OPERATING SYSTEM - I/O SOFTWARES

Interrupts

The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt, performs the necessary processing and executes a interrupt instruction to return the CPU to its execution state.

The basic mechanism of interrurpt enables the CPU to respond to an asynchronous event, such as when a device controller become ready for service. Most CPUs have two interrupt request lines.

- **non-maskable interrupt** - Such kind of interrupts are reserved for events like unrecoverable memory errors.
- **maskable interrupt** - Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

The interrupt mechanism accepts an address - a number that selects a specific interrupt handling routine/function from a small set.In most architectures, this address is an offset stored in a table called the interrupt vector table. This vector contains the memory addresses of specialized interrupt handlers.

Application I/O Interface

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O interfaces with respected to devices.

- **Character-stream / block** - A character-stream device transfers bytes in one by one fashion, whereas a block device transfers a complete unit of bytes.
- **Sequential / random-access** - A sequential device transfers data in a fixed order determined by the device, random-access device can be instructed to seek position to any of the available data storage locations.
- **Synchronous / asynchronous** - A synchronous device performs data transfers with known response time where as an asynchronous device shows irregular or unpredictable response time.
- **Sharable / dedicated** - A sharable device can be used concurrently by several processes or threads but a dedicated device cannot be used.
- **Speed of operation** - Device speeds may range from a few bytes per second to a few gigabytes per second.
- **Read-write, read only, or write only** - Some devices perform both input and output, but others support only one data direction that is read only.

Clocks

Clocks are also called timers. The clock software takes the form of a device driver though a clock is neither a blocking device nor a character based device. The clock software is the clock driver. The exact function of the clock driver may vary depending on operating system. Generally, the functions of the clock driver include the following.

S.N.	Task	Description

1	Maintaining the time of the day	The clock driver implements the time of day or the real time clock function.It requires incrementing a counter at each clock tick.
2	Preventing processes from running too long	As a process is started, the scheduler initializes the quantum counter in clock ticks for the process. The clock driver decrements the quantum counter by 1, at every clock interrupt. When the counter gets to zero , clock driver calls the scheduler to set up another process. Thus clock driver helps in preventing processes from running longer than time slice allowed.
3	Accounting for CPU usage	Another function performed by clock driver is doing CPU accounting. CPU accounting implies telling how long the process has run.
4	Providing watchdog timers for parts of the system itself	Watchdog timers are the timers set by certain parts of the system. For example, to use a floppy disk, the system must turn on the motor and then wait about 500msec for it to comes up to speed.

Kernel I/O Subsystem

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following are some of the services provided.

- **Scheduling** - Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.
- **Buffering** - Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a device with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.
- **Caching** - Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
- **Spooling and Device Reservation** A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.
- **Error Handling** An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Device driver

Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviours on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver are following.

- To accept request from the device independent software above it.
- To see to it that the request is executed.

How a device driver handles a request is as follows: Suppose a request comes to read a block N. If the driver is idle at the time a request arrives, it starts carrying out the request immediately. Otherwise, if the driver is already busy with some other request, it places the new request in the queue of pending requests.