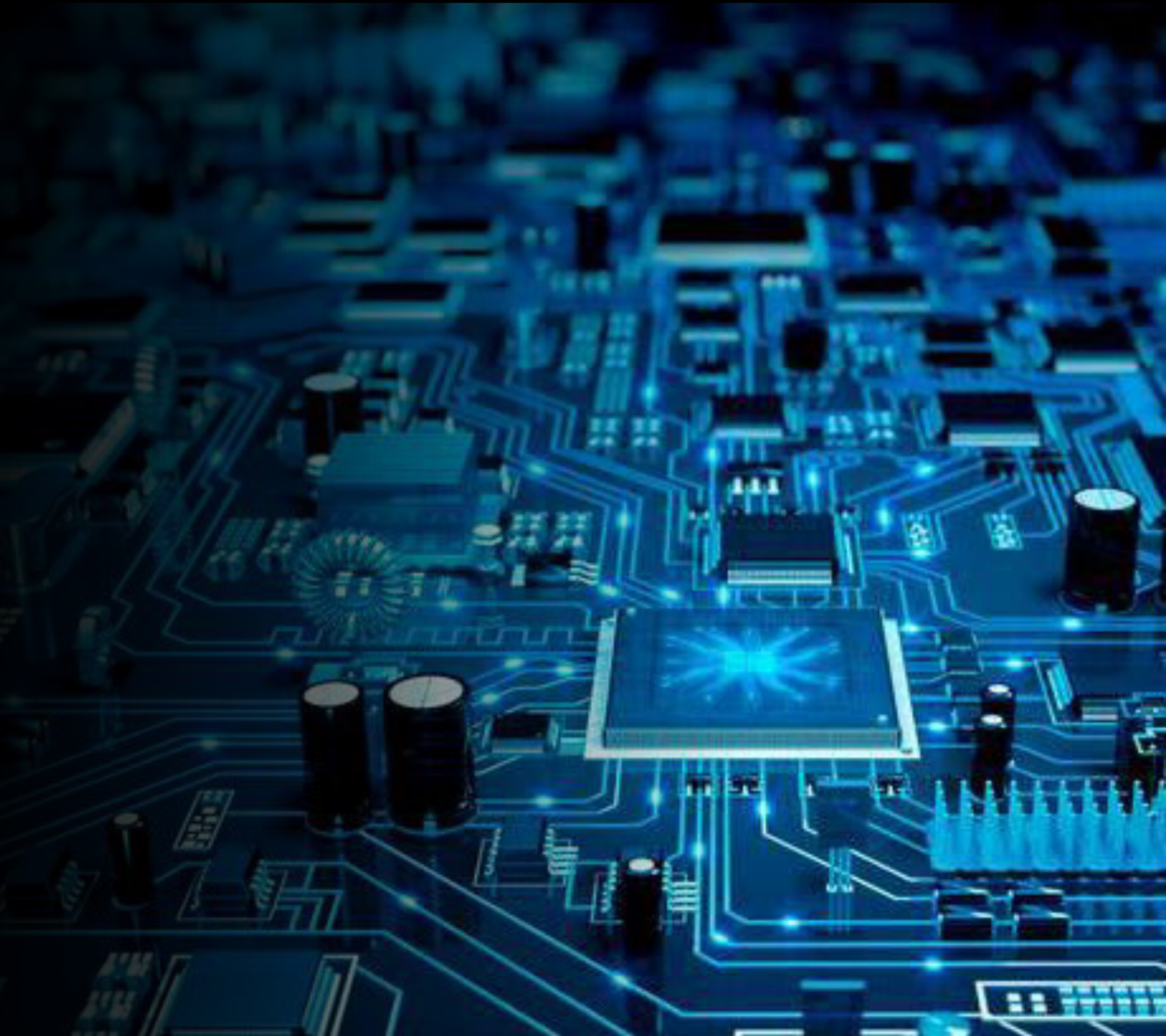
OPERATING SYSTEM OVERVIEW

CHAPTER ONE



INTRODUCTION

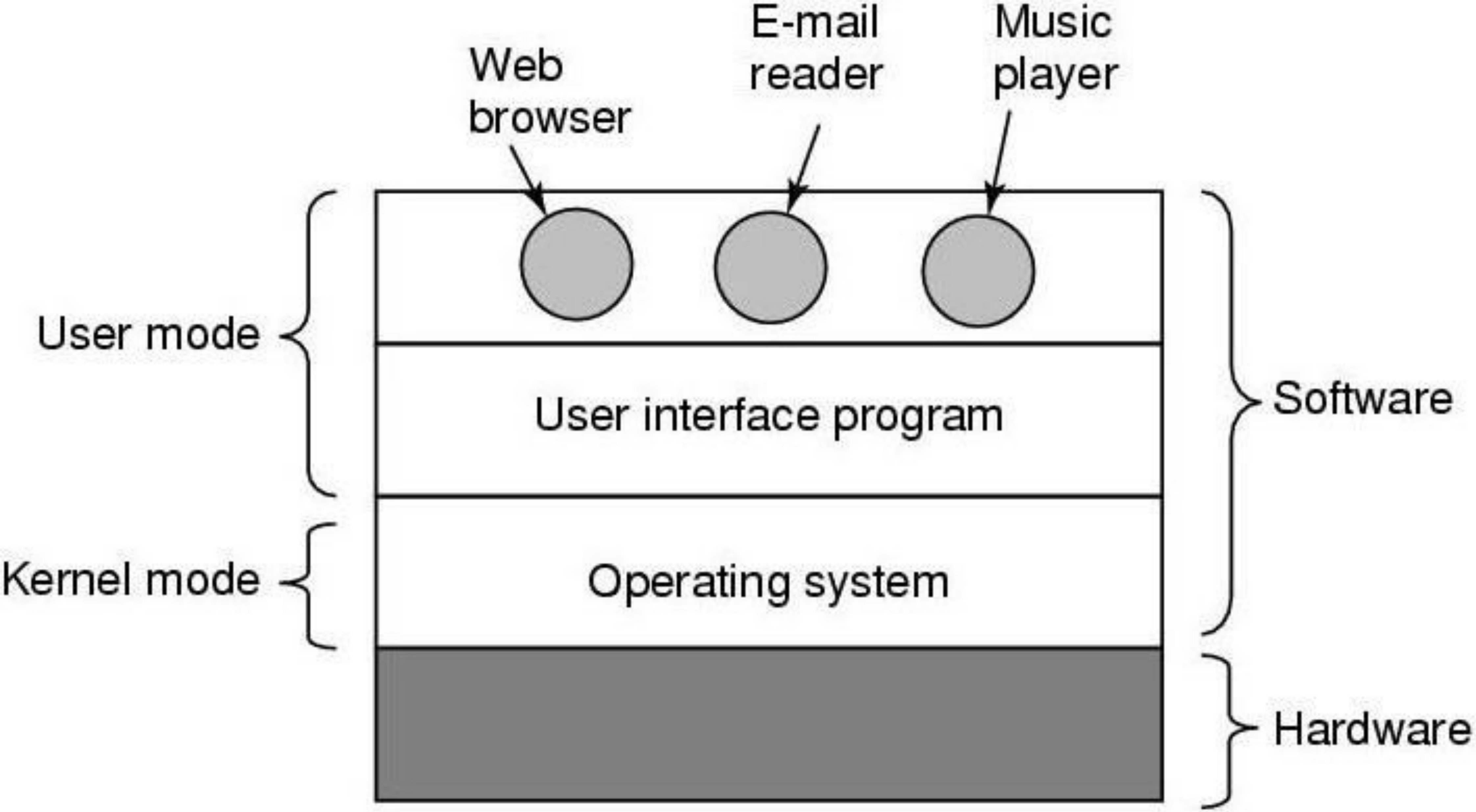
- Modern Computer consist of various devices such as processor, main memory, mouse, display, network interface and other input/output devices which forms a complex system.
- To manage all these components and use them optimally, computers are equipped with a layer of software called **Operating System.**
- Some common operating system are Windows, Linux, Mac OS.

- In computing, a shell is a piece of software that provides an interface for users and provides access to the services of a kernel.
- Operating system shells generally fall into one of two categories: command-line and graphical.
- Command-line shells provide a command-line interface (CLI) to the operating system, while graphical shells provide a graphical user interface (GUI).
- In either category the primary purpose of the shell is to invoke or "launch" another program; however, shells frequently have additional capabilities such as viewing the contents of directories.

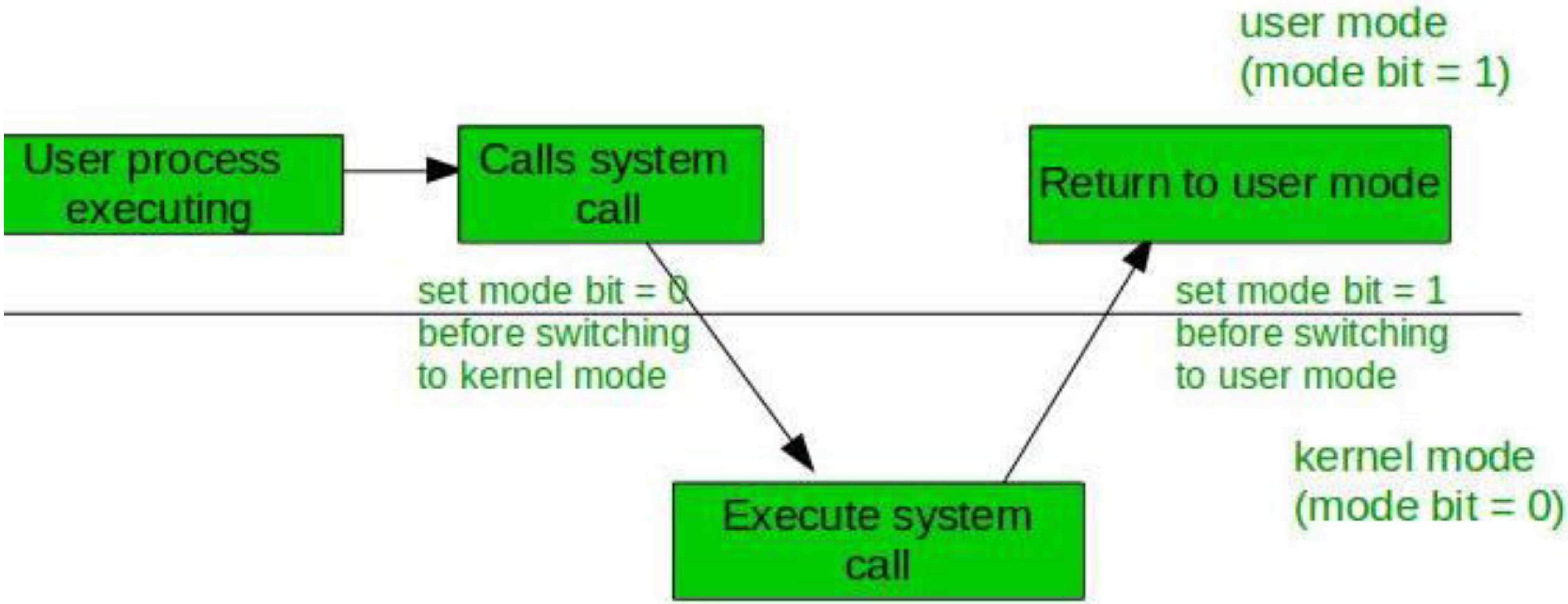
INTRODUCTION

- In computing, a **SHELL** is a user interface for access to an operating system's services.
- In general, operating system shells use either a command-line interface (CLI) or graphical user interface (GUI), depending on a computer's role and particular operation. It is named a shell because it is the outermost layer around the operating system.

[https://en.wikipedia.org/wiki/Shell_(computing)]


Web browser
E-mail reader
Music player
User mode
Kernel mode
User interface program
Operating system
Software
Hardware

- User Mode: When user application is running then the system is in user mode.
- Kernel Mode: When user application request for a service from the operating system or system call, then there will be a transition from the user mode to kernel mode.


user mode
(mode bit = 1)
User process executing
Calls system call
Return to user mode
set mode bit = 0
before switching
to kernel mode
set mode bit = 1
before switching
to user mode
kernel mode
(mode bit = 0)
Execute system call

Two Views of Operating System

- User View
- System View

USER VIEW

- If the user is using a personal computer, the operating system is largely designed to make the interaction easy.
- If the user is using a system connected to a mainframe or a minicomputer, the operating system is largely concerned with resource utilization.
- If the user is sitting on a workstation connected to other workstations through networks, then the operating system needs to focus on both individual usage of resources and sharing though the network.
- If the user is using a handheld computer such as a mobile, then the operating system handles the usability of the device including a few remote operations.

SYSTEM VIEW

- A resource allocator.
- A control program.
- easily communicate with the hardware.
- handling all the application programs.

First Generation (Vaccuum Tubes 1945-55)

- The first computers used vacuum tubes for circuitry and magnetic drums for memory, and were often enormous, taking up entire rooms. First generation computers relied on machine language to perform operations, and they could only solve one problem at a time.

- The Mark-I, EDSAC, EDVAC, UNIVAC-I and ENIAC computers are examples of first-generation computing devices.

Second Generation(Transistors and batch system 1955-65)

- Transistors replaced vacuum tubes allowing computers to become smaller, faster, cheaper, more energy-efficient and more reliable than their first-generation predecessors.
- Still relied on punched cards for input and printouts for output.
- Second-generation computers moved from cryptic binary machine language to symbolic, or assembly, languages, which allowed programmers to specify instructions in words.
- High-level programming languages like COBOL and FORTRAN were used.

Third Generation(1964-1971 Integrated Circuits)

- Integrated circuit was used.
- Transistors were miniaturized and placed on silicon chips, called semiconductors, which increased the speed and efficiency of computers.
- Instead of punched cards and printouts, users interacted through keyboards and monitors and interfaced with an operating system, which allowed the device to run many different applications at one time with a central program that monitored the memory.
- Computers for the first time became accessible to a mass audience because they were smaller and cheaper than their predecessors.

Fourth Generation(Microprocessor 1980-present)

- Microprocessor were used
- In 1981 IBM introduced its first computer for the home user, and in 1984 Apple introduced the Macintosh.
- As these small computers became more powerful, they could be linked together to form networks, which eventually led to the development of the Internet.
- Fourth generation computers also saw the development of GUIs, the mouse and Hand held devices.

FIFTH GENERATION COMPUTER(AI)

- Fifth generation computing devices, based on artificial intelligence, are still in development, though there are some applications, such as voice recognition, that are being used today.
- The goal of fifth-generation computing is to develop devices that respond to natural language input and are capable of learning and self-organization.

TYPES OF OPERATING SYSTEM

- Mainframe Operating System
- Server Operating System
- Multiprocessor Operating System
- Personal Computer Operating System
- Real-Time Operating System
- Embedded Operating System
- Smart Card Operating System

Mainframe Operating System

- Processing many jobs at once, most of which need prodigious amount of I/O.
- Room sized computers with 1000 disks and millions of gigabyte of data.
- Typically offers three kinds of services: Batch, transaction processing and time sharing.

Server Operating System

- Run on server which are either very large personal computers, workstation, or even mainframe.
- Serve multiple user at once over a network and allow the users to share hardware and software resources.
- Provide print service, file service, or web service.

Multiprocessor Operating System

Two or more central processing units (CPUs) control the functions of the computer.

Division of a task among several processors working simultaneously, so that the task is completed more quickly.

Personal Computer Operating System

Supports multiprogramming.

Provide good support to a single user.

Widely used for word processing, spreadsheets and internt access.

Handheld Computer Operating System

- Handheld computer or PDA(Personal Digital Assistant) is a small computer that fits in a pocket and performs a small number of function.
- Symbian OS

Embedded Operating System

- Embedded System run on the computer that control devices that are not generally thought of as computers and which do not accept user installed software.
- Microwave ovens

Sensor Node Operating System

- Tiny computers that communicate with each other and with a base station using wireless communication.
- Small battery-powered computers with build-in radios.
- Each sensor node is a real computer with a CPU, RAM, ROM and one or more environment sensors.
- Event driven, responding to external events or making measurements periodically based on an internal clock.

Real Time Operating System

- System are characterized by having time as a key parameter.
  - Hard real-time system
    - Must provide absolute guarantees that a certain action will occur by a certain time.
  - Soft real-time system
    - Missing an occasional deadline is acceptable and does not cause any damage.

Smart Card Operating System

- Smallest operating system run on smart cards, which are credit card sized devices containing CPU chip.
- Have very severe processing power and memory constraint.

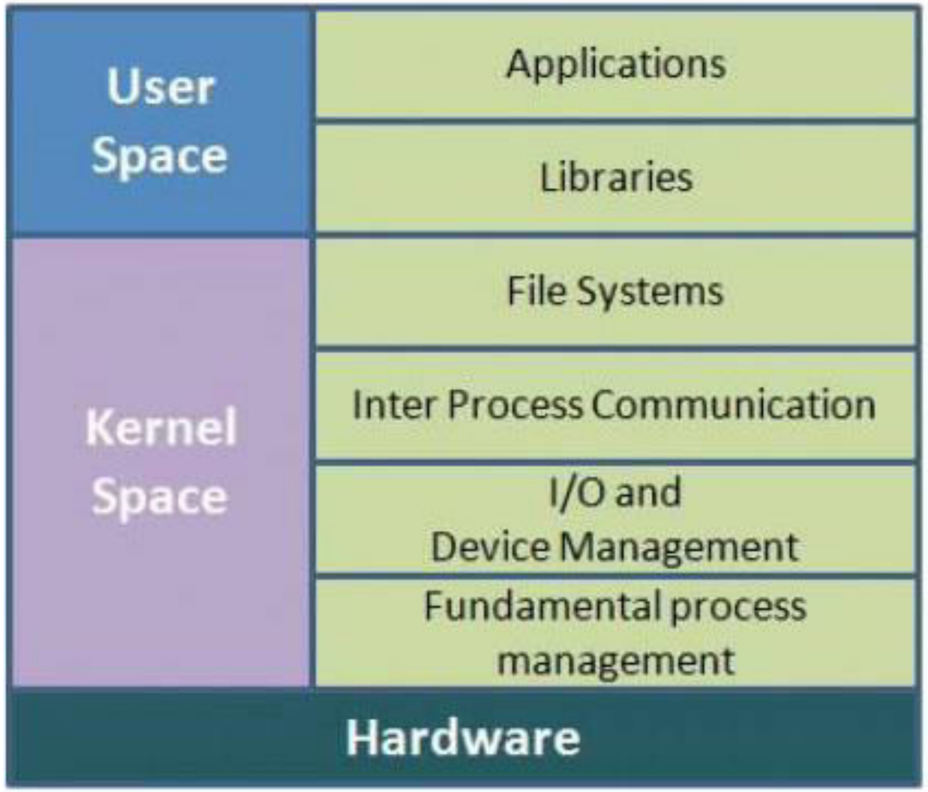
Operating System Structure

- Monolithic Systems
- Layered Systems
- Microkernels
- Client-server Model
- Virtual Machine
- Exokernel

Monolithic Systems

- Entire operating system runs as a single program in kernel mode.

Monolithic Kernel



User Space

Applications

Libraries

Kernel Space

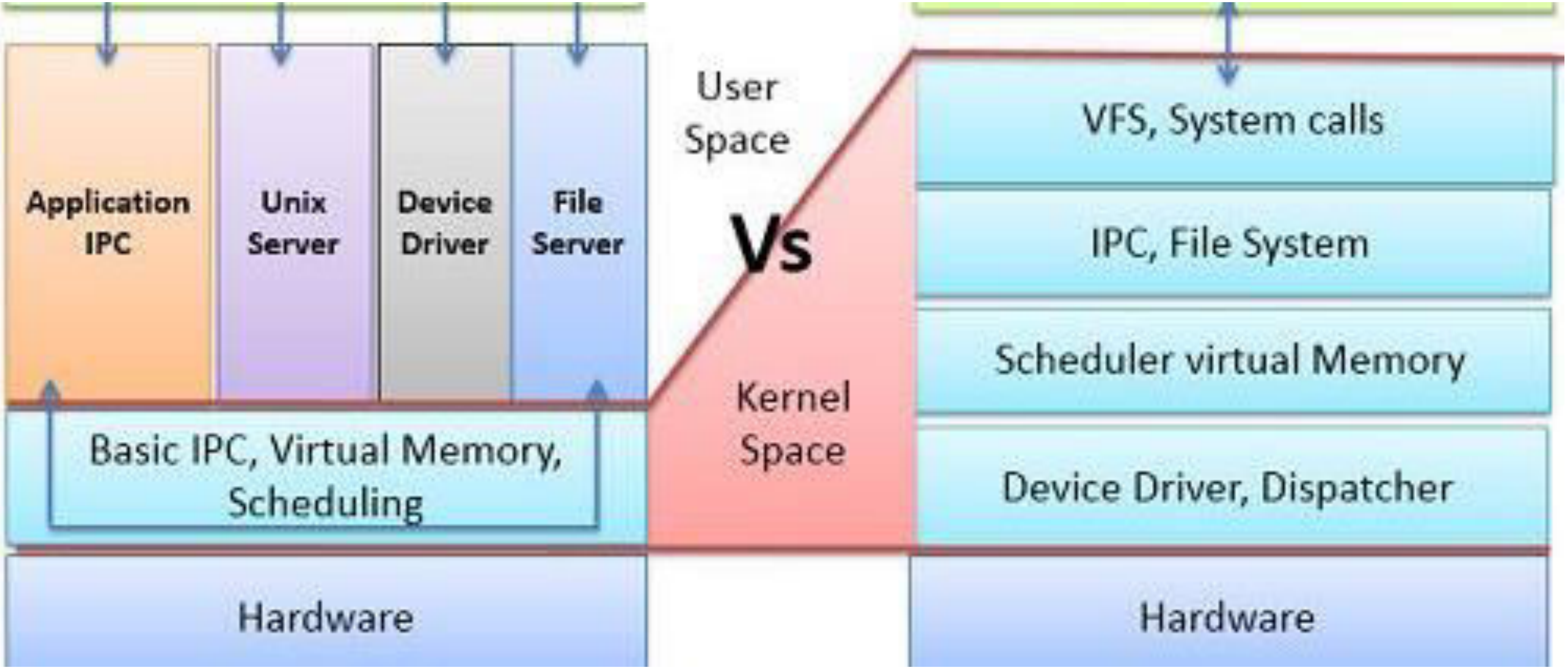
File Systems

Inter Process Communication

I/O and Device Management

Fundamental process management

Hardware


Application IPC
Unix Server
Device Driver
File Server
User Space
Vs
Basic IPC, Virtual Memory, Scheduling
Kernel Space
Hardware
VFS, System calls
IPC, File System
Scheduler virtual Memory
Device Driver, Dispatcher
Hardware

Monolithic System

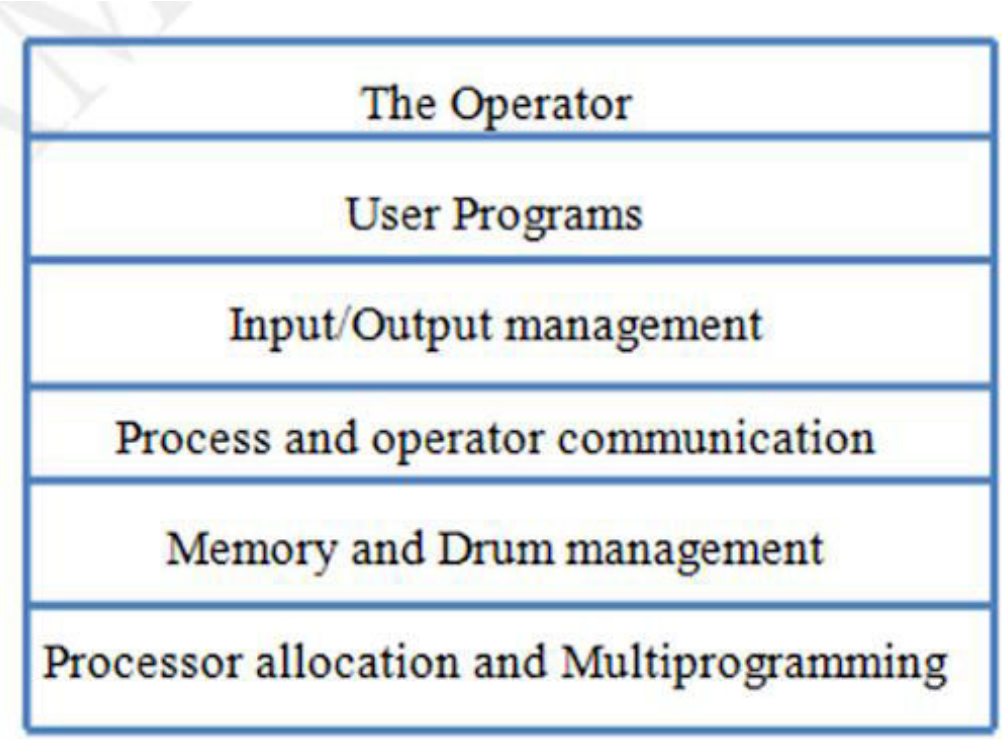
- Written as a collection of procedures, each of which can call any of the other ones whenever it needs to.

- Each procedure is free to call any other one, if the latter provides some useful computation than the former needs.

- In terms of information hiding, there is essentially none – every procedure is visible to every other procedure.

Layered Operating System

- To generalize the monolithic system into the hierarchy of layers, the first system called THE was built by E.W. Dijkstra in 1986.
- Operating system is divided into number of layers, each one constructed upon the one below it.
- Each layer uses functions (operations) and services of only lower-level layers



The Operator

User Programs

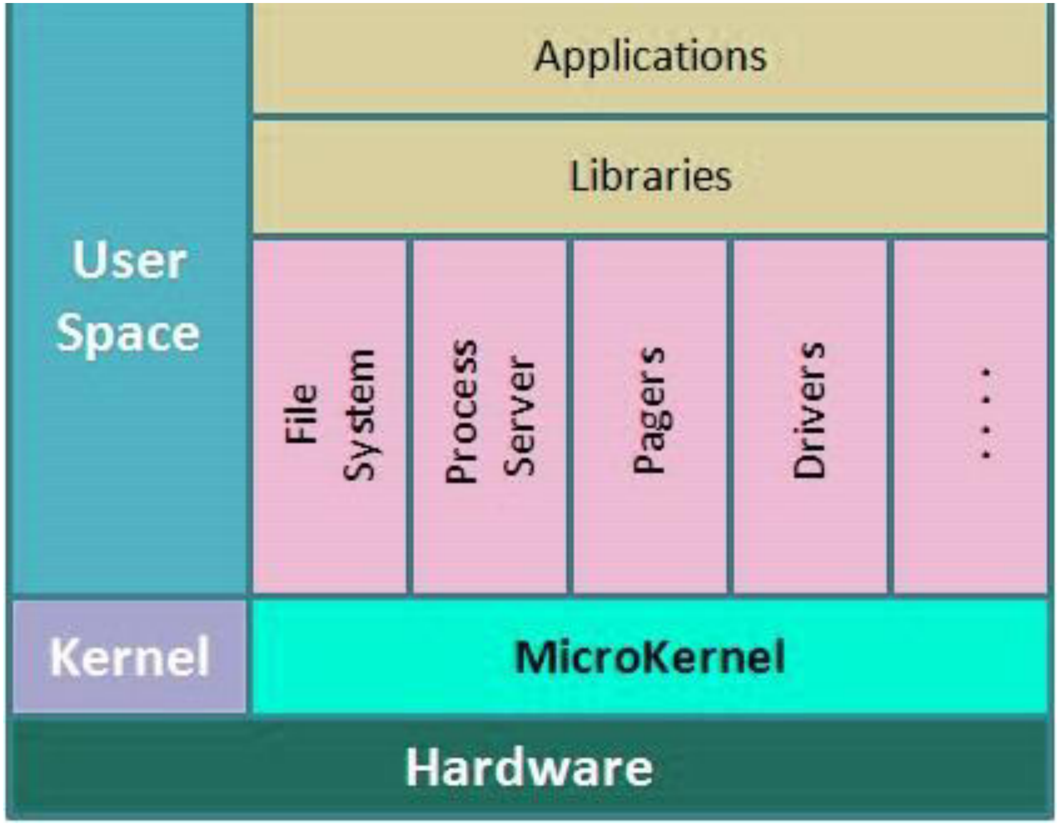
Input/Output management

Process and operator communication

Memory and Drum management

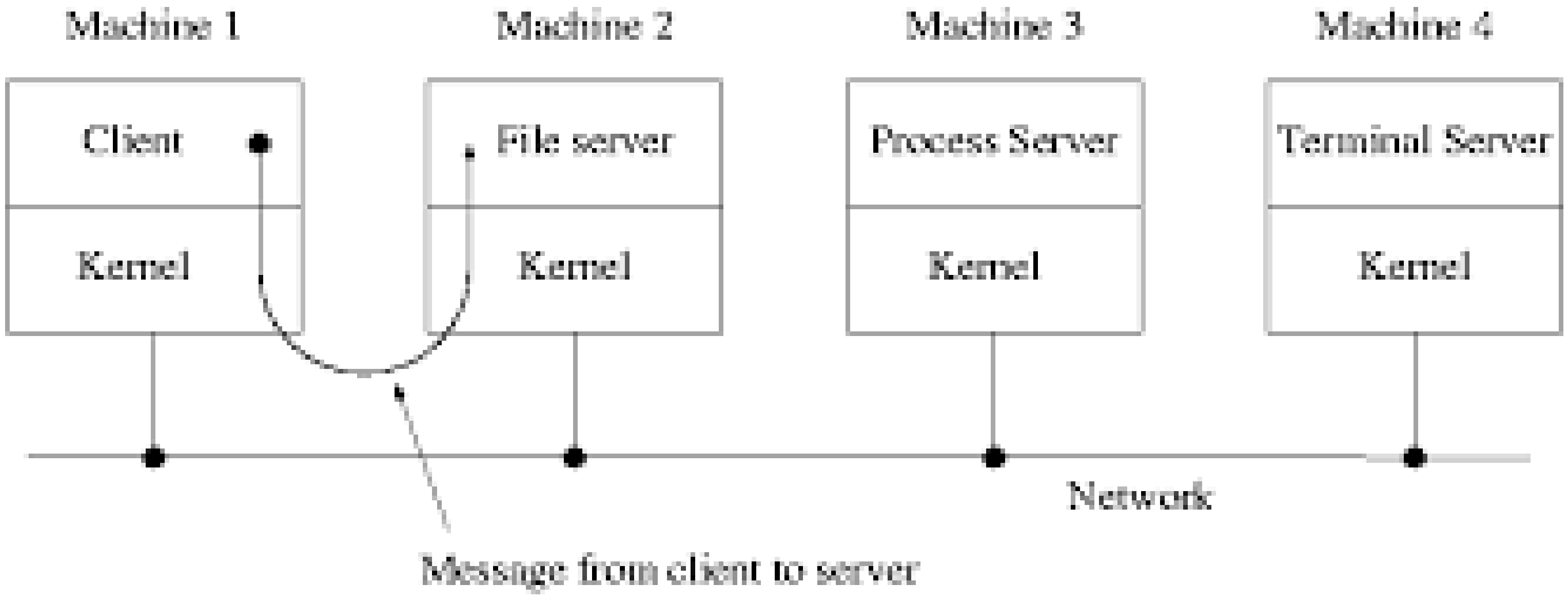
Processor allocation and Multiprogramming

Microkernel



Remove non-essential component from kernel and implement them as system level program.

Client program's request transferred to system program by microkernel called message passing.

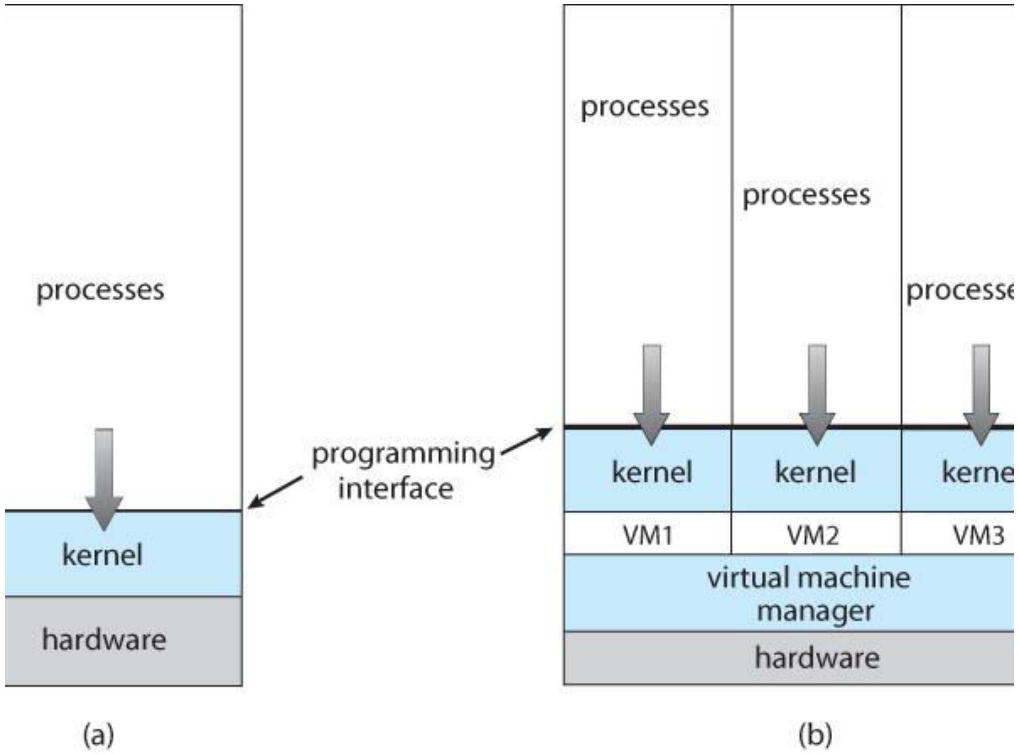


Client-Server Model

- Servers provides some service, and the clients uses these services.
- Often lowest layer is a microkernel, but that is not required.
- Presence of client processes and server processes

Virtual Machine

- Abstract the hardware of a single device into several different execution environment.
- The virtual machines are not the extended machines but are the exact copies of the bare hardware including kernel/user mode, I/O, interrupts and everything else the real machine has.



Exokernels

Partitioning virtual machine ie giving each user a subset of the resource.

At the bottom layer, running in kernel mode is a program called exokernel, its job is to allocate resource to virtual machine .

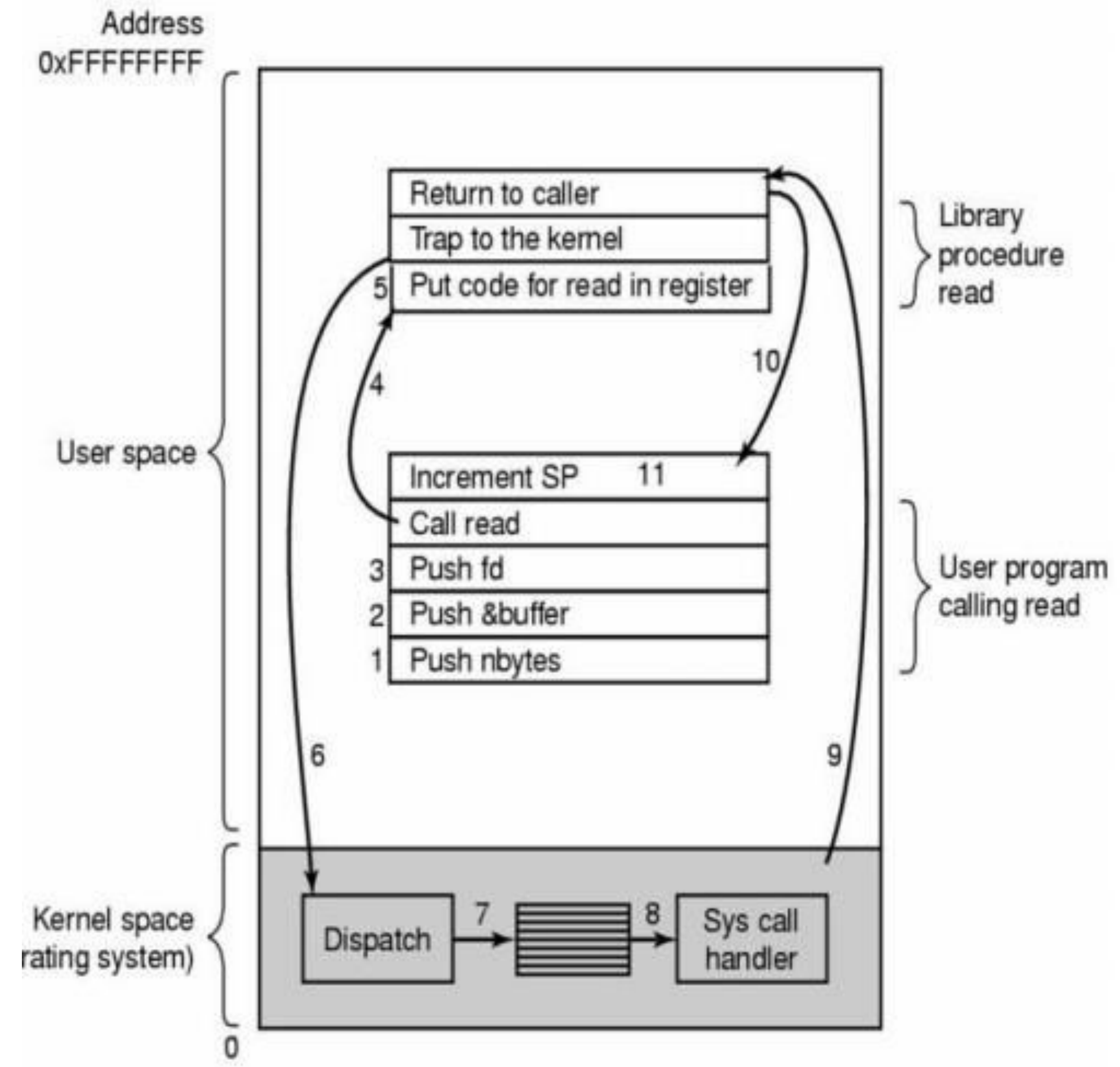
System Calls

- In computing, a system call is the mechanism used by an application program requests a service from an operating system's kernel.
- This may include hardware related services (e.g. accessing the hard disk), creating and executing new processes, and communicating with integral kernel services (like scheduling).

Steps in Making a System Call (example)

There are 11 steps in making the system call read (fd, buffer, nbytes)

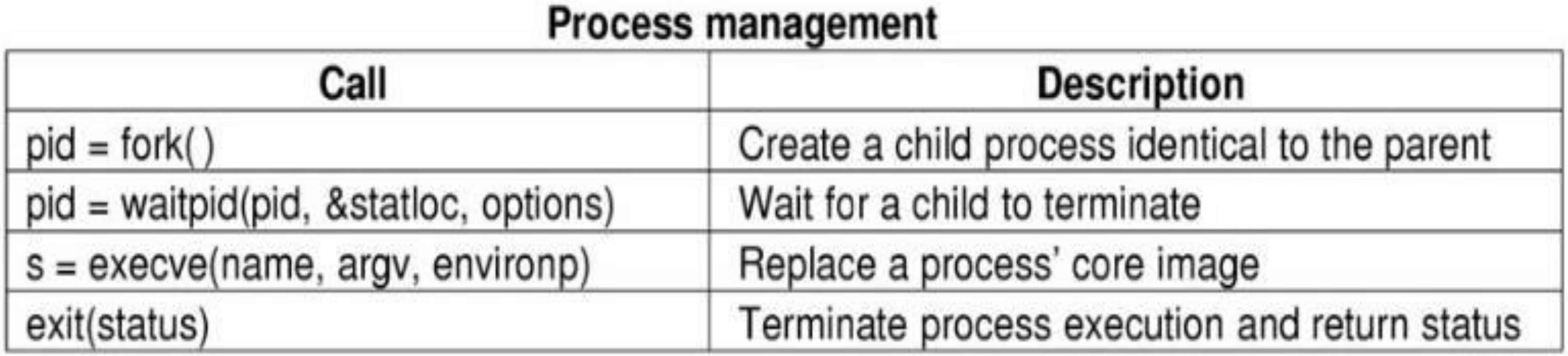
- Push parameter into the stack (1-3)
- Calls library procedure (4)
- Pass parameters in registers (5)
- Switch from user mode to kernel mode and start to execute (6)
- Examine the system call number and then dispatch to the correct system call handler via a table of pointer (7)
- Run System call Handlers (8)
- Once the system call handler completed its work, control return to the library procedure (9)
- This procedure then return to the user program in the usual way (10)
- Increment SP to clean up the stack before call to finish the job. (11)



SYSTEM CALLS

- System calls for process management
- System calls for file management
- System calls for directory management

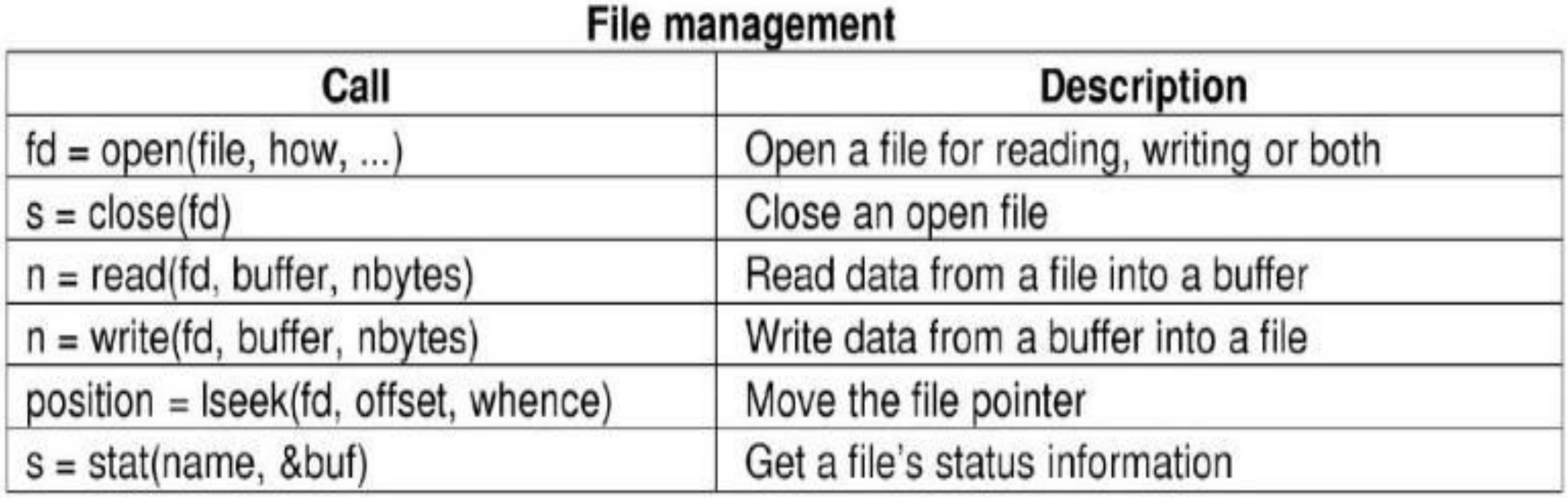
System calls for process management



Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

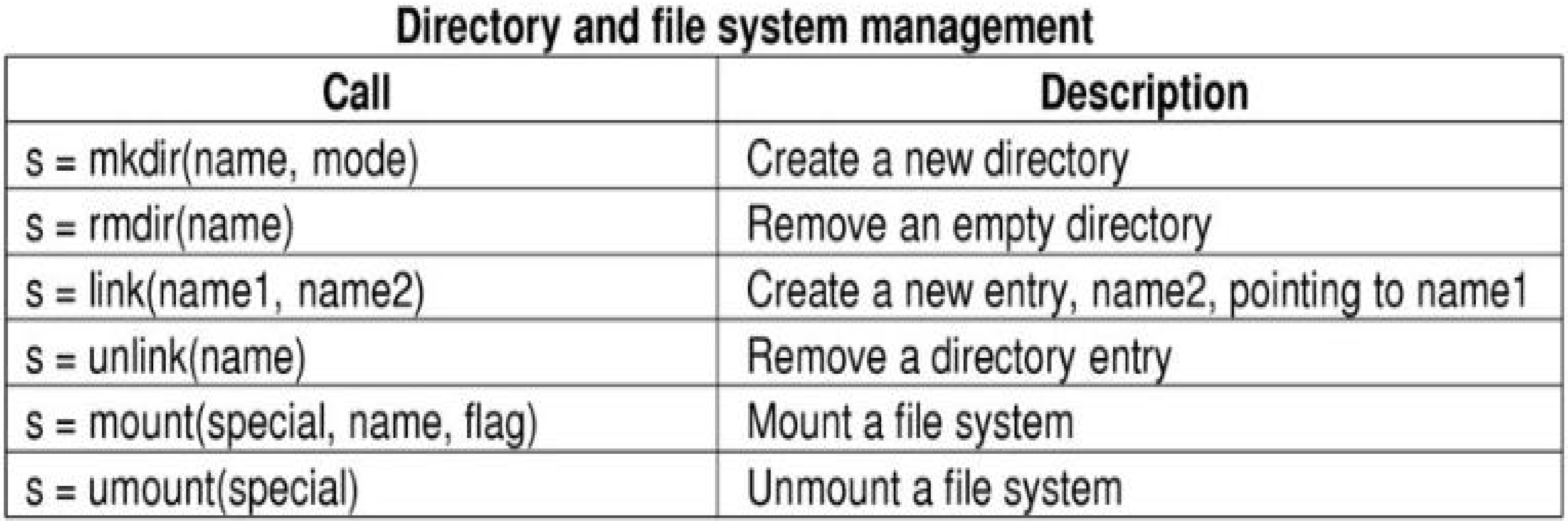
System calls for file management



File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

System calls for directory management



Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system