澳 門 理 工 學 院
Instituto Politécnico de Macau
Macao Polytechnic Institute

School of Applied Sciences (B.Sc. in Computing)

Notes #8: I/O and Disk Scheduling

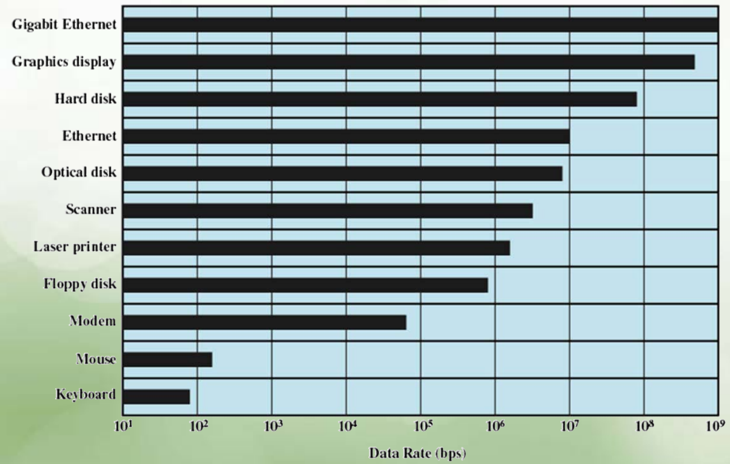
COMP 213
(211/212)
Operating Systems

2019-2020 1st Semester

Differences in I/O Devices

- Data transfer rate
- <u>Application</u>
- Complexity of control
- Unit of transfer – data may be transferred as
  - a stream of bytes (e.g. terminal), stream-oriented
  - in larger blocks (e.g. disk), block-oriented
- Data representation
- Error conditions

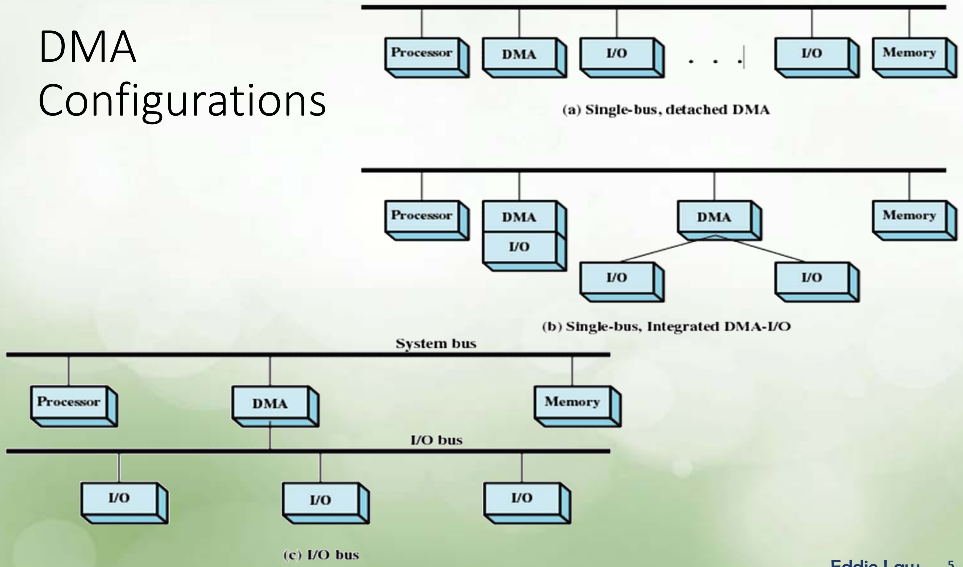
Typical I/O Devices Data Rates



Performing I/O

- Programmed I/O
  - Process is busy-waiting for the operation to complete
- Interrupt-driven I/O
  - I/O command is issued
  - Processor continues executing instructions
  - I/O module sends an interrupt when done
- Direct Memory Access (DMA)
  - DMA module controls exchange of data between main memory and the I/O device
  - Processor interrupted only after entire block has been transferred

DMA Configurations

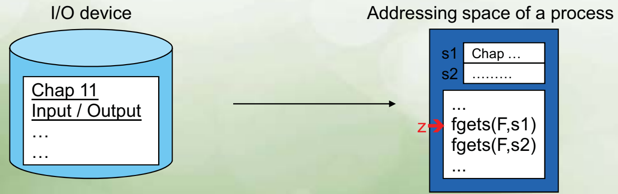


Techniques for Performing I/O

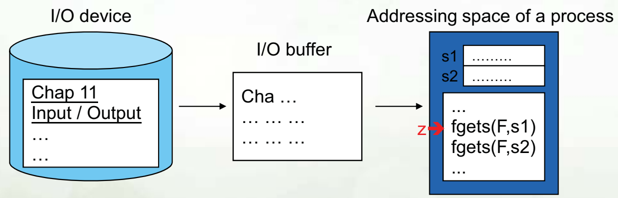
	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		DMA

I/O Buffering

- Reasons for buffering
  - Processes must wait for I/O to complete before proceeding
  - Certain pages must remain in main memory during I/O



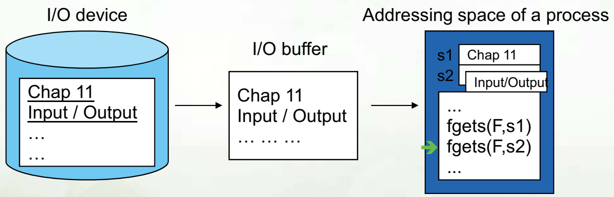
How I/O Buffer Works (1)



To avoid these overheads and inefficiencies, it is sometimes convenient to perform input transfers in advance of requests being made and to perform output transfers sometime after the request is made. This technique is known as buffering.

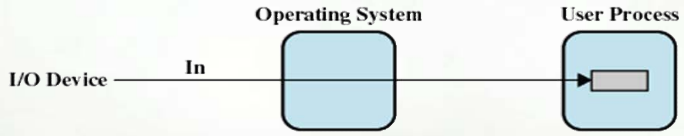
- OS reads data from I/O device to I/O buffer. User process reads from I/O buffer. (similar for data write)
- I/O buffer is not swapped out.

How I/O Buffer Works (2)

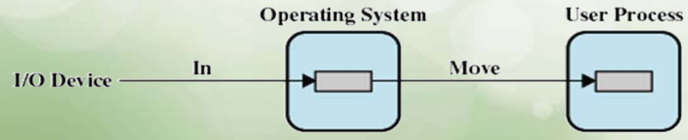


When the process requests the first line, it is blocked. A few lines will be read, in advance, from the I/O device to the I/O buffer. Later, when the process requests the second line, the OS can satisfy the request from the I/O buffer without blocking.

I/O Buffering



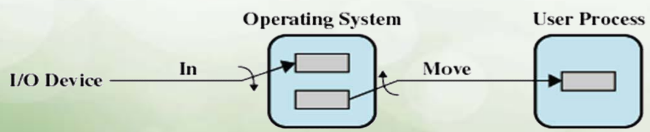
(a) No buffering



(b) Single buffering

Double Buffer

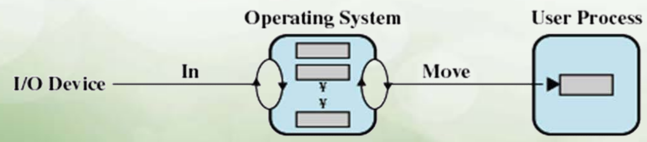
- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer



(c) Double buffering

Circular Buffer

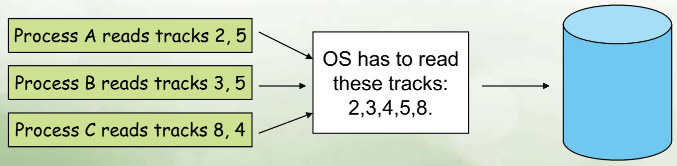
- More than two buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must keep up with process



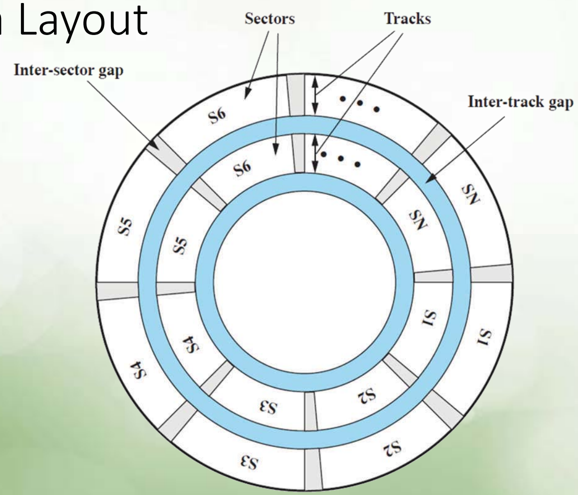
(d) Circular buffering

Disk Scheduling

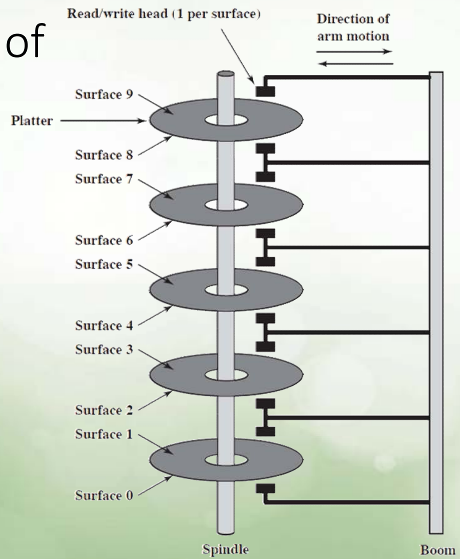
- At runtime, I/O requests for <u>disk tracks</u> come from the processes
- OS has to choose an order to serve the requests



Disk Data Layout



Components of a Disk Drive



Access time

- Total access time = seek time + rotational delay + data transfer time
- Seek time – time required to move the disk arm to the required track
- Rotational delay – time required to rotate the disk to the required sector
- Data transfer time – time to read/write data from/to the disk

Disk Scheduling

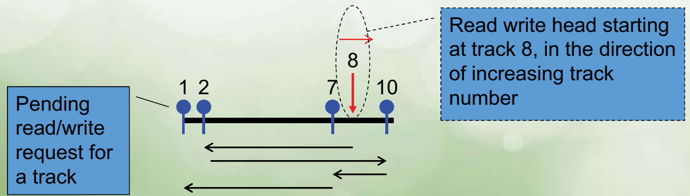
- The order that the read/write head is moved to satisfy several I/O requests
  - determines the total seek time
  - affects performance
  - the OS cannot change the rotational delay or transfer time, but it can try to find a ‘good’ order that spends less time in seek time.
- If requests are selected randomly, we will get the worst possible performance...

Disk Scheduling Policy

- FIFO: fair, but near random scheduling
- SSTF: possible starvation
- SCAN: favor requests for tracks near the ends
- C-SCAN
- FSCAN: avoid “arm stickiness” in SSTF, SCAN and C-SCAN

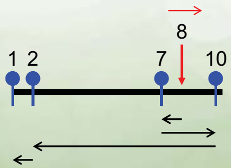
First-in-first-out (FIFO)

- Process requests in the order that the requests are made
- Fair to all processes
- Approaches random scheduling in performance if there are many processes



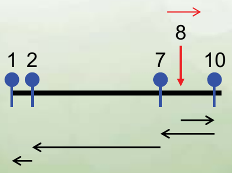
Shortest Service Time First (SSTF)

- Select the disk I/O request that requires the ***least movement*** of the disk arm from its current position
- Always choose the minimum seek time
- New requests may be chosen before an existing request



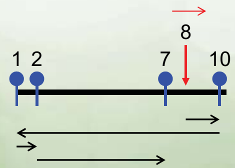
SCAN

- Arm moves in ***one direction only***, satisfying all outstanding requests until there is no more requests in that direction; the ***service direction is then reversed***
- Favor requests for tracks near the ends



C-SCAN (Circular-SCAN)

- ***Restrict scanning to one direction only***
- When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again



FSCAN

- "Arm stickiness" in SSTF, SCAN, C-SCAN in case of repetitive requests to one track
- FSCAN ***uses two queues***
  - When a SCAN begins, all of the requests are in one of the queues, with the other empty
  - During the scan, all new requests are put into the other queue
- Service of new requests is deferred until all of the old requests have been processed

Example

Trace the policies FIFO, SSTF, SCAN, C-SCAN and FSCAN for the following disk requests. Each I/O request on a track takes 5 time units. At time 0, the disk ***starts reading track 10***, and the read/write head was ***moving to the larger track number direction***

Time	0	1	2	3	6	7
Request to access track ..	10	19	3	14	12	9

- Track requests shown on table
- Serving track #10 when starts
- For FIFO
  - 10 → 19, i.e., 9 tracks
  - 19 → 3, i.e., 16 tracks
  - 3 → 14, i.e., 11 tracks
  - 14 → 12, i.e., 2 tracks
  - 12 → 9, i.e., 3 tracks
  - Total 9+16+11+2+3=41 for 5 new requests
  - i.e, 8.2 tracks per request on average

- SSTF
  - Time #0, reading track 10
    - Done by time #5, queued requests: (19, 3, 14)
  - Next to serve: track 14
    - i.e., 10 → 14, i.e., 4 tracks
    - Done by time #10, queued requests: (19, 3, 12, 9)
  - Hence, overall serve order
    (10, 14, 12, 9, 3, 19)
  - Total tracks travelled
    = 4+2+3+6+16 = 31 for the 5 new requests
  - On average, 6.2 tracks per new request

Example (cont'd)

Time	0	1	2	3	6	7
Request to access track ..	10	19	3	14	12	9

- SCAN
  - Among (19, 3, 14), next will be 14
  - Among (19, 3, 12, 9), next will be 19
  - Then the overall serve order is
    (10, 14, 19, 12, 9, 3)
  - Total tracks travelled
    = 4+5+7+3+6 = 25 tracks
  - On averge, 25/5 = 5 tracks / request
- C-SCAN
  - Next is 14 from (19, 3, 14)
  - Next is 19 from (19, 3, 12, 9)
  - Overall serve order is
    (10, 14, 19, 3, 9, 12)
  - Total tracks travelled
    = 4+5+16+6+3 = 34 tracks
  - On average, 34/5 = 6.8 tracks / request

Example (cont'd)

Time	0	1	2	3	6	7
Request to access track ..	10	19	3	14	12	9

- FSCAN
  - Serving track 10 when start
  - Serving queue (19, 3, 14) requests
    - Serve order (10, 14, 19, 3)
    - Requests in another queue (12, 9)
  - Hence, serve order
    (10, 14, 19, 3, 9, 12)
  - Total tracks travelled = 4+5+16+6+3 = 34 for 5 requests
  - On average, 6.8 tracks per request
- On summary

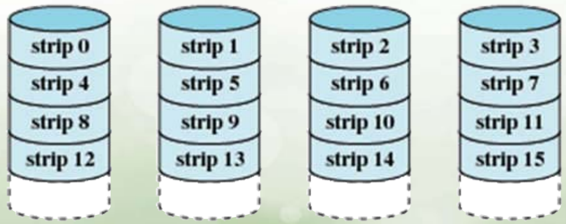
	Track access order	Average seek length
FIFO	10,19,3,14,12,9	(9+16+11+2+3)/5 = 8.2
SSTF	10,14,12,9,3,19	(4+2+3+6+16)/5 = 6.2
SCAN	10,14,19,12,9,3	(4+5+7+3+6)/5 = 5
C-SCAN	10,14,19,3,9,12	(4+5+16+6+3)/5 = 6.8
FSCAN	10,14,19,3,9,12	(4+5+16+6+3)/5 = 6.8

Redundant Array of Independent Disks (RAID)

- A set of physical disk drives viewed by the OS as a single logical drive
- Data are distributed across the physical drives
  - May improve performance
- Redundant disk stores parity information
  - Recoverability, reliability

RAID 0 (Non-Redundant)

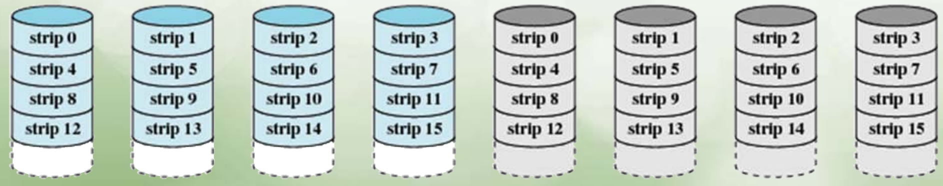
- The logical disk is divided into strips, mapped round robin to consecutive physical disks
- Improve performance in disk read/write
- Not fault tolerant



(a) RAID 0 (non-redundant)

RAID 1 (Mirrored)

- Each disk is mirrored by another disk
- Good performance if the hardware supports concurrent read/write to the mirrored pair
- Reliable, but expensive



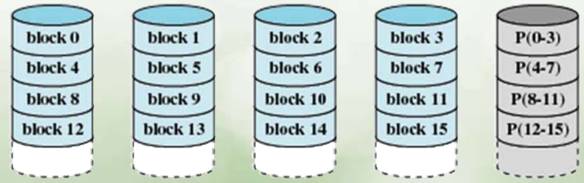
(b) RAID 1 (mirrored)

Parity strip

- Computed and updated at write, verified at read
- Every write results in two read and two write of strips
- A corrupted strip can be recovered

To compute the parity strip...
P(0-3) := b0 ⊕ b1 ⊕ b2 ⊕ b3

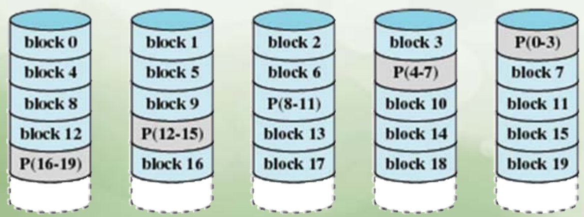
To recover the block 0...
b0 = *P(0-3)* ⊕ b1 ⊕ b2 ⊕ b3



(e) RAID 4 (block-level parity)

RAID 5 (Block-level distributed parity)

- Having all parity strips on one disk may make it a bottleneck. Instead, we can distribute the parity strips among the disks
- If a single disk fails, the system can regenerate the lost data
- Reliable. Good performance with special hardware

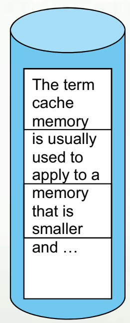


(f) RAID 5 (block-level distributed parity)

Block-Oriented Disk

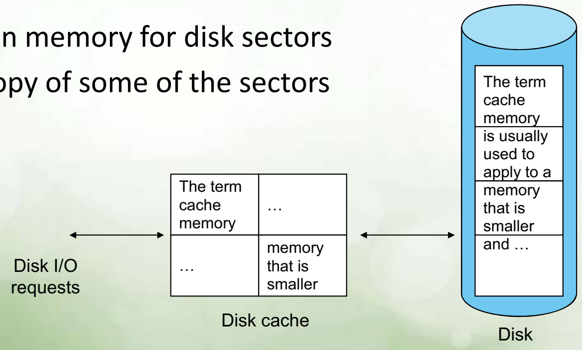
- Disk is block-oriented
- One sector is read/written at a time
- In PC, a sector is 512 bytes

```
while (!feof(F)) {
  // read one char
  fscanf(F, "%c", &c);
  …
}
```



Disk Cache

- Buffer in main memory for disk sectors
- Contains a copy of some of the sectors



Disk Cache, Hit and Miss

- When an I/O request is made for a particular sector, the OS checks whether the sector is in the disk cache
  - If so, (cache hit), the request is satisfied via the cache
  - If not (cache miss), the requested sector is read into the disk cache from the disk

Disk Cache, Replacement

- Least Recently Used (LRU)
  - Replace the block that has been in the cache the longest with no reference
- Least Frequently Used (LFU)
  - Replace the blocks in the set that has experienced the fewest references

Next Topic

- File Management
- Read Ch. 12