I/O System

- In the early days of computing, input and output is at a speed similar to that of processing.
- When processors became electronic, I/O devices still remain mechanical.
    - e.g. hard disk vs CPU
- Today, I/O systems are designed to handle I/O devices which are of many magnitudes slower than the slowest CPU.
    - To provide simple, abstract software interfaces to manage the I/O operations needed.
    - Ensure that there is as much overlap as possible between the operation of the I/O devices and the CPU.

Device Manager Abstraction

- Different devices require different operations in order to work them.
    - What kinds of operations? E.g.?
- It is important to create device drivers, which takes on the task of working these devices, but requiring only a standard set of functions to make them (the devices) work.
- The device manager, in turn, manages the collection of device drivers.
- The manager makes it possible for the OS to then provide a standard set of system calls to application programs, which use the devices.
- Device manager contains:
    - Device-Independent
    - Device-Dependent (To be downloaded) - Device specific

System Call Interface

- Functions available to application programs to call the system call function to operate the devices.
- Abstract all devices (and files) to a few interfaces.
- Devices category:
    - Block devices
    - Character devices
    - Network devices
    - Storage devices (random or sequential)
- Make interfaces as similar as possible
    - Block vs Character
    - Sequential vs direct access
- Device driver implements functions (one entry point per API function)

Examples

BSD UNIX Driver

Commands	Description
open	Prepare dev for operation
close	No longer using the device
ioctl	Character dev specific info

Commands	Description
read	Character dev input op
write	Character dev output op
strategy	Block dev input/output ops
select	Character dev check for data
stop	Discontinue a stream output op

Windows system calls and low-level-I/O

Commands	Description
`_close`	Close file
`_commit`	Flush file to disk
`_creat, _wrcreat`	Create file
`_dup`	Return next available file descriptor for given file
`_dup2`	Create second descriptor for given file
`_eof`	Test for end of file
...	...

Device Status Table

- Device Manager keeps track of the status of the various devices using a **Device Status Table**, which consists of these info:
    - Device ID
    - Device Status (busy, done, idle)
    - Queue of processes waiting for the device

Overlapping the Operation of a Device and the CPU

- Programs which perform IO expects IO operations to complete before the next statement is executed.
- However, performance gains can be gained if we can make the program execute instructions while the IO is taking place.
- This should be done without violating the serial execution order of the program.
- To maximize IO, we can have the CPU operate on another process when the IO is busy with the original process.
    - Does this mean that to the CPU its as if nothing much as changed, just another process/ instruction that is taking resources.
- This increases the efficiency of the computer and reduces the overall time required to execute all the processes.

Buffering

- The speed of I/O is much slower than that of CPU.
- In order to speed up I/O, it is possible to keep I/O devices busy when the processes do not require I/O operations.
- This increase the overlap between I/O and processing.
- A buffer is a temporary memory-based storage area, that stores the data from an I/O operation.

Types of Buffering

- Input buffering
  - Copy the data into memory before the process requests it.
- Output buffering
  - Temporarily stores the data in memory and have it written out to the device when the process resumes execution.
- Hardware buffering
  - Implement buffering in the hardware by having specialized registers to act as buffers. Also known as cache.
- Double buffering
  - Implement hardware buffering and having a separate buffer implemented at the software level.
  - i.e. Using primary memory as a buffer.
- Circular buffering
  - Having a buffer which contains n number of locations. The increased number of locations allow for more to be stored in the buffer.
  - Circular buffering is the technique of managing these locations so that they can be re-used.

Device Class Characteristics

- A typical computer system will handle many types of devices, such as:
  - Character devices (keyboard, monitor)
  - Block devices (printer, USB)
  - Network devices (network, modems)
- Character based devices such as tty (teletype, or terminal) and serial devices are where data stream is transferred and handle one character or byte at a time.
- Block type devices such as hard drives transfer data in blocks, typically a multiple of 256 bytes.

Linux devices

```
brw-rw----  1 root disk    8,  0 Nov   7 07:06 sda
brw-rw----  1 root disk    8,  1 Nov   7 07:06 sda1
brw-rw----  1 root disk    8,  16 Nov  7 07:06 tty0
brw-rw----  1 root disk    8,  0 Nov   7 07:06 tty1
```

- The ones that have a "b" are block type devices and the ones that begins with a "c" are character devices.
- sda, sda1 are naming conventions for disk devices.
- tty0, tty1 are naming conventions for terminal devices or virtual consoles.

Storage Devices

- Randomly accessed storage
    - Flash memory
    - Optical disks
    - Rotating magnetic disk
    - Floppy disk
- Sequentially accessed storage
    - Magnetic tapes

Randomly Accessed Storage Devices

- Allow a driver to access blocks of data in the device in any order. Need not be sequential.
- Non-volatile memory (such as USB memory devices) also fall in this category.
- Rotating disks are still commonly used today.
- Both can be easily managed using a common API

Rotating Disk storage

- Multi-surface disk
- Disk surface
- Cylinders

Rotating Disk Optimizations

- Disk-based devices form the most common randomly accessed storage devices.
- E.g. Hard disks, CD/DVD, Zip disks, MO disks.
- Although they are random, we can improve on their efficiency using disk optimization techniques.
- Their effectiveness is increased in a multi-programming environment, where data blocks can be read from a large range of locations in the disk.

Disk Optimizations

- Transfer time:
    - Time to copy bits from disk surface to memory.
- Disk latency time:
    - Rotational delay waiting for proper sector to rotate under R/W head.
- Disk seek time:
    - Delay while R/W head moves to the destination track/cylinder.
- Access time:
    - Access time = seek + latency + transfer
- Disk optimization techniques for disk based devices:
    - FCFS
    - SSTF
    - SCAN/C-SCAN
    - LOOK/C-LOOK