# “Module 8: Assignment 1 - Operating Systems”

Harsh Siddhapura

“Dr. Dinesh Sthapit”

“November 19, 2023”

**T1**

1. The essential prerequisites of an operating system encompass a variety of functions aimed at ensuring the effective management and coordination of computer resources. These requisites include:
    - Management of Hardware Resources: Efficiently overseeing hardware components like CPU, memory, disk storage, and peripheral devices to optimize their utilization.
    - User Interface: Providing a means of interaction between users and the computer system, which can take the form of a CLI, GUI, or a combination of both.
    - Communication btwn Hardware and Software: “Acting as an intermediary between application software and hardware, enabling smooth communication and ensuring the seamless execution of programs” (Wiley, 2020, p. 299).
    - Error Handling and Security: Implementing robust mechanisms for error handling and security features, detecting and managing errors gracefully, and establishing security protocols to safeguard against unauthorized access and maintain data integrity.

    These fundamental requirements collectively establish the foundation of an operating system, enabling it to provide a stable and user-friendly environment for executing applications and optimizing the use of computer resources.

2. A computer's startup, sometimes referred to as "booting" or "booting," is a sequence of important operations designed to initialize its hardware and software and enable it to function. “There are several steps in this method. Your computer starts POST when you turn it on, and the firmware in the ROM or BIOS then takes over. Important hardware elements including the CPU, memory, storage, and peripherals are verified to be operating by POST” (Wiley, 2020, p. 314).

    Following POST, the boot programme, which is housed in a tiny portion of the computer's boot region, is sought after by the BIOS. A solid state drive or hard disc is frequently used for this kind of storage. The task of loading the operating system into memory falls to the boot software.

The boot programme code is located in the boot sector. The bootloader loads the kernel and the underlying operating system into the computer's memory when it is enabled. “The software that manages the hardware and performs other essential tasks is called the kernel. When used for kernel management, it is in charge of installing and configuring various hardware components, including input/output devices, network interfaces, and storage management” (Wiley, 2020, p. 234).

The system moves to the user area and marks the machine as ready for human interaction after the "init" procedure has finished its task. Users may now login and start using the computer by seeing a login screen, sometimes known as a graphical user interface (GUI). To summarize, the boot process is the first stage that manages the changeover from the boot state to the operating state. It includes setting up hardware, loading devices, and preparing methods for user input.

**3.** In the realm of computer science, processes and threads stand out as fundamental concepts that play a pivotal role in executing programs and optimizing the use of system resources.

- Process: In a computer system, a process is essentially a standalone, self-contained unit of execution. It includes the memory location, execution resources, and execution context in addition to the programme code. To protect the security and integrity of the data, the processes function independently of one another.
- Procedures related to networking, resource allocation, and isolation (IPC) are specific to the process. Process isolation ensures that every process has its own set of services and that they don't communicate with one another. It uses messaging, shared memory, and system communication to coordinate processes. Unfortunately, the need to dedicate memory and resources makes the overhead of creating and maintaining the process significant.

- Threads: On the other hand, they are little and lightweight. Process threads operate independently, although they share the same memory and resources. A single process can contain several threads, each of which concurrently runs a different portion of the programme.

- The basic functions of threads are “to share resources, require less construction and maintenance work than processes, and integrate processes to reduce conflicts while accessing shared data. Talking about this issue helps people communicate, but be careful to prevent nationalism and misinformation” (Wiley, 2020, p. 299).

In essence, processes and threads serve as foundational elements in modern computing. Their understanding is pivotal for developing software applications that are not only efficient but also responsive, given their role in program execution, resource management, and concurrent processing.

**4.** In the domain of computer systems, the loading and execution operations form the foundational steps that transform a program from its stored state to active execution. These operations are crucial in ensuring that programs run seamlessly and efficiently on a computer.

- Loading:loading: The process of moving data from a solid-state or hard disc to the computer's memory is the first step in loading. Throughout this process, the computer's RAM (random access memory) will receive the binary code of the programme, data, and other resources from the operating system or loader.
- Link: Typically, this is a step in the submission process. To develop a single system, it involves merging many code structures and processing data across them. While dynamic links appear during runtime, static links happen before runtime.
- Execution: The programme begins the execution stage as soon as it is loaded into memory. “The CPU processes commands by analyzing and carrying them out one after the other. After retrieving instructions from memory, the CPU decodes them and carries out the specified operations” (Wiley, 2020, p. 423).
- DLL: When DLLs are running on systems that use shared libraries. These libraries save duplication and maximize resource utilization by including reusable code and resources that several applications may access.

The intricacy of installation and operation is something that developers, administrators, and anybody else working in software development or computer administration have to be aware of. For efficient operations, efficient use of resources, and overall performance, these steps formed the basis.

**T2**

1. Disk scheduling refers to the process of determining the order in which pending input/output (I/O) requests are serviced by a disk drive. The primary objective is to minimize seek time, reduce rotational latency, and optimize overall disk access. Below, we delve into the key aspects and significance of disk scheduling in optimizing disk operations:

    - Shorten the search time by "assigning the driver, which needs to be executed before the data in the letter is communicated, the task of reading and writing data on a disc drive, which has several tracks and sectors" (Wiley, 2020, p. 523).

    - Minimize Rotational Latency: "Latency is the duration of time that the disc space takes to spin under the disc head. The disc planning theory aims to decrease rotational delay by taking into account the disc arm's current location. This means estimating where necessary activities are located and planning I/O operations accordingly" (Wiley, 2020, p. 413).

    - Disc scheduling ensures that competing processes or applications share disc resources in an efficient and fair manner. By evaluating service demand and ensuring fairness for all running processes, the disc scheduling strategy keeps one process from controlling disc access.

    - Better delivery: One important performance parameter is the volume of data that is transmitted between discs and systems. By ensuring optimal disc utilization and reducing downtime, effective disc utilization planning aids in hardware optimization.

    - Boost system efficiency: "Disc timing affects the overall efficiency of the system. By reducing access time and latency, disc operations can become more responsive to users and

demands. In situations when there is a time constraint, such as real-time data access, this is extremely important" (Wiley, 2020, p. 534).

In conclusion, disk scheduling is a critical component of disk management, aiming to optimize the performance of disk operations in terms of seek time, rotational latency, fairness, throughput, and system responsiveness. Effective disk scheduling algorithms contribute to a more efficient and responsive computing environment, ensuring that disk resources are utilized judiciously for diverse workloads and user interactions.

**T3**

1. Network Operating Systems (NOS) are pivotal in the management and facilitation of communication and resource sharing within networked environments. These systems provide a range of critical services that are fundamental to the smooth operation of networked computing.
    - File sharing and printing: One of the main features offered by NOS is the ability to share files and printers over a network. The utilization of networked printers and data printing capabilities enhances resource efficiency and facilitates collaborative work.
    - User authentication and authorization: Network operating systems use strong authentication techniques to provide safe access to network services. To gain access, users must enter legitimate credentials. "Based on user roles and authorizations, the NOS manages user permission and grants access privileges. In order to maintain a centralized database of network data, including user accounts, devices, and resources, NOS offers directory services. It is simpler to manage resources, authenticate users, and run the network when there is only one directory" (Wiley, 2020, p. 623).
    - The NOS must integrate security components for access control and security. These characteristics protect network systems from possible security issues and illegal access. They include access control, logging, and electronic configuration protocols.

- When it comes to allocating and managing network resources, the network operating system (NOS) is essential. “The internet has resources available. Ensuring equitable distribution of network resources, optimizing bandwidth, and managing IP addresses are necessary for sustaining network performance” (Wiley, 2020, p. 354).

In summary, Network Operating Systems offer a comprehensive suite of services that enable effective communication, resource sharing, security, and management within a networked environment. These services contribute to the seamless operation of networks, supporting collaboration, data integrity, and efficient utilization of computing resources.

**T4**

1. Type 1 (native) and Type 2 (hosted) hypervisors are two distinct categories of virtualization technologies, each designed for specific use cases with differences in architecture, performance, and deployment scenarios.
    - Frequently referred to as a native or full hypervisor, a Type 1 hypervisor installs itself directly into the host hardware and functions apart from the host operating system. They function effectively because they have direct access to the body's resources. “Data centers and commercial settings frequently use type 1 hypervisors for server virtualization. One key advantage of Type 1 hypervisors is their ability to achieve high performance by eliminating host overhead by operating at the bare metal level. Due to minimal latency and separation from the host process, they are often more secure” (Wiley, 2020, p. 387).
    - Hypervisor type 2: Often referred to as a host hypervisor, it replaces the host operating system. “They utilize the host operating system's resources for virtualization while functioning as applications on it. Type 2 hypervisors include Parallels Desktop, Oracle VirtualBox, and VMware Workstation. Testing environments and desktop virtualization are two common uses for these hypervisors. They are easier to use and more accessible for desktop virtualization for developers and ordinary users” (Wiley, 2020, p. 398).

Comparison:

- Performance: Type 1 hypervisors generally offer better performance as they run directly on the hardware, while Type 2 hypervisors have additional layers, impacting performance to some extent.
- Use Cases: Type 1 hypervisors are ideal for server virtualization in data centers, where performance and resource efficiency are critical. Type 2 hypervisors are often used for development, testing, or running virtual machines on desktops.
- Resource Utilization: “Type 1 hypervisors have more direct access to hardware resources, resulting in more efficient utilization. Type 2 hypervisors share resources with the host OS, leading to slightly higher resource overhead” (Wiley, 2020, p. 287).

In conclusion, the choice between Type 1 and Type 2 hypervisors depends on the specific requirements of the virtualization environment. Type 1 hypervisors are well-suited for performance-critical server environments, while Type 2 hypervisors provide a more user-friendly experience for desktop and testing purposes.

**2.** Advantages of Virtual Machines are as follows:

- Resource Utilization: By “enabling several virtual machines (VMs) to operate on a single physical server, virtualization increases resource utilization. This provides efficiency and financial advantages over operating several physical servers” (Wiley, 2020, p. 23)..
- Offers total segregation between virtual machines. It increases security and stability since programmes and processes operating on a virtual machine are separate from those operating on other virtual machines. It also makes controlled testing and development possible.
- Scalability and flexibility: Virtualization makes the setup, use, and maintenance of virtual machines easier. The capacity to adapt allows for dynamic resource scalability as required. It is possible to quickly offer new virtual computers, which facilitates task shift.

Limitations of Virtual Machines are as follows:

- Performance overhead: “Virtualization offers some of the lowest performance overhead since the virtual machine shares resources with the underlying hypervisor. Performance-sensitive applications will continue to suffer, even if advancements in virtualization technology have reduced this cost” (Wiley, 2020, p. 376).
- Discussion of Service: In a virtualized environment, several virtual computers share the same hardware. If several virtual machines fight for the same resources, it might lead to decreased performance.
- Complexity: Large installations, in particular, can be difficult to manage in virtualized systems. Effective planning and management are crucial for improving resource allocation, tracking performance, and preserving security.

In summary, while virtual machines offer numerous advantages in terms of resource utilization, flexibility, and isolation, there are considerations such as performance overhead, resource contention, and security concerns that organizations need to weigh when adopting virtualization.

**T5**

1. The kernel is the core component of an operating system, serving as the central hub that manages system resources and facilitates communication between hardware and software. It is responsible for executing essential functions that enable the proper functioning of the operating system. The primary kernel functions can be broadly categorized into several key roles.
    - Process Management: One of the fundamental roles of the kernel is process management. “It oversees the creation, scheduling, and termination of processes within the system. This includes managing the execution of multiple processes concurrently, allocating resources to them, and ensuring their proper execution” (Wiley, 2020, p. 376).
    - Memory management is yet another essential element of vital processing. It is in charge of allocating and freeing up memory for the process and optimizing resource utilization. Each process has its own memory space thanks to the kernel's handling of virtual memory.

- File system administration: “This basic file management process includes the creation, reading, writing, and deleting of files. It perceives the file system that applications ask for as a low-level storage device operation” (Wiley, 2020, p. 478).
- System calls act as a link between the kernel and user applications. These features let applications make requests for specialized services including networking, data processing, and process rendering. Students receive these requests, give them their approval, and complete the necessary assignments.
- Handling of interruptions: The system handles interruptions brought on by external events or hardware. “The master pauses the execution flow in response to an interrupt, allowing the event to conclude on schedule. This means keeping the present state, starting an interrupt, and then going back to the original state” (Wiley, 2020, p. 386).

In summary, the primary kernel functions are crucial for the smooth operation of an operating system. Process management, memory management, file system management, device drivers, system calls, and interrupt handling collectively ensure the efficient utilization of resources, secure execution of processes, and effective communication between software and hardware components.

**T6**

1. In an operating system, the concept of process states refers to the various stages that a process goes through during its lifecycle. The understanding of these states is crucial for managing and scheduling processes efficiently. The typical process states include New, Ready, Running, Blocked, and Terminated.
    - New State: The process starts in the "new" state when it is initially created. In this state, the operating system allocates the necessary resources, initializes the Process Control Block (PCB) with relevant information, and prepares the process for execution.

- Ready State: Once the process is set up in the new state, it transitions to the "ready" state. Here, the process is loaded into the main memory and awaits its turn to be executed. Multiple processes can be in the ready state simultaneously, waiting for the CPU to be allocated.
- Running State: “When the operating system scheduler assigns the CPU to a process from the ready queue, it enters the "running" state. In this state, the process actively executes its instructions and utilizes the CPU for computation” (Wiley, 2020, p. 387).
- Blocked (Wait) State: Processes move to the "blocked" or "wait" state when they need to wait for a particular event or resource. For instance, “a process might enter this state when waiting for user input or data from external devices. In the blocked state, the process is temporarily halted until the event it is waiting for occurs” (Wiley, 2020, p. 476).
- Exit State: The final state in the process lifecycle is the "terminated" or "exit" state. A process enters this state when it completes its execution or is explicitly terminated. “In the terminated state, the operating system releases all resources allocated to the process, and the process is removed from the process table” (Wiley, 2020, p. 265).

The transitions between these states are influenced by various events and system decisions. These events include a process completing its execution, issuing an I/O request, encountering an error, or waiting for a specific event to occur. State transitions are often represented in a state diagram, providing a visual representation of how a process moves through different states based on specific conditions.

**2.** Threads are fundamental units of execution within a process and represent the smallest sequence of programmed instructions that can be managed by an operating system's scheduler. This concept of threads introduces several advantages over independent processes.

- The resource efficiency of threads is one of its best features. Because “they share the same resources, threads inside the same process are less burdened than those in separate processes. Compared to maintaining separate processes, thread creation, termination, and content

changes are frequently faster and need less resources. When completing many tasks at once, this performance is really helpful" (Wiley, 2020, p. 476).

- The process's execution threads must share the same memory area in order to exchange data and communicate easily. "Unlike distinct processes, which usually need communication methods like message forwarding or shared memory, threads can communicate directly across a shared network. It is simpler for threads operating in the same process to communicate and coordinate when there is a shared memory region" (Wiley, 2020, p. 487).
- Reactivity: Adding news to your app makes it more reactive. In a multithreaded environment, the execution of multiple threads within the same process does not interfere with the operation of a single thread. "The programme will respond more quickly overall if other threads can operate while one thread waits for an I/O operation. For systems that require instantaneous or interactive processing, this response is essential" (Wiley, 2020, p. 344).

In summary, threads provide a more efficient and flexible approach to concurrent execution within a process. Their advantages include resource efficiency, shared memory space, improved responsiveness, simplified communication, enhanced throughput, and lightweight characteristics. These attributes make threads a valuable concurrency model in the design and implementation of modern operating systems and applications.

**T7**

1. Because it guarantees smooth system operation and optimal use of CPU resources, CPU scheduling is a crucial part of performance management. "Enhancing process integrity, maximizing resource utilization, cutting down on turnaround times, and enhancing performance are the main goals of CPU scheduling" (Wiley, 2020, p. 476).

   Fairness, or giving an equitable share of CPU time resources to each trip and preventing a single trip in resource management, is one of the main goals. All processes will have adequate CPU time to execute, regardless of demand or priority, thanks to this fairness. Making the most use of the

resources at hand is another crucial goal. CPU scheduling aims to optimize system performance by keeping the CPU busy and reducing idle time.

Shortening turnaround times is necessary to improve performance. “Turnaround time is the total amount of time it takes for a transaction to finish from submission to fulfillment. CPU scheduling that is effective reduces turnaround time, enabling processes to complete tasks faster. Moreover, CPU scheduling aims to improve performance by reducing the amount of time that processes in queues must wait. Using a task scheduler, which ranks the work according to predetermined standards like milestones, allows for this” (Wiley, 2020, p. 486).

For the purpose of ensuring integrity, increasing resource utilization, and reducing resource consumption, CPU scheduling is essential. It's time to shift gears and boost productivity at work; each of these elements plays a part.

**T8**

1. Virtual memory and demand paging are integral components of modern operating systems, contributing to efficient memory management and system performance.
    - Virtual Memory: It is a memory management method that offers "the greatest possible resolution of storage resources available on a particular machine." It enables computers to compensate for lack of physical memory by combining RAM (random access memory) and disc space. The operating system simulates a bigger memory footprint and binds to processes, allowing them to operate as if they had more memory than the system has.
    - Demand Paging: The request page is a mechanism that functions similarly to virtual memory. It entails only loading data into memory when a certain operation requests it or "asks" for it during execution. The operating system does not load all programmes into memory at launch; instead, it only loads the pages required by the programme while it is executing.

In summary, “virtual memory, coupled with demand paging, allows for efficient utilization of memory resources by providing processes with an illusion of a larger memory space and loading data into memory only when it is needed. This contributes to improved system performance and responsiveness” (Wiley, 2020, p. 387).

**2.** The concept of locality plays a crucial role in memory management and significantly influences the efficiency of a computer system. Locality refers to the tendency of a program to access a relatively small portion of its address space for a certain period. There are two main types of locality: temporal locality and spatial locality.

- Temporal Locality: The concept of temporal locality, “which is often referred to as the peripheral time principle, asserts that a particular memory location will very certainly recur in the future. The premise behind this approach stems from the fact that software repeatedly accesses the same memory area. Modern systems frequently employ caching techniques to save time by storing the most recent requests for data in the cache. This lessens the need to repeatedly retrieve the same item from slow main memory” (Wiley, 2020, p. 467).
- Spatial Locality: The concept of spatial locality posits that upon accessing one memory location, other memory areas are also immediately accessible. Programmers often access data that is close to data that was previously available. By arranging data to take into consideration the possibility of adjacent memory accesses, efficient memory management maximizes the amount of local space that is accessible. To do this, you might need to group files together, optimize your memory layout, or use prefetch techniques to obtain nearby files.

Impact on Memory Management:

- Cache efficiency: The locality of the cache system determines its performance. “It is possible to decrease cache hits and increase the speed of data retrieval by prioritizing localized and long-term data caching” (Wiley, 2020, p. 365).

- Page replacement policy: “To determine which pages to save in physical memory while using virtual memory, the operating system makes use of this policy. Space has an impact on this code since large pages are more likely to be kept for later use” (Wiley, 2020, p. 397).
- Optimized data structures: Native knowledge is common when programmers and compilers create data structures. For example, the purpose of arrays and associated structures is to improve geolocation and ensure that crucial data is consistently or reliably retained in RAM.

In conclusion, the concept of locality, encompassing both temporal and spatial aspects, is a cornerstone in memory management strategies. It guides the design of memory systems, caches, and algorithms to exploit predictable patterns in program behavior, ultimately enhancing the overall performance and responsiveness of computer systems.

**T9**

1. Advantages of Docker containers are as follows:

    - Portability: Docker containers provide for consistency across environments by encapsulating software and their dependencies. This modification simplifies the deployment process and solves the "works on my computer" issue.

    - Insulation: “Working in containers is bright and well-insulated. With its own system files, libraries, and processes, each container operates independently, removing any issues with dependencies or applications” (Wiley, 2020, p. 469).

    - Because “Docker containers include the operating system kernel, they are more efficient than virtual computers. They are less demanding and use fewer resources, which makes better use of those resources possible” (Wiley, 2020, p. 467).

Limitations are as follows:

- Security problems: If the necessary procedures are not followed, security risks persist even in the case when the containers are divided. directing The possibility of escape boxes and the coordination of essential activities are crucial factors.
- Persistence and Stateful Applications: Because Docker containers are designed to be stateless, it can be challenging to retain persistent data. High-end programmes may need extra settings or alternative ways to manage persistent data.
- Learning Curve: “Using Docker may need a learning curve for businesses that are unfamiliar with container principles. A successful deployment requires an understanding of Dockerfile creation, image management, and container orchestration” (Wiley, 2020, p. 487).

In summary, Docker containers offer numerous advantages in terms of portability, isolation, resource efficiency, and rapid deployment. However, users should be mindful of security considerations, persistence challenges for stateful applications, and potential compatibility issues for certain workloads.

**2.** Docker containers and traditional virtualization are both technologies that address the challenges of deploying and managing applications, but they differ in their approach and architecture.

- Running several virtual machines (VMs) on a hypervisor—a software layer that sits between the operating system and the hardware is a common practice associated with virtualization. “Every virtual machine includes programmes and executable files specific to it. Multiple processes and functionalities can run on the same hardware thanks to virtualization, which enables strict application separation” (Wiley, 2020, p. 587).
- Docker containers, on the other hand, operate at the application level. They come in a compact, portable bundle with a programme and all of its dependencies. Although they function as separate processes, containers share the operating system's kernel and resources. This suggests that containers start up faster and weigh more than virtual machines.

Their extra role is the basis for the relationship between virtualization and containers. “To boost flexibility and efficiency, virtualization technologies are commonly used with containers. Docker containers provide further isolation and application protection, while virtualization allows hardware separation and makes it easier to integrate several operating systems” (Wiley, 2020, p. 597).

## References

Irv Englander. (2020). *ARCHITECTURE OF COMPUTER HARDWARE, SYSTEMS SOFTWARE, AND NETWORKING : an information... technology approach*. John Wiley.