Input/Output management & 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, Controller
3. Communication
   - Suitable for communicating with remote devices
   - Modem, digital line drivers

Differences in I/O devices

1. Data rate
   - There may be differences of magnitude bet" data transfer rate
2. Application
   - The use to which a device is put has an influence on the software
3. Complexity of control
   - The effect on the OS 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

Organization of the I/O function

- Goal is to be general as possible

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
   → If non blocking
   - Processor continues to execute instructions from the process that issued the I/O command
   → If blocking
   - The next instruction the process 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 exchange of data between main memory and an I/O module

Evolution of the I/O function

- Processor directly controls a peripheral device
- A controller or I/O module is added
- Some configuration as step above, but now interrupts are employed
- The I/O module is given direct control of memory via DMA
- The I/O module is enhanced to become a separate processor with a specialized instruction set tailored for I/O
- The I/O module has a local memory of its own and is in fact a computer in its own right

Design Objectives

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

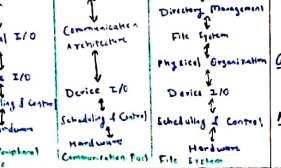
Generality

- Desirable to handle all devices in 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 i/p transfers in advance of requests being made and perform output transfers some time after the request is made

Block oriented device

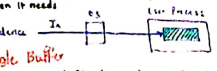
- Stores information in blocks that are usually of fixed size
- Transfers are made one block at a time
- Possible to reference data by its block number
- Ex: Disks & USB keys

Stream oriented device

- Transfers data in & out as a stream of bytes
- No block structure
- Ex: Terminals, Printers, Communication Ports and most other devices which are not secondary storage

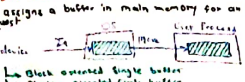
No buffer

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



Single Buffer

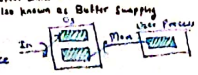
- OS assigns a buffer in main memory for an I/O request



→ Block oriented single buffer
→ Stream oriented single buffer

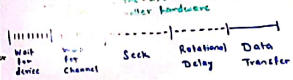
Double buffer

- Use 2 system buffers instead of 1
- A process can transfer data to or from one buffer while the OS fills the other buffer
- Also known as Buffer Swapping



Disk Scheduling Algorithms

- The actual details of disk I/O operation depends on
  → Computer system
  → The I/O channel and other hardware



Seek Time

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

Rotational Delay

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

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

Transfer Time

- Time taken to transfer the data

Disk Response Time

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

Disk Access Time = Rotational Latency + Seek Time + Transfer Time

Goal → Fairness
→ High throughput
→ Minimal Traveling head time

Algorithms → FCFS
→ SSTF
→ SCAN
→ C SCAN
→ LOOK
→ C LOOK

Mega Watch

FCFS Scheduling Algorithm

- Simplest disk scheduling algo
- 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

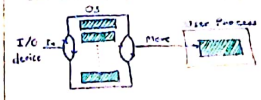
- The scheme does not optimize the seek time
- The request may come from different processes therefore there is the possibility of inappropriate movement of the head

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 variety of I/O and process activity to service, buffering can increase the efficiency of the OS and the performance of individual processes

Circular Buffer

- 2 or more buffers are used
- Each individual buffer is 1 unit in a circular buffer
- Used when I/O operation must keep up with process



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 closest track in service queue

Disadvantages

- It may cause starvation for some request
- Switching direction on the frequent basis slows the working of algorithm
- It is not the most optimal algorithm

Scan Algorithm

- Also known as Elevator algorithm
- In this algo, 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 this 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 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 Algorithm

- 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 cylinder 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 what any request is to be serviced on that cylinder or not

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

- Similar to other unix implementation
- Associates a special file with each I/O device driver
- Block, character and network devices are recognized
- Default disk scheduler in linux is basic 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

- Dirty pages can be collected and written out efficiently
- Pages in the page cache are likely to be referenced again due to temporal locality

Disk Management

- The disk management activities of the OS includes
  → Disk initialization
  → Booting from disk
  → Bad block recovery

Mega Watch