CmpE 322

January 29, 2022

Contents

1 Discussion I

1) We said the OS Systems acts on behalf of the application programs as an intermediary. Why is that so? Why should we prevent the applications accessing the hardware directly?

Answer: Operating Systems are designed to be convenient and efficient. OS is responsible for the proper share of resources and compatibility of hardware with the applications. That's why OS acts as an intermediary between application programs and hardware.

OS provides functionalities for such basic operations so that it becomes easier for the programmer and the user to solve the main problems with their applications rather than tackling with hardware related problems. We should prevent the applications from accessing the hardware directly because slight changes in the hardware may require us to modify our application programs one by one.

OS is also used to prevent user from interacting directly with hardware.

2) Which of the following component(s) is (are) part of an OS? (You may choose more than one option)

(a) Kernel
(b) Hard disk
(c) Scheduler
(d) File browser
(e) File system
(f) Web browser
(g) Device drivers
(h) All of the above

Answer: **Kernel**, **Scheduler** and **File System**.

Kernel is the core of the operating system. Hard disk is a hardware component. File browser is a system program. Web browser is an application program. Drivers are not part of the operating system. They are just utility programs to make user's work easier. Scheduler can be included as a part of the OS. The task scheduler, sometimes called process scheduler, is the part of the kernel that decides which task to run next.

3) Is it possible to modify bootstrap?

Answer: Bootstrap is typically stored in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM). Therefore it is not possible to modify bootstrap.

OS Concepts pg.9: EEPROM can be changed but cannot be changed frequently and so contains mostly static programs.

Bootstrap is modifiable as long as the hardware that stores it allows the modification. In modern systems, bootstrap can be modified since it is stored in EEPROM. BIOS update can be considered as a modification in the bootstrap.

4) If we already have the disk, then why do we also have a disk controller? What does the disk controller do? Why not connect the disk directly without a controller?

Answer: Disk controller is one of the device controllers in charge of the disk. It moves data from disk to the disk buffer and informs the CPU that it finished its operation by causing an interrupt.

Disk controller takes the responsibility of dealing with different type of disks rather than putting the burden on the shoulders of CPU. Disk controller also provides abstraction between disk and the CPU.

Disk controller is responsible for the communication between CPU and disk. Transmission from disk to buffer or disk to memory is so slow and it occupies the transmission lines, namely data bus and address bus. Keeping the buses busy for a long time prevents the communication between CPU and other controllers. Therefore we use a controller to fetch data from disk to its buffer. During this period, CPU can perform its own tasks. Transmission from disk buffer to memory is relatively faster because the buffer itself in kind of a memory.

Disk buffer is an embedded memory in a hard disk drive.

5) What is an interrupt? Why do you use interrupts? If the concept of an interrupt was not invented, how could we read an array of 100 bytes from the disk into a variable in the main memory?

Answer: Interrupt is a signal emitted by hardware or software (trap/exception) when a process or an event needs immediate attention. Interrupts are essential because a processor cannot process all the programs at once, interrupts enable it perform other tasks until a specific task needs attention. With the help of interrupt

mechanism, CPU can return to the exact position where it left the previous task after handling with the interrupt service routing.

Operating System is interrupt driven. If the concept of an interrupt was not invented, CPU had to wait for the transfer from I/O device to the buffer of the controller so that it transforms the data from buffer to the main memory. Obviously it would make the use of CPU extremely inefficient.

If the concept of an interrupt was not invented, CPU waste countless processor cycles by repeatedly checking the command-ready little bit of each device since device drivers are not able to emit interrupt signal.

6) What are the strengths and drawbacks of polling as an interrupt handling mechanism?

Answer: Polling mechanism involves checking each device in the computer system for an interrupt. One of the drawbacks of polling is that it cannot change the order of the devices, therefore most of the time devices with smaller IDs are prioritized. Order of polling defines an implicit precedence. It's not a realistic scenario for our computers because from time to time priority of a device may change based on our usage.

Another drawback is that it's a slow process since we have to check them one by one. Most of the time, we unnecessarily poll the devices to check if there is an interrupt emitted by them. Vectored interrupt system is a fast process.

Polling has the disadvantage that if there are too many devices to check, the time required to poll them can exceed the time available to service the I/O device.

One strength can be its relatively simple implementation. Vectored interrupt system requires a harder implementation since it should send the index of the interrupt service routing alongside with the interrupt.

Other strength is that we do not need any additional chip to store the index and other information of interrupt service routine as we do in vectored interrupt system. Transmission reliability is another strength.

7) Order these storage devices from the fastest to slowest.

- Optical disk
- SSD
- Main memory
- HDD
- Register
- Cache
- Magnetic tape

Answer: Fastest to slowest:

Register > Cache > Memory > SSD > HDD > Optical Disk > Magnetic tapes

As the speed of the storage device increases, amount of memory it can hold decreases. Register, cache and memory are volatile. SSD, HDD, optical disk and magnetic tapes are nonvolatile.

- Optical disk: CD-ROM - laser technology
- SSD: Semiconductor cells
- Main memory: Random Access Memory- Accessing to any part of the memory takes same amount of time.
- HDD: Rotating platters (magnetic storage)
- Register: Don't have any moving parts like HDD. Stored in the CPU, alongside with the control unit.
- Cache: Copying information for a faster storage system. Located very close to the CPU.
- Magnetic tapes: Only for back-up processes

8) What is the DMA good for? What is the drawback of not having DMA in a computer system? Would the system completely fail or perform poorly? Explain.

Answer: With the use of DMA, device controller can transfer an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU. Only one interrupt is generated per block, to tell the device driver that the operation has completed, rather than one interrupt per byte generated for low-speed devices. While the device controller is performing these operations, the CPU is available to accomplish other work.

Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read/write operation, and is thus unavailable to perform other work. However, with DMA there may occur some cache coherence problems.

Absence of DMA causes the system to perform poorly.

9) What is the difference between multiprogramming and multitasking?

Answer: Multiprogramming organizes jobs so that CPU always has one job to execute. Jobs are selected and run via job scheduling. When CPU has to wait, OS switches to another job (context switching). Whereas in multitasking, CPU switches between jobs in a frequent manner rather than focusing only on one job for a limited amount of time.

Main purpose of multiprogramming is to increase the **CPU utilization**, while the main purpose of multitasking is to increase **interactivity**.

10) Why do we have kernel and user modes? When the system administrator (root user) runs a process, does it execute in user mode or in kernel mode?

Answer: Dual mode operation provides us with the means for protecting the operating system from errant users. If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the OS.

Such protection can include I/O protection (user can only access I/O by sending a request to the OS), memory protection (user can access the memory that the OS has allocated) aand CPU protection (user can only have as much CPU time as the OS allocates.)

A normal application running with root privileges still exists in user mode and only switches to kernel mode when a kernel system call is made and then switches right back to user mode.

11) What is so special about the Program Counter (PC) register? What if we change the restore it before the other registers, say R1, R2, ...

Answer: PC contains the memory address or location of the instruction being executed. After an interrupt is serviced, the saved return address is loaded into the **program counter**, and the interrupted computation resumes as though the interrupt had not occurred.

Restoring other registers before restoring the program counter would lead to the errors in our system. We should restore R1 and R2 in our previous task after restoring the program counter.

If we restore the PC and then restore the other registers, the next instructions can be executed wrongly. Because when the PC turns back the value before the jump or interrupt, it uses the values in the registers which are not restored, so it uses wrong values.