I/O DEVICE MANAGEMENT

The role of the operating system in computer I/O is to manage and control I/O operations and I/O devices. Devices connected to the computer (I/O devices) vary so widely in their function and speed and therefore, different methods are needed to control them. These methods form I/O subsystem of the operating system which separates the rest of the operating system from the complexities of managing I/O devices.

I/O device technology exhibits two conflicting trends in which, on one hand, there is rise of improved device generations, and on the other hand, there is increasingly broad variety of I/O devices, all of which must be incorporated into the existing computers and operating systems. The basic I/O hardware elements such as ports, buses and device controllers, accommodate a wide variety of I/O devices. This is done through the use of **device drivers** which present a uniform device access interface to the I/O subsystem.

I/O Hardware

Computers operate different kind of devices which fit into general categories of storage devices (disks and tapes), transmission devices (network connections, Bluetooth) and human-interface devices (screen, keyboard, mouse, and audio in and out). Generally, they can be categorized into the following categories;

1. **Human readable –** these are devices suitable for communicating with computer user. E.g., printers, video display terminals, keyboards etc.
2. **Machine readable –** these are devices suitable for communicating with electronic equipment. E.g., disk and tape drives, sensors, controller etc.
3. **Communication –** these are devices suitable for communicating with remote devices. E.g., modems, digital line drivers etc.

I/O devices also vary in terms of speed, throughput, and operations. Some of the differences between I/O devices are;

- **Data rate** – there are several orders of magnitude between data transfer rates.
- **Application-** different devices have different use in the system.
- **Complexity of control –** other devices such as disks require more complex control interface while other devices such as printers require simple control interface.
- **Unit of transfer –** data may be transferred as a stream of bytes, characters or in larger blocks.
- **Data representation** – different data encoding schemes are used for different devices

- **Error conditions** – the nature of errors differs widely from one device to another.

How processor accomplishes an I/O transfer

A device communicates with a computer system by sending signals over a cable or even through air. For a device to communicate with the machine, a connection point referred to as a **port** is used, e.g. serial port, USB port etc. If devices share a common set of wires, the connection is called a bus. A **bus** is a set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires. Buses are used widely in computer architecture and vary in their signaling methods, speed, throughput, and connection methods and thus controllers are used. A **controller** is a collection of electronics that can operate a port, a bus or a device. They control signals on the wires of a port or a bus. For example, a serial-port controller is a simple device controller that control signals on the wires of a serial port.

The processor accomplishes an I/O transfer through issuing commands and data to a controller. A controller has one or more registers for data and control signals. The processor communicates with the controller by reading and writing bit patterns in these registers. I/O instructions issued by the processor to the controllers, triggers bus lines to select the proper device and to move bits into or out of a device register. Alternatively, memory-mapped I/O can be supported where device control registers are mapped into the address space of the processor. The CPU then executes I/O requests by reading and writing device-control registers at their mapped locations in physical memory.

An I/O port typically consists of four registers, called the status, control, data-in, and data-out registers.

- The **data-in register** is read by the host to get input.
- The **data-out register** is written by the host to send output.
- The **status register** contains bits that can be read by the host. These bits indicate states, such as whether the current command has completed, whether a byte is available to be read from the data-in register, and whether a device error has occurred.
- The **control register** can be written by the host to start a command or to change the mode of a device. For instance, changing into different speeds supported or word length from smaller bits to higher bits.

Direct Memory Access (DMA)

For devices that perform large transfers, such as disk drives, it seems wasteful to use an expensive general-purpose processor to watch status bits and to feed data into controller registers one byte at a time. Many computers avoid burdening the CPU by offloading some of this work to a special-purpose processor called **direct-memory-access controller.** Usually, the host writes the pointer to the source of transfer, a pointer to the destination, and the count of the number of bytes to be transferred to the DMA controller, which in turn operates memory bus directly by placing addresses on the bus to perform transfer without the help of the main CPU. In general DMA allow transfer of blocks of data directly between an external device and the main memory without continuous intervention by the processor.