CO - 1

Session : 2

COURSE NAME : SYSTEM DESIGN AND INTRODUCTION TO CLOUD

COURSE CODE : 23AD2103A

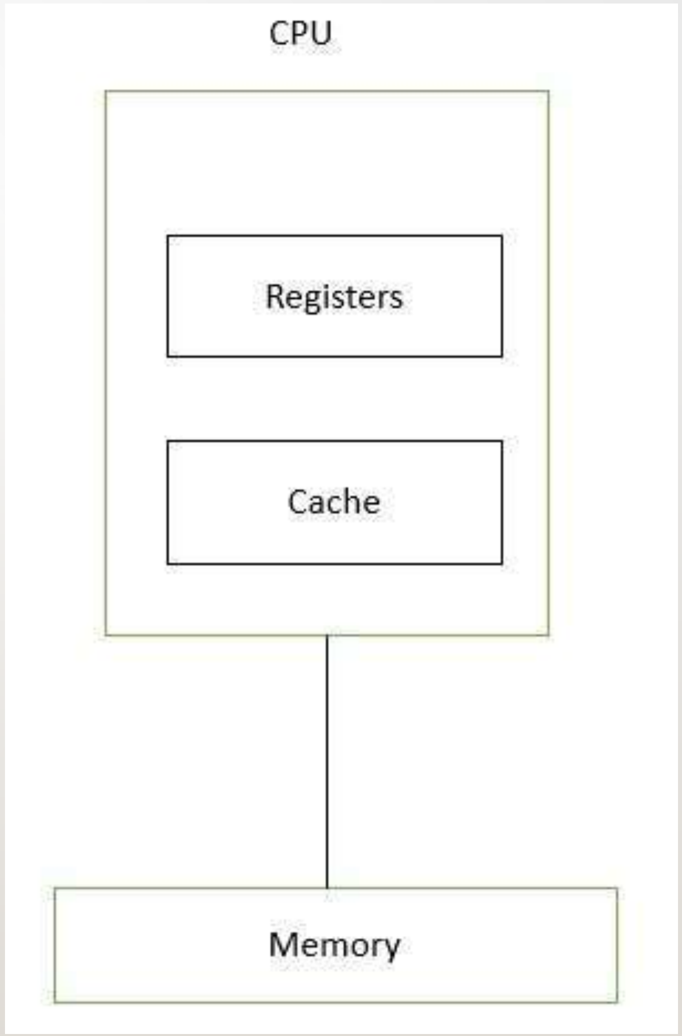
TOPICS : COMPUTER ARCHITECTURE SUPPORT TO OPERATING SYSTEMS

SESSION DESCRIPTION

- What is the computer architecture that supports the operating system?
- Computer Architecture Basics
- Generic computer architecture
- Architectural features motivated by OS services

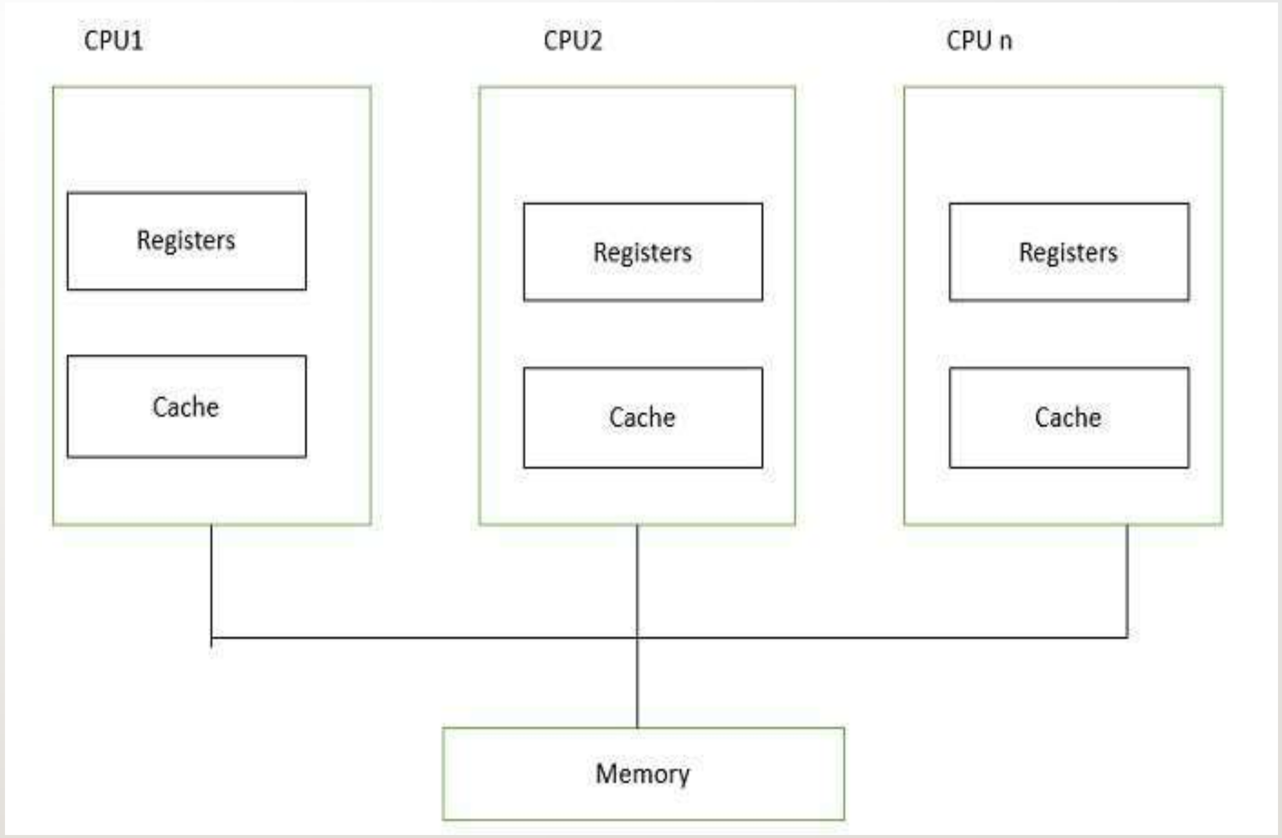
WHAT IS THE COMPUTER ARCHITECTURE THAT SUPPORTS THE OPERATING SYSTEM?

- There are different operating systems for different kinds of computers and processors. They are divided into different categories.
- **Single processor system**
- Single CPU or processor that manages the computer and it runs on a different operating system, and performs a number of tasks using one processor called a single processor system.



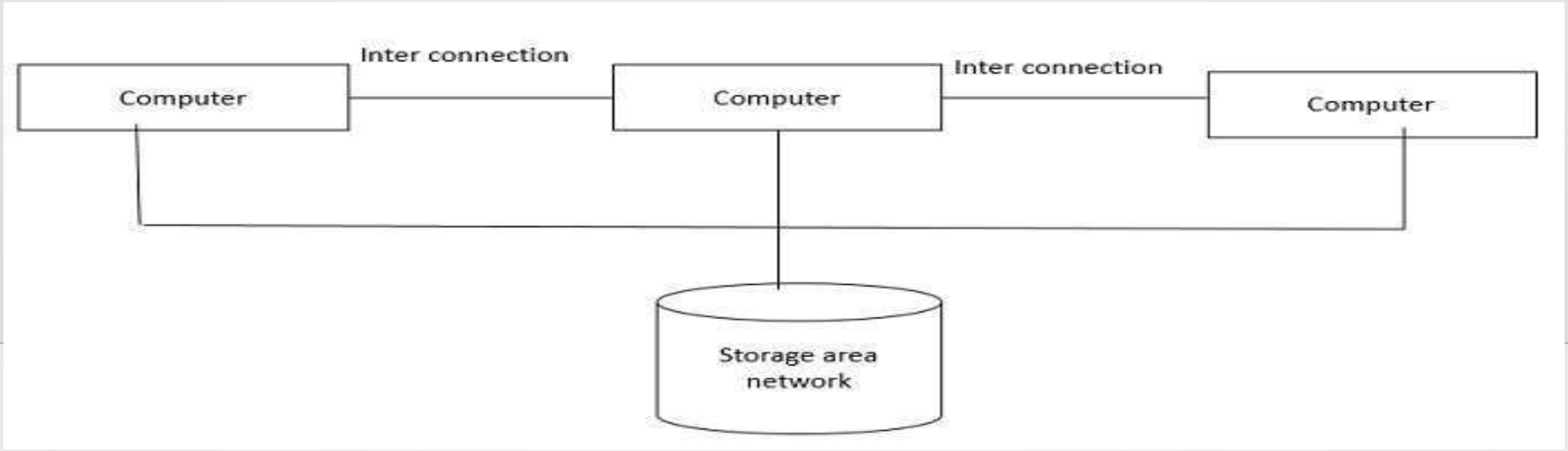
COMPUTER ARCHITECTURE THAT SUPPORTS THE OPERATING SYSTEM

- **Multiprocessor system**
- When one task can be performed by using multiple processors then it is called a multiprocessor system.
- In a multiprocessor system a number of processors are involved to perform a particular task so that the task can be performed easily.
- In a multiprocessor system the task is divided into sub tasks and these sub tasks are executed by using different processors.
- The multiprocessor system is used for time calculations, arithmetic and logic calculation, memory calculation, error detection, CPU system.

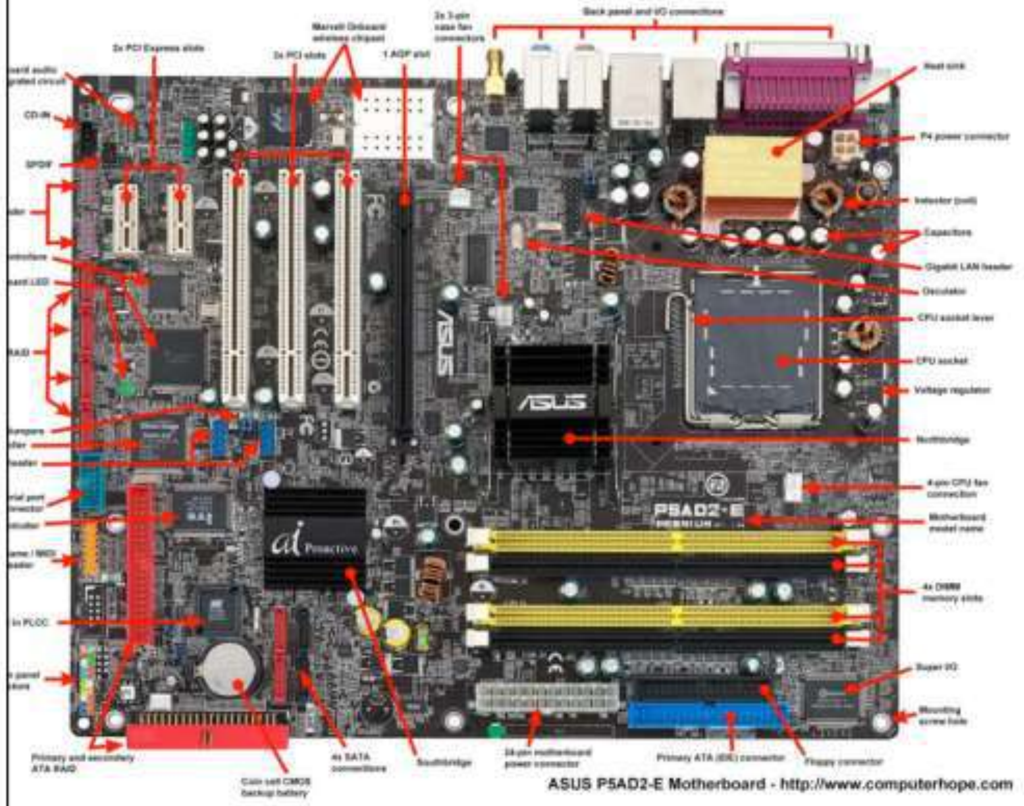


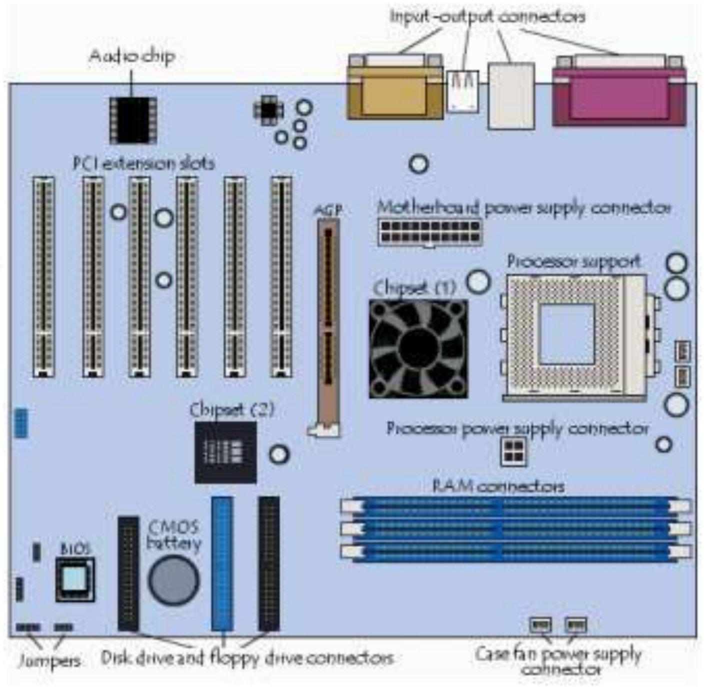
COMPUTER ARCHITECTURE THAT SUPPORTS THE OPERATING SYSTEM

- **Cluster processor system**
- In cluster processor systems independent systems share common storage and are connected by a high speed internet connection so a number of computer systems are using a processor by the help of cluster systems.
- The cluster system is a type of network which can be established by the help of different computer processors and these processors are used to find out the result of a particular task.
- In a cluster system the user will be able to use a network and by the help of the network the user connects with a number of processors or a processor connects with a number of users.



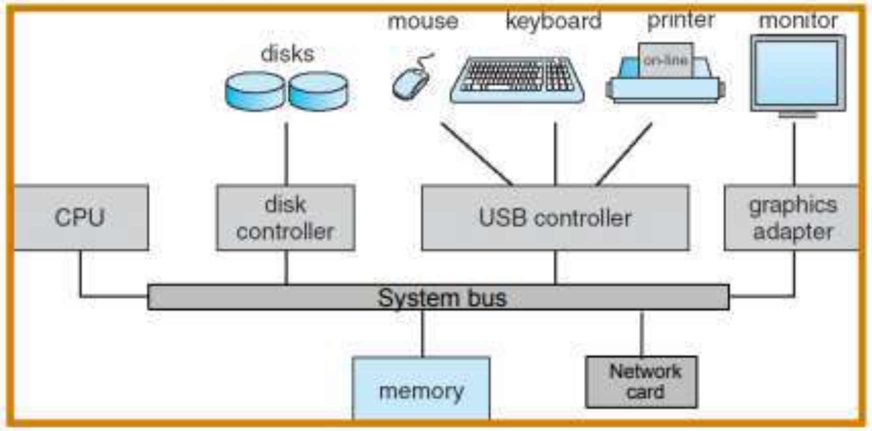
Computer Architecture Basics




Input-output connectors
Audio chip
PCI extension slots
AGP
Motherboard power supply connector
Processor support
Chipset (1)
Chipset (2)
Processor power supply connector
RAM connectors
BIOS
CMOS battery
Jumpers
Disk drive and floppy drive connectors
Case fan power supply connector

- Picture of a motherboard/logicboard

Generic Computer Architecture



- **CPU:** the processor that performs the actual computation
  - Multiple "cores" common in today's processors
- **I/O devices:** terminal, disks, video board, printer, etc.
  - Network card is a key component, but also an I/O device
- **Memory:** RAM containing data and programs used by the CPU
- **System bus:** communication medium between CPU, memory, and peripherals

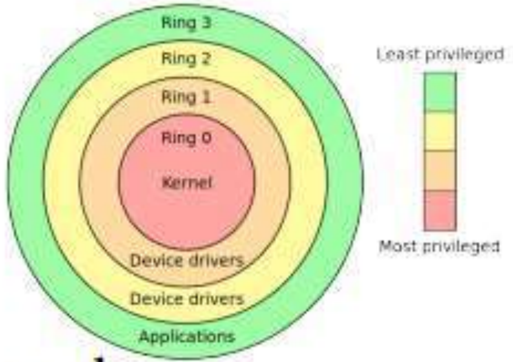
Architectural Features Motivated by OS Services

OS Service	Hardware Support
Protection	Kernel/user mode, protected instructions, base/limit registers
Interrupts	Interrupt vectors
System calls	Trap instructions and trap vectors
I/O	Interrupts and memory mapping
Scheduling, error recovery, accounting	Timer
Synchronization	Atomic instructions
Virtual memory	Translation look-aside buffers

Protection

Kernel mode vs. User mode: To protect the system from aberrant users and processors, some instructions are restricted to use only by the OS. Users may not

- address I/O directly
- use instructions that manipulate the state of memory (page table pointers, TLB load, etc.)
- set the mode bits that determine user or kernel mode
- disable and enable interrupts
- halt the machine



but in kernel mode, the OS can do all these things.

The hardware must support at least kernel and user mode.

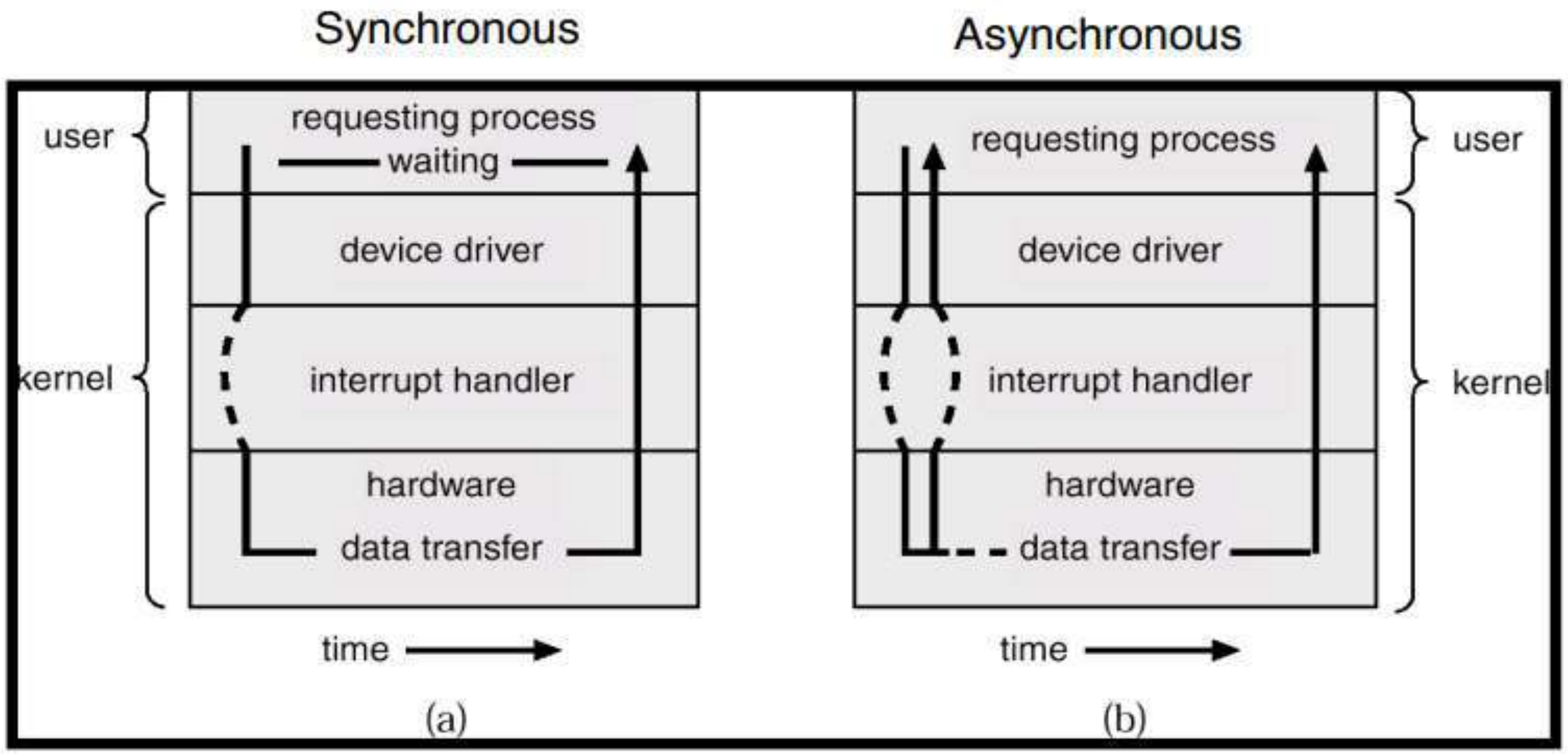
- A status bit in a protected processor register indicates the mode.
- Protected instructions can only be executed in kernel mode.

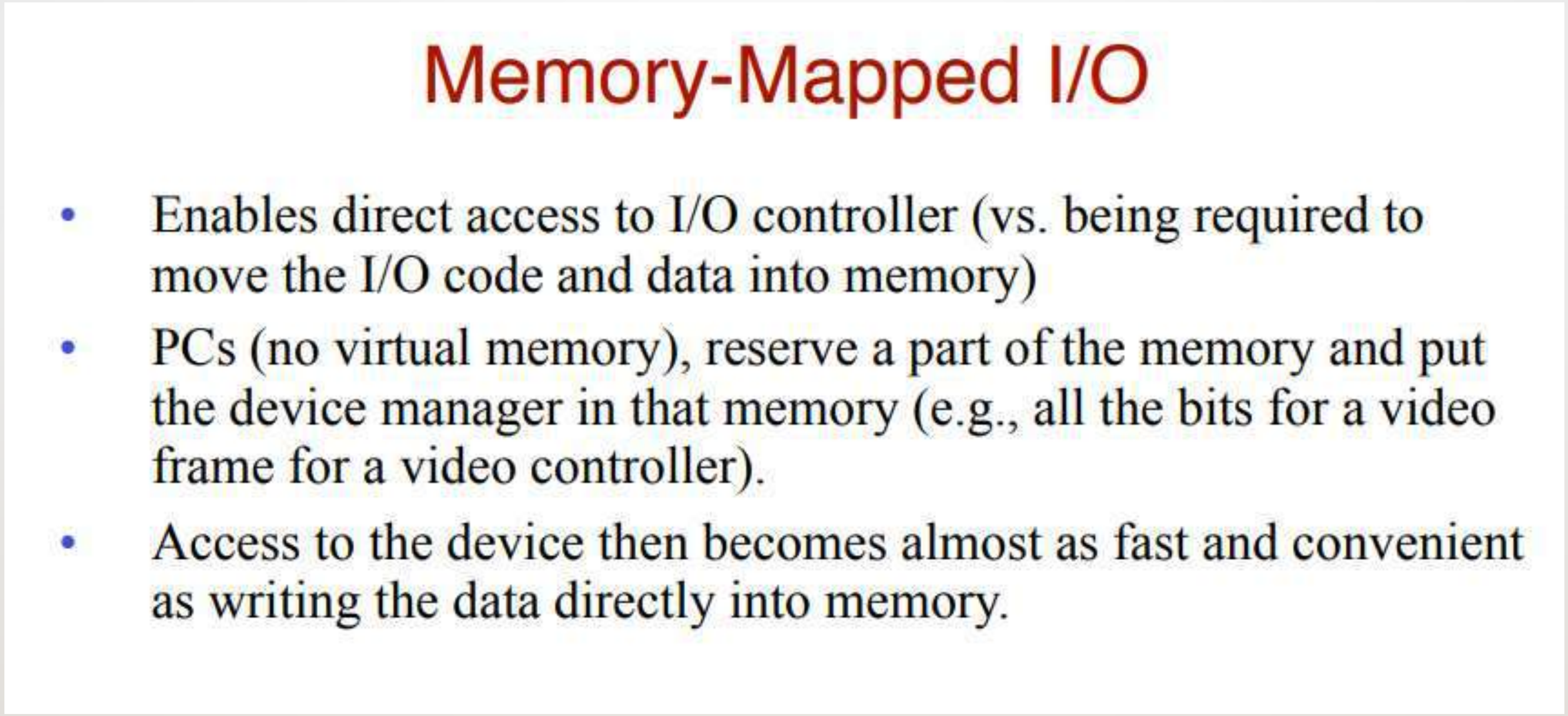
I/O Control

- Each I/O device has a little processor inside it that enables it to run autonomously.
- CPU issues commands to I/O devices, and continues
- When the I/0 device completes the command, it issues an interrupt
- CPU stops whatever it was doing and the OS processes the I/O device's interrupt

Three I/O Methods

- Synchronous, asynchronous, memory-mapped





Memory-Mapped I/O

- Enables direct access to I/O controller (vs. being required to move the I/O code and data into memory)
- PCs (no virtual memory), reserve a part of the memory and put the device manager in that memory (e.g., all the bits for a video frame for a video controller).
- Access to the device then becomes almost as fast and convenient as writing the data directly into memory.

Synchronization

- Interrupts interfere with executing processes.
- OS must be able to synchronize cooperating, concurrent processes.

→ Architecture must provide a guarantee that short sequences of instructions (e.g., read-modify write) execute atomically. Two solutions:

1. Architecture mechanism to disable interrupts before sequence, execute sequence, enable interrupts again.
2. A special instruction that executes atomically (e.g., test&set)

Virtual Memory

- Virtual memory allows users to run programs without loading the entire program in memory at once.
- Instead, pieces of the program are loaded as they are needed.
- The OS must keep track of which pieces are in which parts of physical memory and which pieces are on disk.
- In order for pieces of the program to be located and loaded without causing a major disruption to the program, the hardware provides a translation lookaside buffer to speed the lookup.

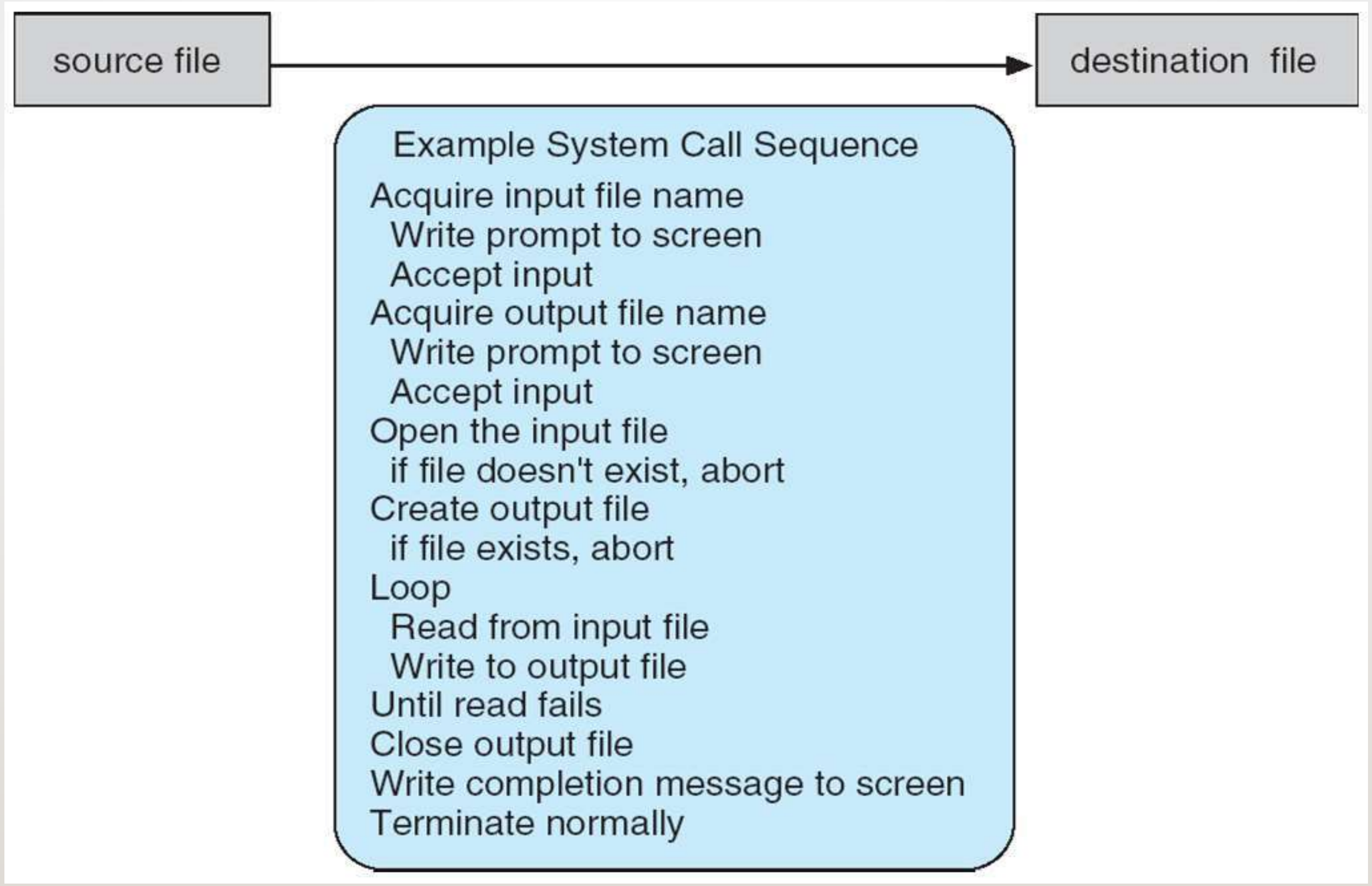
WHAT IS A SYSTEM CALL?

- A system call is a mechanism used by programs to request services from the operating system (OS). In simpler terms, it is a way for a program to interact with the underlying system, such as accessing hardware resources or performing privileged operations.

WHAT IS A SYSTEM CALL?

- A system call is a way for programs to **interact with the operating system**.
- System call **provides** the services of the operating system to the user programs via Application Program Interface(API).
- It provides an interface between a process and operating system to allow user-level processes to request services of the operating system.
- System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

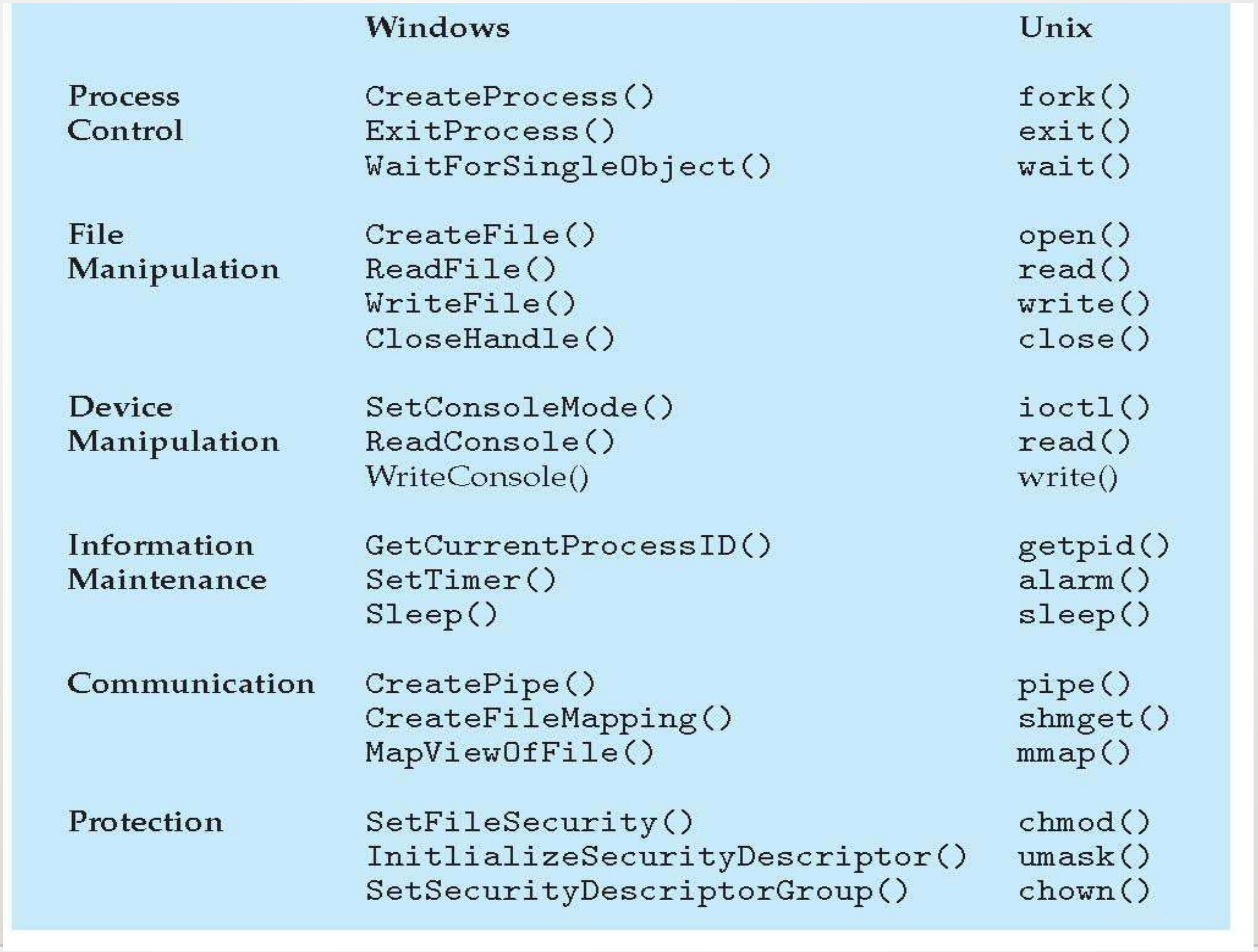
EXAMPLE OF SYSTEM CALLS



Example System Call Sequence

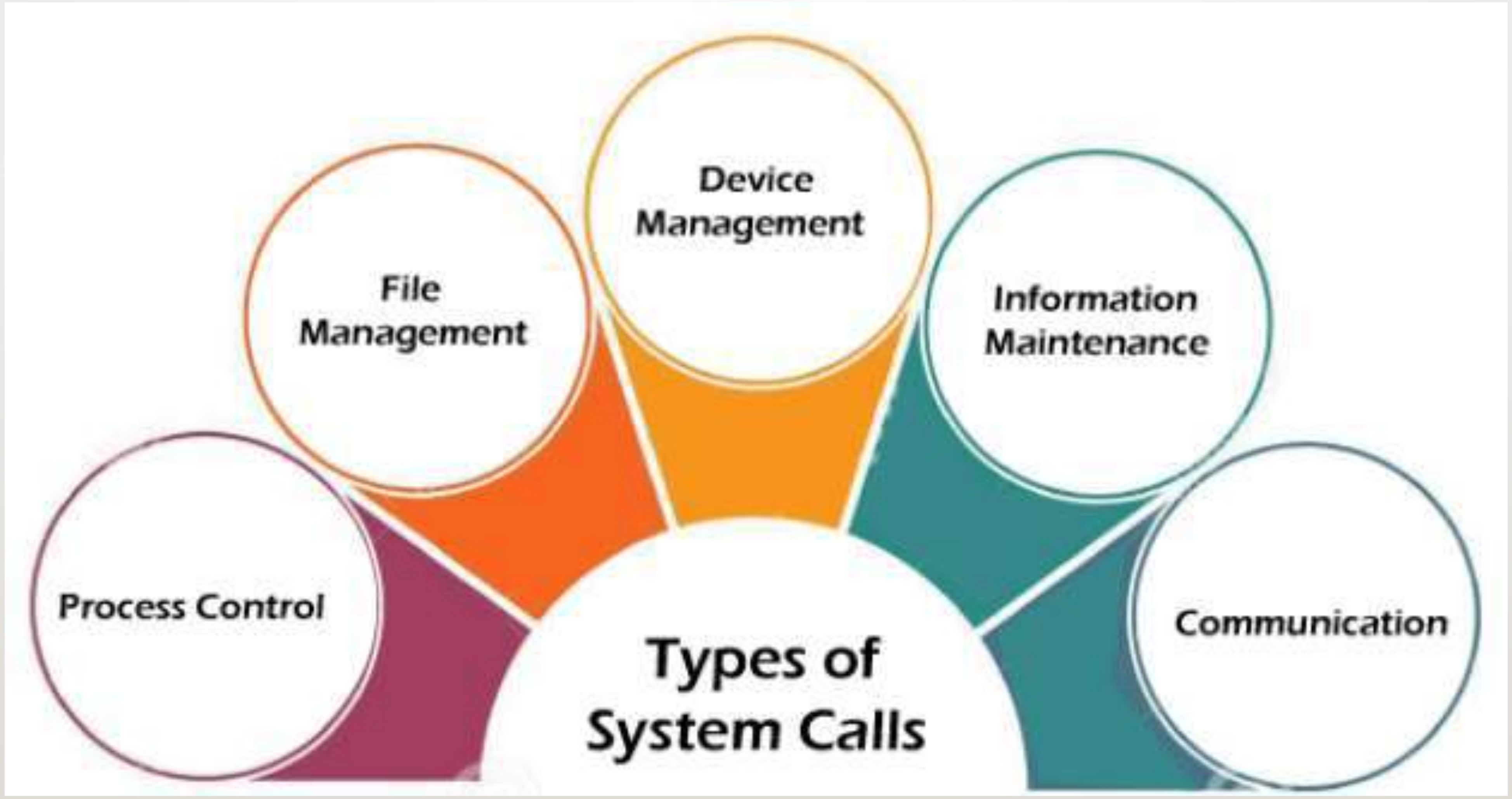
```
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS



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

TYPES OF SYSTEM CALLS



- **Process Control**

  Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

- **File Management**

  File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

- **Device Management**

  Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

- **Information Maintenance**

  Information maintenance is a system call that is used to maintain information.

- **Examples of information maintenance :**

  Including getting system data, set time or date, get time or date, set system data, etc.

- **Communication**

  Communication is a system call that is used for communication.

- **Examples of communication :**

  Including create, delete communication connections, send, receive messages, etc.

- **open()**

The **open()** system call allows you to access a file on a file system. It allocates resources to the file and provides a handle that the process may refer to. Many processes can open a file at once or by a single process only. It's all based on the file system and structure.

- **read()**

It is used to obtain data from a file on the file system. It accepts three arguments in general:

  - A file descriptor.
  - A buffer to store read data.
  - The number of bytes to read from the file.
  - The file descriptor of the file to be read could be used to identify it and open it using **open()** before reading.

- **wait()**

In some systems, a process may have to wait for another process to complete its execution before proceeding. The **wait()** system call is used to suspend the parent process. Once the child process has completed its execution, control is returned to the parent process.

- **write()**

It is used to write data from a user buffer to a device like a file. This system call is one way for a program to generate data. It takes three arguments in general:

  - A file descriptor.
  - A pointer to the buffer in which data is saved.
  - The number of bytes to be written from the buffer.

- **fork()**

Processes generate clones of themselves using the **fork()** system call. It is one of the most common ways to create processes in operating systems.

- **close()**

It is used to end file system access. When this system call is invoked, it signifies that the program no longer requires the file, and the buffers are flushed, the file information is altered, and the file resources are de-allocated as a result.

- **exec()**

When an executable file replaces an earlier executable file in already executing process, this system function is invoked.

- **exit()**

The **exit()** is a system call that is used to end program execution.

SERVICES PROVIDED BY SYSTEM CALLS

- Process Creation and Management
- Main Memory Management
- File Access, Directory, and File System Management
- Device Handling(I/O)
- Protection
- Networking, etc.
  - Process Control: end, abort, create, terminate, allocate, and free memory.
  - File Management: create, open, close, delete, read files,s, etc.
  - Device Management
  - Information Maintenance
  - Communication

THANK YOU

Team – System Design & Introduction to Cloud