Ch1 : Introduction

Operation System: A program that acts as an intermediary between a user of a computer and the computer hardware

**goals** of the operating system: -

* Execute user program.
* Make computer system convenient to use.
* Use the computer hardware in an efficient manner.

Computer system **components**: -

* Hardware: Provide basic computing resources
* CPU
* Memory
* I/O
* Operating system: controls the use of hardware among various applications and users.
* Application programs: solve computing problems for the users.
* Word processors
* Compilers
* Web browsers
* Database systems
* Video games
* Users
* People
* Machines
* Other computers

**Diagram** of Components of Computer: -

A diagram of a computer system

Description automatically generated

Users need: -

* Convenience
* Ease of use
* Good performance

Mobile user interface (UI): -

* Touch screen
* Voice recognition

Notes: -

* The user doesn’t care about resource utilization.
* shared computers (Ex: mainframe or minicomputer) must keep all users happy.
* Dedicated system users (Ex: workstation) have dedicated resources but frequently use shared resources from servers.
* Mobile devices are resource-poor that are optimized for usability and battery life.
* embedded computers in devices and automobiles almost have no interface because it’s run without user intervention.
* Parallel execution of CPU and devices compete for memory cycles

Computer system organization: one or more CPUs connected to a bus providing access to shared memory.

Computer-System Operation: -

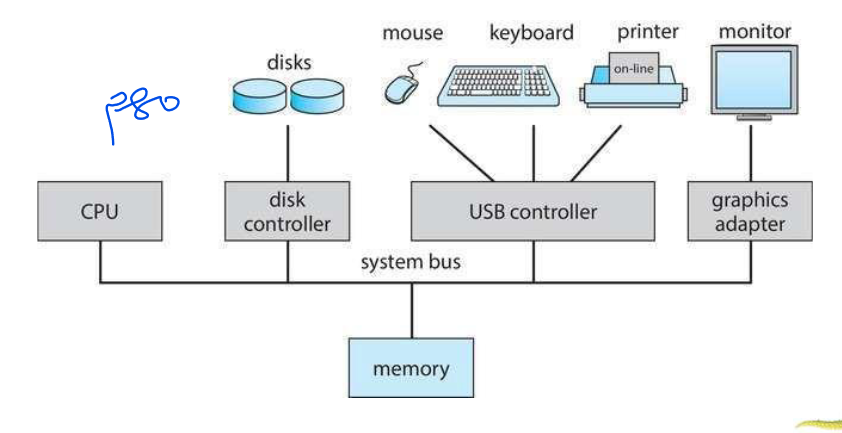
* I/O and CPU can execute concurrently, each device controller has a local buffer and a charge of a particular device type which has a device driver (OS) to manage it.
* CPU moves between data, main memory, and local buffer.
* I/O from the device to the local buffer of the controller.
* Device controller causes an interruption.

Bootstrap program: Simple code to initialize the system and load the **kernel**.

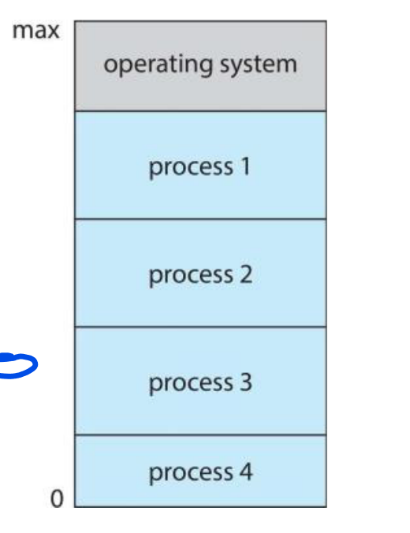
Kernal loads: -

* Start system daemons.
* Kernal interrupt driven.
* Hardware Interrupt
* Software Interrupt
* Software error.
* System call
* Other process problems.

Bus diagram: -



Multitasking: Logical extension of batch system.

Criteria:-

* Response time should be < 1 second.
* Each user has a process.
* several jobs ready to run at the same time.
* Swamping.
* Virtual memory.

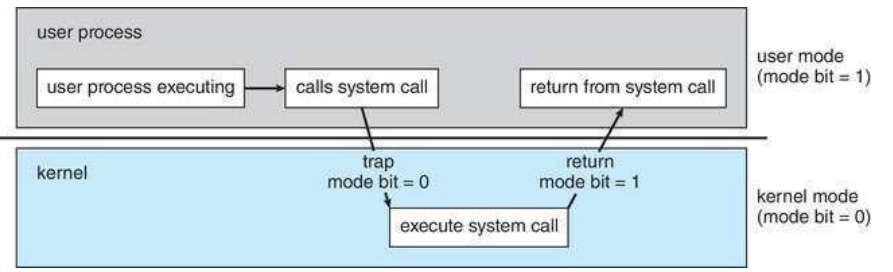
Dual mode: allows OS to protect itself and other

system components.

It has kernel mode and User mode.

Some instructions only execute on kernel mode.

System call: changes mode to kernel, return from call resets it to user.

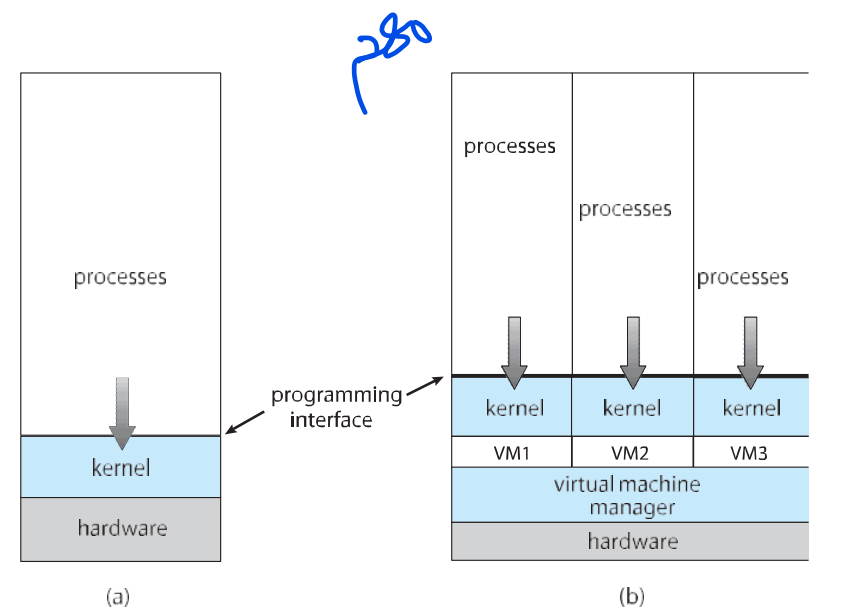


Virtualization: Allows operating systems to run applications within other OSes.

Emulation: used when source CPU type is different from target type. (Generally slowest method)

Interpretation: is used When computer language is not compiled to native code.

VMM (virtual machine Manager): provides virtualization services such as VirtualBox.



**What is the common approach for processors in most systems?**

Most systems typically use a single general-purpose processor.

**Are special-purpose processors common in systems?**

Yes, most systems also incorporate special-purpose processors.

**What is the trend in the use of multiprocessor systems?**

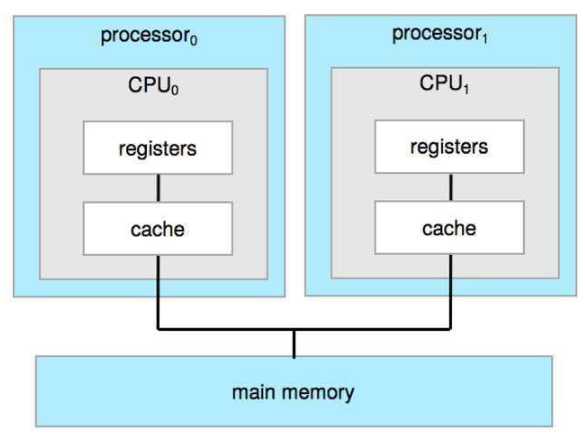
Multiprocessor systems, also known as parallel systems or tightly-coupled systems, are growing in both use and importance.

**What are the advantages of using multiprocessor systems?**

The advantages include increased throughput, economy of scale, and increased reliability through features like graceful degradation or fault tolerance.

**What are the two types of multiprocessor systems?**

The two types are Asymmetric Multiprocessing, where each processor is assigned a specific task, and Symmetric Multiprocessing, where each processor is capable of performing all tasks.



**What functions do operating systems provide in the execution environment for programs and services to users?**

Operating systems provide a range of services for the execution of programs and user interactions:

1. **User Interface (UI):**
   * Most operating systems include a User Interface (UI) which can take various forms such as Command-Line (CLI), Graphics User Interface (GUI), touch-screen, or Batch.
2. **Program Execution:**
   * Operating systems must load programs into memory, execute them, and handle termination, whether normal or abnormal (indicating errors).
3. **I/O Operations:**
   * Running programs may require Input/Output (I/O) operations involving files or devices.
4. **File-System Manipulation:**
   * Operating systems facilitate file-system operations, including reading and writing files, managing directories, searching, listing file information, and permission management.

**What additional functions do operating systems provide to assist users?**

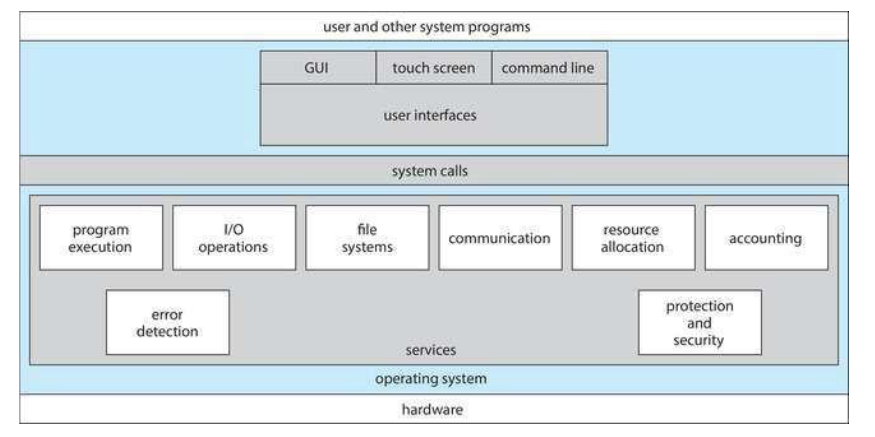
Continuing from the previous set of services, operating systems offer:

1. **Communications:**
   * Processes may exchange information, either on the same computer or between computers over a network, through shared memory or message passing facilitated by the OS.
2. **Error Detection:**
   * Operating systems are designed to detect and handle errors occurring in the CPU and memory hardware, I/O devices, or user programs. Debugging facilities enhance the user's and programmer's efficiency.

**What functions does the operating system perform to ensure the efficient operation of the system itself through resource sharing?**

In addition to user-centric services, operating systems perform functions related to the efficient operation of the system:

1. **Resource Allocation:**
   * When multiple users or jobs run concurrently, the OS allocates resources such as CPU cycles, main memory, file storage, and I/O devices.
2. **Logging:**
   * Operating systems maintain logs to track resource usage by users, keeping records of how much and what types of computer resources are utilized.
3. **Protection and Security:**
   * Operating systems ensure protection and security by controlling access to system resources. This involves user authentication and defending external I/O devices from unauthorized access attempts.



**What characterizes a general-purpose operating system (OS) in terms of its structure?**

A general-purpose OS can be structured in various ways:

1. **Simple Structure:**
   * Examples like MS-DOS feature a straightforward design.
2. **More Complex Structure:**
   * UNIX represents a more intricate design, offering a broader range of functionalities.
3. **Layered Structure:**
   * OS with a layered structure involves abstraction, organizing into layers with specific functions.
4. **Microkernel Structure:**
   * Operating systems like Mach adopt a minimal kernel, with additional functionalities implemented as separate modules.

**How was the original UNIX operating system structured, and what are its key components?**

The original UNIX operating system faced limitations imposed by hardware functionality, resulting in a relatively simple structure. The UNIX OS comprises two distinct parts:

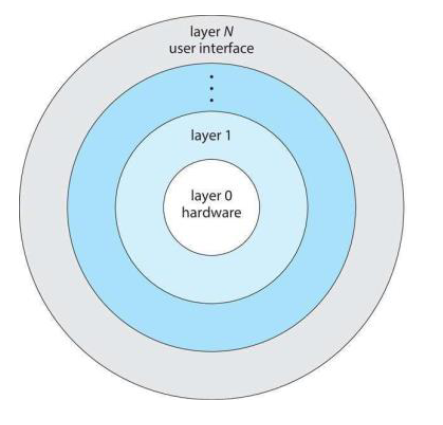
1. **Systems Programs:**
   * Programs designed to perform various system-related tasks.
2. **The Kernel:**
   * The kernel encompasses everything below the system-call interface and above the physical hardware.
   * It provides essential operating-system functions, including the file system, CPU scheduling, memory management, and others.
   * Despite its limited structuring, the UNIX kernel handles a substantial number of functions at a single level.

**How is the operating system structured in terms of layers, and what is the significance of this layering?**

The operating system is organized into multiple layers or levels, with each layer built upon lower layers. The layering is hierarchical, ranging from the hardware at the bottom (layer 0) to the user interface at the top (layer N).

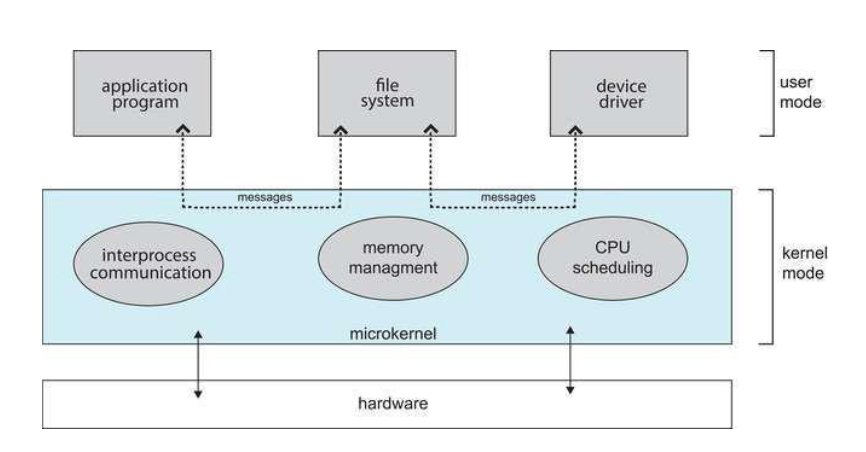
**Key Points:**

1. **Layer Hierarchy:**
   * Layer 0 represents the hardware, while layer N is the user interface.
2. **Modularity and Layer Selection:**
   * The modularity of the system is maintained by selecting layers in such a way that each layer uses only the functions, operations, and services of the lower-level layers.



**What characterizes a microkernel, using Mach as an example, and how does it impact the operating system?**

A microkernel, exemplified by Mach, involves moving significant functionality from the kernel to user space. Notably, the Mac OS X kernel (Darwin) is partially based on Mach. Key features of microkernels include:

1. **Communication via Message Passing:**
   * User modules communicate using message passing, enhancing modularity.
2. **Benefits of Microkernels:**
   * Easier extensibility, facilitating the addition of new features.
   * Enhanced portability to new architectures.
   * Increased reliability as less code operates in kernel mode.
   * Improved security measures.
3. **Drawbacks of Microkernels:**
   * Performance overhead due to communication between user space and kernel space.

**CH2 : Processes & Threads**

**How does an operating system handle the execution of programs, and what components constitute a process?**

An operating system is responsible for executing various programs, each running as a process. Key characteristics and components of a process include:

1. **Definition of a Process:**
   * A process is a program in execution, and its execution progresses sequentially. There is no parallel execution of instructions within a single process.
2. **Components of a Process:**
   * Various parts include the program code (text section), the current activity (program counter, processor registers), a stack (temporary data like function parameters, return addresses, and local variables), a data section (global variables), and a heap (memory dynamically allocated during runtime).
3. **Program vs. Process:**
   * A program is a passive entity stored on disk in an executable file, while a process is active. The program becomes a process when loaded into memory.
4. **Process Initialization:**
   * The execution of a program is initiated through GUI interactions, mouse clicks, command line entries, etc.
5. **Multiplicity of Processes:**
   * One program can give rise to several processes, particularly evident when multiple users execute the same program.

**How does the state of a process change during execution, and what are the different states a process can be in?**

As a process executes, it undergoes various states:

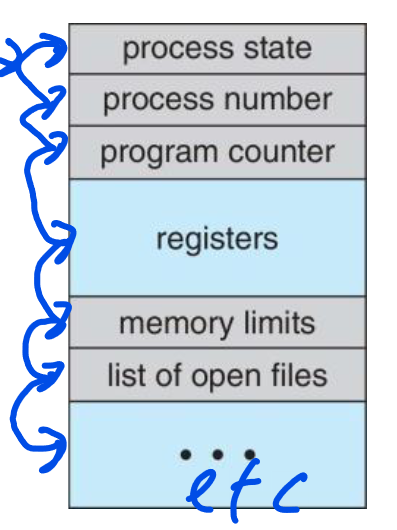
1. **New:**
   * The process is being created.
2. **Running:**
   * Instructions are actively being executed.
3. **Waiting:**
   * The process is in a state of anticipation, waiting for a specific event to occur.
4. **Ready:**
   * The process is prepared and waiting to be assigned to a processor for execution.
5. **Terminated:**
   * The process has completed its execution and is in a terminated state.

A diagram of a running process

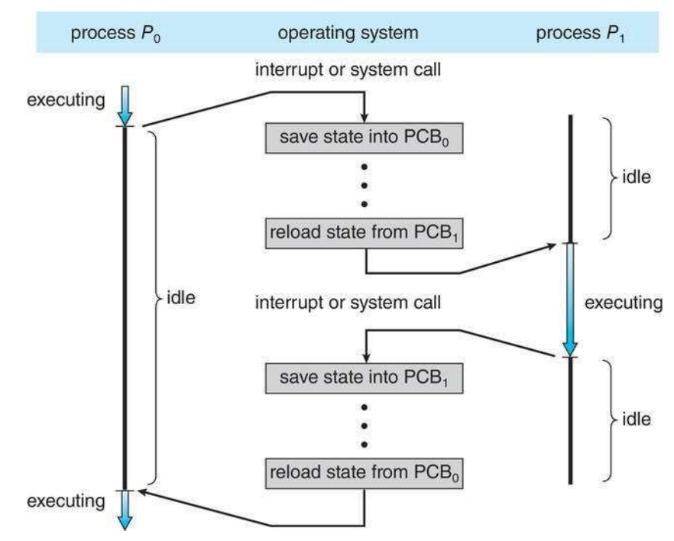
Description automatically generated

**What information is associated with each process in an operating system, typically stored in a task control block?**

The task control block (TCB) or process control block (PCB) contains crucial information for each process:

1. **Process State:**
   * Indicates the current state of the process, such as running, waiting, etc.
2. **Program Counter:**
   * Stores the location of the next instruction to be executed by the process.
3. **CPU Registers:**
   * Encompasses the contents of all process-centric registers.
4. **CPU Scheduling Information:**
   * Includes details like priorities and scheduling queue pointers, guiding the process's position in the scheduling queue.
5. **Memory-Management Information:**
   * Specifies the memory allocated to the process.
6. **Accounting Information:**
   * Records CPU usage, clock time elapsed since start, and time limits associated with the process.
7. **I/O Status Information:**
   * ****Enlists I/O devices allocated to the process and maintains a list of open files related to the process.

context switch occurs when the CPU switches from one process to another

****