What is interrupts? Explain with Example

An interrupt is an occurrence that causes the processor's execution sequence to be disrupted. An interrupt can be planned (i.e., requested by the presently running programme) or unplanned (i.e., not requested by the currently running programme) (caused by an event that might or might not be related to the currently running program).

Pressing a key on the keyboard or moving the mouse, for example, causes hardware interrupts, which cause the processor to read the keystroke or mouse location. Hardware interruptions can occur asynchronously to the processor clock and at any point during the execution of an instruction.

What is the purpose of Interrupts?

Interrupts are frequently used to service hardware timers, transport data to and from storage (e.g., disc I/O) and communication interfaces (e.g., UART, Ethernet), manage keyboard and mouse events, and respond to any other time-sensitive events that the application system requires.

What are the benefits of using interrupts in a computer?

Interrupts are useful because they allow the user to have more control over the computer. Without interruptions, a user may have to wait for a certain application to be assigned a higher priority over the CPU before it can be executed. This guarantees that the CPU will take care of the process right away.

Why is disabling the interrupts in a multicore operating system not considered to be an efficient approach?

Step 1:

When interruptions are disabled, other programs are effectively stopped from running. When interrupts are disabled, the CPU is unable to switch processes, and programs can use shared variables without being accessed by another process.

Disabling interruptions requires a change in the way the processor handles instructions, which means it can't be done while instructions are being run. Before the processor can turn off interrupts, the pipeline must be "flushed," meaning all in-flight instructions must be completed.

Step 2:

In multiprocessor systems, interrupts are insufficient because disabling interrupts only prevents other processes from running on the processor where interrupts were disabled; there are no restrictions on what processes can run on other processors, so the process disabling interrupts cannot guarantee mutually exclusive access to program state.