RU CT questions 2021

Class Test #1

1. Define Operating System. What are the different services of an operating system? Discuss them.

Answer:

An operating system (OS) is a software component that manages computer hardware and software resources and provides common services for computer programs. It acts as an intermediary between the user and the computer hardware, enabling users to interact with the system and execute applications efficiently. The OS also manages system resources, such as memory, processors, devices, and file systems.

The services provided by an operating system can vary, but here are some of the key services commonly offered:

1. Process Management: The OS manages the execution of processes (or tasks), which are programs in execution. It allocates system resources, schedules processes for execution, and provides mechanisms for interprocess communication and synchronization.

2. Memory Management: This service is responsible for managing the computer's memory resources. It includes allocating memory to processes, tracking memory usage, and implementing memory protection to prevent one process from interfering with another.

3. File System Management: The OS provides a file system that organizes and manages files on storage devices, such as hard drives or solid-state drives. It handles file creation, deletion, and access, as well as directory management and file permissions.

4. Device Management: Operating systems interact with hardware devices, including input/output devices (e.g., keyboards, mice, printers) and storage devices. They provide device drivers to communicate with these devices, handle device interrupts, and manage device queues.

5. User Interface: The operating system provides a user interface (UI) that allows users to interact with the computer system. This can include command-line interfaces (CLI), graphical user interfaces (GUI), or a combination of both, enabling users to run programs, manage files, and configure system settings.

6. Network Management: In modern operating systems, network management services facilitate communication between computers in a network. This includes managing network connections, protocols, and addressing, as well as providing network security mechanisms.

7. Security: Operating systems implement security measures to protect the system and its resources from unauthorized access, malware, and other threats. This includes user authentication, access control mechanisms, encryption, and auditing.

8. Error Detection and Handling: The OS detects and handles errors that occur during system operation. It includes error reporting, logging, and recovery mechanisms to minimize the impact of errors on system stability and data integrity.

9. System Utilities: Operating systems often provide utility programs to perform various system management tasks. These utilities can include disk defragmentation, backup and restore tools, system monitoring tools, and software installation tools.

Overall, the services provided by an operating system are designed to facilitate efficient and secure execution of computer programs, manage system resources effectively, and provide a user-friendly interface for users to interact with the computer system.

2. *Differentiate program & process. What do you mean by Program Control Block? Discuss different components of a Program Control Block.*

Answer:

Program vs. Process:

A program is a set of instructions or code written in a programming language. It is a passive entity that resides on the storage media, such as a hard drive, and does not perform any actions by itself. A program becomes active when it is loaded into memory and executed by the operating system.

On the other hand, a process is an active entity that represents an instance of a program in execution. It is created by the operating system when a program is loaded into memory and starts executing. A process consists of the program's code, data, and resources allocated to it during runtime. Each process has its own memory space and execution context.

Program Control Block (PCB):

A Program Control Block (PCB), also known as a Process Control Block, is a data structure maintained by the operating system for each process. It contains essential information about the process and provides the operating system with the necessary details to manage and control the execution of the process. The PCB is typically stored in the operating system's memory and is updated as the process progresses.

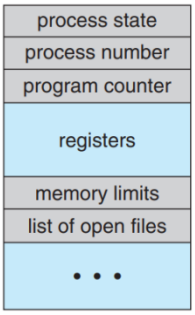


Figure 3.3 Process control block (PCB).

Components of a Program Control Block:

1. Process Identifier (PID): A unique identification number assigned to each process, allowing the operating system to differentiate between different processes.

2. Process State: It indicates the current state of the process, such as running, ready, blocked, or terminated. The state is updated by the operating system as the process transitions between different states.

3. Program Counter (PC): The PC stores the address of the next instruction to be executed within the process. It helps in maintaining the execution flow of the process.

4. CPU Registers: The PCB contains information about the process's CPU registers, which include general-purpose registers, stack pointers, and instruction registers. This information is crucial for context switching and restoring the process's state.

5. Memory Management Information: The PCB stores information about the memory allocated to the process, including the base address and limit of the process's memory space. It helps in managing the process's memory and protecting it from accessing unauthorized memory areas.

6. Priority: Some operating systems assign priorities to processes to determine their order of execution. The PCB may include a priority field that indicates the priority level of the process.

7. Open Files: If a process has any open files or input/output (I/O) devices associated with it, the PCB maintains a list or pointers to these resources. It ensures proper management and control of file and device access.

8. Accounting Information: The PCB may include fields for collecting accounting-related data, such as CPU usage, execution time, and I/O statistics. This information is useful for performance monitoring and resource allocation decisions.

9. Scheduling Information: The PCB may contain details related to the process's scheduling, such as the scheduling algorithm used, time slices assigned, and the process's scheduling history.

The Program Control Block serves as a repository of critical information about a process, enabling the operating system to manage and control processes efficiently, facilitate context switching, and ensure proper resource allocation and protection.

3. What is context switch? Discuss context switching between two processes.

Answer:

A context switch is the process of saving and restoring the state of a process or thread so that it can be paused and another process can be scheduled and resumed by the operating system. When a context switch occurs, the operating system saves the current execution context of a process, including the values of CPU registers, program counter, and other relevant information, and restores the saved context of another process to continue its execution.

Context switching is necessary for multitasking, where multiple processes or threads are running concurrently on a single processor. Here's a step-by-step overview of how a context switch between two processes typically occurs:

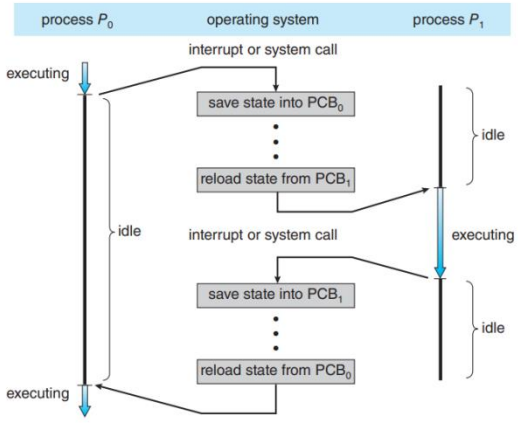


Figure 3.6 Diagram showing context switch from process to process.

1. Save Current Process State: When the operating system decides to switch from the currently running process (let's call it Process A), it saves the current state of Process A, which includes the values of CPU registers, program counter, and other relevant information. This information is stored in the PCB of Process A.

2. Load Next Process State: The operating system selects the next process (Process B) to be executed and retrieves its saved state from its PCB.

3. Update Memory Management: If the memory mappings of Process A and Process B are different, the operating system updates the memory management structures, such as page tables, to reflect the memory mappings required by Process B.

4. Restore Process State: The saved state of Process B is loaded into the CPU registers, including the program counter. This allows the CPU to resume execution from the point where Process B was previously paused.

5. Execute Process B: The CPU begins executing instructions from Process B. Process B continues its execution until a context switch is triggered again, or it voluntarily yields the CPU.

6. Save Process B State (if necessary): If Process B yields the CPU or if a context switch is triggered due to a higher-priority process becoming ready, the current state of Process B is saved in its PCB.

7. Repeat Steps 2-6: The operating system repeats the process of selecting the next process to execute, saving its state if necessary, loading the state of the next process, and resuming execution.

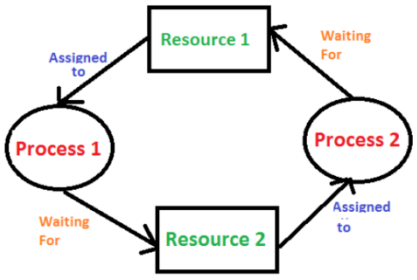
Context switching is an essential mechanism for achieving concurrency in modern operating systems. It allows multiple processes or threads to appear as if they are running simultaneously, even though the processor executes them sequentially in quick succession. Context switches incur some overhead due to the saving and restoring of process states, but they are crucial for efficient and fair process scheduling, resource allocation, and providing a responsive multitasking environment.

Class Test #2

1. Define deadlock. What are the 4 necessary conditions to have deadlock in a system? Discuss.

Answer:

Deadlock refers to a situation in a computer system where two or more processes are unable to proceed further because each process is waiting for a resource held by another process, resulting in a circular dependency and a state of permanent blocking.



To have a deadlock occur in a system, four necessary conditions, known as the Coffman conditions, must be satisfied simultaneously:

1. Mutual Exclusion: At least one resource must be held in a non-sharable mode, meaning that only one process can use the resource at a time. Mutual exclusion is necessary to ensure the integrity and correctness of the resource.

2. Hold and Wait: A process holding at least one resource is waiting to acquire additional resources that are currently held by other processes. This condition arises when a process acquires a resource and keeps holding it while waiting for other resources to be available, preventing other processes from using those resources.

3. No Preemption: Resources cannot be forcibly taken away from a process; they can only be released voluntarily by the process holding them. In other words, once a process acquires a resource, it cannot be forcibly interrupted or preempted by the operating system.

4. Circular Wait: There must exist a circular chain of two or more processes, where each process in the chain is waiting for a resource held by the next process in the chain. This creates a cycle of dependencies among the processes, making it impossible for any of them to proceed.

When these four conditions are met simultaneously, a deadlock can occur. The processes involved in the deadlock become stuck in a state of waiting indefinitely, and the system cannot make progress.

To resolve a deadlock, at least one of these conditions must be eliminated. This can be achieved through various techniques such as resource allocation strategies, deadlock detection algorithms, and deadlock recovery methods. These techniques aim to break one or more of the necessary conditions, ensuring that deadlocks cannot occur or can be resolved if they do occur.

2. **A cycle in a resource allocation graph is necessary condition to have deadlock in a system, but not sufficient" - explain with example

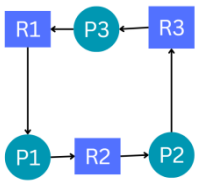
Answer:

A cycle in a resource allocation graph is a necessary condition for deadlock in a system, but it is not sufficient on its own to guarantee the presence of deadlock. Let me explain with an example.

Consider a simplified scenario with three processes (P1, P2, P3) and three resources (R1, R2, R3). Each process needs to acquire two resources to complete its execution. The resource allocation graph is as follows:

R1 R2 R3

P1 ---> P2 ---> P3 ---> P1

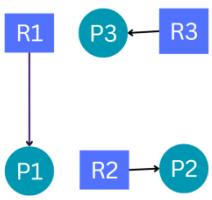


In this case, there is a cycle in the resource allocation graph, which is P1 → P2 → P3 → P1. According to the necessary condition for deadlock, a cycle implies the potential for deadlock. Initially all processes are wants one resource:

P1 (R1)

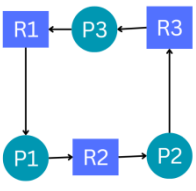
P2 (R2)

P3 (R3)



Initially holds 1

→



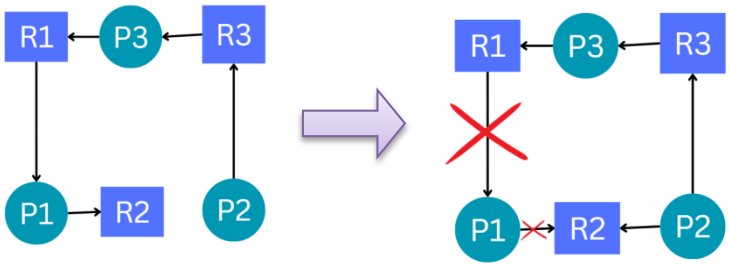
Then requests for the other for execution

At this point, none of the processes can proceed further because they are waiting for a resource. Although there is a cycle in the resource allocation graph, deadlock has not occurred yet because the hold and wait condition is not satisfied. (Then they understand there is need of another resource to execute the process)

Processes are not holding any resources while waiting for additional resources.

Now, if one of the processes, let's say P2, releases its allocated resource (R2) before requesting the second resource, the system can progress:

P1 (R1) P2 P3 (R3)



P1 execution complete. Then P3 will execute and lastly P2

P1 can now acquire the released resource (R2) and complete its execution.

Afterward, P2 can proceed to acquire the remaining resource (R3), and finally, P3 can acquire the remaining resource (R1). The system has reached a stable state without any deadlock.

This example illustrates that even though there is a cycle in the resource allocation graph, deadlock does not occur if the hold and wait condition is not satisfied. Deadlock occurs when all four necessary conditions—mutual exclusion, hold and wait, no preemption, and circular wait—are satisfied simultaneously.

Therefore, while a cycle in the resource allocation graph is a necessary condition for deadlock, it is not sufficient to guarantee deadlock. The additional conditions must also be present for deadlock to occur in a system.

3. If resource preemption is required to deal with deadlock, then what issues should be addressed? Discuss. (2019 4c 2nd page of my note- resource preemption)

Answer:

When considering resource preemption as a strategy to deal with deadlock, several issues need to be addressed to ensure its effectiveness and minimize potential problems. Here are the key issues that should be considered:

1. Selecting Preemption Candidates: The decision of which processes and resources to preempt requires careful consideration. Not all processes or resources may be suitable for preemption. The selection criteria should be well-defined and take into account factors such as process priority, resource utilization, and potential impact on system performance.

2. Preemption Criteria: Determining when to preempt a resource from a process is essential. Preempting resources too frequently or inappropriately can lead to inefficient resource utilization and increased overhead. Defining clear criteria for preemption, such as resource allocation policies or priority-based rules, can help guide the preemption process effectively.

3. Resource Rollback and Recovery: Preempting a resource from a process may require rolling back the process to a previous state before the resource was acquired. This rollback process can be complex, especially if the process has made modifications or updates based on the acquired resource. It is crucial to ensure proper rollback and recovery mechanisms are in place to maintain the integrity of the system and the affected processes.

4. Handling Process Starvation: Preemption can potentially lead to a situation where a process is continually preempted, preventing it from making progress. This is known as process starvation. To mitigate this issue, fairness mechanisms, such as aging or priority adjustments, can be implemented to ensure that preempted processes have opportunities to acquire the required resources and complete their execution.

5. Communication and Coordination: Preemption may require communication and coordination between processes, the operating system, and other system components. It is important to establish proper mechanisms for process notification, resource availability updates, and synchronization to ensure smooth preemption operations without introducing race conditions or inconsistencies.

6. Performance Impact: Resource preemption introduces additional overhead in terms of context switching, rollback operations, and potential delays caused by resource unavailability. These factors can impact system performance and response times. It is crucial to analyze the trade-offs between preventing deadlock and the potential performance impact of resource preemption.

7. Deadlock Detection and Resolution: Resource preemption is often used as part of a broader deadlock detection and resolution strategy. The integration of resource preemption with deadlock detection algorithms and other resolution techniques, such as resource reallocation or process termination, should be carefully designed to effectively and efficiently handle deadlocks in the system.

Addressing these issues helps ensure that resource preemption is applied appropriately and effectively in dealing with deadlock situations. It requires careful consideration of system design, policy definition, and coordination mechanisms to strike a balance between preventing deadlocks and maintaining system performance and fairness.

4. Define logical & physical address. Explain how a physical address is mapped into its physical address

Answer:

Logical Address: A logical address, also known as a virtual address, is a memory address generated by a program or process. It is independent of the underlying physical memory layout and represents the address space visible to the process. The logical address is used by programs during their execution, providing an abstract view of memory.

Physical Address: A physical address represents the actual location of data in the physical memory of a computer system. It refers to the specific memory location in the hardware. The physical address is the address that is used by the memory management unit (MMU) to access the physical memory.

To map a logical address to its corresponding physical address, a process known as address translation is performed by the hardware and operating system. This mapping process typically involves the use of a memory management unit (MMU) and a page table.

Here's a high-level overview of how a logical address is mapped to its physical address:

1. Paging and Page Tables: The operating system divides the logical address space of a process into fixed-size units called pages. Similarly, the physical memory is divided into frames of the same size. Each page of the logical address space corresponds to a frame in the physical memory.

2. Page Table: For each process, the operating system maintains a page table. The page table is a data structure that contains the mapping between the logical pages and the physical frames. Each entry in the page table holds the physical address corresponding to a logical page.

3. Translation Lookaside Buffer (TLB): The MMU includes a cache called the Translation Lookaside Buffer (TLB) to store recently used page table entries. The TLB acts as a fast lookup table for address translation, avoiding the need to access the page table for every memory access.

4. Address Translation: When a program generates a logical address, it goes through the MMU for translation. The MMU checks the TLB first to see if the mapping for the logical page is available. If it is, the physical address is retrieved directly from the TLB. If the mapping is not found in the TLB, a page table lookup is performed.

5. Page Fault Handling: If the page table lookup fails, indicating a page fault, the operating system is notified. The operating system handles the page fault by bringing the required page into physical memory from secondary storage (such as a hard disk) if it is not already presents. It updates the page table accordingly.

6. Physical Address Mapping: Once the mapping between the logical page and physical frame is obtained, the offset within the logical page is combined with the base address of the physical frame to obtain the final physical address.

By using this mapping process, the logical address generated by a process is translated into its corresponding physical address, allowing the processor to access the correct memory location in the physical memory. This mechanism provides the illusion of a large and contiguous logical address space to each process while efficiently utilizing the available physical memory resources.

5. What is dirty or modify bit? Explain its uses & benefits while swapping out some pages from memory to backing store.

Answer:

The dirty or modify bit, also known as the dirty bit or modified bit, is a flag associated with each page in a computer's memory management system. It is used to track whether a particular page has been modified (written to) since it was last loaded into memory or swapped in from the backing store. The dirty bit provides several uses and benefits, particularly when swapping out pages from memory to the backing store (such as a disk).

Dirty bit: An MMU bit used to indicate that a frame has been modified (and therefore must have its contents saved before page replacement)]

Here are the uses and benefits of the dirty or modify bit in the context of swapping pages:

1. Efficient Page Replacement: The dirty bit helps in making more informed decisions during page replacement algorithms, such as the commonly used LRU (Least Recently Used) or LFU (Least Frequently Used) algorithms. When a page needs to be replaced to free up memory space, the dirty bit indicates whether the page has been modified. If the dirty bit is set, it implies that the page has been modified, and its contents need to be written back to the backing store before eviction. Pages that have not been modified (dirty bit not set) can be evicted without the need for a write operation, improving the efficiency of page replacement.

2. Optimized Writebacks: The dirty bit allows for optimized writeback strategies. Rather than writing back all pages to the backing store when swapping them out, only the modified pages (dirty bit set) need to be written back. This reduces the number of unnecessary write operations, conserving disk I/O and improving performance.

3. Minimizing Data Loss: The dirty bit helps to minimize data loss in the event of a system crash or power failure. If a modified page is swapped out without the writeback operation, the modifications would be lost if the system crashes before the page is written back. By using the dirty bit, the operating system can ensure that modified pages are written back to the backing store before swapping them out, preserving the changes made by processes.

4. Caching and Prefetching: The dirty bit also plays a role in caching and prefetching strategies. If a page is marked as modified (dirty bit set), it indicates that the page is likely to be accessed or modified again in the future. This information can be used by the caching or prefetching algorithms to prioritize the retention or loading of such pages, improving overall system performance.

5. Optimized Memory Management: By using the dirty bit, the operating system can make more efficient decisions regarding memory management. It can prioritize swapping out pages that are less likely to be modified, reducing the overhead of unnecessary write operations and enhancing system responsiveness.

Overall, the dirty or modify bit is a valuable mechanism in memory management systems. It allows for efficient page replacement, optimized writebacks, data integrity, and improved performance by selectively identifying and handling modified pages during swapping operations.