Mustafa Alawadi
ID: 40217764
Comp346
2024-10-16

Theory Assignment 1

1.

I. An operating system is the most important software that manages computer hardware and software resources and provides essential services for computer programs. It serves as a bridge between users and computer hardware. Furthermore, an operating system improve performance for the computer through many mechanisms such as memory management, I/O buffering, user interface optimization, etc.

II. Batch: Crucial for efficiency in multiprogramming

. Tasks are collected to be processed in groups

. Efficient for processing large amounts of similar tasks that need little interaction

. There is no direct user interaction during the execution

. Performance is increased when CPU and I/O devices are busy at all times

Time Sharing: Allows multiple users to interact simultaneously with the computer

. Use of CPU scheduling and multiprogramming for interactive use of the system

. Rapid switch from user to user

. Encourage multitasking and provides a user-friendly interface for many users

Dedicated: Designed for a single purpose program

. Minimal overhead

. Not flexible and could lack support for general purpose programs

. Optimized for a particular program

. Very reliable and efficient for the designated tasks

Parallel: Management of multiple processors that execute multiple processes

. Execution of multiple processes simultaneously improve performance

. Communication is in the system bus

. For systems that contain multiple CPU

Multiprogramming: Execution of multiple programs simultaneously on one CPU

. Programs are executed while they are in memory

. Use of process scheduling to switch between processes

. Risk of conflicts

III. A user would be better off using a timesharing system when there is more than one user who need to access and work on shared resources, the tasks are intensive, and the hardware is fast. The task may be resolved quicker than on a personal computer since users can access remotely and can

depend on a much powerful machine. A personal computer is more suitable for a user's own needs, when the job is small, and the performance is sufficient.

2.

a) In a single programmed OS, only one process can be executed at a time.

Process	CPU	I/0	CPU
P1	15	10	10
P2	10	5	15

Therefore, we need to traverse by going through all the processes

15	10	10	10	5	15

Summing them all gives us the total time which is 65 units.

b) In a multiprogrammed OS, while a process is waiting for I/O operation, another can use the CPU. Thus the processes can overlap the I/O.

15	10	10	15

Summing them all gives us the total time which is 50 units.

c) Throughput is the number of processes completed per unit time.

For part a) : 2/65 = 0.0308 tasks/unit time

For part b): 2/50 = 0.04 tasks/unit time

This suggests that it allows more efficient performance and reduce CPU time. Multiprogramming increases throughput as it allows for overlapping and making the CPU non-idle, it takes advantage of I/O wait times to execute other processes. In this case, the time saved was significant.

3.

I. Polling and interrupt differ mainly in how the CPU interacts with devices that are attached. In polling, the processor keeps on checking if the device requires attention, and this can lead to a wasted CPU time which could increase the heat due to constant activity. Interrupt, however, use a scripted mechanism that allows the devices to signal the CPU only when the action is truly needed, and this can minimize unnecessary processing. Interrupt also supports the use of multiple devices and can easily handle complex time complexities. Polling can be advantageous over interrupts when a system has a limited hardware capabilities, where predictable timing is the priority, debugging and testing, or for devices that generate events continuously.

II. Yes, it is possible to use a DMA controller in a system that does not support interrupts by using polling, but we might encounter challenges. First the CPU is going to need to poll the DMA controller to ensure the data transfer is complete. Since there's no interrupts, the CPU is responsible for managing this which can slow things down. Having the CPU continuously monitoring the status can significantly reduce the efficiency as there will be an increase in workload.

III. a) Context switching must be atomic to prevent race conditions and inconsistencies in the CPU and memory. If there is an interrupt, then we might lose data and the state could get corrupted. For instance, if process A registers are saved but there is an interrupt in the switch to process B, and then A is resumed before it is restored correctly, A could end up having only partial values.

b) In practice, we can achieve atomicity by disabling interrupts only at the beginning and make them available later and we can also use mutexes or spinlocks to ensure only one processor can perform a context switch.

4.

I. We can't allow user processes to perform I/O directly as this would cause a lot of risks. For example, direct I/O access can allow the user processes to manipulate hardware resources and memory directly which can lead to potential breaches of security. Sensitive data can be overwritten if there are malicious programs and the system could crash entirely. Another reason is because user programs might intentionally misconfigure hardware which then leads to an unstable system. Finally, the operating system handles exceptions in I/O operations and having direct access to it could lead to inconsistencies in handling those errors.

II. a) In case the system does not have strict memory protection, a user can modify the interrupt vector and use the pointers to redirect execution to his malicious code. Of course he would have written a user-mode program for this. Another loophole

could be if the user somehow manages to execute privileged instructions or access hardware directly because then they can control the processor state and switch to the monitor mode. There are many disastrous effects such as the user gaining unrestricted access and modifying or deleting data, launch malicious attacks that overload resources and crash the system, and finally installing malware that can compromise the system to gain more information on other systems.

b) Remedies for the loophole could be strict memory protection to prevent user processes from gaining kernel memory, and also code verification to prevent unauthorized code execution.

5.

a) If we can execute N-1 processes simultaneously without increasing the waiting time, the overall completion time can still be minimized to match the maximum execution time of the longest process.

b) The total execution time can exceed the sum of individual process execution times if there's overhead from context switching, I/O operations or synchronization delays. The accumulation of those delays can result in a total execution time greater than the sum of individual process execution times.

6.

I. Because it only provides information about current time and there is no control over the time that you read, this is not privileged.

II. Because it can affect the integrity of the system and you don't want memory to be cleared randomly, this is privileged.

III. User level processes need to read from their own memory space often, this is not privileged.

IV. Similar to reading, user processes should be able to write to their own memory, this is not privileged.

V. Copying data between registers is a basic operation that does not affect system stability or security, this is not privileged.

VI. This means the system can't respond to events and manage I/O, this is privileged.

VII. This means that the process can reach kernel mode and utilize the whole CPU, this is privileged.

7. Network OS is used to manage networked computers. There is a server that will manage one more computers. An example could be a college or university. Distributed OS is when all computers are connected and can share tasks. For example, if there's a program installed on somebody's computer, you can use it if you're using distributed OS, which is why distributed OS requires more RAM and higher speed processor.