**SISTEM OPERASI**

**PRACTICE EXERCISES**

****

DISUSUN OLEH:

Nauril Putri Hadining Tyas

3124521012

Dosen pembimbing : Dr Ferry Astika Saputra ST, M.Sc

**PROGRAM STUDI D3 TEKNIK INFORMATIKA PSDKU LAMONGAN**

**DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2025**

1. **What are the three main purposes of an operating system?**

**Answer:**

|  |
| --- |
| The three main purposes of an operating system are:   1. **Resource Management**: The operating system allocates hardware resources such as CPU time, memory, and storage among various applications and users. It ensures that no single application monopolizes resources and isolates applications to protect them from errors or security vulnerabilities. 2. **User and Application Interface**: The OS provides an interface for users (via GUI or CLI) and applications to interact with hardware. It abstracts complex hardware details, making it easier for programmers to develop software and for users to operate the system. 3. **Service Provision**: Operating systems offer common services such as file management, networking, and device control. These services allow applications to run on different hardware without needing modification, ensuring compatibility and efficiency. |

1. **We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to "waste" resources? Why is such a system not really wasteful?**

**Answer:**

|  |
| --- |
| It's appropriate to "waste" resources when doing so improves:   * **Responsiveness:** Allocating extra resources to critical tasks for faster response times. * **User Experience:** Simplifying tasks for users, even if it means using more resources. * **Security:** Implementing security measures that consume resources but protect the system.   Not Really Wasteful**:** The resources are being used to achieve a higher-level goal, like better performance or enhanced security. |

1. **What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?**

**Answer:**

|  |
| --- |
| Meeting Strict Timing Constraints: Real-time systems require tasks to be completed within specific timeframes. The programmer must ensure the OS can guarantee these deadlines. |

1. **Keeping in mind the various definitions of operating system, consider whether the operating system should include applications such as web browsers and mail programs. Argue both that it should and that it should not, and support your answers.**

|  |
| --- |
| **Arguments for Including Web Browsers and Mail Programs in an OS:**   1. **Integrated User Experience:**  * Including core applications like web browsers and email clients by default provides a seamless user experience. Users can immediately access essential internet functions without needing to search for and install additional software. This integration allows for deeper optimization, enhancing performance and stability.  1. **Standardization and Consistency:**  * Operating system developers can ensure standardization and consistency across platforms by including these applications. This reduces compatibility issues and fragmentation, and allows for optimization tailored to specific hardware.  1. **Enhanced Security and Updates:**  * Integrating security features directly into these applications allows for better protection against threats. Centralized updates through the OS ensure users have the latest security patches, and the OS developer has full control to respond quickly to vulnerabilities.   **Arguments Against Including Web Browsers and Mail Programs in an OS:**   1. **Bloatware and Resource Usage:**  * Including too many applications can lead to bloatware, consuming excessive system resources. This wastes storage and memory, potentially slowing down performance, especially on resource-constrained devices.  1. **User Choice and Flexibility:**  * Users have diverse preferences for applications. Including specific applications by default limits user choice and flexibility. Users may prefer alternative third-party applications with different features or interfaces.  1. **Antitrust Concerns and Competition:**  * Including applications by default can raise antitrust concerns, giving an unfair advantage to the OS developer. This can stifle competition from third-party application developers and limit innovation.  1. **Core OS Focus:**  * "The core function of an OS is to manage hardware resources and provide essential services. Adding complex applications can distract from this core function. Keeping the OS lean allows for more efficient updates and increased stability." |

1. **How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security)?**

**Answer:**

|  |
| --- |
| Kernel mode allows the OS to execute privileged instructions and access all hardware resources, while user mode restricts applications to a subset of instructions and resources. This prevents user programs from interfering with the OS or other programs, enhancing system stability and security. |

1. **Which of the following instructions should be privileged?**
   1. **Set value of timer.**
   2. **Read the clock**
   3. **Clear memory**
   4. **Issue a trap instruction**
   5. **Turn off interrupts**
   6. **Modify enteries in device-status table.**
   7. **Switch from user to kernel mode.**
   8. **Access I/O device**

**Answer:**

|  |
| --- |
| These instructions are considered privileged because they have the potential to directly impact the system's hardware or core operating system functions. If user-level programs were allowed to execute these instructions without restriction, it could lead to system instability, security breaches, or data corruption.  a) Set value of timer: Modifying the timer can disrupt the operating system's scheduling and timekeeping mechanisms.  c) Clear memory: Allowing user-level programs to clear memory could result in the deletion of critical operating system data or data belonging to other applications.  e) Turn off interrupts: Disabling interrupts can prevent the operating system from responding to hardware events or handling critical errors.  f) Modify entries in device-status table: Altering device status entries can lead to conflicts with device drivers and cause hardware malfunctions.  g) Switch from user to kernel mode: This is a fundamental operating system function that must be strictly controlled to maintain system security.  h) Access I/O device: Direct access to I/O devices could allow user-level programs to bypass the operating system's control and potentially cause conflicts or damage hardware. |

1. **Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.  
   Answer:**

|  |
| --- |
| Two difficulties that could arise with such a scheme are:   1. Limited OS Flexibility: If the OS cannot modify its own code or data, it becomes difficult to implement updates, patches, or dynamic changes. This can lead to a stagnant and potentially insecure system. 2. Resource Inefficiency: A fixed memory partition may not be optimally sized for all scenarios. If it's too small, the OS might lack necessary space; if too large, it wastes valuable memory. |

1. **Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?  
   Answer:**

|  |
| --- |
| Two possible uses of multiple modes of operation are:   1. **Virtualization:** Additional modes can be used to create virtual machines, where each VM runs in its own isolated mode, enhancing security and resource management. 2. **Security Levels:** Multiple modes can implement fine-grained security policies, where different levels of privilege are assigned to various system components or applications. |

1. **Timers could be used to compute the current time. Provide a short description of how this could be accomplished.  
   Answer:**

|  |
| --- |
| A timer can be used to compute the current time by:   1. Initializing a counter with a starting time: At system startup, the OS initializes a counter with a known starting time (e.g., the Unix epoch). 2. Setting the timer to generate interrupts at regular intervals: The timer hardware is configured to generate interrupts at fixed intervals (e.g., every millisecond). 3. Incrementing the counter on each interrupt: Each time the timer interrupt occurs, the OS increments the counter. 4. Calculating the current time: The current time is calculated by adding the elapsed time (based on the counter) to the starting time. This can be done by converting the counter value to a human-readable format. |

1. **Give two reasons why caches are useful.**

|  |
| --- |
| **Two reasons why caches are useful:**   * **Speed Improvement:** Caches store frequently accessed data, reducing the need to access slower main memory or storage devices. * **Reduced Latency:** By providing faster access to data, caches minimize the time it takes for the system to retrieve information.   **What problems do they solve?**   * **Memory/Storage Latency:** Caches mitigate the delay caused by accessing slower main memory or storage devices. * **Bandwidth Bottlenecks:** They reduce the load on memory or storage buses by serving data from a faster, local source. |

**What problems do they cause?**

|  |
| --- |
| * Cache Coherence: In multi-core or multi-processor systems, ensuring that all caches have consistent data can be challenging. * Complexity: Caches add complexity to the system architecture and require sophisticated algorithms for data management and replacement. * Cost: Larger and faster caches add to the overall system cost. |

**If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?Answer:**

|  |
| --- |
| While a cache as large as the device it's caching would theoretically eliminate the need for the device in terms of data retrieval speed, there are several reasons why this is not practical:   1. Cost: Caches are typically made of faster, more expensive memory (e.g., SRAM) than the devices they cache (e.g., DRAM or hard drives). Making a cache as large as a disk would be prohibitively expensive. 2. Volatility: Caches are often volatile, meaning they lose data when power is lost. Disks, on the other hand, provide persistent storage. 3. Durability and Reliability: Devices like disks are designed for long-term data storage and are more durable than typical cache memory. 4. Power Consumption: Larger caches consume more power, which can be a significant issue in portable devices or data centers. 5. Technological Limitations: There are physical and technological limitations to how large and fast a cache can be made. |

1. **Distinguish between the client-server and peer-to-peer models of distributed systems.**

**Answer:**

|  |
| --- |
| Client-Server Model:   * Structure: In a client-server model, there is a centralized server that provides resources or services to multiple clients. * Roles: Clients request services, and the server provides them. * Centralization: The server is a central point of control and data storage. * Scalability: Scalability is limited by the server's capacity. * Reliability: The system's reliability depends on the server's availability. |

|  |
| --- |
| Peer-to-Peer (P2P) Model:   * Structure: In a P2P model, all nodes (peers) in the system have equal roles and capabilities. * Roles: Each peer can act as both a client and a server, sharing resources and services directly with other peers. * Decentralization: There is no central server, and control is distributed among the peers. * Scalability: P2P systems can be highly scalable as the workload is distributed among the peers. * Reliability: The system is more resilient to failures as there is no single point of failure. |

1. **How do clustered systems differ from multiprocessor systems?**

|  |
| --- |
| * **Multiprocessor systems:** These systems contain multiple CPUs within a single computer, sharing resources like memory and I/O devices. They are tightly coupled, meaning the processors have direct access to the same resources and communicate with each other quickly. * **Clustered systems:** These systems consist of multiple independent computers (nodes) connected through a network. Each node has its own CPU, memory, and I/O devices. They are loosely coupled, meaning the communication between nodes happens over the network, which can be slower than in multiprocessor systems. |

**What is required for two machines belonging to a cluster to cooperate in providing a highly available service?**

**Answer:**

|  |
| --- |
| Requirements for High Availability in a Cluster:   * Shared Storage: Both machines must have access to shared storage to access the same data. * Heartbeat Mechanism: A mechanism to detect when a node fails. * Failover Mechanism: When a node fails, the other node must take over its workload. * Cluster Management Software: Software to coordinate the cluster's operation. |

1. **Consider a computing cluster consisting of two nodes running a database. Describe two different ways in which the two nodes could share access to the data on the disk. Discuss the benefits and disadvantages of each.**

**Answer:**

|  |
| --- |
| **Shared Disk:**   * Both nodes have direct access to the same disk. * **Benefits:** Simple to implement, high performance. * **Disadvantages:** Requires hardware that supports concurrent access, risk of data corruption if not managed correctly.   **Shared Nothing:**   * Each node has its own disk, and data is replicated between the nodes. * **Benefits:** More scalable, less risk of data corruption. * **Disadvantages:** More complex to implement, higher latency. |

1. **What is the purpose of interrupts? How does an interrupt differ from a trap? Can traps be generated intentionally by a user program? If so, for what purpose?**

**Answer:**

|  |
| --- |
| **Purpose of Interrupts:** To signal the CPU that an event has occurred, requiring attention.  **Interrupts vs. Traps:**   * **Interrupts** are generated by hardware. * **Traps** are generated by software.   **User-Generated Traps:** Yes, user programs can generate traps intentionally.   * **Purpose:** To request services from the OS (system calls). |

1. **Explain how the Linux kernel variables 'HZ' and 'jiffies' can be used to determine the number of seconds the system has been running since it was booted.**

**Answer:**

|  |
| --- |
| HZ: The number of timer interrupts per second.  Jiffies: A counter that increments with each timer interrupt.  Calculation: Seconds since boot = jiffies / HZ. |

1. **Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU's execution load.**

|  |
| --- |
| * **How does the CPU interface with the device to coordinate the transfer?**   **Answer:**  **CPU Interface:** The CPU programs the DMA controller with the source and destination addresses, the number of bytes to transfer, and the direction of the transfer.   * **How does the CPU know when the memory operations are complete?**   **Answer:**  **Completion Signal:** The DMA controller generates an interrupt when the transfer is complete**.**   * **The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.**   **Answer:**  **Interference**: Yes, the DMA controller can interfere with the execution of user programs.  **Forms of Interference**: Memory bus contention, cache coherence issues. |

1. **Some computer systems do not provide a privileged mode of operation in hardware. Is it possible to construct a secure operating system for these computer systems? Give arguments both that it is and that it is not possible.**

**Answer:**

|  |
| --- |
| **Arguments that it is possible**:   1. Software-Based Isolation: Security can be enforced through software techniques like memory segmentation, process isolation, and access control lists. These techniques can create virtual boundaries between processes, preventing unauthorized access to memory or other resources. 2. Virtualization: Even without hardware support for privileged modes, a hypervisor can create virtual machines, each with its own isolated environment. This allows multiple operating systems or applications to run securely on the same hardware. 3. Language-Based Security: Using memory-safe languages can prevent memory corruption and other vulnerabilities. These languages provide built-in mechanisms to ensure that programs cannot access memory outside of their allocated regions.   **Arguments that it is not possible**:   1. Lack of Hardware Enforcement: Without hardware support for privileged modes, it's difficult to prevent malicious software from directly accessing or manipulating system resources. Software-based security mechanisms can be bypassed or subverted. 2. Performance Overhead: Software-based security mechanisms can introduce significant performance overhead, as they require additional checks and validations at runtime. 3. Complexity: Building a secure OS without hardware support requires complex software techniques, making it more prone to bugs and vulnerabilities. This increases the risk of security flaws that could be exploited by attackers. |

1. **Many SMP systems have different levels of caches; one level is local to each processing core, and another level is shared among all processing cores. Why are caching systems designed this way?**

**Answer:**

|  |
| --- |
| Caching systems are designed this way to balance performance and efficiency.   1. **Local Caches (L1):** Provide very fast access to data for each core, reducing latency and contention. 2. **Shared Caches (L2/L3):** Allow data sharing and communication between cores, reducing redundancy and improving overall performance.   This hierarchical design optimizes data locality and reduces the need to access slower main memory. |

1. **Arrange the following storage systems from slowest to fastest:**

|  |
| --- |
| The storage systems ranked from slowest to fastest are:  f. Magnetic tapes  c. Optical disk  a. Hard-disk drives  e. Nonvolatile memory  d. Main memory  g. Cache  b. Registers |

1. **Consider an SMP system similar to the one shown in Figure 1.8. Illustrate with an example how data residing in memory could in fact have a different value in each of the local caches.**

**Answer:**

|  |
| --- |
| Imagine two cores, Core A and Core B, in an SMP (Symmetric Multiprocessing) system. Both have their own local caches. Let's say a variable 'x' is initially stored in main memory with a value of 10.   1. **Core A reads 'x':** Core A's cache now stores 'x' = 10. 2. **Core B reads 'x':** Core B's cache also stores 'x' = 10. 3. **Core A modifies 'x':** Core A changes the value of 'x' to 20. Its local cache is updated to 'x' = 20, but main memory and Core B's cache still hold 'x' = 10.   Now, if Core B tries to read 'x', it will read the outdated value of 10 from its cache, not the updated value of 20. This is a classic example of the cache coherence problem, where different caches hold inconsistent values of the same data. |

1. **Discuss, with examples, how the problem of maintaining coherence of cached data manifests itself in the following processing environments:**

|  |
| --- |
| a. **Single-processor systems**   * Manifestation: In single-processor systems, cache coherence is primarily a concern when dealing with I/O operations. If an I/O device modifies data in main memory, the cache may hold a stale (outdated) copy of that data. * Example: Imagine a scenario where a program reads a file from disk into its cache. If another process or an I/O operation directly modifies that file on the disk, the cached copy will not reflect the changes until the cache is invalidated or updated.   b. **Multiprocessor systems**   * Manifestation: In multiprocessor systems, multiple cores can have their own local caches. If these caches contain copies of the same memory location, and one core modifies the data, the other caches will hold inconsistent data. * Example: Consider two cores running threads that access a shared variable. If one thread updates the variable, the cache of the other core will not automatically reflect this change, leading to inconsistencies.   c. **Distributed systems**   * Manifestation: In distributed systems, data can be replicated across multiple machines. If one machine modifies a piece of data, the copies on other machines become inconsistent. * Example: In a distributed database, if one node updates a record, the replicas of that record on other nodes must also be updated to maintain consistency. This requires complex protocols to ensure all copies are eventually consistent. |

1. **Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.**

**Answer:**

|  |
| --- |
| A common mechanism is memory segmentation and paging. The OS divides memory into segments or pages and maintains a table that maps virtual addresses to physical addresses, along with access permissions (read, write, execute). When a program tries to access memory, the MMU (Memory Management Unit) checks the permissions. If the access is unauthorized, a trap is generated, and the OS terminates the program. |

1. **Which network configuration—LAN or WAN—would best suit the following environments?**

**Answer:**

|  |
| --- |
| **a.A campus student union**   * **LAN (Local Area Network)**: A student union is typically a single building or a small area, making LAN the most suitable.   **b. Several campus locations across a statewide university system**   * **WAN (Wide Area Network)**: Connecting campuses across a state requires a WAN to cover the large geographical distances.   **c. A neighborhood**   * **LAN or WAN**: Depending on the size of the neighborhood. A smaller neighborhood might use a LAN, while a larger one might use a WAN or a combination of both. |

1. **Describe some of the challenges of designing operating systems for mobile devices compared with designing operating systems for traditional PCs.**

**Answer:**

|  |
| --- |
| * Resource Constraints: Mobile devices have limited CPU, memory, and battery resources. * Power Management: OS must optimize power usage to extend battery life. * Touchscreen Interface: OS must provide a user-friendly touch interface. * Wireless Connectivity: OS must handle various wireless technologies (Wi-Fi, cellular, Bluetooth). * Security: Mobile devices are vulnerable to malware and data theft, requiring robust security measures. |

1. **What are some advantages of peer-to-peer systems over client-server systems?**

**Answer:**

|  |
| --- |
| * Scalability: P2P systems can handle a large number of users without a central server bottleneck. * Resilience: P2P systems are more resilient to failures, as there is no single point of failure. * Cost-effectiveness: P2P systems can be cheaper to set up and maintain, as they don't require expensive servers. |

1. **Describe some distributed applications that would be appropriate for a peer-to-peer system.**

**Answer:**

|  |
| --- |
| * **File Sharing:** Applications like BitTorrent allow users to share files directly with each other. * **Content Delivery:** P2P networks can distribute content (e.g., videos, software updates) efficiently. * **Decentralized Social Networks:** P2P systems can enable social networks without central control. *  **Distributed Computing:** P2P networks can harness the collective computing power of many devices. |

1. **Identify several advantages and several disadvantages of open-source operating systems. Identify the types of people who would find each aspect to be an advantage or a disadvantage**

**Answer:**

|  |
| --- |
| **Advantages of Open-Source Operating Systems:**   1. **Transparency and Customization:**    * **Description:** The source code is publicly available, allowing users to inspect, modify, and distribute it.    * **Advantage for:** Developers, advanced users, and organizations requiring specific functionalities. 2. **Community Support:**    * **Description:** Large communities of developers and users provide support, documentation, and troubleshooting.    * **Advantage for:** Users who need assistance, developers seeking collaboration, and organizations with limited support resources. 3. **Cost-Effectiveness:**    * **Description:** Open-source OS are typically free of licensing fees, reducing costs.    * **Advantage for:** Individuals, small businesses, and organizations with budget constraints. 4. **Security:**    * **Description:** The open nature of the code allows for rapid identification and patching of security vulnerabilities by the community.    * **Advantage for:** Security-conscious users and organizations.   **Disadvantages of Open-Source Operating Systems:**   1. **Compatibility Issues:**    * **Description:** Hardware and software compatibility can be inconsistent due to the wide range of distributions and configurations.    * **Disadvantage for:** Users seeking seamless integration with specific hardware or software. 2. **Lack of Formal Support:**    * **Description:** There is no single entity responsible for providing formal support or warranties.    * **Disadvantage for:** Businesses and organizations that require guaranteed support and service-level agreements. 3. **Learning Curve:**    * **Description:** Configuring and troubleshooting open-source OS may require technical expertise.    * **Disadvantage for:** Non-technical users and those seeking a plug-and-play experience. 4. **Potential for Fragmentation:**    * **Description:** The existence of multiple distributions and forks can lead to fragmentation, making it difficult to maintain consistency and compatibility.    * **Disadvantage for:** Developers targeting a unified platform and users seeking a standardized experience. |