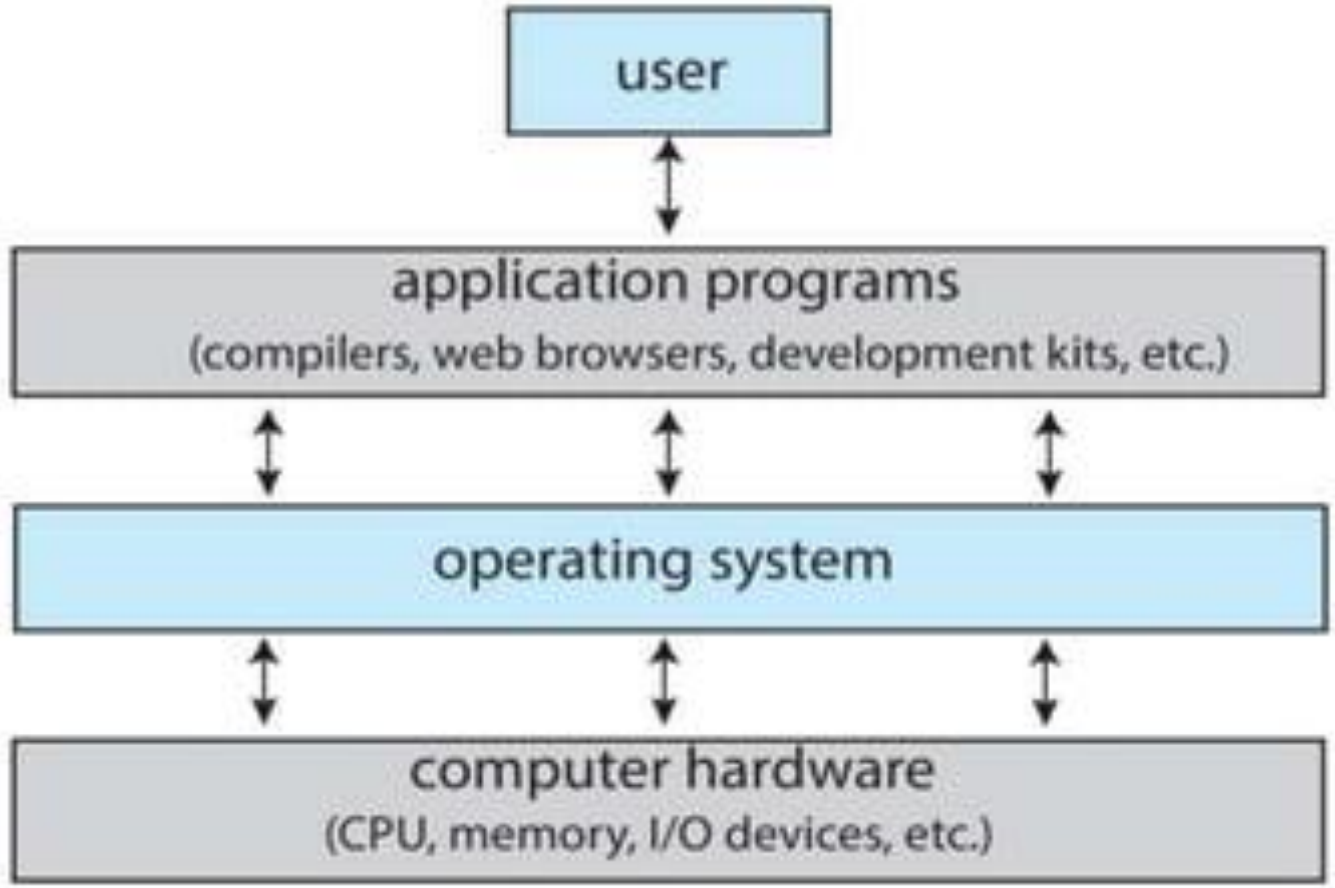
OPERATING SYSTEM UNIT I

Topics To be Covered

- Introduction to OS
- History and Generations of OS
- Objectives/Functions of OS
  - -- OS Concepts
  - -- OS Services
- Types of OS

Introduction to OS

Abstract View of Components of Computer



Basic Elements of Computer System

Processor

I/O Modules

Main Memory

System Bus

Top-Level View

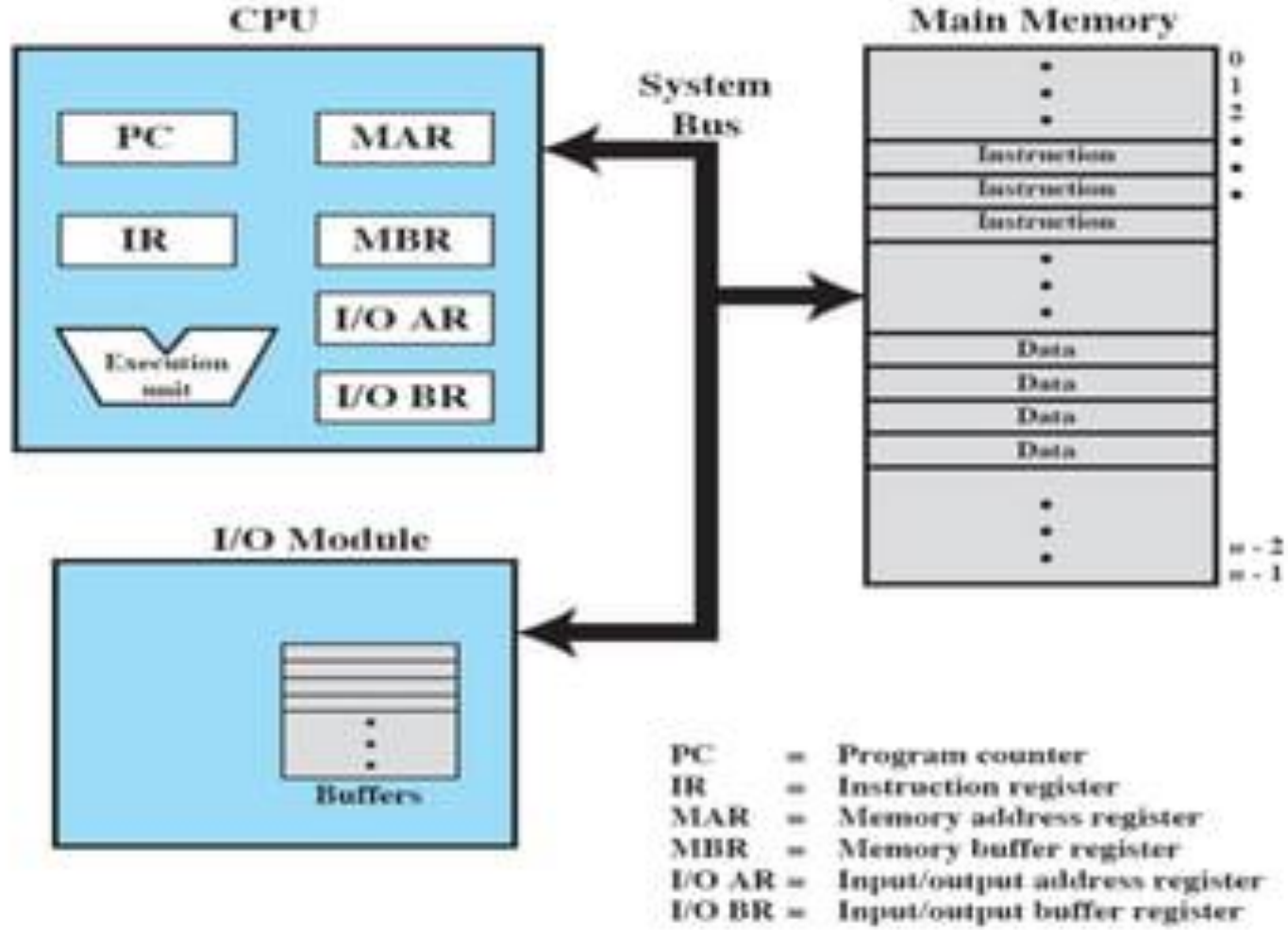


Figure 1.1 Computer Components: Top-Level View

What Is An Operating System

- A modern computer consists of:
  - One or more processors
  - Main memory
  - Disks
  - Printers
  - Various input/output devices
- It is impossible for every application programmer to understand every details.
- A layer of computer software is introduced to provide a better, simpler, cleaner model of the resources and manage them - **the operating system**

What Is An Operating System

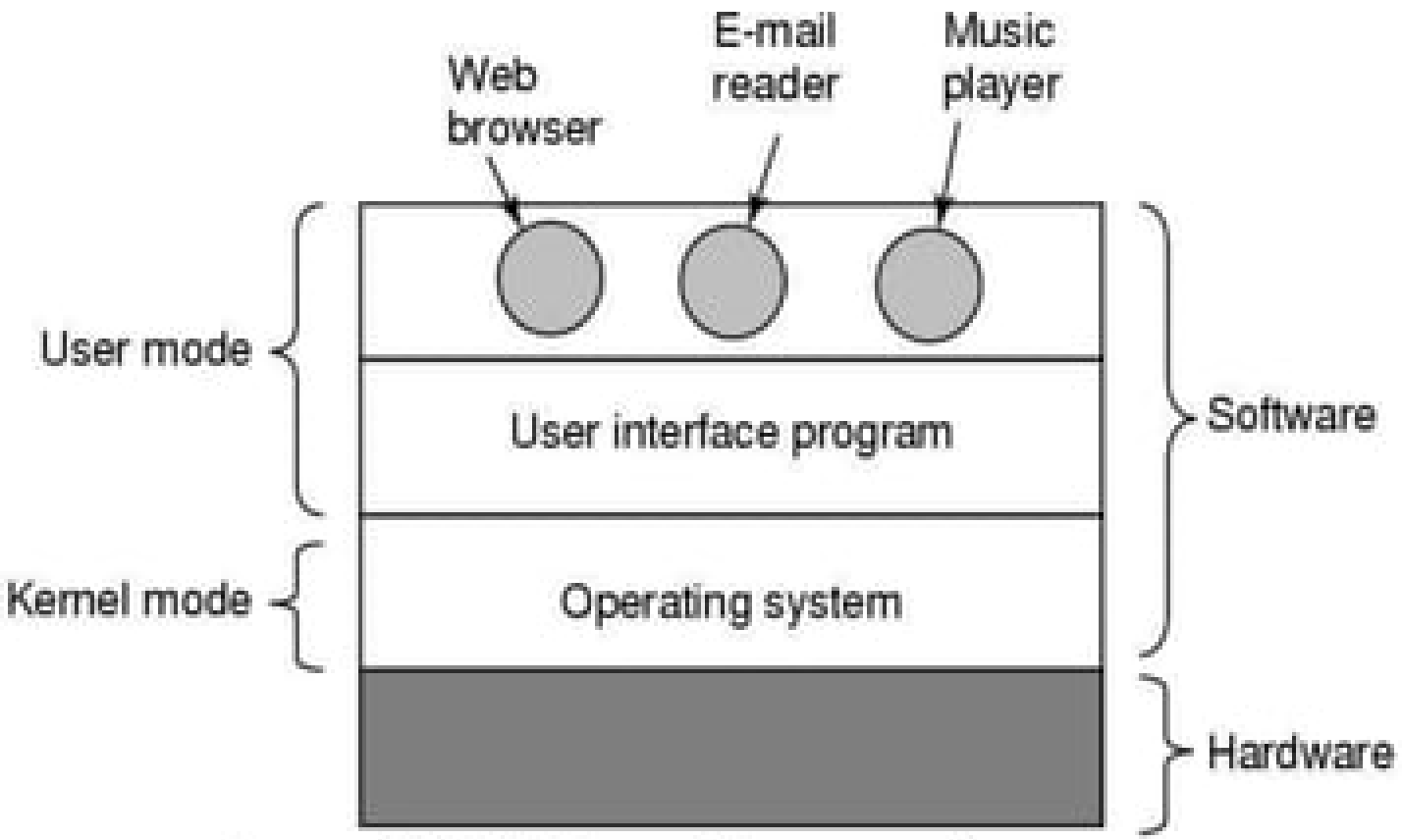


Figure 1-1. Where the operating system fits in.

- On top of hardware is OS
- Most computers have two modes of operation:
  - Kernel mode and user mode
  - OS runs in kernel mode, which has complete access to all hardware and can execute any instruction
  - Rest of software runs in user mode, which has limited capability
  - Shell or GUI is the lowest

Thus, Operating System is..

- 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
- "The one program running at all times on the computer" is the **kernel**, part of the operating system. Everything else is either
  - **A system program** (ships with the operating system, but not part of the kernel) , or
  - **An application program**, all programs not associated with the operating system

software that controls the hardware

its an interface between the application program and the hardware level.

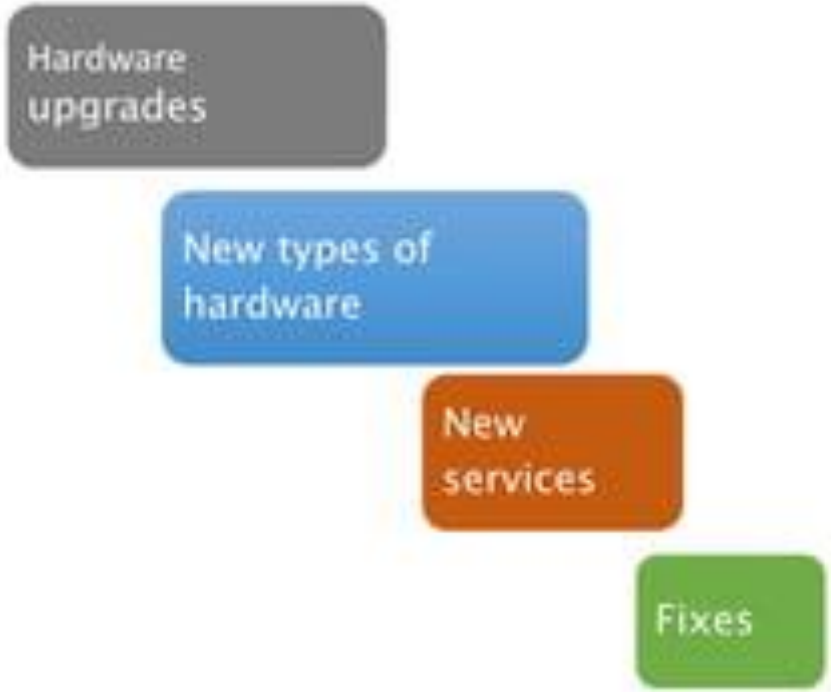
Main objectives of an OS:
Convenience
Efficiency
Ability to evolve

History and Generations of Operating System

In the 1960s, Moore's Law predicted that the number of transistors on an integrated circuit would double every 18 months, and that prediction has held true.

Evolution of Operating Systems

- A major OS will evolve over time for a number of reasons:



- Operating Systems are among the most complex pieces of software ever developed
- Major advances in development include:
  - Processes
  - Memory management
  - Information protection and security
  - Scheduling and resource management

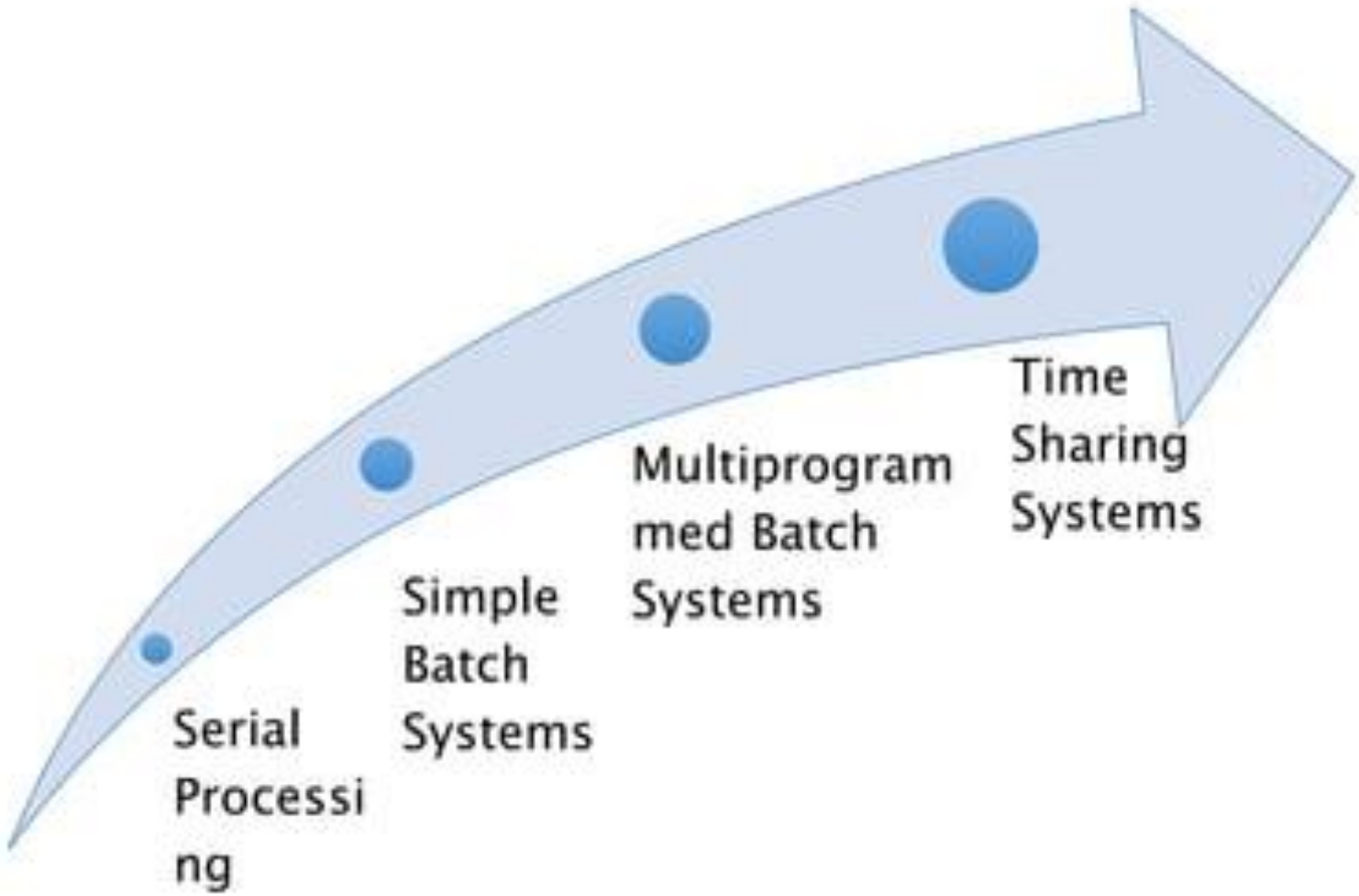
History of Operating Systems

Generations:

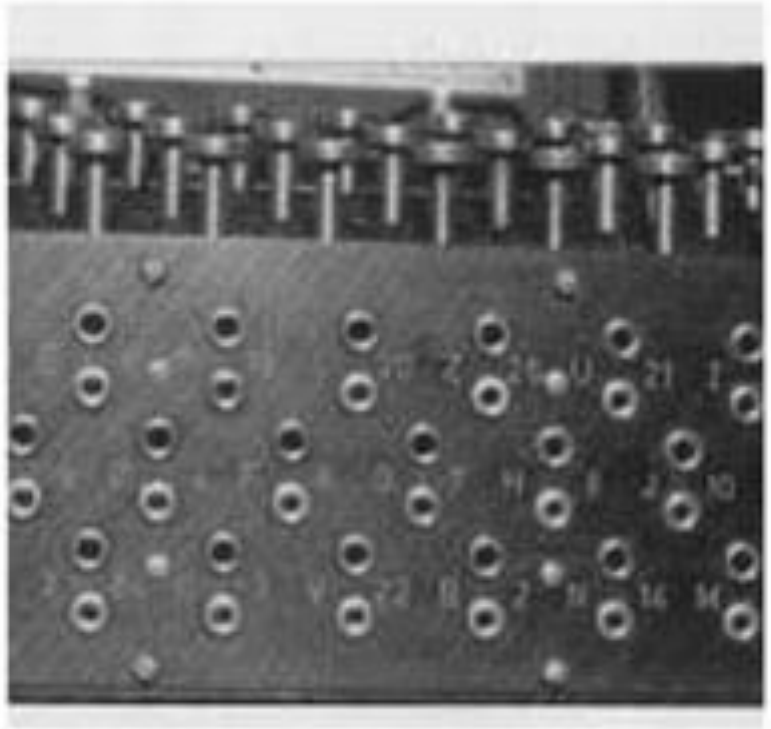
- (1945–55) Vacuum Tubes – **First Generation**
- (1955–65) Transistors and Batch Systems – **Second Generation**
- (1965–1980) ICs and Multiprogramming – **Third Generation**
- (1980–Present) Personal Computers – **Fourth Generation**

Evolution of Operating Systems

Stages include:



First Generations – (1945-55) Vacuum Tubes



- When electronic computers were first introduced in the 1940's they were created **without any operating systems.**
- **Large and slow**
- Engineers design, build, operate and maintain the computer
- All programming was done in **absolute machine language**, often by wiring up plug boards to control the machine's basic functions.
- During this generation computers were generally **used to solve simple math calculations**, operating systems were not necessarily needed.

Serial Processing

Earliest Computers:

- No operating system
  - programmers interacted directly with the computer hardware
- Computers ran from a console with display lights, toggle switches, some form of input device, and a printer
- Users have access to the computer in "series"

Problems:

- time allocations could run short or long, resulting in wasted computer time
- a considerable amount of time was spent just on setting up the program to run

Second Generation – (1955–65) Transistors and Batch Systems

- Also called mainframes
- Computers are managed by professional operators
- Programmers **use punch card to run programs**; operators operate (load compiler etc) and collect output to the user
- Complains soon come:
  - Human Operation between computer operation
  - Lead to **batch system**
  - Collect a batch of jobs in the input room, then read them into a magnetic tape; the same for output

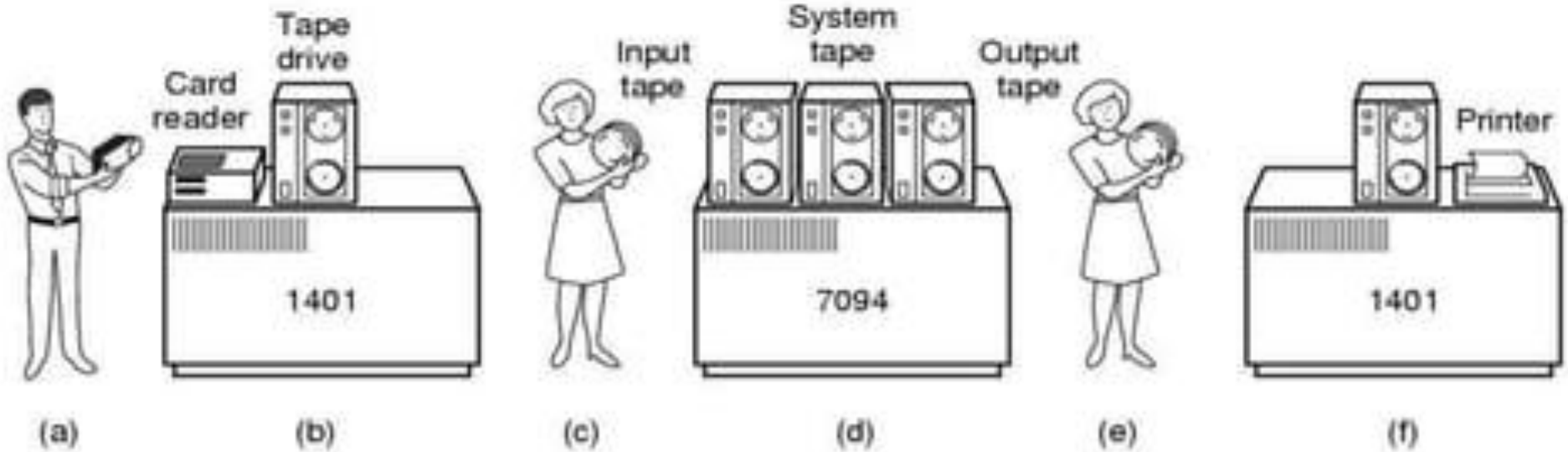


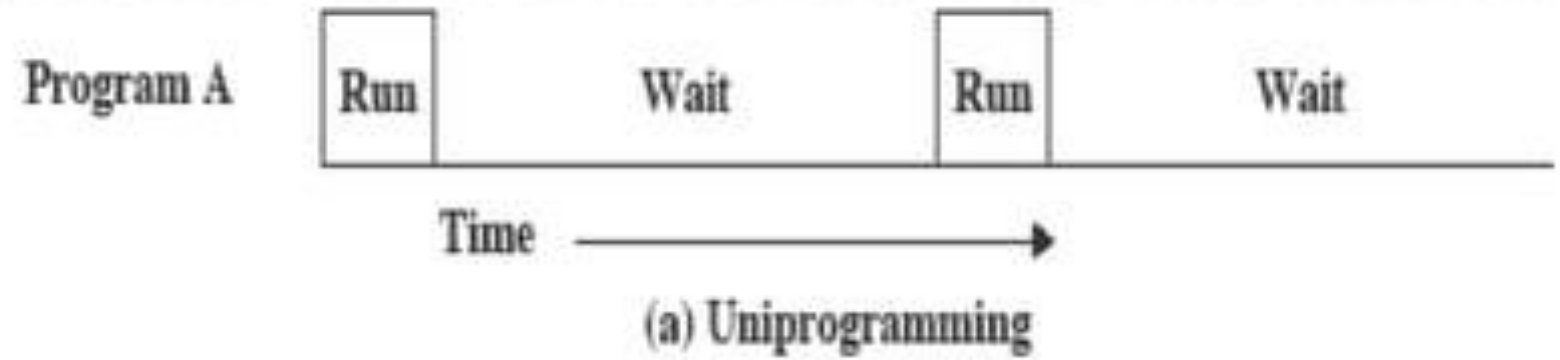
Figure 1-3. An early batch system.
(a) Programmers bring cards to 1401.
(b) 1401 reads batch of jobs onto tape.
(c) Operator carries input tape to 7094.
(d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

Simple Batch Systems

Early computers were very expensive

- important to maximize processor utilization
- job is submitted to computer operator who batches them together and places them on an input device
- program branches back to the monitor

wl



(a) Uniprogramming

Third Generation – (1965–1980) ICs and Multiprogramming

- By the late 1960's operating systems designers were able to develop the **system of multiprogramming in which a computer program will be able to perform multiple jobs** at the same time.
- It allowed a CPU to be busy nearly 100 percent of the time that it was in operation.
- Another major development during the third generation was the **phenomenal growth of minicomputers**,(less than 5 percent of the price of a 7094), it sold like hotcakes.
- These minicomputers help create a whole new industry and the development of more PDP's. These PDP's helped lead to the creation of personal computers which are created in the fourth generation.

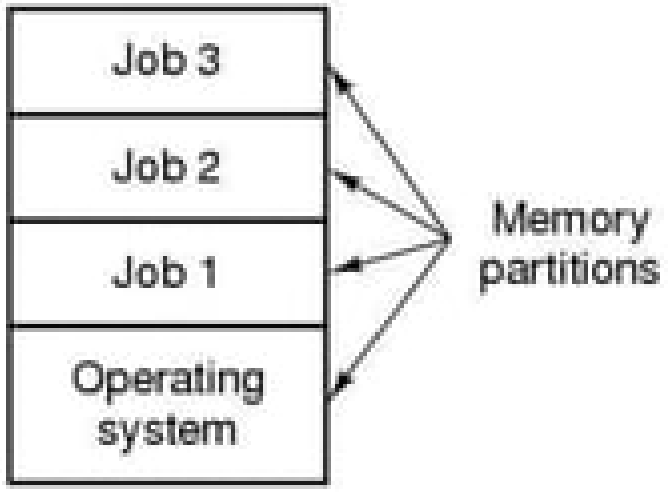
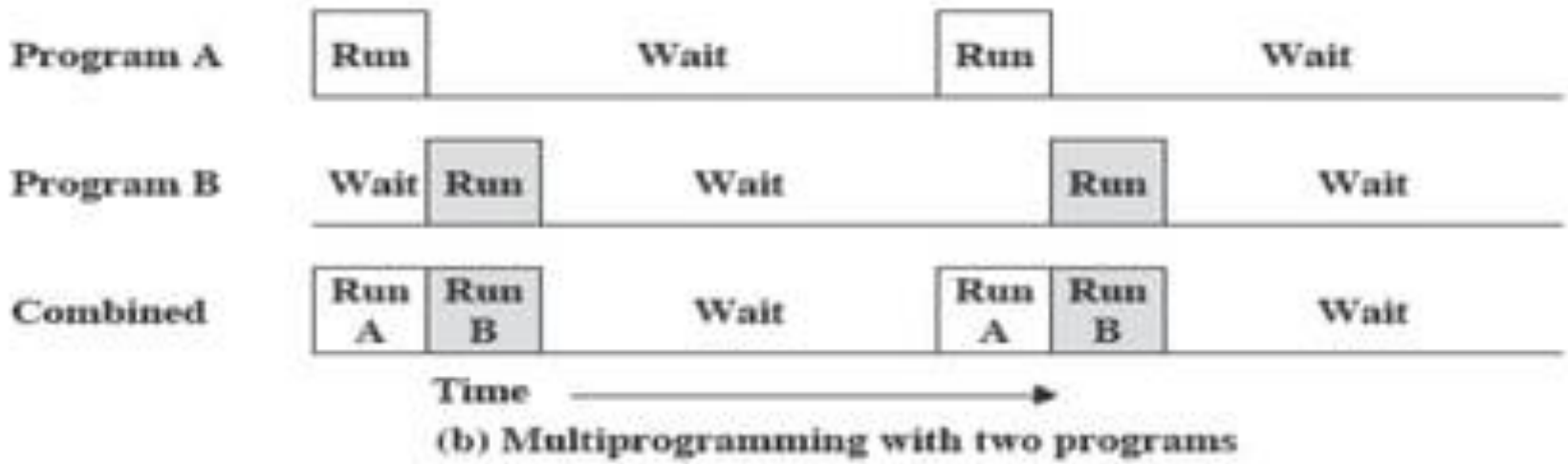


Figure 1-5. A multiprogramming system with three jobs in memory.

- 3rd generation OS was **well suited for big scientific calculations and massive data processing**
- **OS/360**: a dinosaur stuck in a tar pit
  - Aims to adapts 1401/7904, covers all trades of life
  - However, OS/360 introduces several key techniques
    - Multi-programming: solve the problem of CPU idling
    - Spooling: simultaneous peripheral operation on line
- A system to be remembered: **MULTICS**
  - A machine that would support hundreds of simultaneous timesharing users
  - Introduces many brilliant ideas but enjoys no commercial success
  - Its step-child is the well-known and time-honored UNIX -System V/ FreeBSD, MINIX

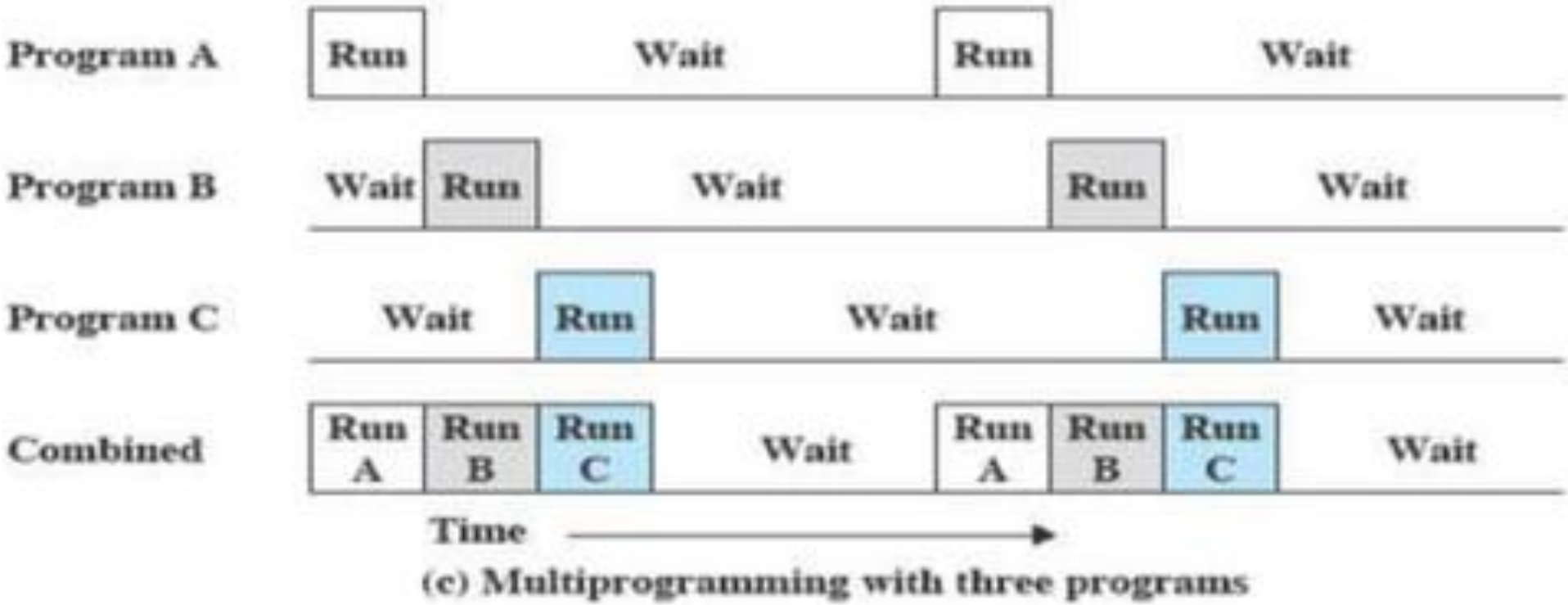
Multiprogramming



(b) Multiprogramming with two programs

There must be enough memory to hold the OS (resident monitor) and one user program

When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O



(c) Multiprogramming with three programs

memory is expanded to hold three, four, or more programs and switch among all of them

Fourth Generation - (1980-Present) Personal Computers

- The fourth generation of operating systems saw the creation of personal computing.
- One of the major factors in the creation of personal computing was the birth of **Microsoft and the Windows operating system**.
- Bill Gates introduced the MS-DOS in 1981 although **it was effective - difficult** for people who tried to understand its cryptic commands.

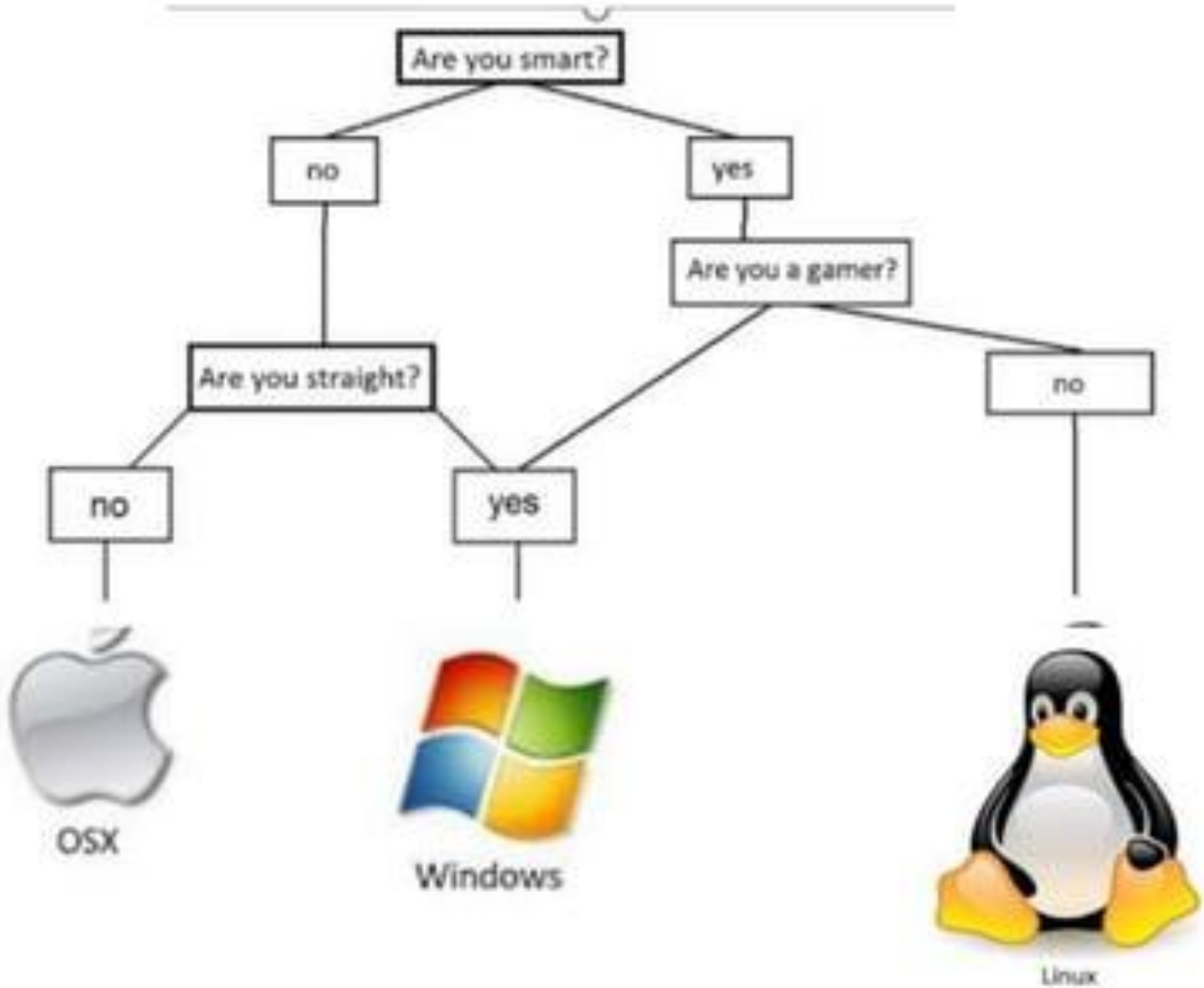
- Windows went on to become the largest operating system used in technology today with releases of Windows 95, Windows 98, Windows XP (Which is currently the most used operating system to this day), and their latest operating system Windows 11.

- Along with Microsoft, Apple is the other major operating system created in the 1980's. Steve Jobs, co-founder of Apple, created the Apple Macintosh which was a huge success due to the fact that it was so user friendly.

Time-Sharing Systems (Multi-Tasking)

- Can be used to handle multiple interactive jobs
- Processor time is shared among multiple users
- Multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation


All available tasks appear to be executing ...
Task 1 Executing
Task 2 Executing
Task 3 Executing
t1
t2
Time
tn
... but only one task is ever executing at any time.
Task 1 Executing
Task 2 Executing
Task 3 Executing
t1
t2
Time
tn


Are you smart?
no
yes
Are you a gamer?
Are you straight?
no
no
yes
OSX
Windows
Linux

Objectives/ Functions of Operating System

Functions of operating system

Two functions:

- **From top to down**: provide application programmers a clean abstract set of resources instead of hardware ones

(OS as Extended Machine)

- **From down to top**: Manage these hardware resources

(OS as Resource Manager)

As an extended machine

- The **architecture** (instruction set, memory organization, I/O, and bus structure) of most computers at the machine-language level is **primitive and awkward** to program, especially for input/output.
- The job of the operating system is **to create good abstractions** and then implement and manage the abstract objects thus created.
- Abstraction:
  - CPU—process
  - Storage – files
  - Memory– address space
- Operating systems turn the ugly into the beautiful, as

The Operating System as an Extended Machine

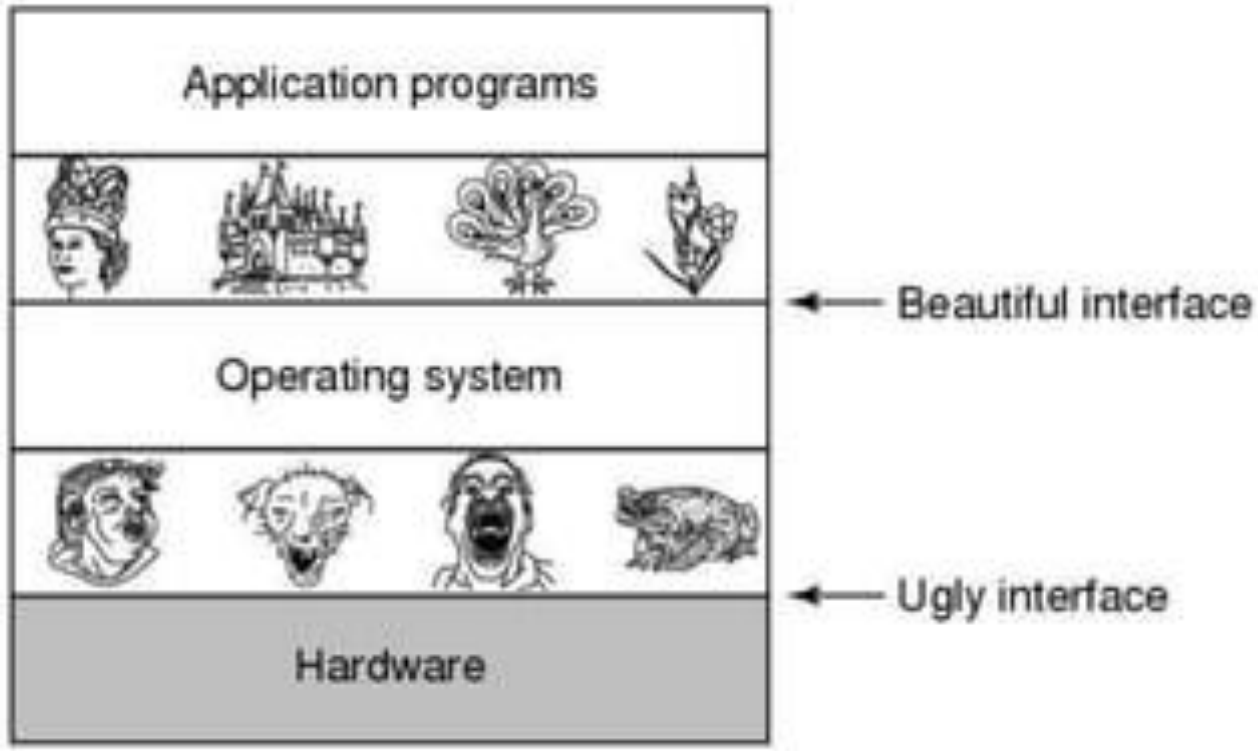


Figure 1-2. Operating systems turn ugly hardware into beautiful abstractions.

Operating System as a resource manager

- Modern OS runs multiple programs of multiple users at the same time
  - Imagine what would happen if several programs want to print at the same time?
  - How to account the resource usage of each process?
  - Resources can be multiplexed:
    - How to ensure fairness and efficiency?

The Operating System as a Resource Manager....

- In the **bottom-up view,** the job of the operating system is to provide for an **orderly and controlled allocation** of the processors, memories, and I/O devices among the various programs wanting them.
- In this view operating system keeps track of which programs are using which resource, to grant resource requests, to account for usage, and to mediate conflicting requests from different programs and users.
- OS manages and protect memory, I/O devices, and other resources
- Resource management includes multiplexing (sharing) resources in two different ways: in time and in space.

In time: different programs or users take turns using resources. First one of them gets to use the resource, then

As a resource manager

Process Management

- A process is a **program in execution.** It is a unit of work within the system.
- Program is a passive entity; process is an active entity.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization Data
- Process termination requires reclaim of any reusable resources
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs

As a resource manager

Process Management Activities

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

As a resource manager

Memory Management

- To execute a program all (or part) of the **instructions must be in memory.**
- Memory management **determines what is in memory and when**
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory

As a resource manager

File-system Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

As a resource manager

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection

As a resource manager

Caching

- Caching is Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

As a resource manager

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Types of Operating System

Types of OS
Operating System Zoo

Types of OS

- Batch operating system
- Time-sharing operating systems
- Distributed operating System
- Network operating System
- Real Time operating System

Batch operating system

- The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator.
- To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

Problems with Batch Systems are as follows −

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- Difficult to provide the desired priority.

Time-sharing operating systems

- Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.
- The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.
- Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response.
- The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Time-sharing operating systems

Advantages of Timesharing operating systems are as follows −

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Time-sharing operating systems are as follows −

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Distributed operating System

- Distributed systems use multiple central processors to serve multiple real-time applications and multiple users.
- Data processing jobs are distributed among the processors accordingly.
- The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines).

The advantages of distributed systems are as follows −

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Network operating System

- A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions.
- The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.
- Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Network operating System

The advantages of network operating systems are as follows −

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are as follows −

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Real Time operating System

- A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment.
- The time taken by the system to respond to an input and display of required updated information is termed as the response time. So in this method, the response time is very less as compared to online processing.
- Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application.
- A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

Real Time operating System

Hard real-time systems

- Hard real-time systems guarantee that critical tasks complete on time.

Soft real-time systems

- Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes.

Other elements in The Operating System Zoo

- Mainframe operating systems
- Multiprocessor operating systems
- Personal computer operating systems
- Handheld operating systems
- Embedded operating systems
- Smart card operating systems

Operating System Concepts

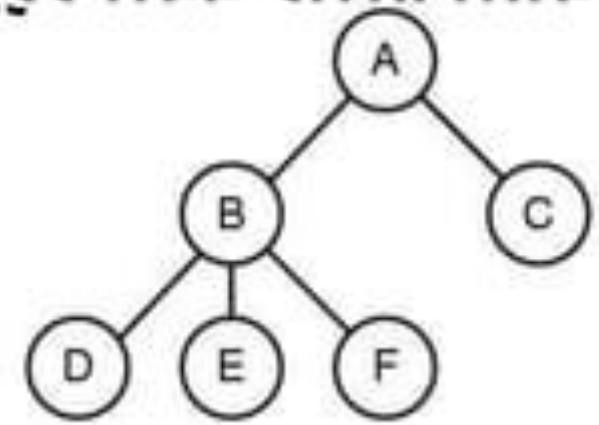
- Processes
- Address spaces
- Files
- System Call

Processes

- A process is basically a **program in execution.**
- A process is fundamentally **a container** that holds information for a program to run.
- Each process is associated with set of resources commonly including **registers ,a list of open files, outstanding alarms, lists of related processes**, and all the other information needed to run the program.
- Mainly process consists of its **address space and its process table** entry.
- Associated with each process is its **address space** which is a , a list of memory locations from 0 to some maximum, which the process can read and write.
  - Address space contains executable program, program's data, and its stack
- When a process is temporarily suspended, all the pointers must

Processes

- A process can create one or more other processes and these processes in turn can create other child processes so that we get a **process tree structure** as shown below.



- Related processes that are cooperating to get some job done often need to communicate with one another and synchronize their activities. This communication is called **interprocess communication**, and will be addressed it in detail in unit 3.

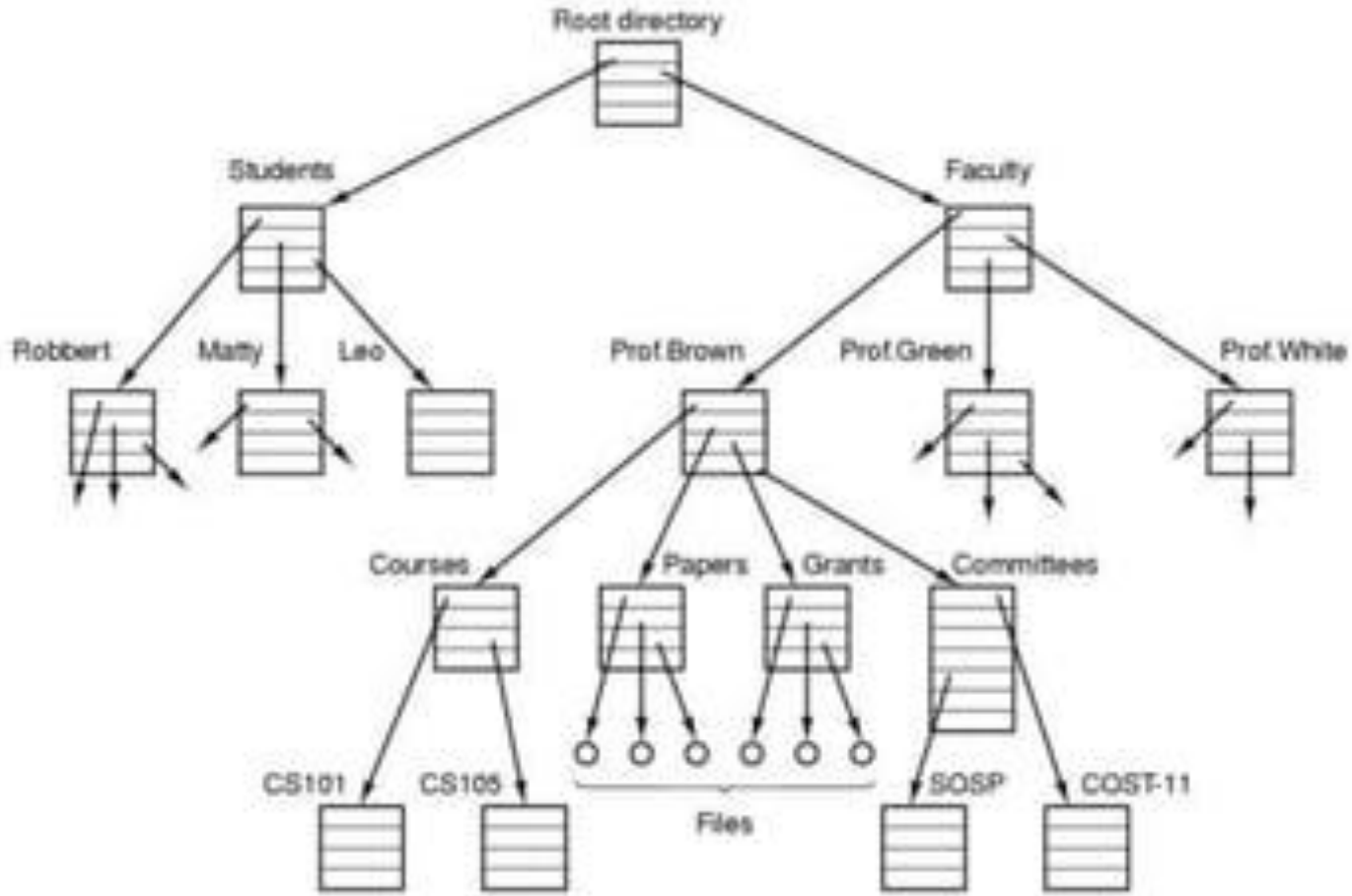
Address Space

- Conceptually it is **the memory used by a process**
- Normally, each process has some set of addresses it can use, typically running from 0 up to some maximum , less than the main memory.
- What happens if a process has more address space than the computer has main memory and the process wants to use it all?
- **Virtual memory** exists, in which the operating system keeps part of the address space in main memory and part on disk.
- In essence, the operating system **creates the abstraction of an address space** as the set of addresses a process may reference.

Files

- Operating system hides the peculiarities of the disks and presents the programmer with a nice, clean abstract model of device-independent **files**.
- To provide a place to keep files, most PC operating systems have the concept of a **directory** as a way of grouping files together.
- System calls are obviously needed to create files, remove files, read files, and write files ,to create and remove directories and also to put an existing file in a directory and to remove a file from a directory.
- Directory entries may be either files or other directories. This model also gives rise to a hierarchy—the **file system** as below

Files (1)


Root directory
Students
Faculty
Robbert
Matty
Leo
Prof.Brown
Prof.Green
Prof.White
Courses
Papers
Grants
Committees
CS101
CS105
SOSP
COST-11
Files

Files (2)

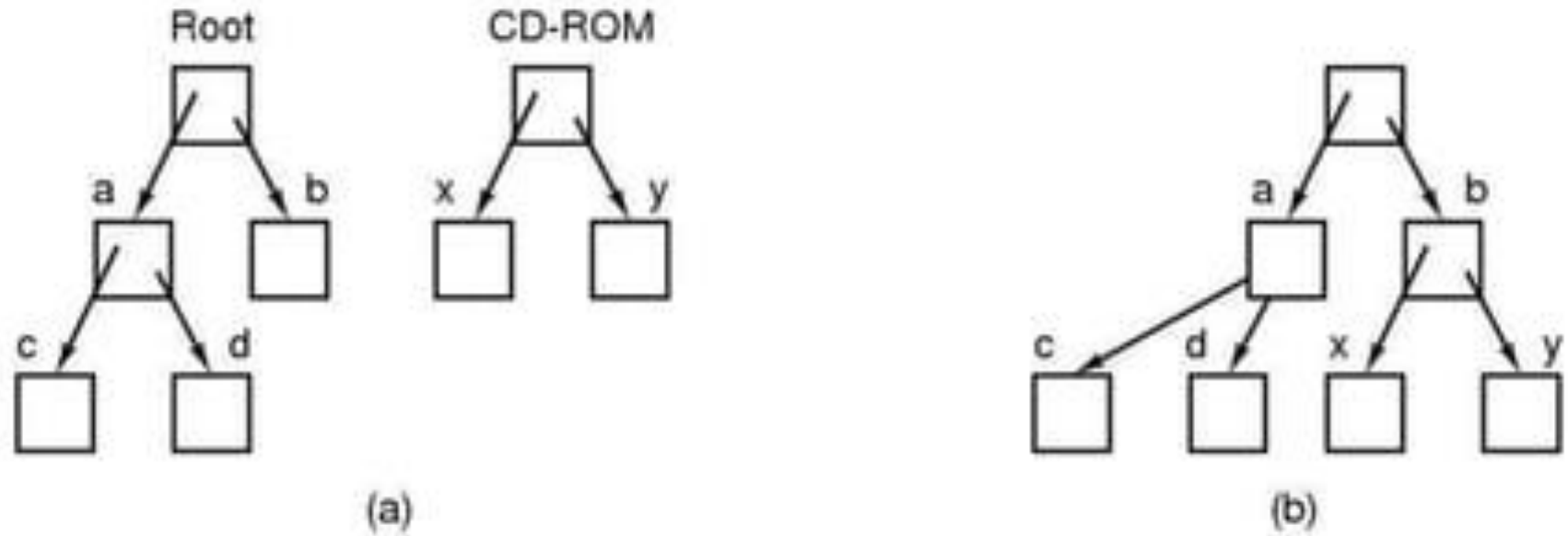


Figure 1-15. (a) Before mounting, the files on the CD-ROM are not accessible. (b) After mounting, they are part of the file hierarchy.

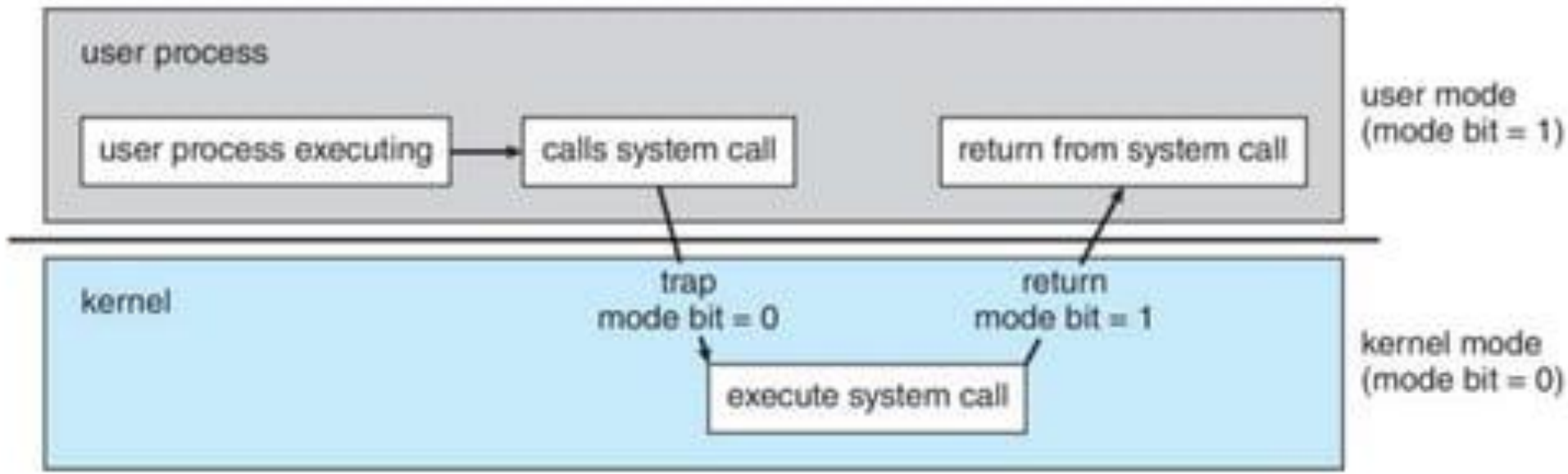
Files (3)

- A **pipe** is a sort of pseudofile that can be used to connect two processes, as shown in Fig.
- When process *A* wants to send data to process *B*, it writes on the pipe as though it were an output file. Process *B* can read the data by reading from the pipe as though it were an input file.
- Thus, communication between processes in UNIX looks very much like ordinary file reads and writes.

Figure : Two processes connected by a pipe.

System calls

- System calls is the interface users contact with OS and hardware
- Any single-CPU computer can execute only one instruction at a time. If a process is running a user program in user mode and needs a system service, such as reading data from a file, it has to execute a trap instruction to transfer control to the operating system-known as **system call.**
- System calls vary from system to system, but the underlying concepts are similar


user process
user process executing
calls system call
return from system call
user mode
(mode bit = 1)
kernel
trap
mode bit = 0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

System Calls

System call **has three parameters**: the first one specifying the file, the second one pointing to the buffer, and the third one giving the number of bytes to read.
count = read(fd, buffer, nbytes);

Step 1-3: the calling program **first pushes the parameters onto the stack.**
Step 4 -5: Then comes the actual call to the library procedure- **puts the system-call number in** a place where the operating system expects it, such as **a register.**
Step 6 : It executes a TRAP instruction to switch from user mode to kernel mode and **start execution at a fixed address within the kernel**.

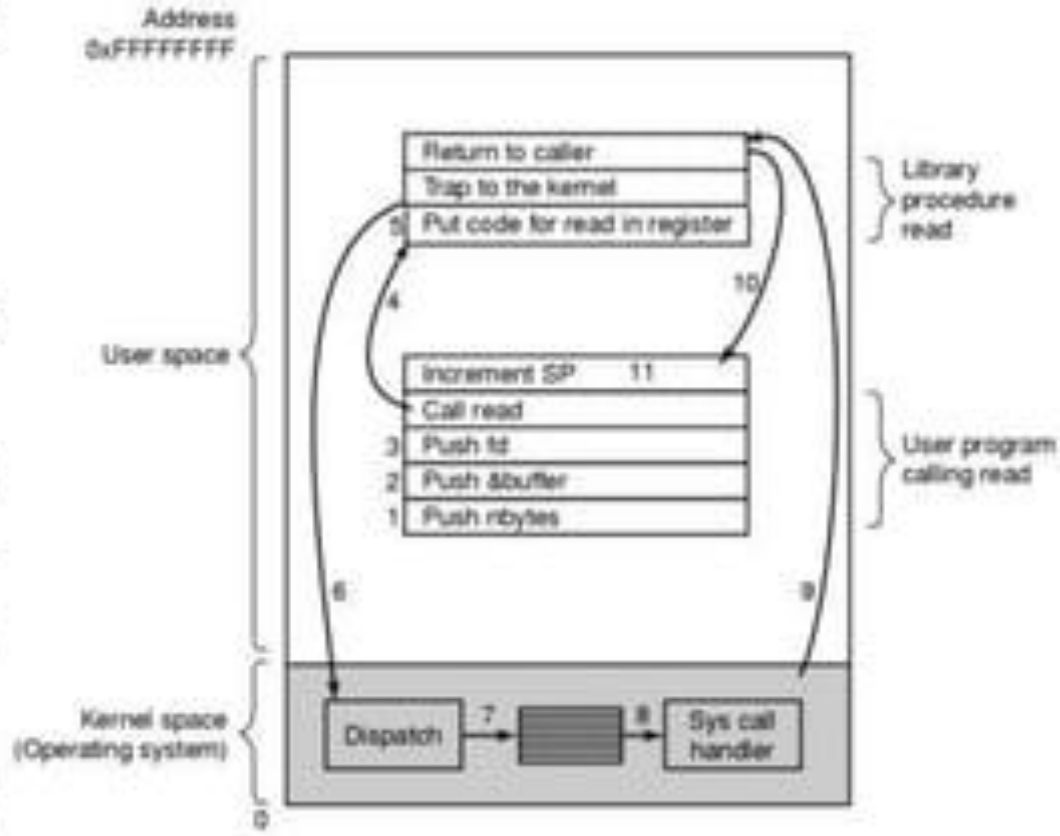


Figure 1-17. The 11 steps in making the system call

System Calls

Step 7: The kernel code that starts following the TRAP examines the system-call number and then **dispatches to the correct system-call handler.**

Step 8 : The **system-call handler** runs.

Step 9: Once it has completed its work, control
may be returned to the user-space library procedure at the instruction following the TRAP instruction .

Step 10: This procedure then returns to the user program in the usual way procedure calls return.

Step 11: To finish the job, the user program has to clean up the stack, as it does after

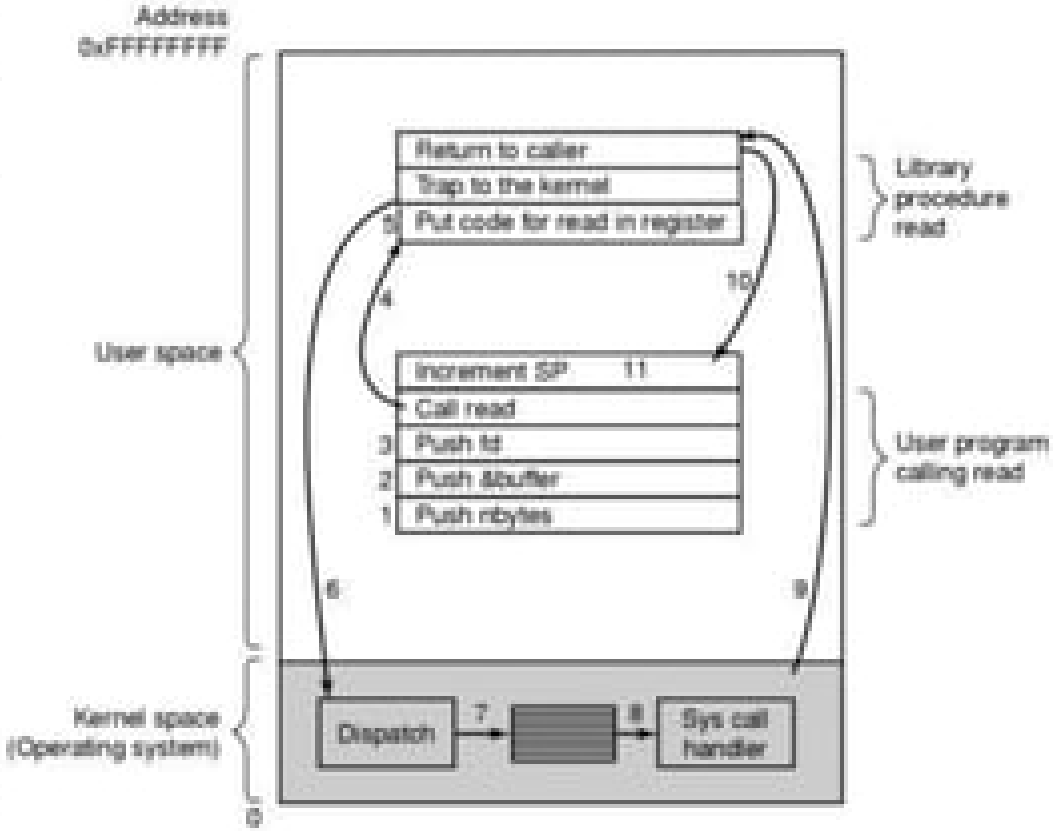


Figure 1-17. The 11 steps in making the system call