Theory assignment 1

Jordan Hubscher

ID: 27019696,

Diego Kulibaba

ID: 27010540

Question 1:

I.

An operating system is software. It’s function is to abstract the hardware of the system directly from the user through a visual interface and other supplementary application programs.

II.

- Batch; Sequentially performs each task (job) within a collective batch. Saves times, maximizes throughput.

- TimeSharing; Performs parts of individual tasks within a given time slice (part of a process for a small amount of time, unlike batch OS).

- Dedicated/Embedded; Separate computer “dedicated” to specific externalized hardware in which the user is now the computer itself rather than the human user.

- RealTime; Must return a result within a very specific time frame (i.e. less than a few milliseconds such as in a car’s braking control system).

III.

If there is a large job to be executed, typically the timesharing system would be more beneficial to use, rather than the single-user pc since timesharing systems connect multiple users to a single, larger, more powerful unit, which dwarfs the power of a single-user pc in comparison.

Question 2:

I.

The advantage of having interrupts is that you are not constantly polling the devices, checking if they’re ready. In the case of large jobs, for example, if the cpu is constantly polling the device, and the job isn’t done, this would take up unnecessary amounts of processing time from the cpu.

Polling is quite advantageous when the amount of interrupts from a device happens very often (often enough so that the interrupt time is greater than the amount of time polling). This is because the time it takes to poll is negligible when compared to the amount of time taken to process an interrupt service routine.

II.

Yes it’s possible. But again, unless the system is constantly being interrupted (i.e. the amount of times polling a device greatly exceeds the amount of times interrupting it, only then it would be remotely viable, otherwise the excessive polling from checking if the device is free wastes CPU cycles)

III.

a.) A context switch must be atomic in order to maintain proper values of the given thread’s resources and register values. Else, if you by any chance context switch *during* a context switch, you may lose all register values at that moment, for that thread.

b.) One can achieve atomicity within a context switch by way of locking access from other threads during the critical operations. Such as disabling interrupts during the execution of the save operations (or using a semaphore to have the interrupting threads wait until the current thread is ready)

Question 3:

1. If user processes could directly perform I/O and therefore obtain direct access to main memory, then there is great risk of those processes wiping crucial parts of system memory since at that point the user has direct access to system code in even the reserved parts of memory (i.e. bootstrap loader).

II.

a.) He could have intentionally caused a buffer overflow, overwriting the interrupt vector leading to system access. He could also have exploited a system call by breaking it once the mod-bit was added.

b.) By correctly initializing variables/ using a strong static type checking programming languages with safe common libraries. (checking boundaries of arrays)

Question 4:

Turning off interrupts should be a privileged action since you are disabling the system from reacting to certain errors/faults and thus may compromise vital parts of memory during execution of a program.

Monitor mode should definitely be privileged. In monitor mode, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address. Monitor mode is generally reserved for the lowest-level, most trusted functions of the operating system. Crashes in kernel mode are catastrophic; they will halt the entire PC.

Clearing memory should also be a privileged instruction in case a program decides to accidentally wipe, say, the bootstrap loader (since it has full access and rights to the entire system memory). This instruction should be trusted only low-level and validated operations.