OS CHAPTER 6 Input/Output management and Disk scheduling

Types of I/O devices

① Human readable
- Suitable for communicating with the computer user.
- Printers, terminals, video display, keyboard, mouse

② Machine readable
- suitable for communicating with electronic equipment
- disk drives, USB keys, sensors, controllers

③ Communication
- suitable for communicating with remote devices
- modems, digital line drivers

Differences in I/O devices

① Data rate
- There maybe differences of magnitude between the data transfer rates

② Application
- The use to switch a device is put has an influence on the software

③ Complexity of control
- The effect on the operating system is filtered by the complexity of the I/O module that controls the device

④ Unit of transfer
- data may be transferred as a stream of bytes or characters or in larger blocks

⑤ Data Representation
- Different data encoding schemes are used by different devices

⑥ Error conditions
- The nature of errors, the way in which they are reported, their consequences and available range of responses differs from one device to another

I/O Techniques

	No Interrupts	Use of Interrupts
I/O to memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O to memory tranfer		Direct memory Access (DMA)

Organization of the I/O Function

Three ways to perform I/O are:

① Programmed I/O
- The processor issues an I/O command on behalf of a process to an I/O module; that process then busy waits for the operation to be completed before proceeding

② Interrupt-driven I/O
- The processor issues an I/O on behalf of the process

i] If non blocking:
→ Processor continues to execute instructions from the process that issued the I/O command.

ii] If blocking:
→ The next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process

③ Direct Memory Access (DMA)
- It controls the execution of data between main memory and an I/O module

Evolution of the I/O Function

① Processor directly controls a peripheral device

② A controller or I/O module is added

③ Same configuration as step 2, but now, interrupts are employed

④ The I/O module is given direct control of memory via DMA

⑤ The I/O module is enhanced to become a separate processor with a specialized instruction set tailored for I/O

⑥ The I/O module has a local memory of its own and is in fact a computer in its own right.

Design Objectives

① Efficiency

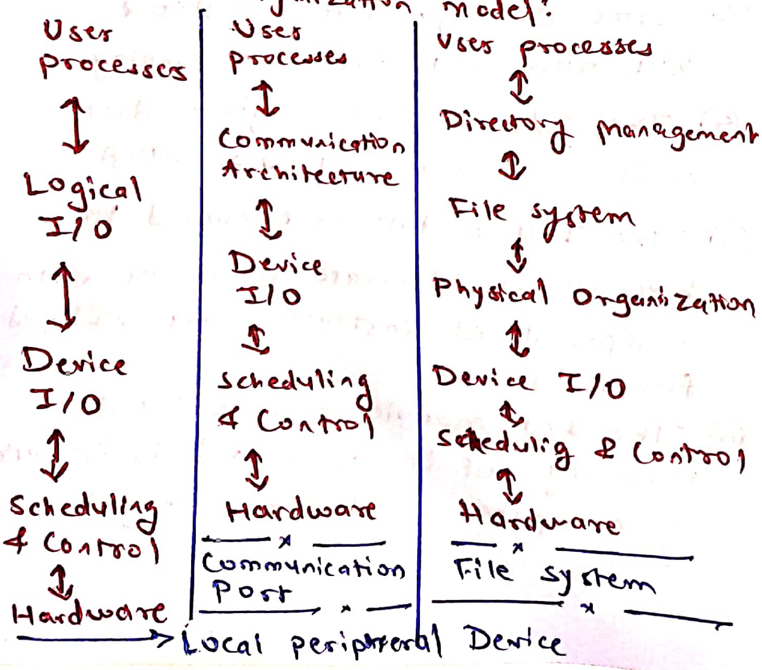
- Major effort in I/O design
- Important because I/O operations often form a bottleneck
- Most I/O devices are extremely slow compared with main memory and the processor
- The area that has received the most attention is disk I/O

② Generality

- Desirable to handle all devices to a uniform manner
- Applies to the way processes view I/O devices and way the OS manages I/O devices and operations
- Diversity of devices makes it difficult to achieve true generality

Hierarchical Design

- Functions of the OS should be separated according to their complexity their characteristic time scale and their level of abstraction
- Leads to an organization of the OS into a series of layers
- Each layer performs a related subset of the functions required of the OS.
- Layers should be defined so that changes in one layer do not require changes in other layers.
- An I/O organization model:



Buffering

- Perform input transfers in advance of requests being made and perform output transfers some time after the request is made

① Block oriented device:

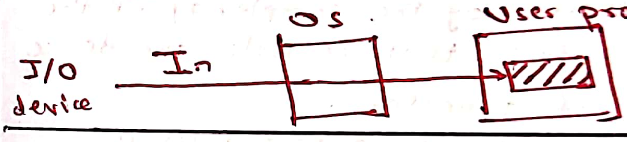
- Stores information in blocks that are usually of fixed size
- Transfers are made one block at a time
- Possible to reference data by its block number
- Disks and USB keys are examples

② Stream oriented device:

- Transfers data in and out as a stream of bytes
- No block structure
- Examples - Terminals, printers, communication ports, and most other devices which are not secondary storage

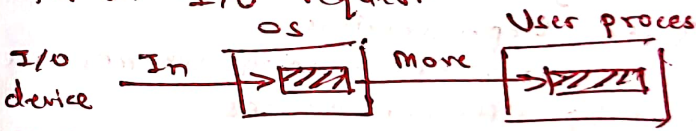
No Buffer

- Without a buffer, the OS directly access the device when it needs



Single Buffer

- OS assigns a buffer in main memory for an I/O request



① Block oriented single buffer

- Input transfer are made to the system buffer
- Reading ahead input:
  - i] is done in the expectation that the block will eventually be needed
  - ii] when the tranfer is complete, the process moves the block into user space and immediately requests another block
- Generally provides a speedup compared to the lack of system buffering
- Disadvantage:
  - i] Complicates the logic in the OS
  - ii] Swapping logic is also affected

II] stream oriented single buffer

① Line at a time operation

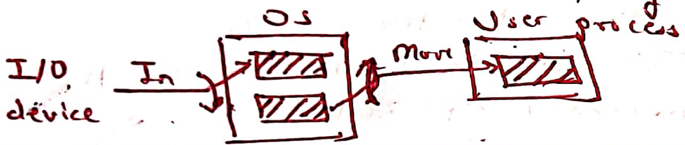
- Appropriate for scroll mode terminals
- User input is one line at a time with a carriage return signaling the end of a line
- Output to the terminal is one line at a time

② Byte at a time operation

- Used on forms mode terminals
- When each stroke is significant
- Other peripherals such as sensors and controllers.

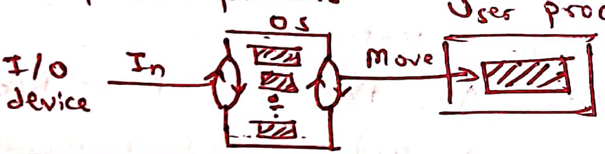
Double Buffer

- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the OS fills the other buffer
- Also known as buffer swapping



Circular Buffer

- Two or more buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must keep up with process

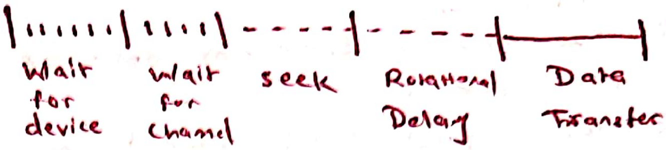


The Utility of Buffering

- Technique that smoothes out peak in I/O demand
- With enough demands eventually all buffers become full and their advantage is lost
- When there is a variety of I/O and process activities to service, buffering can increase the efficiency ~~of~~ the os and the performance of individual processes.

Disk Scheduling Algorithms ③

- The actual details of disk I/O operation depends on
  - → computer system
  - → os
  - → Nature of the I/O channel and disk controller hardware



① seek time

- It is the time to move the arm to the proper cylinder.

② Rotational Delay

- The time for the proper sector to rotate under the head

③ Actual data transfer time

- For most of the disk, seek time is greater than rotational delay and actual data transfer time
- Hence by minimizing mean seek time it is possible to enhance system performance significantly

Disk Scheduling

Transfer time

→ Time taken to transfer the data

Disk Access Time

= Rotational Latency

+ seek Time

+ Transfer time

Disk Response time

→ Average of time spent by each request waiting for I/O operation

Goal : ① Fainess

② High throughout

③ minimal travelling head time.

Algorithms:

→ FCFS

→ SSTF

→ Scan

→ C scan

→ Look

→ C-Look

FCFS scheduling Algorithm

- Simplest Disk Scheduling Algorithm
- It services the I/O requests in the order in which they arrive.
- There is no starvation in this algo, every request is serviced.
- Disadvantages:

① The scheme does not optimize the seek time

② The request may come from different processes therefore there is the possibility of inappropriate movement of the head.

SSTF Scheduling Algorithm

- Shortest seek Time First algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction.
- It reduces the total seek time as compared to FCFS.
- It allows the head to move to the closest ~~track~~ in service queue
- Disadvantages:

① It may cause starvation for some request

② Switching direction on the frequent basis slows the working of algorithm

③ It is not the most optimal algo.

Scan Algorithm

- Also known as Elevator algorithm.
- In this algo, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path and then it turns back and moves in the reverse direction satisfying requests coming in its path.
- It works in the way an elevator works, elevator moves in the direction completely till the last floor of that direction and then turns back.

C-SCAN Algorithm

- In C-SCAN algo, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder then it jumps to a last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests

Look Scheduling Algorithm

- It is like scan scheduling algo to some extent except the difference that in this algo, the arm of the disk stops moving inwards or outwards when no more request in that direction exists.
- This algo tries to overcome the overhead of SCAN algo which forces disk arm to move in one direction till the end regardless of knowing if any requests exists in the direction or not

C Look Scheduling

- C LOOK is similar to C-SCAN Algo. to some extent.
- In this algo, the arm of the disk moves outwards servicing requests until it reaches the highest request cylinder, then it jumps to lowest request cylider without servicing any request then it again start moving outwards servicing the remaining requests
- This algo tries to overcome the overhead of C-SCAN algo which forces the disk arm to move till the last cylinder regardless of knowing whether any request is to be serviced on that cylinder or not

Disk Management

- The disk management activities of the OS includes disk initialization, booting from disk and bad block recovery

① Disk Formatting

- Magnetic disks come in various sizes and so each use different disk drives.
- Different computers have different ways to organizing data on disk surface, so they define their own tracks, sectors, etc
- Same case for disk manufacturer and causes problem for other computer users.
- The concept of disk formatting is used to overcome these problems

② Boot Block

- For a computer to start running it needs to have an initial program to run, This program is called bootstrap program
- Bootstrap program initializes the system, from CPU registers to device controllers and the contents of main memory and then starts the OS.
- Bootstrap is stored in ROM.
  → ROM needs no initialization
  → ROM cannot be infected by computer virus.

③ Bad Blocks

- Disk sector/blocks may become defective. Can no longer store data
- System could not put data there
- Some sectors/blocks of disk can be reserved for mapping. This is called sector sparing

Disk Cache

- Cache memory is used to apply to a memory that is smaller and faster than main memory and that is interposed between main memory and the processor.
- Reduces average memory access time by exploiting the principle of locality
- Disk cache is buffer in main memory for disk sectors.

Linux I/O

- Very similar to other UNIX implementatio
- Associates a special file with each I/O device drivers
- Block, character and network devices are recognized
- Default disk scheduler in linux is Linux Elevator.

Anticipatory I/O scheduler:

- Elevator and deadline scheduling can be counterproductive if there are numerous synchronous read requests
- Is superimposed on deadline scheduler.
- When a read request is dispatched the anticipatory scheduler causes the scheduling system to delay

Linux Page Cache.

- For Linux, there is single unified page cache for all traffic between disk and main memory
- Benefits.

① Dirty pages can be collected and written out efficiently

② Pages in the page cache are likely to be referenced again due to temporal locality