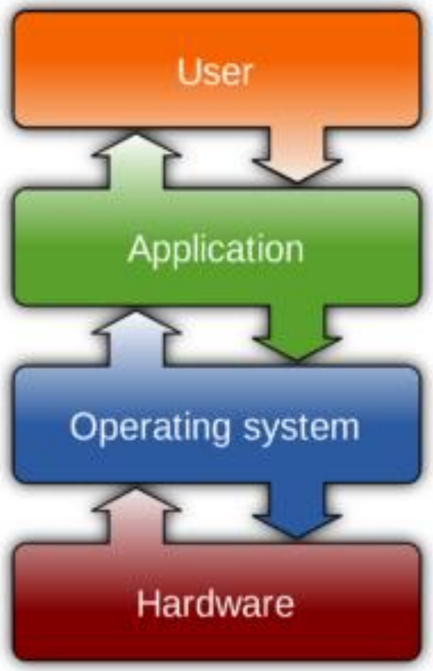
Operating Systems

Module:1	Introduction	3 hours
Introduction to OS: Functionality of OS - OS design issues - Structuring methods (monolithic, layered, modular, micro-kernel models) - Abstractions, processes, resources - Influence of security, networking, and multimedia.		

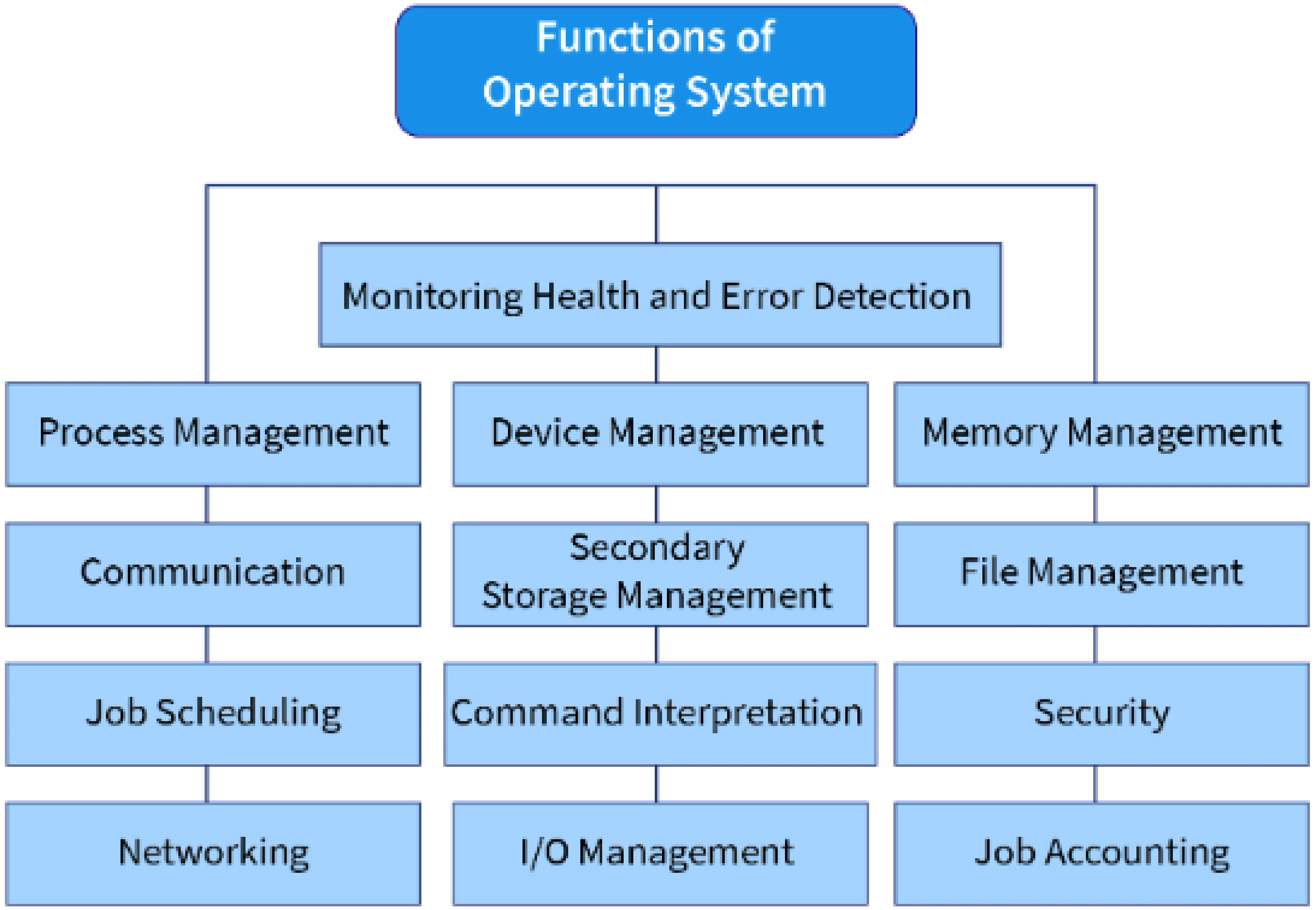
1. Introduction to OS

An operating system (OS) is a program that acts as an interface between the system hardware and the user. Moreover, it handles all the interactions between the software and the hardware. All the working of a computer system depends on the OS at the base level. Further, it performs all the functions like handling memory, processes, the interaction between hardware and software, etc.

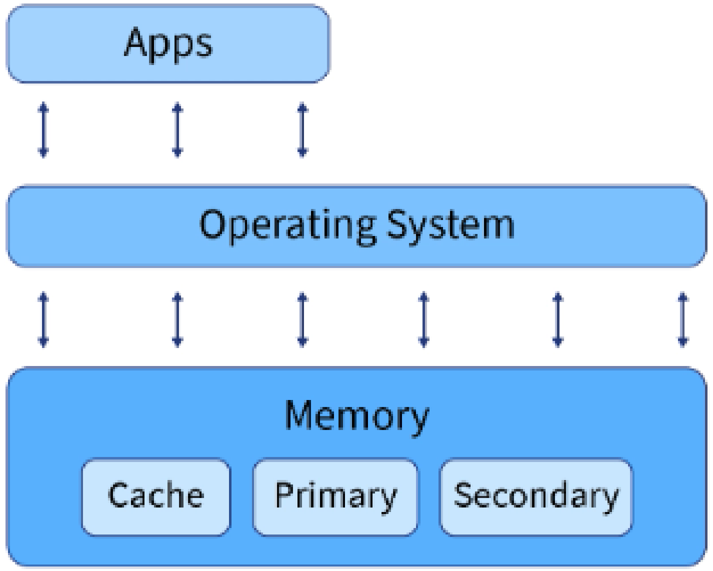


Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers


Functions of Operating System
Monitoring Health and Error Detection
Process Management
Device Management
Memory Management
Communication
Secondary Storage Management
File Management
Job Scheduling
Command Interpretation
Security
Networking
I/O Management
Job Accounting

Memory Management



- Allocates and deallocates the memory.
- Keeps a record of which part of primary memory is used by whom and how much.
- Distributes the memory while multiprocessing.

Processor Management/Scheduling

- Processor management is an execution unit in which a program operates.
- The operating system determines the status of the processor and processes, selects a job and its processor, allocates the processor to the process, and de-allocates the processor after the process is completed.

Device Management

- An operating system regulates device connection using drivers. The processes may require devices for their use. This management is done by the OS. The OS:
- Allocates and deallocates devices to different processes.
- Keeps records of the devices.
- Decides which process can use which device for how much time.

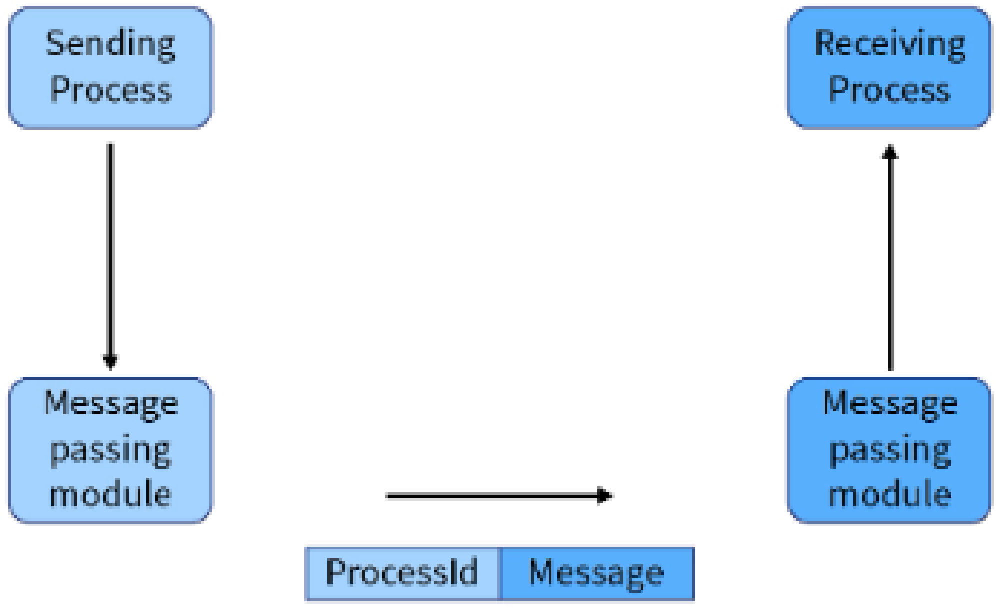
File Management

- It specifies which process receives the file and for how long. It also keeps track of information, location, uses, status, and so on. These groupings of resources are referred to as file systems. The files on a system are stored in different directories. The OS:
- Keeps records of the status and locations of files.
- Allocates and deallocates resources.
- Decides who gets the resources.

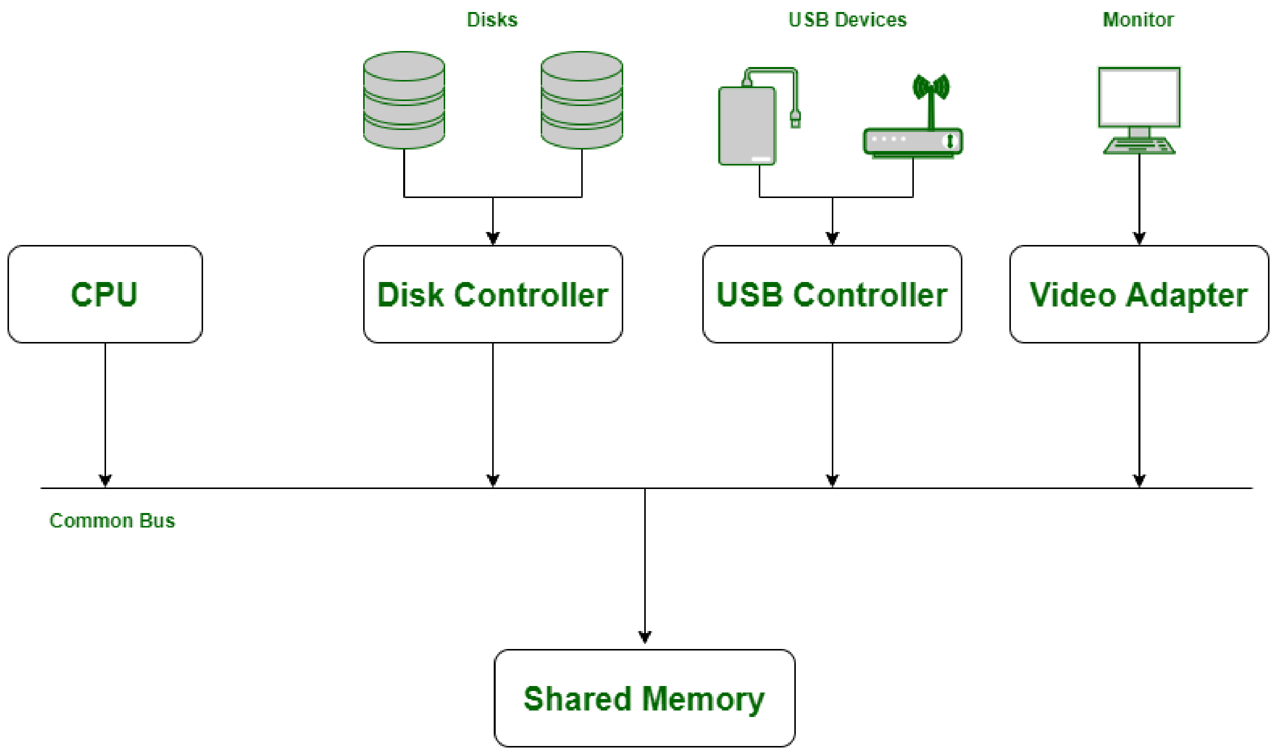
Storage Management

- Storage management is a procedure that allows users to maximize the utilization of storage devices while also protecting data integrity on whatever media on which it lives.

Communication



I/O Management



- Shared memory in I/O (Input/Output) management of an operating system serves as a communication mechanism between different processes or threads running on the system.

- **Security** – For security, modern operating systems employ a firewall.
- **Job Accounting** – As the operating system keeps track of all the functions of a computer system. Hence, it makes a record of all the activities taking place on the system.

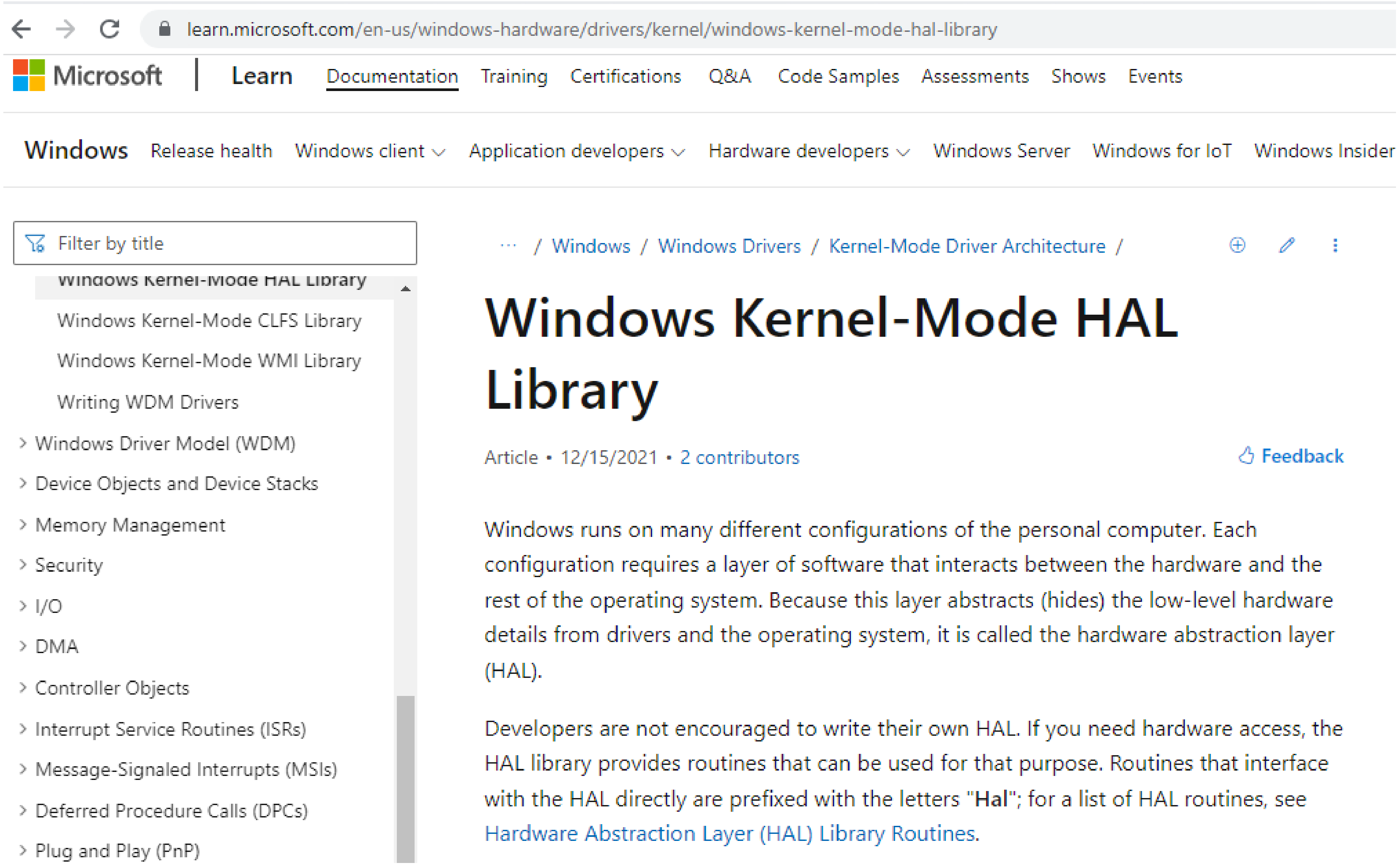
3. OS Design Issues

- Efficiency:
- Throughput and Resource Utilization: This aspect of efficiency measures how much useful work the operating system can accomplish in a given amount of time while utilizing the available resources optimally. It focuses on maximizing the system's throughput, which is the number of tasks completed or processed per unit of time. A highly efficient operating system can handle a significant amount of workload without wasting resources or causing unnecessary delays.

- CPU Utilization and Response Time: In a time-sharing system or a multi-user environment, efficiency can be measured by how well the operating system utilizes the CPU and how quickly it responds to user requests. CPU utilization is the ratio of the time the CPU spends executing tasks compared to the total time. A highly efficient OS aims to keep the CPU busy as much as possible, avoiding idle time. Additionally, response time is the time it takes for the system to respond to a user's request. Low response times indicate a more efficient system, as it means users experience less delay in getting their tasks processed.

- Robustness:
- Robust software is designed to handle unexpected situations and errors gracefully without crashing or causing significant disruptions.
- Fault Tolerance: The software can handle unexpected errors or failures gracefully without causing a complete system crash. If one component fails, the system can continue functioning with minimal impact on other components.
- Error Handling: Robust software is designed to handle errors effectively, either by recovering from them or providing informative error messages to users for troubleshooting.
- Isolation: Failures in one application or process should not bring down the entire system. Robust systems are structured to provide isolation between different processes, ensuring that problems in one part do not affect others.
- Security: Robustness often goes hand-in-hand with security. A secure system can withstand attacks and attempts to compromise its integrity.

- Flexibility:
- A flexible system can be easily extended, upgraded, or reconfigured to meet changing requirements without significant constraints or limitations.
- Portability:
- Portability, in the context of software and operating systems, refers to the ability of a program or the operating system itself to run on different platforms or hardware configurations with minimal changes or recompilation.
- For an operating system to be portable, it means the OS is designed and implemented in a way that allows it to be easily adapted to different hardware architectures. This includes making sure that the OS kernel and core components are hardware-independent and using proper abstraction layers to interact with the hardware.



learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-hal-library

Microsoft | Learn
Documentation Training Certifications Q&A Code Samples Assessments Shows Events
Windows Release health Windows client Application developers Hardware developers Windows Server Windows for IoT Windows Insider
Filter by title

Windows Kernel-Mode HAL Library
Windows Kernel-Mode CLFS Library
Windows Kernel-Mode WMI Library
Writing WDM Drivers
Windows Driver Model (WDM)
Device Objects and Device Stacks
Memory Management
Security
I/O
DMA
Controller Objects
Interrupt Service Routines (ISRs)
Message-Signaled Interrupts (MSIs)
Deferred Procedure Calls (DPCs)
Plug and Play (PnP)

… / Windows / Windows Drivers / Kernel-Mode Driver Architecture /

Windows Kernel-Mode HAL Library

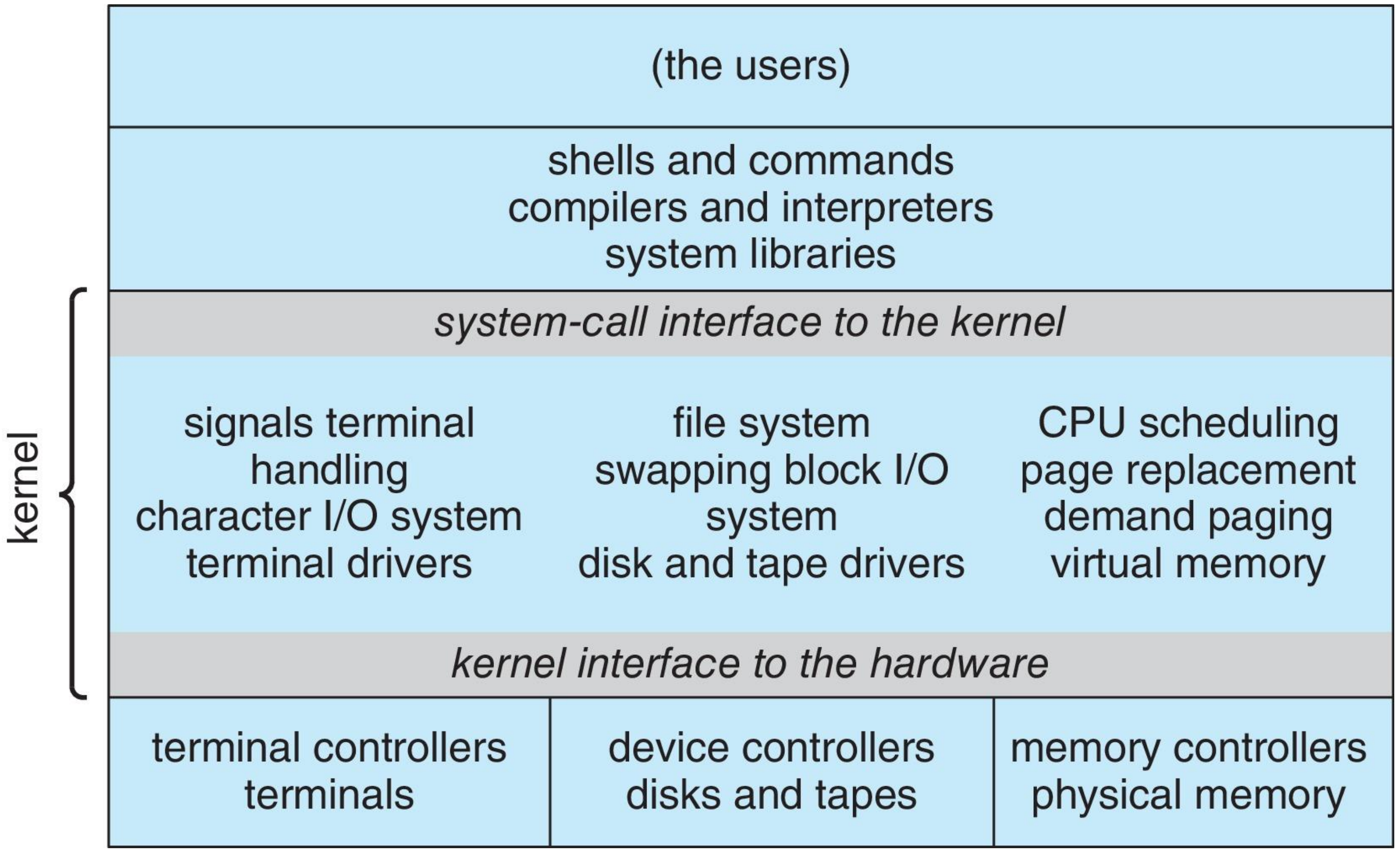
Article • 12/15/2021 • 2 contributors

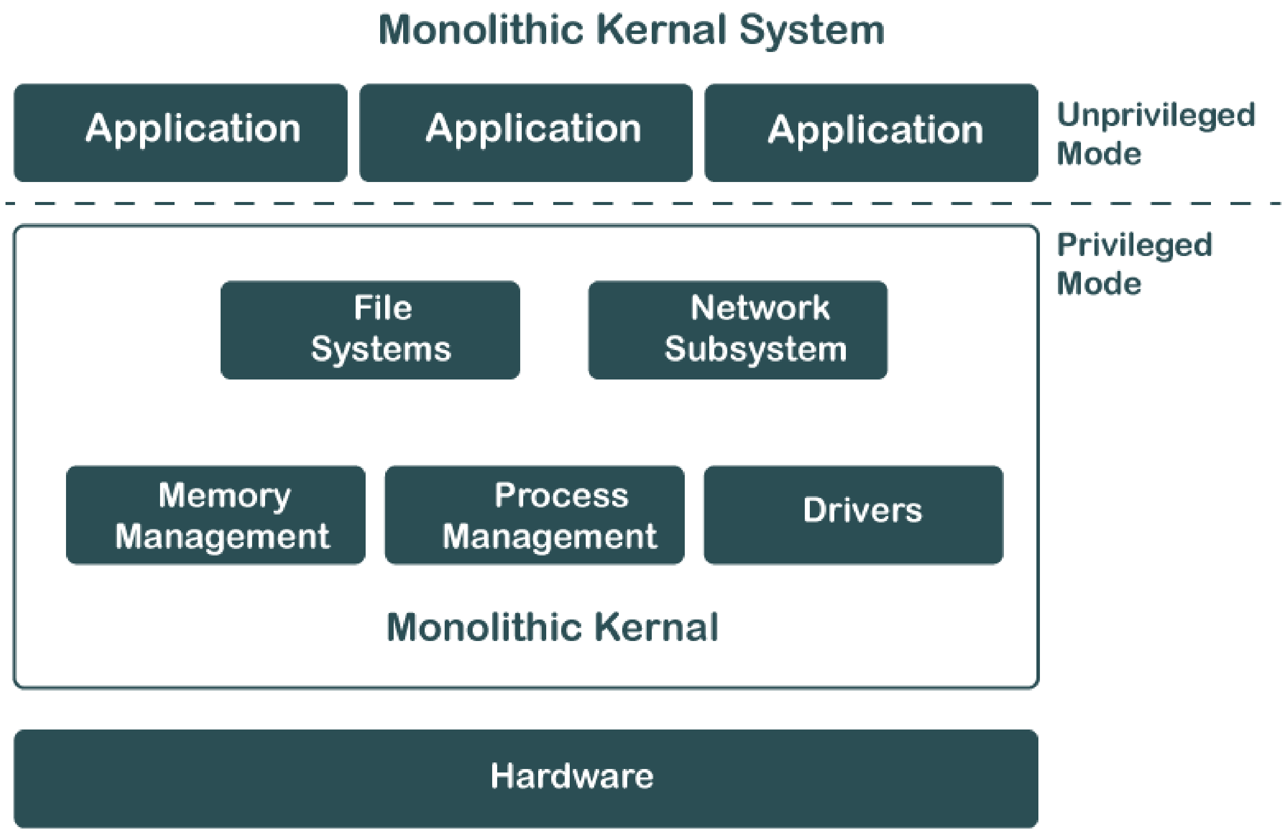
Windows runs on many different configurations of the personal computer. Each configuration requires a layer of software that interacts between the hardware and the rest of the operating system. Because this layer abstracts (hides) the low-level hardware details from drivers and the operating system, it is called the hardware abstraction layer (HAL).

Developers are not encouraged to write their own HAL. If you need hardware access, the HAL library provides routines that can be used for that purpose. Routines that interface with the HAL directly are prefixed with the letters "**Hal**"; for a list of HAL routines, see Hardware Abstraction Layer (HAL) Library Routines.

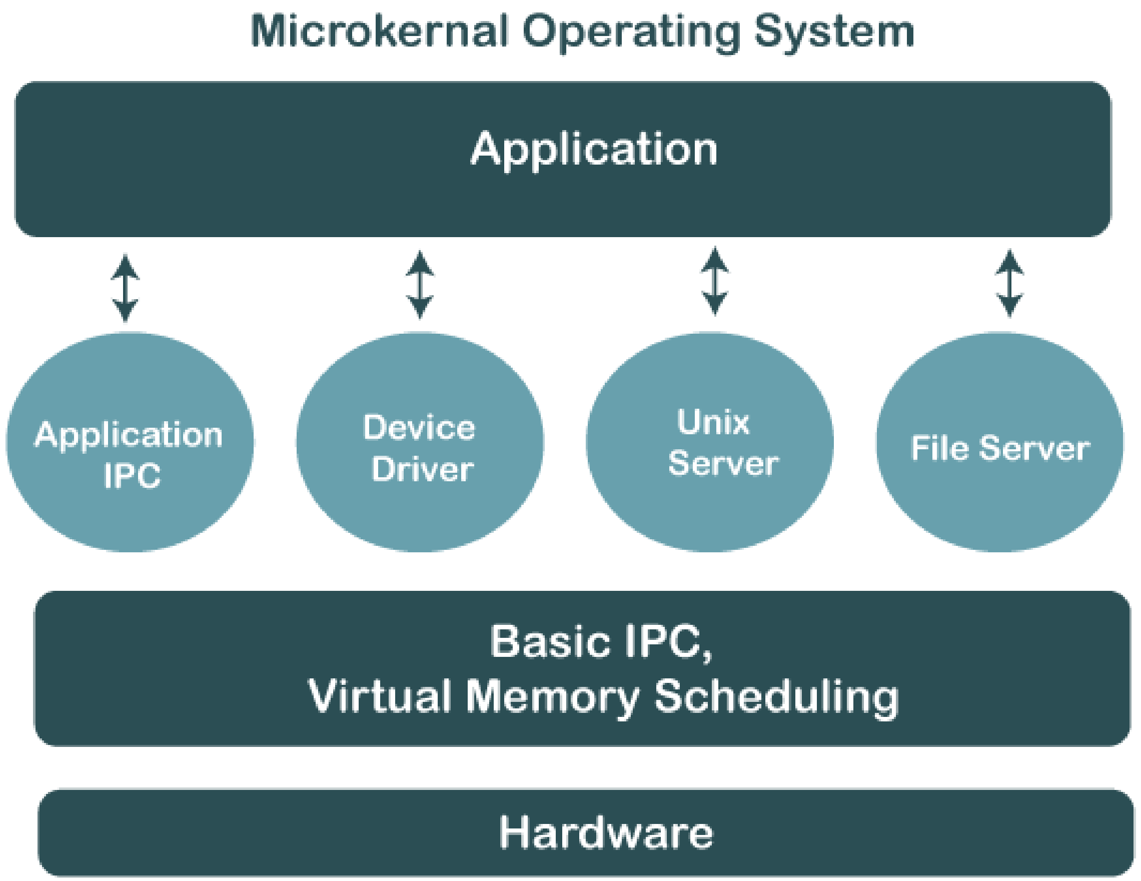
4.Structuring methods (monolithic, layered, modular, micro-kernel models)

- The overall organization and arrangement of the operating system's components are referred to as the operating system structure. This structure defines how various components interact with each other and how they are integrated into the operating system as a whole.

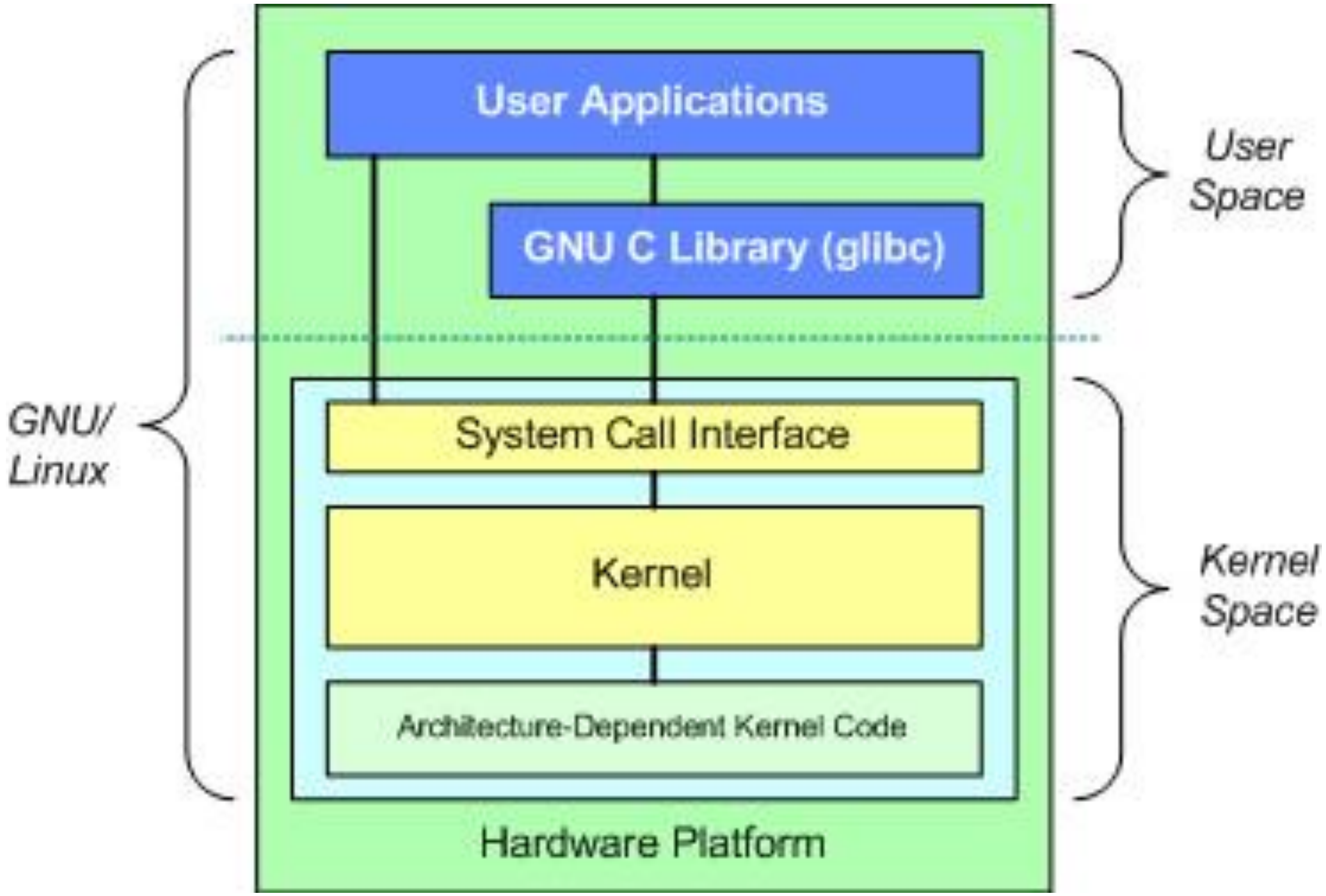




operating system services and functionalities are implemented in a single, large, and unified kernel space.



only the essential services, such as process scheduling, memory management, and inter-processcommunication (IPC), remain in the kernel, while other functionalities, such as device drivers, file systems, and networking protocols, are implemented as separate user-space processes.


User Applications
GNU C Library (glibc)
User Space
GNU/ Linux
System Call Interface
Kernel
Architecture-Dependent Kernel Code
Kernel Space
Hardware Platform

- Key features of a monolithic kernel include:
- Kernel Space: In a monolithic kernel, all the components and functionalities of the operating system run in a single privileged kernel address space. This means that any process or component within the kernel can access and interact with all other parts of the kernel directly.
- Efficiency: Monolithic kernels often provide faster communication between kernel components because there is no need for inter-process communication (IPC) mechanisms, which can introduce some overhead.
- Simplicity: The monolithic design is relatively straightforward compared to other kernel architectures, as it treats the entire kernel as a single program without a clear separation between kernel services.
- Tight Integration: All components in a monolithic kernel are closely linked, enabling them to share data structures and functions easily. This tight integration can lead to efficient system calls and resource management.

- However, there are some drawbacks to the monolithic kernel design:
- Lack of Modularity: Since all functionalities are packed into a single kernel space, adding or modifying individual features can be challenging. A bug or failure in one component can affect the entire system's stability.
- Limited Fault Isolation: A failure in one kernel component can potentially crash the whole system, as there is no strict isolation between components.
- Kernel Size: As all features are included in the kernel, it can lead to larger kernel sizes, consuming more memory.
- Examples of operating systems that use monolithic kernels include early versions of Linux (before advancements like loadable kernel modules), older versions of Windows (e.g., Windows 95, 98), and BSD-based operating systems like FreeBSD.

Advantages of Monolithic Structure:

- Because layering is unnecessary and the kernel alone is responsible for managing all operations, it is easy to design and execute.
- Due to the fact that functions like memory management, file management, process scheduling, etc., are implemented in the same address area, the monolithic kernel runs rather quickly when compared to other systems. Utilizing the same address speeds up and reduces the time required for address allocation for new processes.

Disadvantages of Monolithic Structure:

- The monolithic kernel's services are interconnected in address space and have an impact on one another, so if any of them malfunctions, the entire system does as well.
- It is not adaptable. Therefore, launching a new service is difficult.

- Key features of a microkernel include:
- Modularity: The microkernel design emphasizes modularity, with the core kernel providing only the most basic and essential services. Additional functionalities are implemented as separate user-space processes, allowing for easy extension and modification without modifying the kernel itself.
- Minimalism: By keeping the kernel small and focused, the microkernel design aims to reduce its complexity and increase its reliability and security. The idea is to have a minimal trusted computing base (TCB) in the kernel, reducing the potential attack surface and making it easier to verify and maintain.
- Fault Isolation: Since most operating system services are implemented as user-space processes, a failure in one component does not directly impact the entire system. This fault isolation enhances system stability and makes it easier to recover from failures.
- Inter-Process Communication (IPC): Since user-space processes handle most functionalities, communication between different components requires IPC mechanisms. This introduces some overhead but enhances system security and robustness.
- Portability: The microkernel design can be more easily ported to different hardware architectures because the kernel itself is small and hardware-independent. Most hardware-specific code resides in user-space drivers.

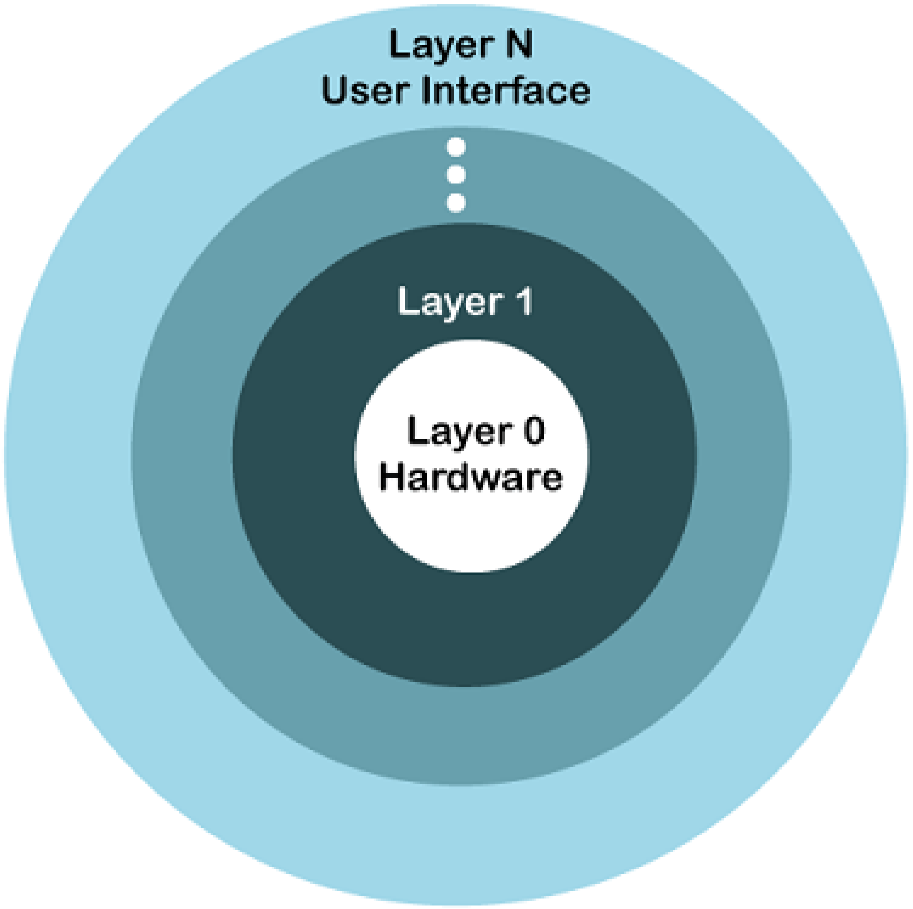
Advantages of Micro-Kernel Structure:

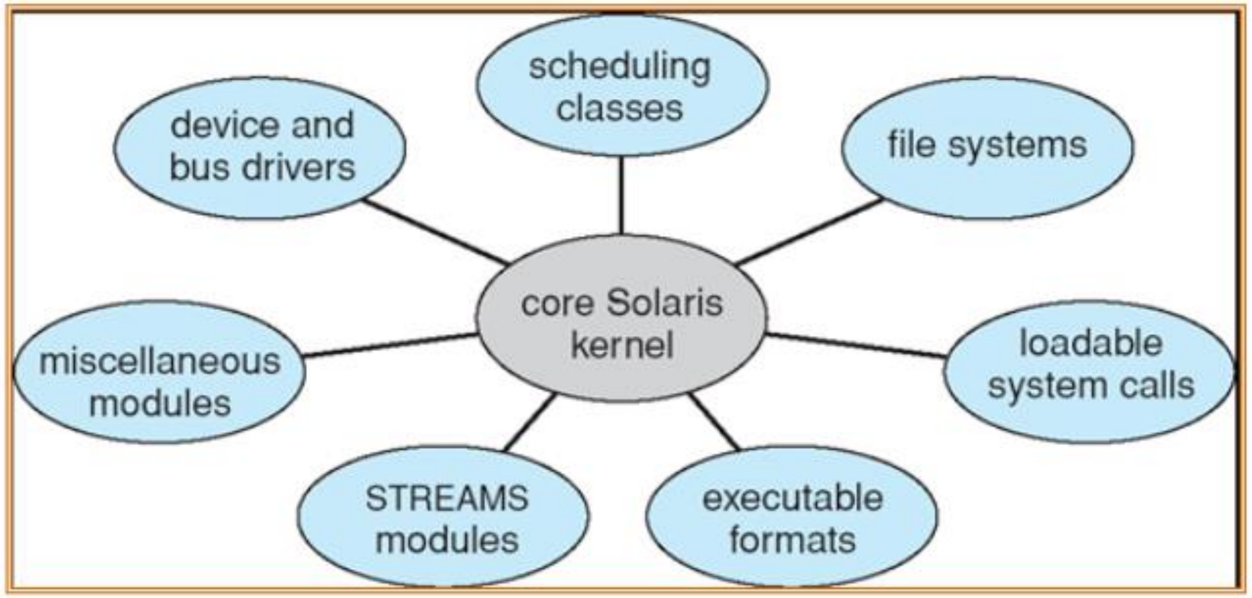
- It enables portability of the operating system across platforms.
- Due to the isolation of each Micro-Kernel, it is reliable and secure.
- The reduced size of Micro-Kernels allows for successful testing.
- The remaining operating system remains unaffected and keeps running properly even if a component or Micro-Kernel fails.

Disadvantages of Micro-Kernel Structure:

- The performance of the system is decreased by increased inter-module communication.
- The construction of a system is complicated.

Solaris Modular Approach





The operating system is divided into a number of layers (levels), each built on top of lower layers.

- The bottom layer (layer 0), is the hardware;
- The highest (layer N) is the user interface.

A *modular kernel* allows an administrator to add functionality only when required. Keeping only what is necessary in kernel memory reduces the kernel's *memory footprint*

Layered Approach

- Advantage
  - Simplicity of construction and debugging
  - The design and implementation is simplified.
  - Each layer is implemented with only those operations provided by lower-level layers.
  - A layer does not need to know how these operations are implemented.
- Disadvantage
  - Appropriately defining the various layers
  - Because a layer can use only lower-level layers, careful planning is necessary.

Examples:

Windows NT;

MS DOS

Modular Kernel Structure

- The Modular kernel approach is <u>similar to</u> the Layered approach
  - The modular kernel approach requires subsystems to interact with each other through carefully constructed interfaces

- However, the Modular kernel approach <u>differs from</u> the Layered approach
  - The layered kernel imposes a strict ordering of subsystems such that <u>subsystems at the lower layers are not allowed to invoke operations corresponding to the upper-layer subsystems</u>.
  - There are no such restrictions in the modular-kernel approach, wherein *<u>modules are free to invoke each other without any constraints</u>*.

Example

Linux

5. Abstractions, processes, resources

- The operating system provides a level of abstraction or a layer of abstraction with the help of OSAL. OSAL stands for Operating System Abstraction Layer, a set of application programming interfaces (APIs) that a developer can use to quickly develop an application without considering the type of operating system, hardware, and background complexity of the operating system.

- Virtualization, on the other hand, is a technique that enables the creation of multiple virtual instances of a resource or system on a single physical machine. It abstracts the underlying hardware and resources, allowing multiple virtual environments to run independently on the same physical infrastructure.

Virtualizing the CPU -- Processes

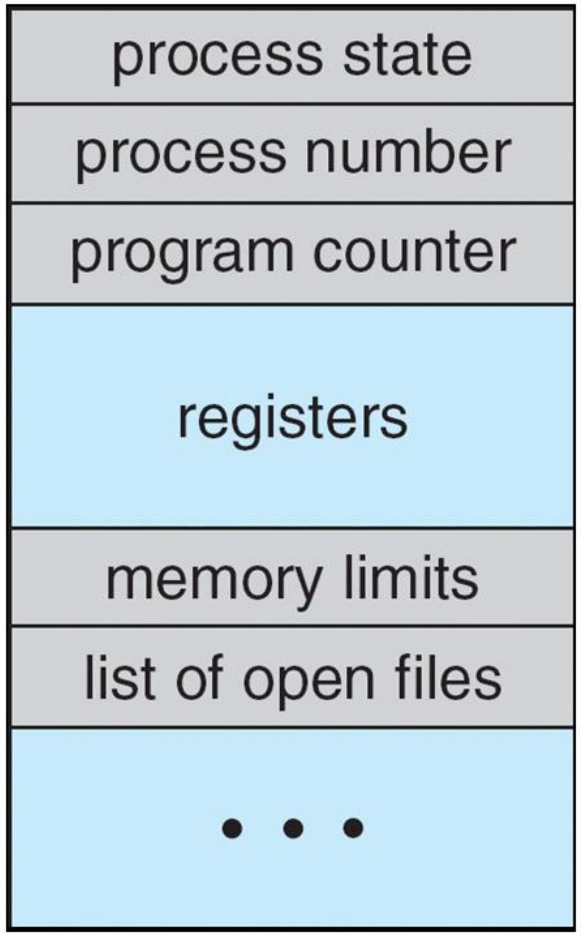
- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Process Control Block (PCB)

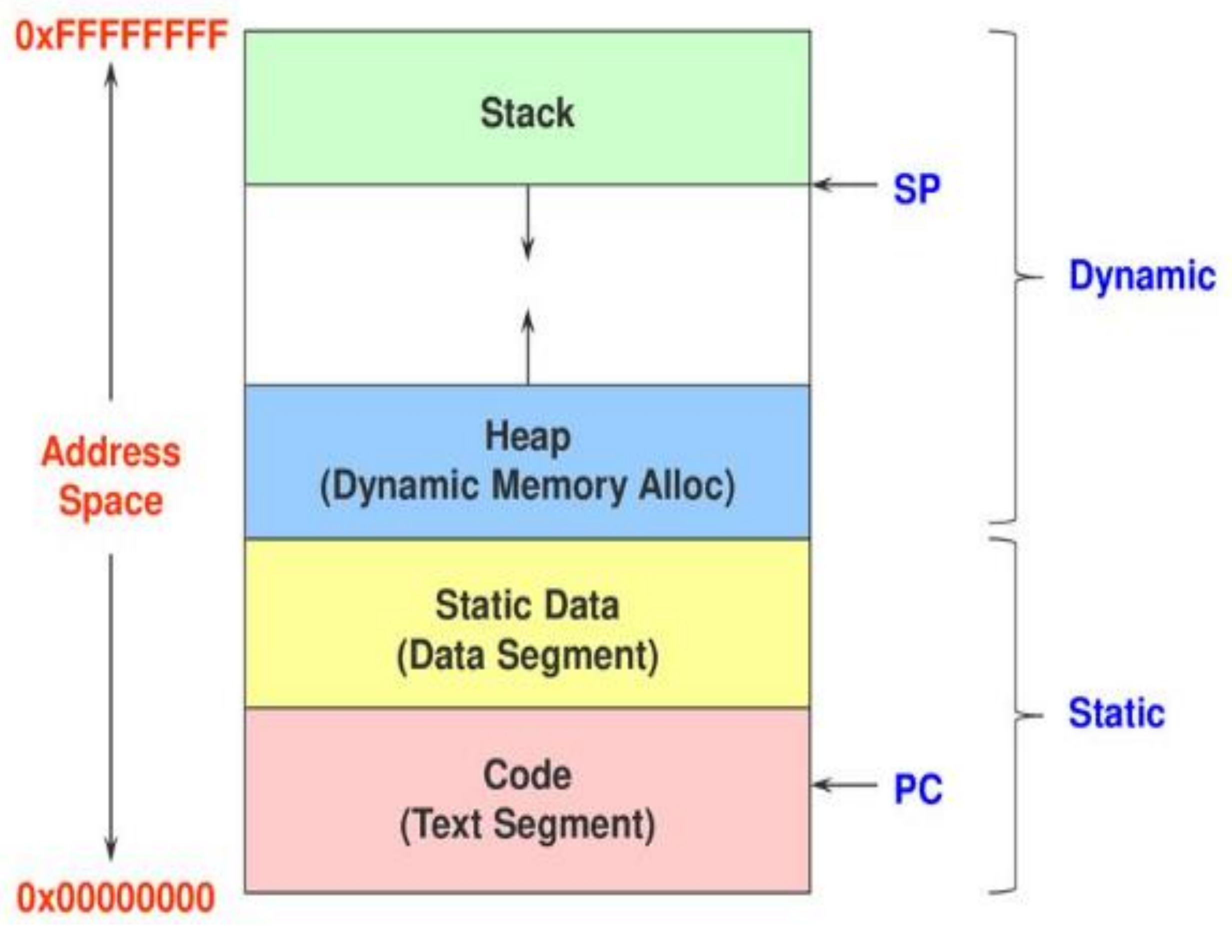
Information associated with each process

(also called **task control block**)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files



Address Space (memory abstraction)



- Address space refers to the range of memory addresses that a process can use to access memory.
- Each process in an operating system has its own address space, which represents a private and isolated memory region.
- The address space is typically divided into different sections, such as code segment, data segment, stack, and heap.

6. Influence of security, networking and multimedia

- Security in Operating Systems:
- User Authentication and Access Control: Operating systems implement various authentication methods, such as passwords, biometrics, and two-factor authentication, to verify the identity of users. Access control mechanisms restrict users' permissions and privileges, ensuring that they can only access resources they are authorized to use.
- System Hardening: Operating systems employ security measures like firewalls, intrusion detection systems, and antivirus software to safeguard against external and internal threats. These measures help detect and prevent malicious activities, such as unauthorized network access, malware infections, and data breaches.

- **Secure Communication: Operating systems provide secure communication protocols, such as Transport Layer Security (TLS) and Virtual Private Networks (VPNs), to ensure the confidentiality, integrity, and authentication of data transmitted over networks.**

- **Encryption: Operating systems offer encryption mechanisms to protect sensitive data stored on disk or transmitted over networks. Encryption algorithms and protocols help prevent unauthorized access and data theft.**

- **Networking in Operating Systems**
- Internet Connectivity: Operating systems provide networking protocols and drivers to establish connections to the internet, enabling users to browse the web, access online services, and communicate over email or messaging platforms.
- Network File Sharing: Operating systems support network file sharing protocols, such as Server Message Block (SMB) or Network File System (NFS), allowing users to access shared files and folders on remote systems.

- Printing and Peripheral Devices: Operating systems include networking protocols for printing and connecting peripheral devices, such as printers and scanners, over a network. This allows users to share these devices across multiple computers.

- Remote Access: Operating systems offer remote access capabilities, such as Remote Desktop Protocol (RDP) or Secure Shell (SSH), which enable users to access and control remote systems over a network. This is particularly useful for system administration and troubleshooting purposes.

- **Multimedia in Operating Systems**
- Streaming and Video Conferencing: Multimedia technologies enable real-time streaming of audio and video, facilitating applications like video conferencing and online media streaming.

- User Interfaces and Experience: Multimedia elements enhance user interfaces, making them more engaging and intuitive through the use of graphics, animations, and interactive elements.

- Gaming and Virtual Reality: Multimedia is foundational in gaming and virtual reality applications, creating immersive and interactive experiences.

- Entertainment Industry: Multimedia technologies have revolutionized the entertainment industry, from digital audio and video production to special effects in movies.