate interrupt signals that cause the CPU to set aside its current activity and address the cause of the interrupt.
  - To manage the task of recognizing and responding to incoming interrupts, the various interrupt signals are assigned priorities so that the more important tasks can be taken care of first.
- The highest priority interrupt is usually associated with a power failure. Such an interrupt signal is generated if the computer's power is unexpectedly disrupted. The associated interrupt routine directs the CPU through a series of "housekeeping" chores during the milliseconds before the voltage level drops below an operational level.

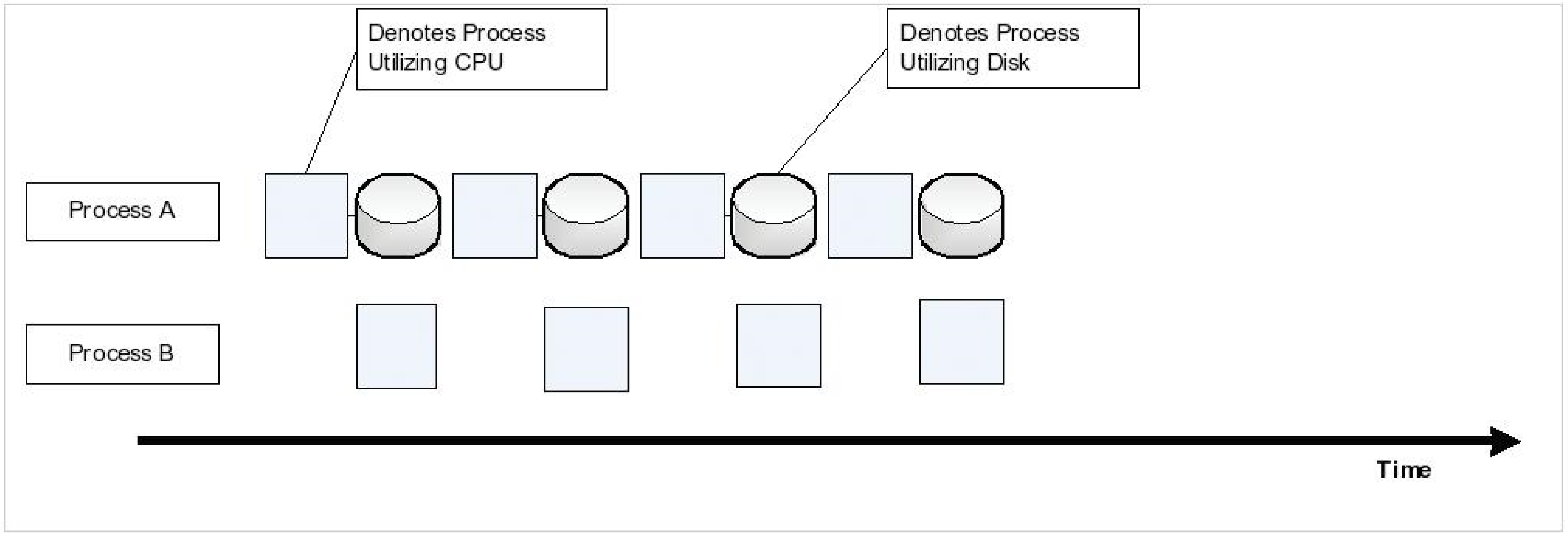
Figure 3.6 **Time-sharing between process A and process B**



Scheduler and dispatcher in action

- The effect of an interrupt signal is to preempt the current process and transfer control back to the dispatcher.
- At this point, the dispatcher first allows the scheduler to update the process table (for instance, the priority of the process that has just completed its time slice may need to be lowered and the priorities of other processes may need to be raised).
- Then, the dispatcher:
  - selects the process from the process table that has the highest priority among the ready processes
  - restarts the timer circuit
  - allows the selected process to begin its time slice.

Two processes sharing the CPU



Context Switch

- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process
- Context-switch time is overhead; the system does no useful work while switching
  - Its speed varies from machine to machine, depending on the memory speed, the number of registers which must be copied, and the existence of special instructions (such as a single instruction to load or store all registers).
  - Typically, the speed ranges from 1 to 1000 microseconds.
- Time dependent on hardware support
- Context switching may become a **performance bottleneck**
  - If there are more active processes than there are register sets, the system resorts to copying register data to and from memory, as before.

Efficiency of time-sharing

- In closing, we should note that the use of time-sharing has been found to increase the overall efficiency of a machine.
- This is somewhat counterintuitive since the shuffling of processes required by time-sharing introduces an overhead.
- However, without time-sharing each process runs to completion before the next process begins, meaning that the time that a process is waiting for peripheral devices to complete tasks or for a user to make the next request is **wasted**.
  - **Time-sharing allows this lost time to be given to another process.**

End of lesson 6

- Readings
  - Book: chapter 3.