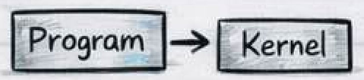
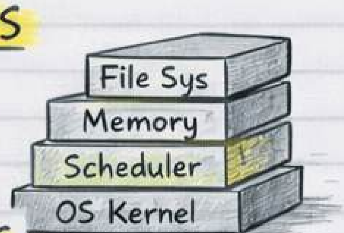
Operating System

Module-1 Notes

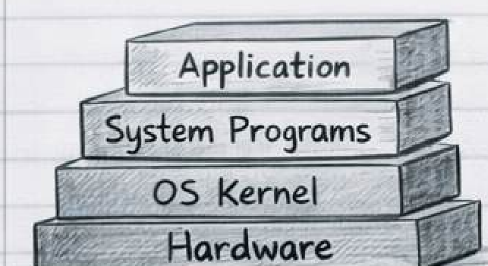
by pyqfort.com

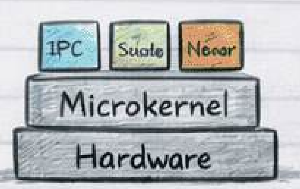
Contents Covered:

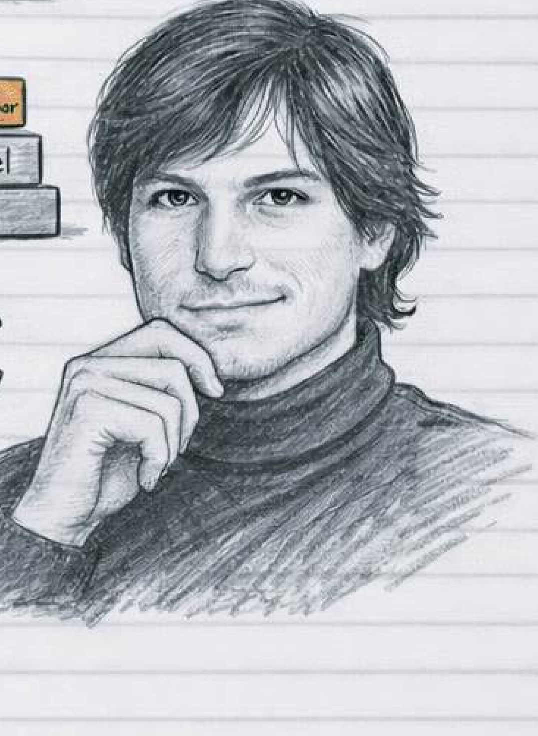
- Intro to OS 
- Types of OS 
- OS Services 
- User View & System View 
- System Calls 

- Monolithic Structure of OS
- Layered Structure of OS 

- Microkernel Structure of OS



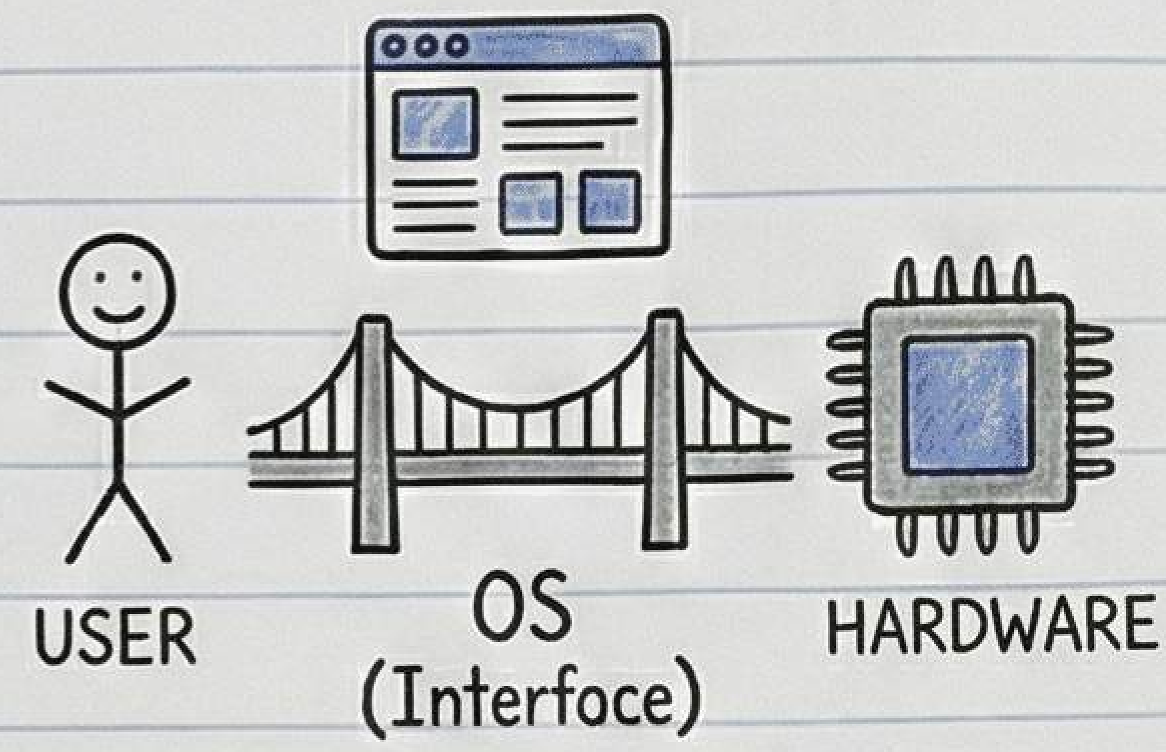




Module-1: Concept of Operating Systems

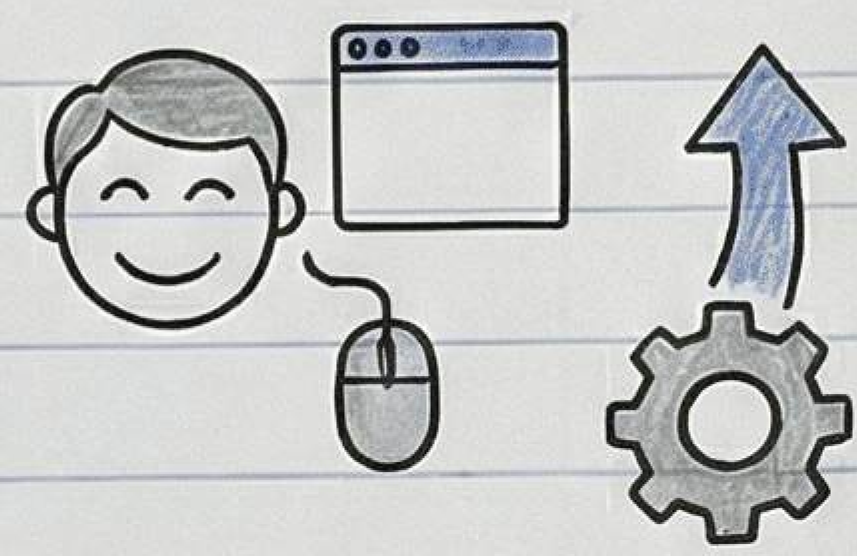
• What is an OS?

- An OS is a software acting as an interface between users and hardware.
- It provides a working environment for apps and is a resource manager.



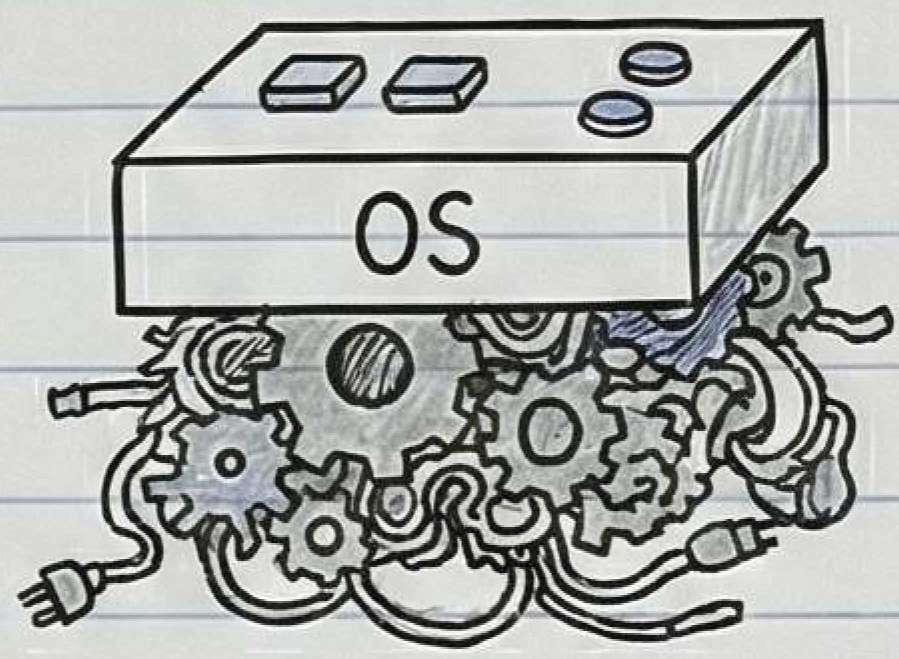
• Goals of an OS:

- Prime goal is user convenience (a friendly environment, like GUI).
- Users don't want to deal with hardware details.
- Another goal is efficient utilization of hardware resources.



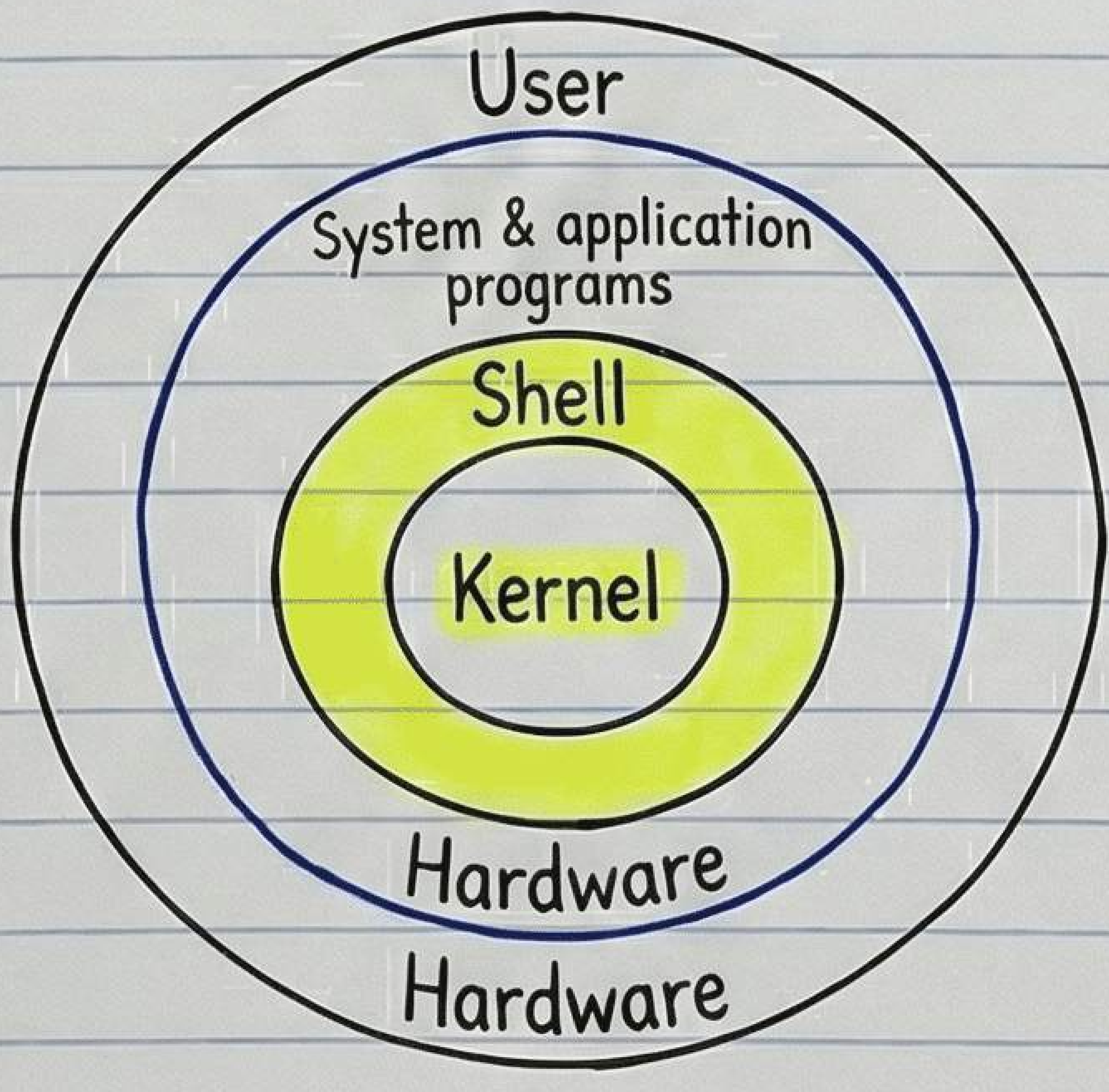
• OS as an Extended Machine:

- Acts as an extended machine manager.
- It hides hardware complexity, presenting a simpler machine to the user.



• Key Components & Structure:

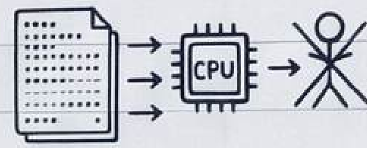
- Two generic components: Shell and Kernel.
- Shell: A command interpreter that reads user control statements.
- Kernel: Contains only the essential modules of the OS.



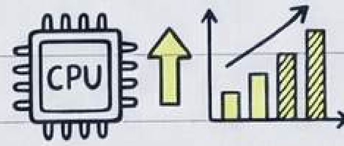
Types of Operating Systems

Batch Processing Systems

- Takes a sequence of jobs in a batch and executes them one by one without any intervention of the user.

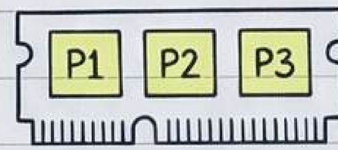


- The main advantage is to increase the CPU utilization.

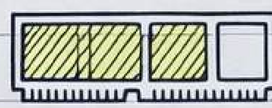


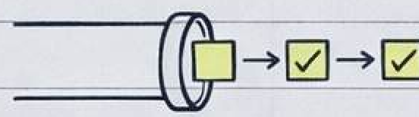
Multi-programming Systems

- A central concept: placing more than one program in the main memory.



- Major Benefits:
  * Less execution time
  * Increased utilization of memory
  * Increased throughput (jobs/time).

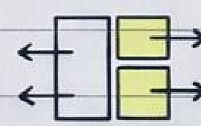


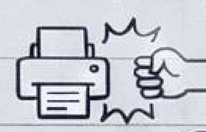


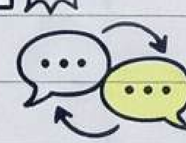
Problems & Solutions

Multi-programming originated many problems
Solutions led to new OS modules:

- Memory management (partitioning, allocation).
- Process protection (must be protected).
- Competing for limited I/O devices.
- Process communication & synchronization.



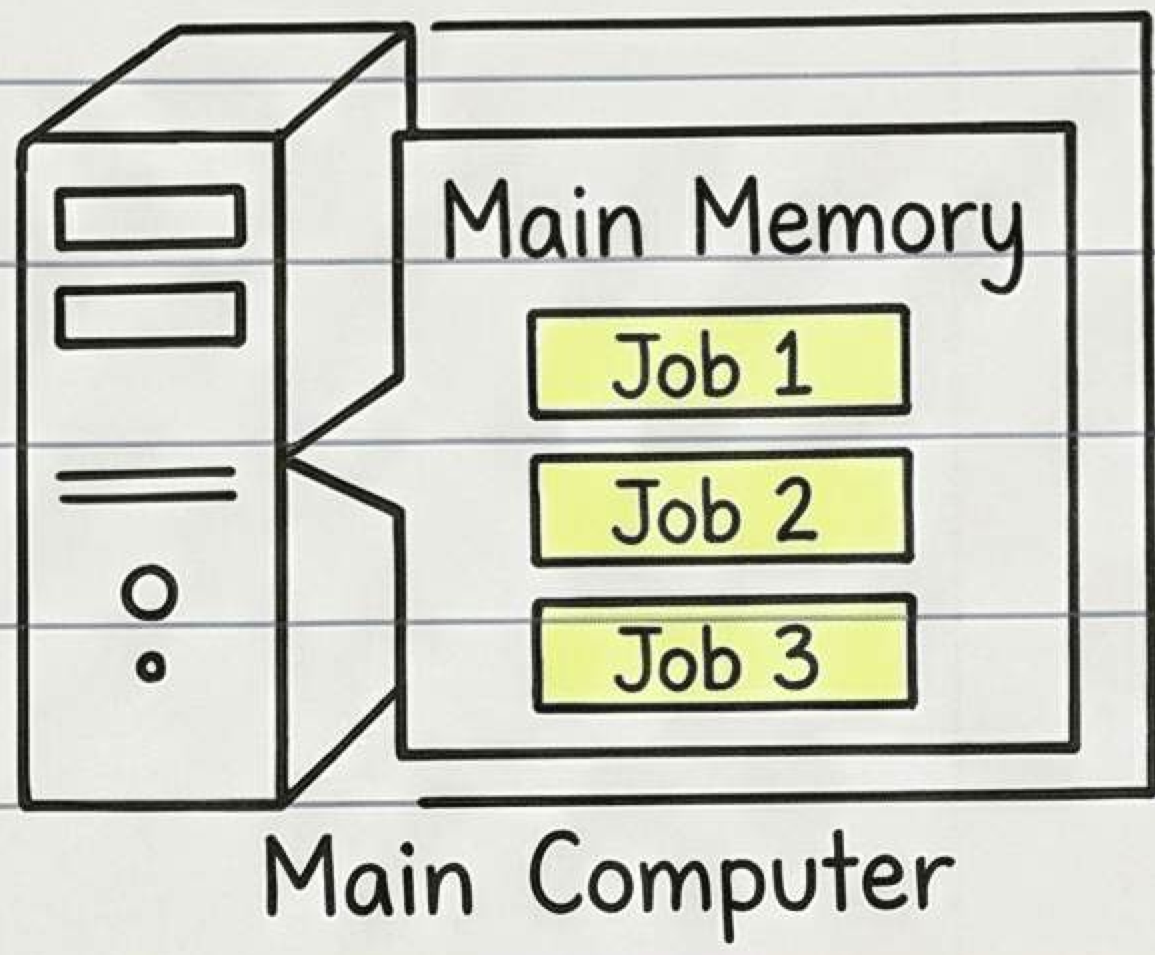




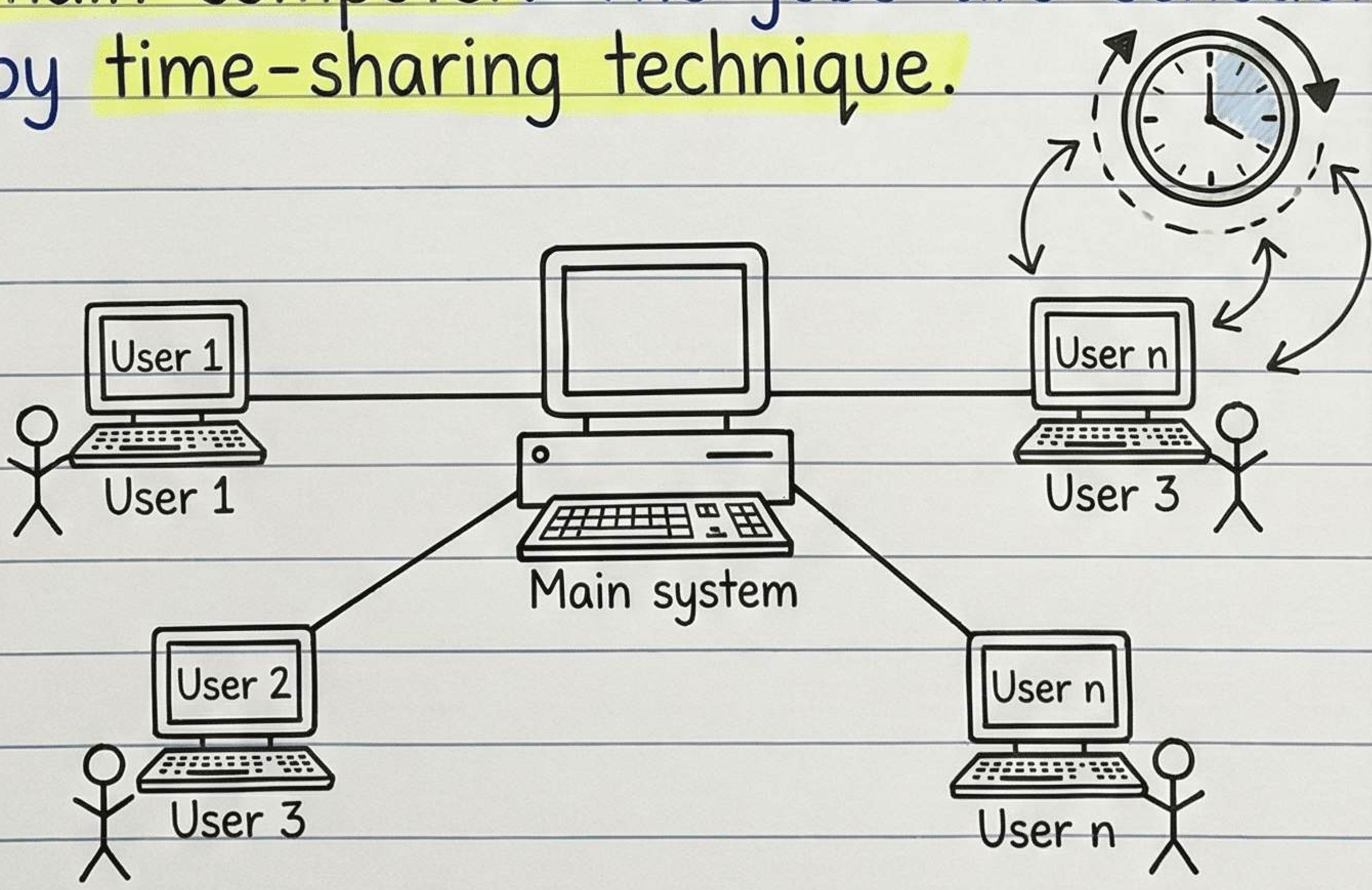
Therefore, modules like process scheduling, device management, protection, etc., were developed.

Multi-user Time-sharing Systems

- Multi-user systems place more than one job/program/task in the main memory of the main computer system.

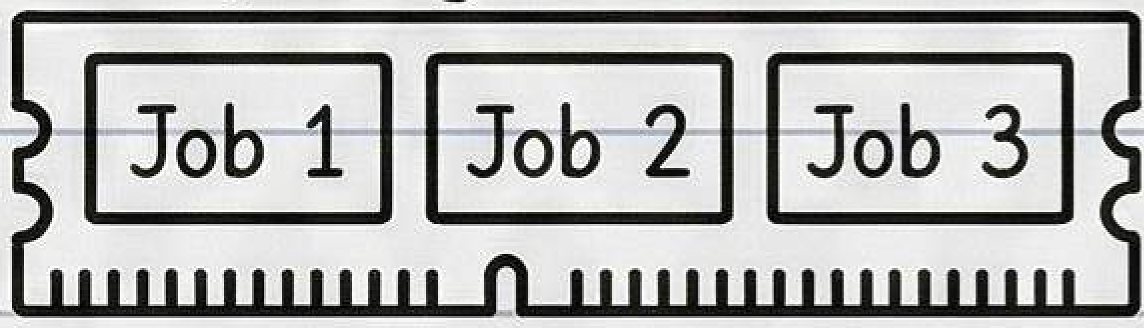


- The jobs are of different users who are connected through terminals to the main computer. The jobs are to the main computer. The jobs are scheduled by time-sharing technique.

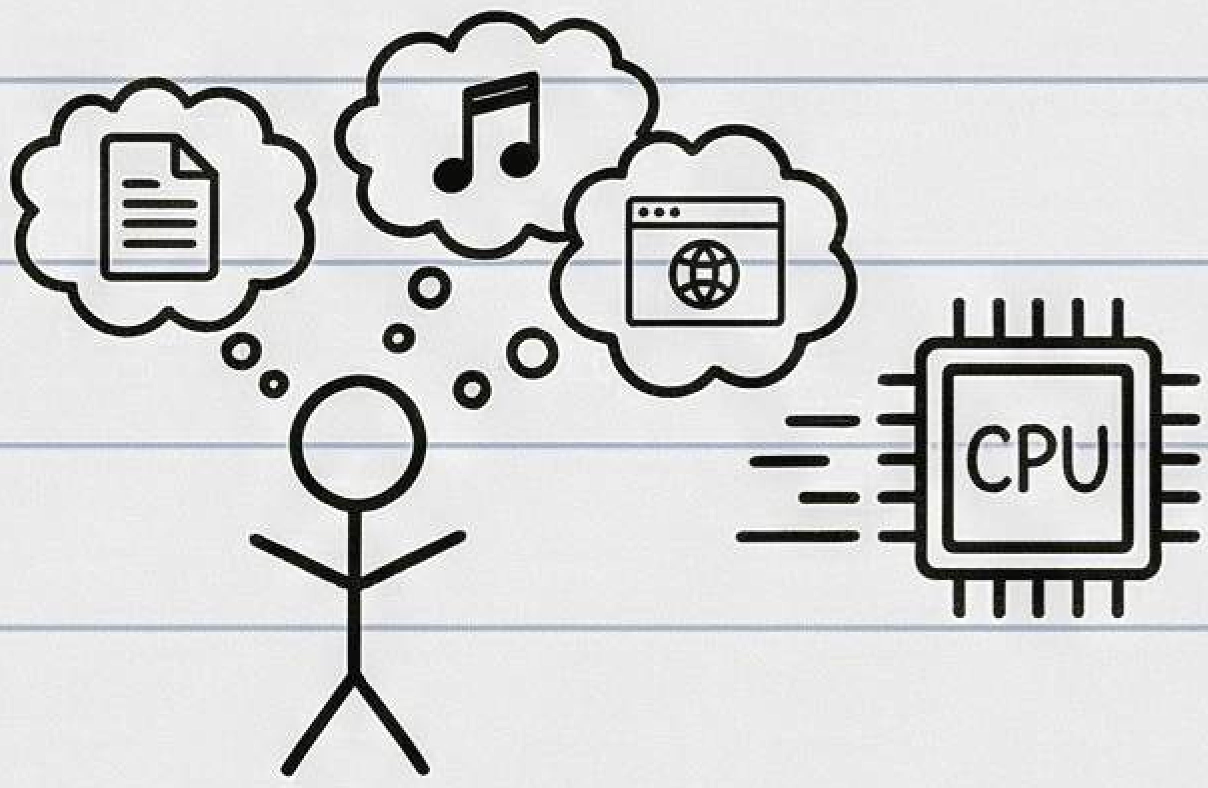


Multi-Tasking Systems

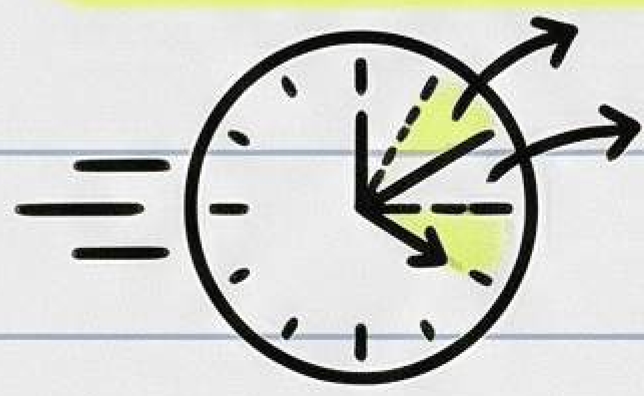
- Places more than one job/program/task.
  Places more than one job/program/task in the main memory.



- User gets the illusion of working in parallel on multiple tasks due to a high speed processor.

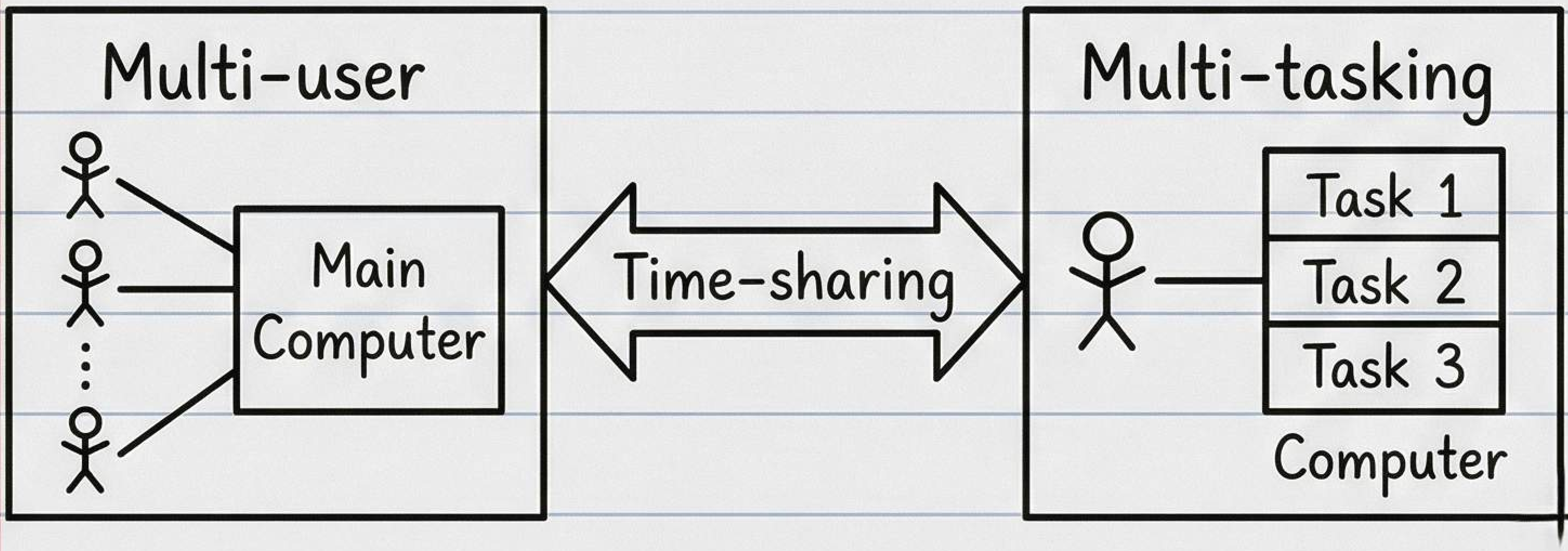


- This is made possible by the time-sharing scheduling technique.



- All jobs are from a single user.

- Multi-user and Multi-tasking are different terms, but both use time-sharing.

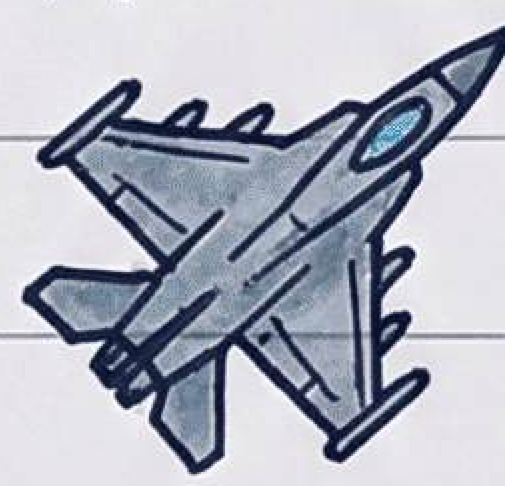
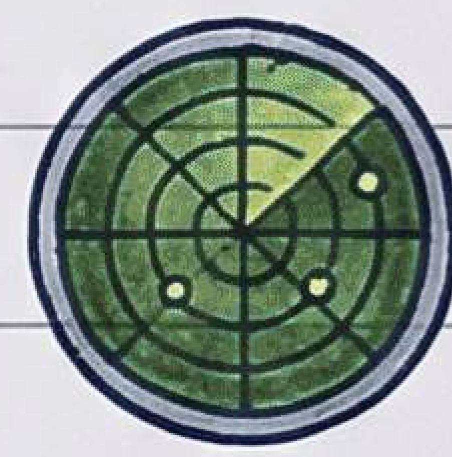


Real-Time Operating Systems (RTOS)

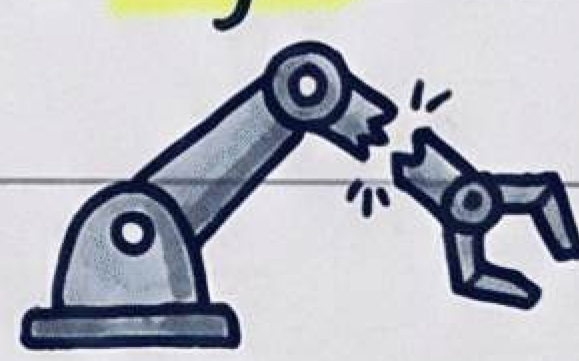
- Response to a user request must be immediate or within a fixed time frame, otherwise the application will fail.


- This is known as real-time processing.

- Largely useful in defence applications which are mission specific.

- If no timely response, there might be loss of equipment and even life.



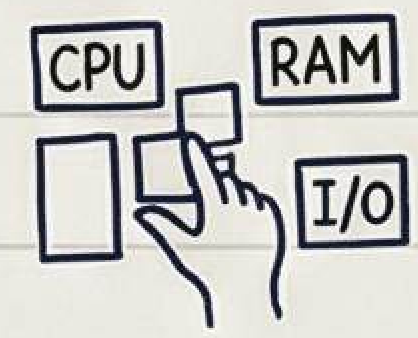

- Must meet deadlines of time to prevent failures.


OS Services

Primary Goals & Convenience

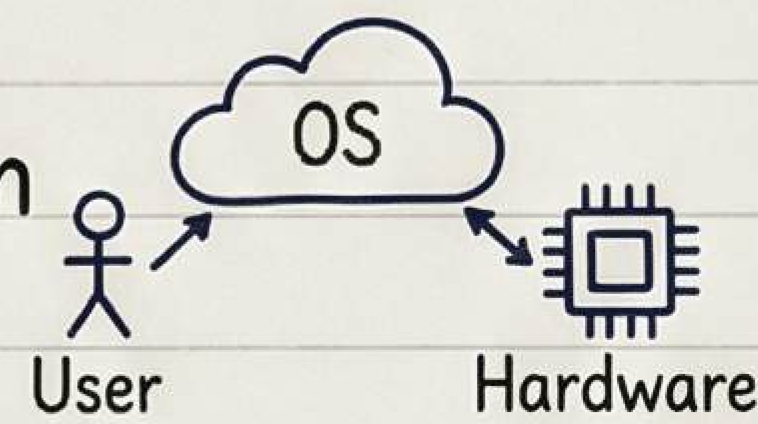
- Primary goals are convenience of the user and best utilization of hardware.
- Convenience is the prime goal for the user performing a job.

- Resource Utilization/Management : Efficiency is a major goal; takes care of best allocation of resources.



Abstraction & Interface

- Hardware abstraction/virtual machine : Provides an abstraction layer between user and hardware.



- Easy-to-use interface : Relieves user from remembering commands (like older UNIX/DOS).

- Provides a convenient programming environment.

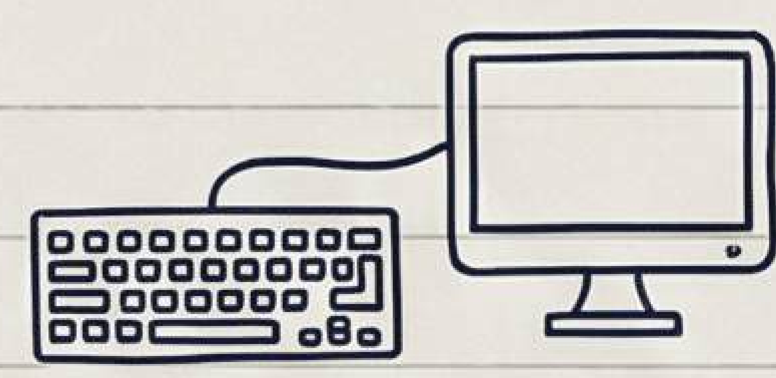
Protection & Response

- Protection : Protects one user's task from another and the OS from any user.

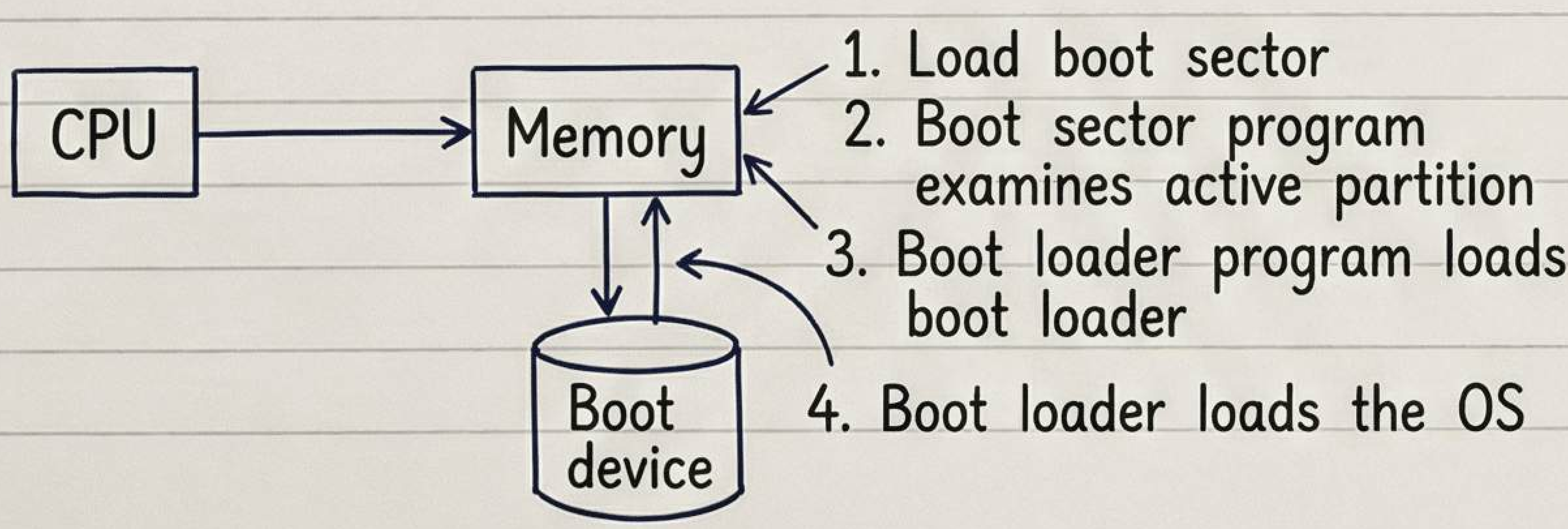
- Response time.

BIOS & Booting

- BIOS : Software with low-level I/O functions (keyboard, screen, ports).



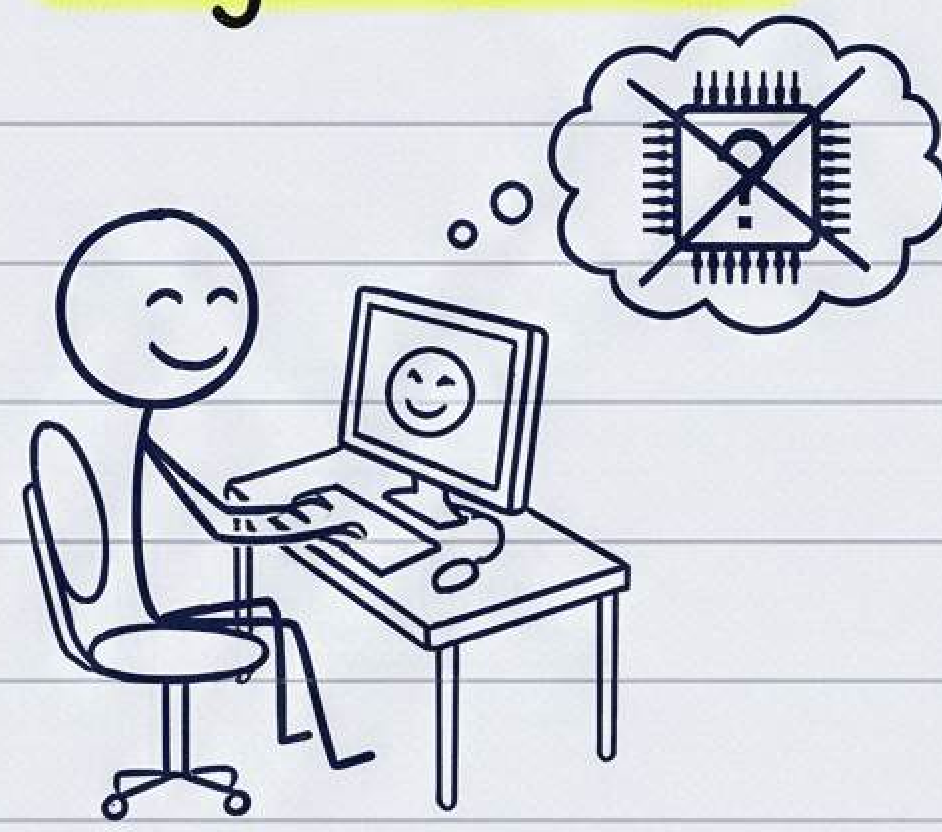
- Booting : Loading the OS into RAM using Boot software/loader.



User View & System View of OS

User View

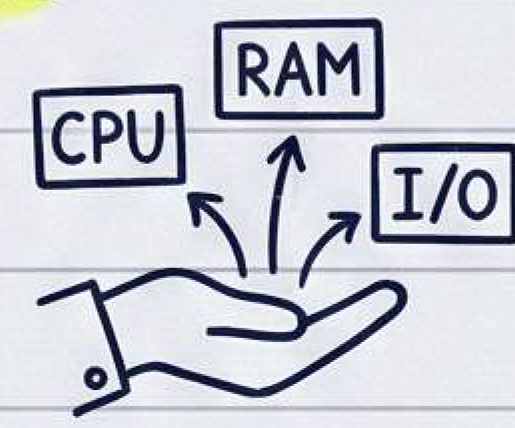
- From user's viewpoint, OS acts as an easy interface between user & computer.



- Presents a friendly environment to work efficiently.
- User doesn't worry about hardware details or configuration.

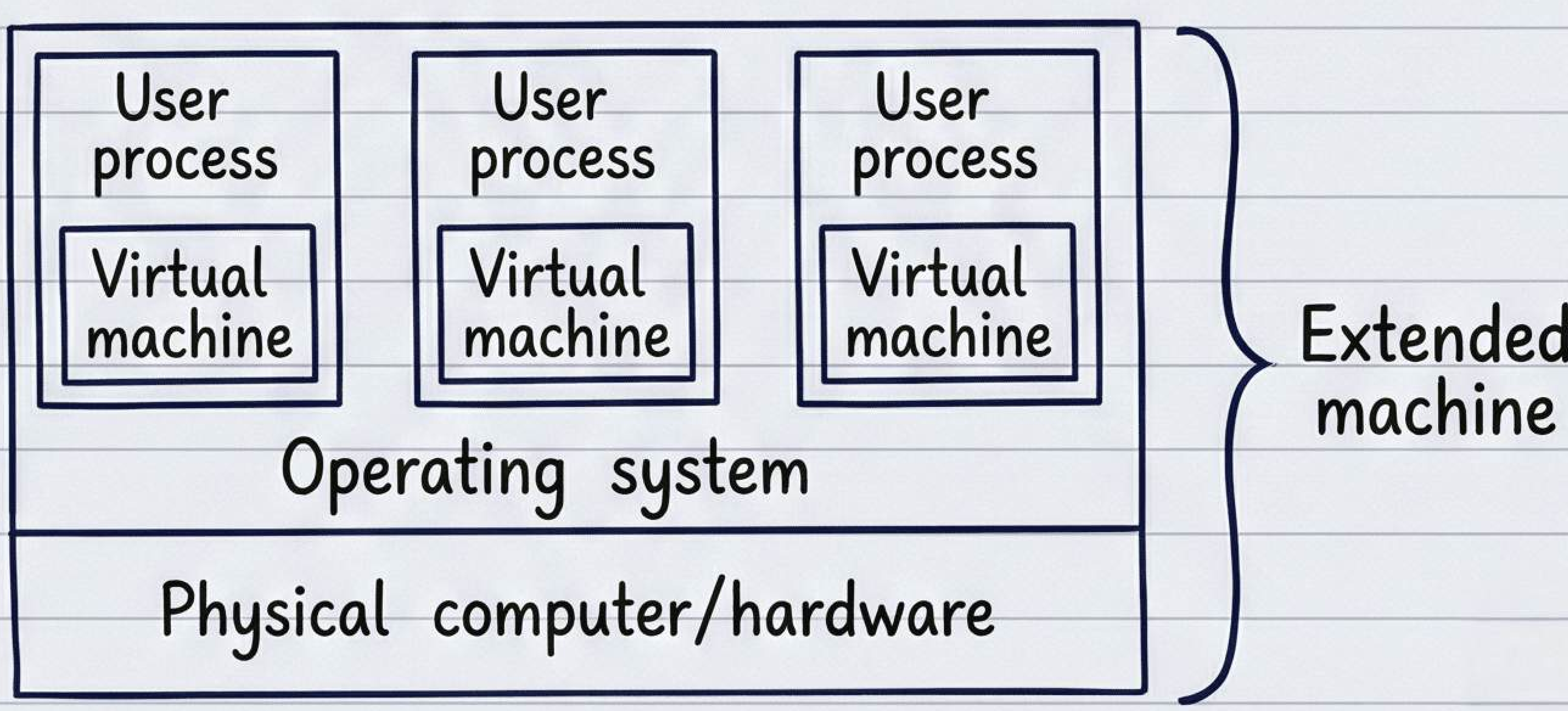
System View

- From system's viewpoint, OS acts as a resource manager, control program, & virtual machine manager.



- Resource manager : Schedules & manages allocation of all resources.
- Control program : Controls user activities, I/O access, etc.

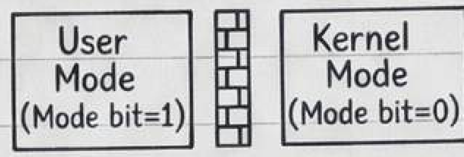
- Virtual machine manager : Provides an abstraction layer on actual hardware.
  Creates an illusion of a virtual machine performing the work, hiding physical hardware.



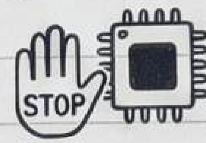
System Calls

Dual Mode & Privileged Instructions

- Modern OSs separate user code from OS code, termed dual mode operation: user mode & kernel mode.

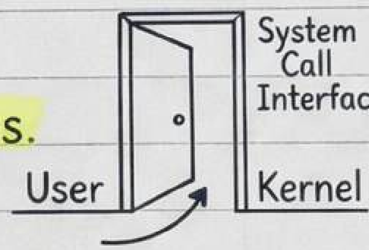


- Instructions interacting with hardware (like I/O instructions) are privileged instructions.
- User programs cannot directly execute them.



What is a System Call?

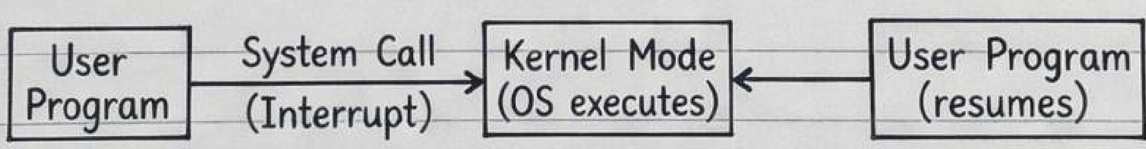
- It is the interface for user-mode programs to access kernel-mode functions.



- A user request to the OS to perform a task on its behalf.

How it Works

1. User process makes a system call.
2. This generates an interrupt.
3. Interrupt changes mode to kernel mode (bit 0) & control passes to OS.
4. OS executes the privileged instruction.
5. Control returns to user mode.

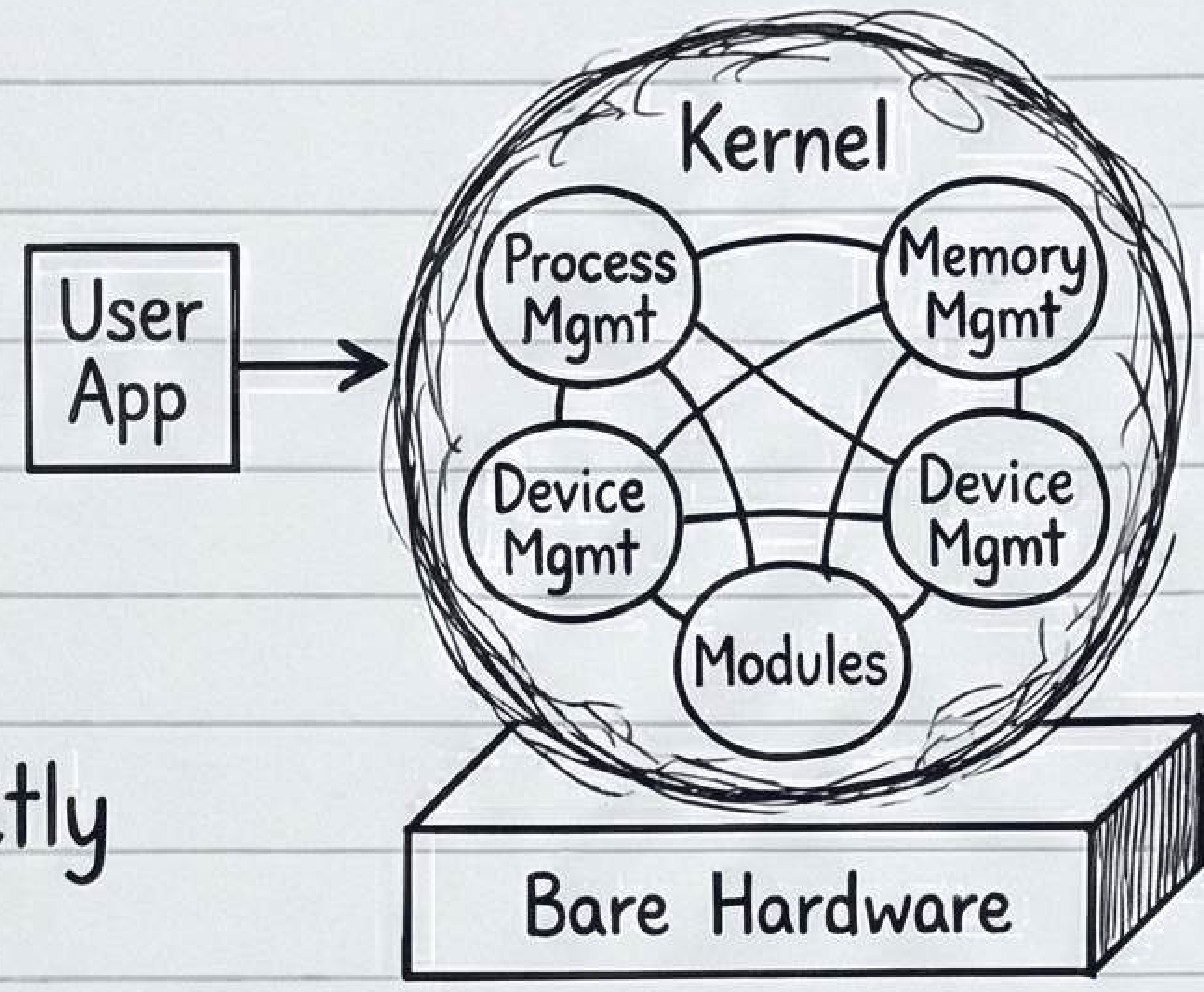


- **Difference**: A system call enters the kernel, a normal function call does not.

Monolithic Structure of an OS

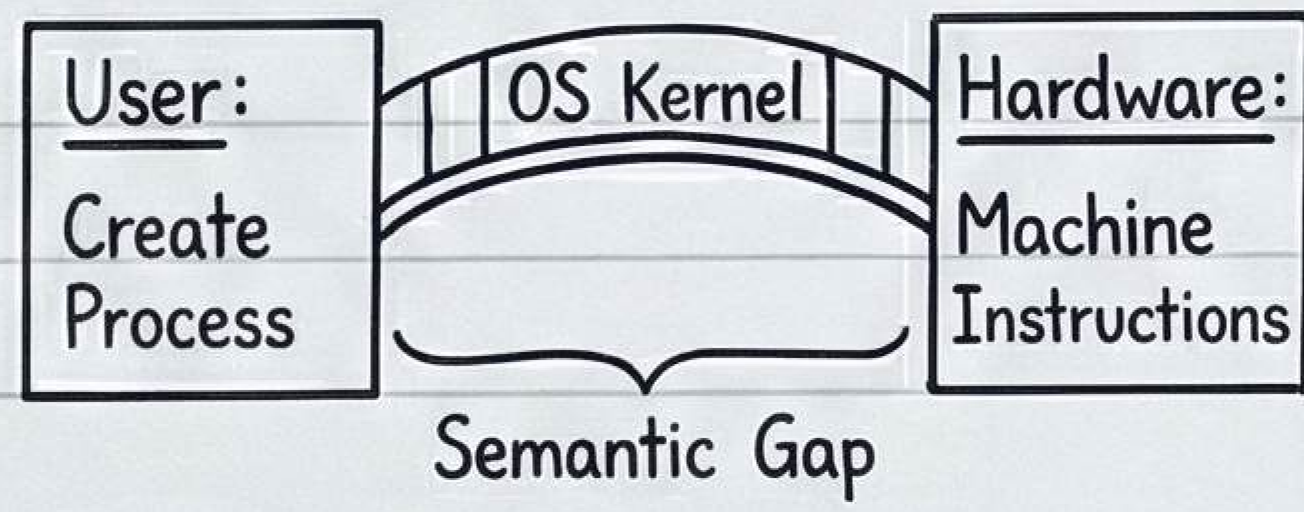
Origin & Structure

- Initial, unplanned architecture where all functionalities were added in the kernel only.



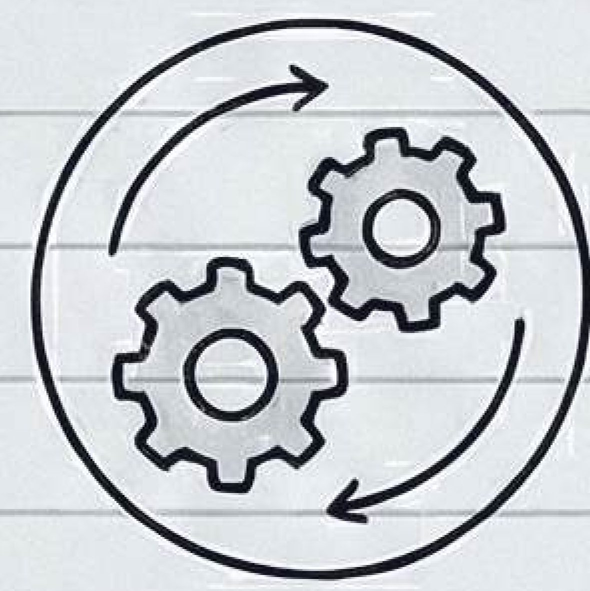
- User applications interact with this single kernel layer, which directly accesses bare hardware.

- Helps bridge the large semantic gap between user operations and machine instructions.



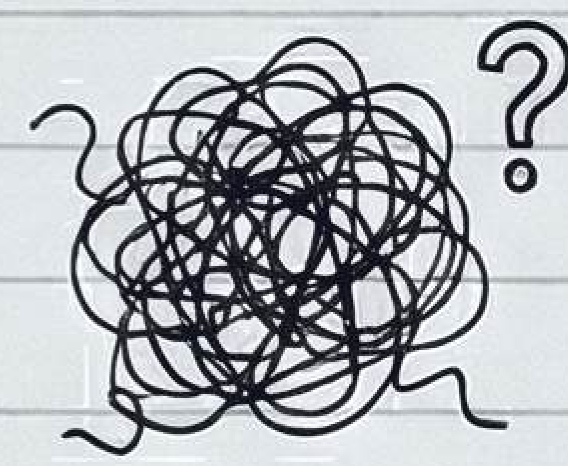
Advantage

- Efficient intercommunication between modules as they are all in the kernel together.



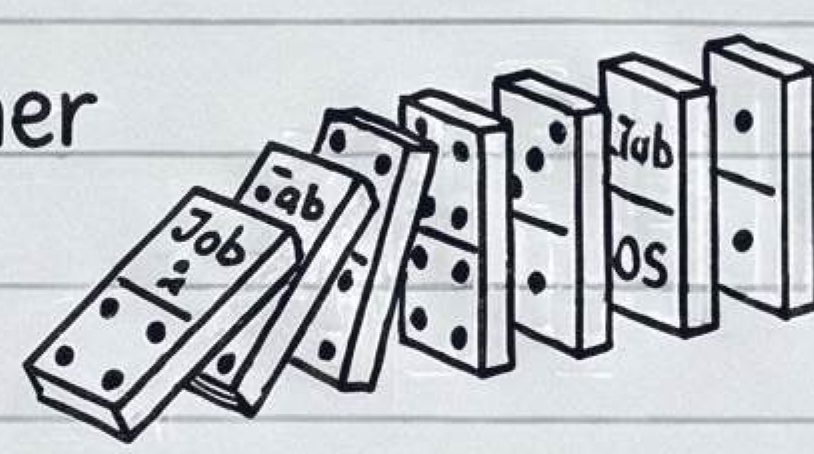
Disadvantages

- Difficult to do modifications because a change in one module can affect many others.
- Consequently, debugging... became a difficult job.

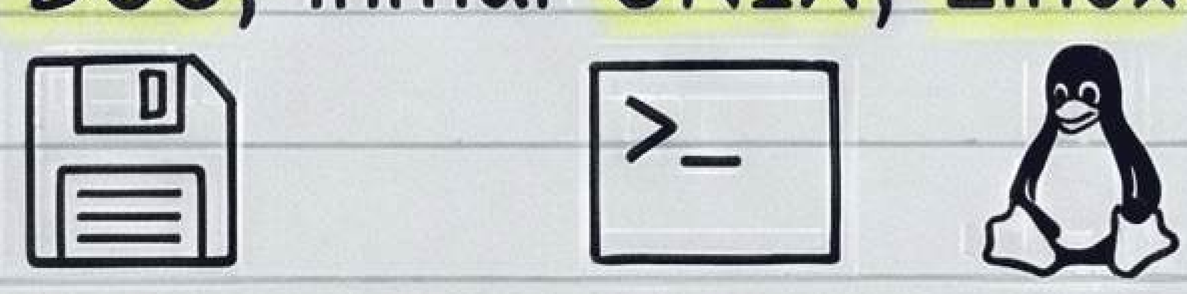


- No protection : There is unrestricted access, creating a danger of malicious code.

- Instability : Any user's job can corrupt any other user's job and even OS.



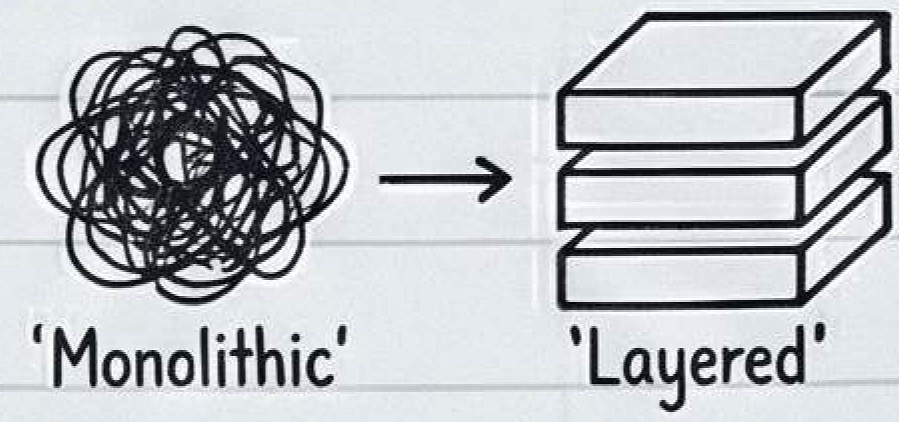
- Examples: DOS, initial UNIX, Linux.



Layered Structure of an OS

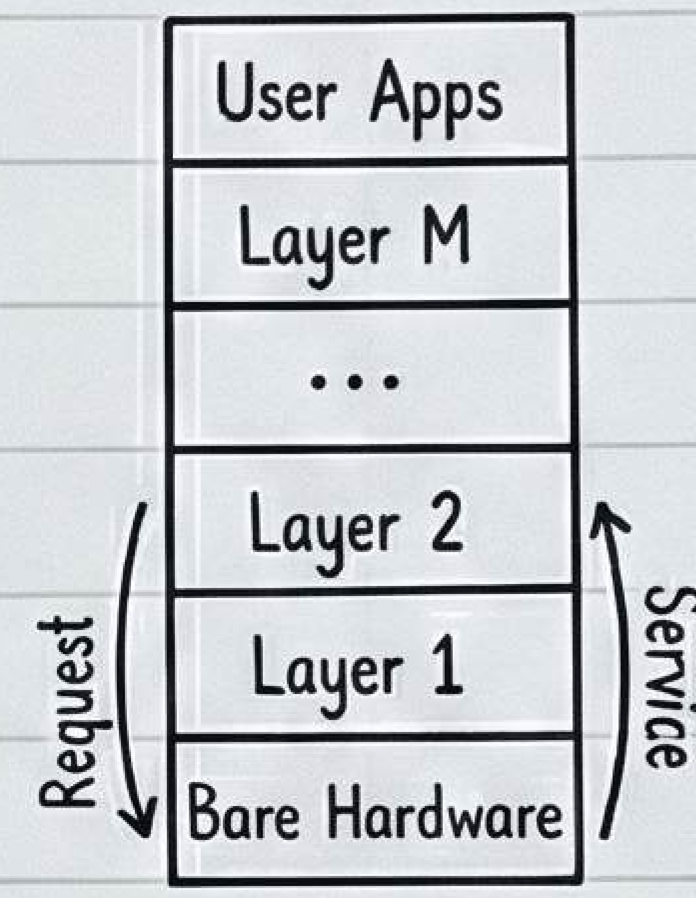
Origin & Need

- Monolithic structures became complex and obsolete.
- Need arose to design OS with no single layer having all functionalities.
- Resulted in layered architectures.



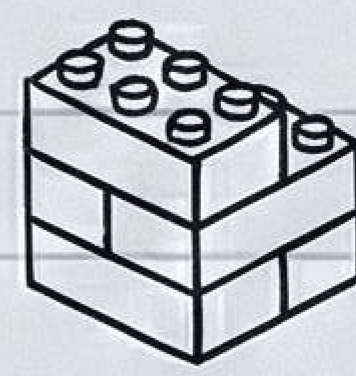
Core Principles & How it Works

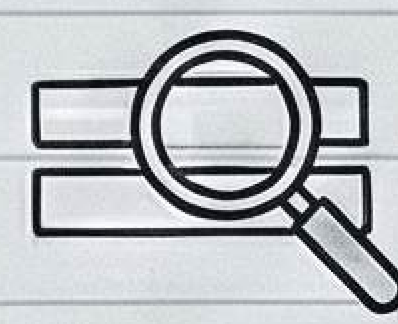
- Based on grouping of functions into a layer & hierarchy of layers.
- Functions of a category are grouped into a defined layer (e.g., process management layer).
- Topmost layer interfaces with user applications.
- Lowest layer interacts with underlying hardware.
- Each layer communicates only with adjacent layers (immediately above or below).
- A layer uses services of the layer below it.

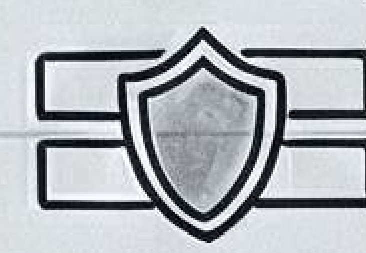


Advantages

- Simple interfacing compared to monolithic.
- Provides modularity.
- Easy debugging & modification as changes are localized to one layer.
- Protection between layers; prevents malicious code spread.

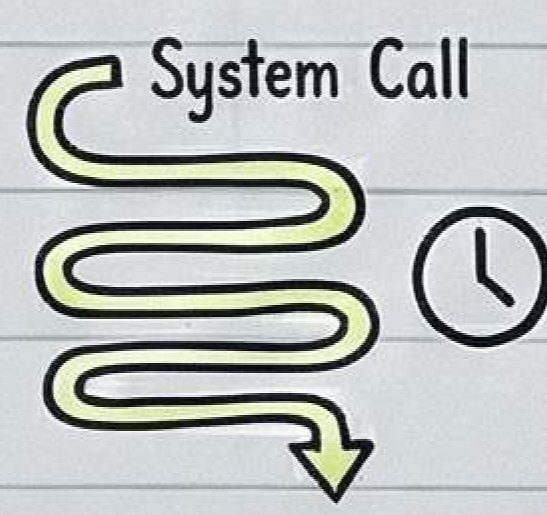


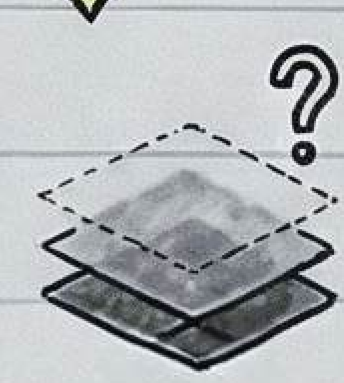




Disadvantages

- System calls must pass through all layers, which takes time.
- May suffer from efficiency problems with many layers.
- Difficult to group modules and isolate functionalities.

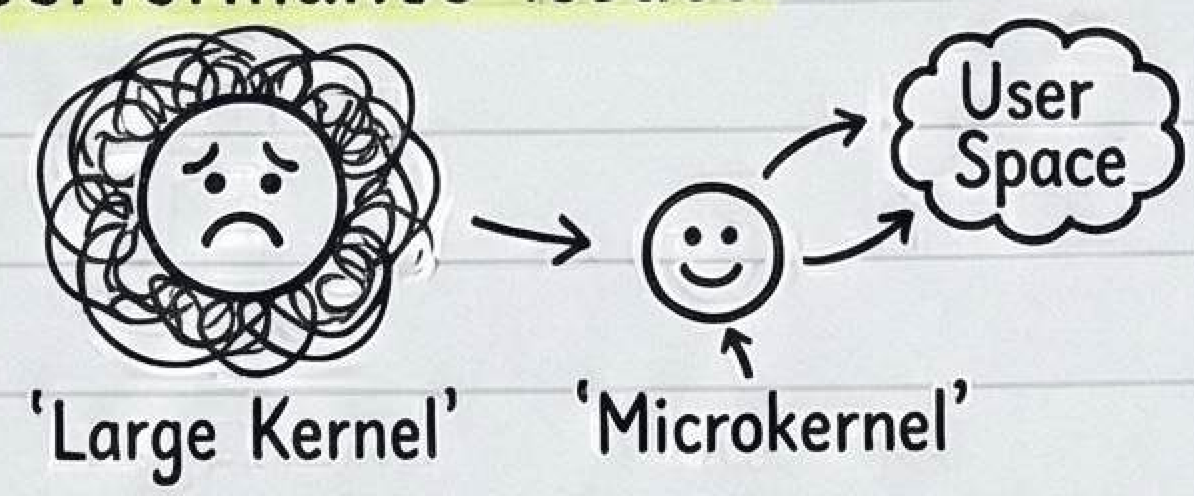




Microkernel Structure of an OS

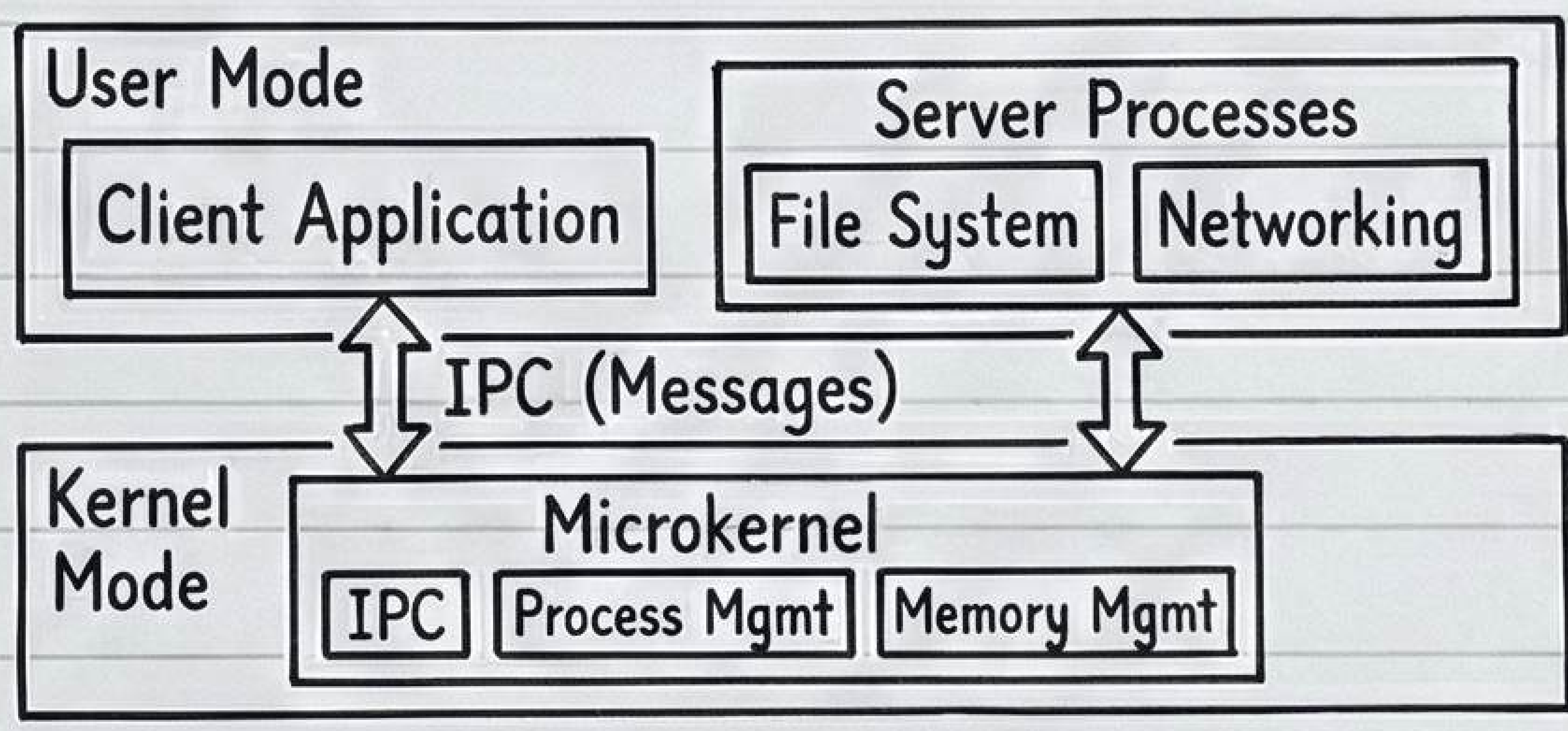
Origin & Need

- As OS demands increased, monolithic/layered kernels grew large and became difficult to manage, extend, and rely on.
- Large kernels were prone to errors and had performance issues.
- Large kernels were prone to errors and had performance issues.
- Adding/deleting support for new devices was hard due to dependencies.



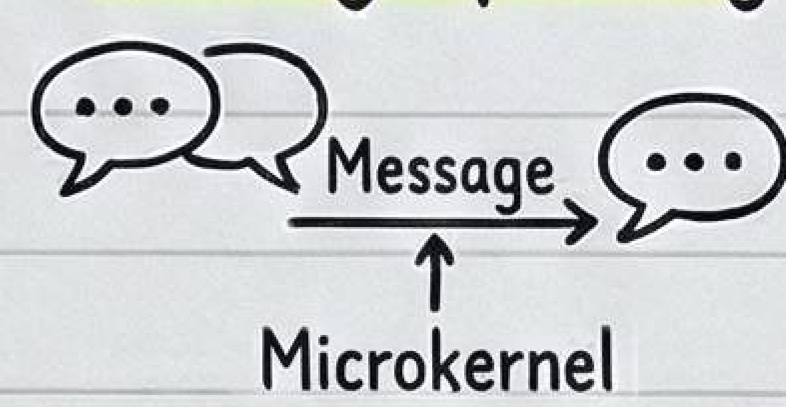
Core Concept & Structure

- The kernel is minimized to only essential functionalities (the 'essential core') like process management, IPC, low-level memory management.
- Non-essential modules are moved to user space as server processes.
- Also known as client-server architecture. cite

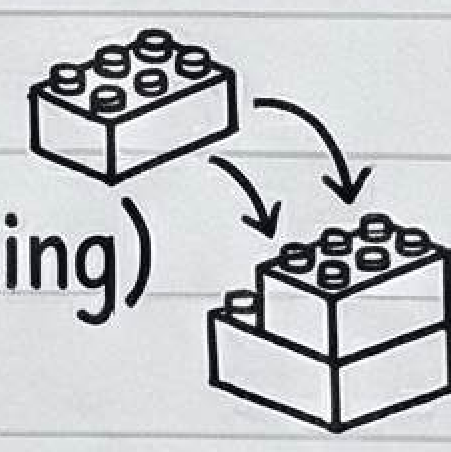


Communication (IPC)

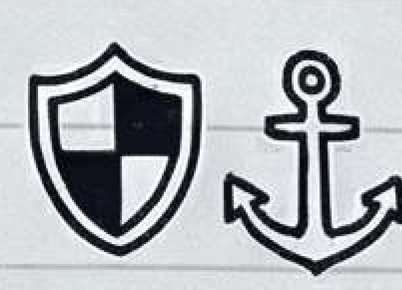
- Communication between clients and servers is through message passing, called Inter-process Communication (IPC).
- The microkernel facilitates IPC by validating and passing messages.



Advantages & Examples



- Adaptable/Extensible : Easily add/remove modules (e.g., networking) by starting/stopping servers without recompiling the kernel.
- Reliable/Robust : If a server module fails, it can be stopped without crashing the entire OS.



- Examples : Mach (first & successful), PARAS (C-DAC, India).
- Difficulty : Deciding what goes into the essential core.