Unix SVR4 I/O Management and Disk Scheduling

IT540 Operating Systems

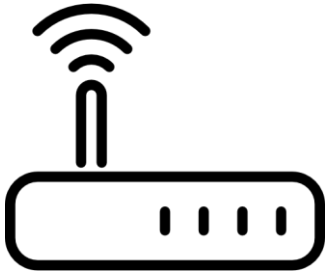
Berat Kaan Çelen
Advisor: Prof. Hasan Hüseyin Balık

Outline

- I/O Devices
  - Categories of I/O Devices
  - Differences in I/O Devices
  - Techniques for Performing I/O
    - Direct Memory Access
  - Design Objectives
  - I/O Buffering
- Disk Drive
  - Disk Performance Parameters
  - Disk Scheduling Algorithms
  - RAID
  - Disk Cache
- UNIX SVR4 I/O

Categories of I/O Devices

- Human readable
  - Communicating with the user
  - Printers, terminals, monitör, keyboard, Mouse
- Machine readable
  - Communicating with the electronic equipment
  - Disk drives, USB sticks, sensors
- Communication
  - Communicating with the remote devices
  - Modems, digital line drivers



Differences in I/O Devices

- Data Rate
- Application
- Complexity of Control
- Unit of Transfer
- Data Representation
- Error Conditions

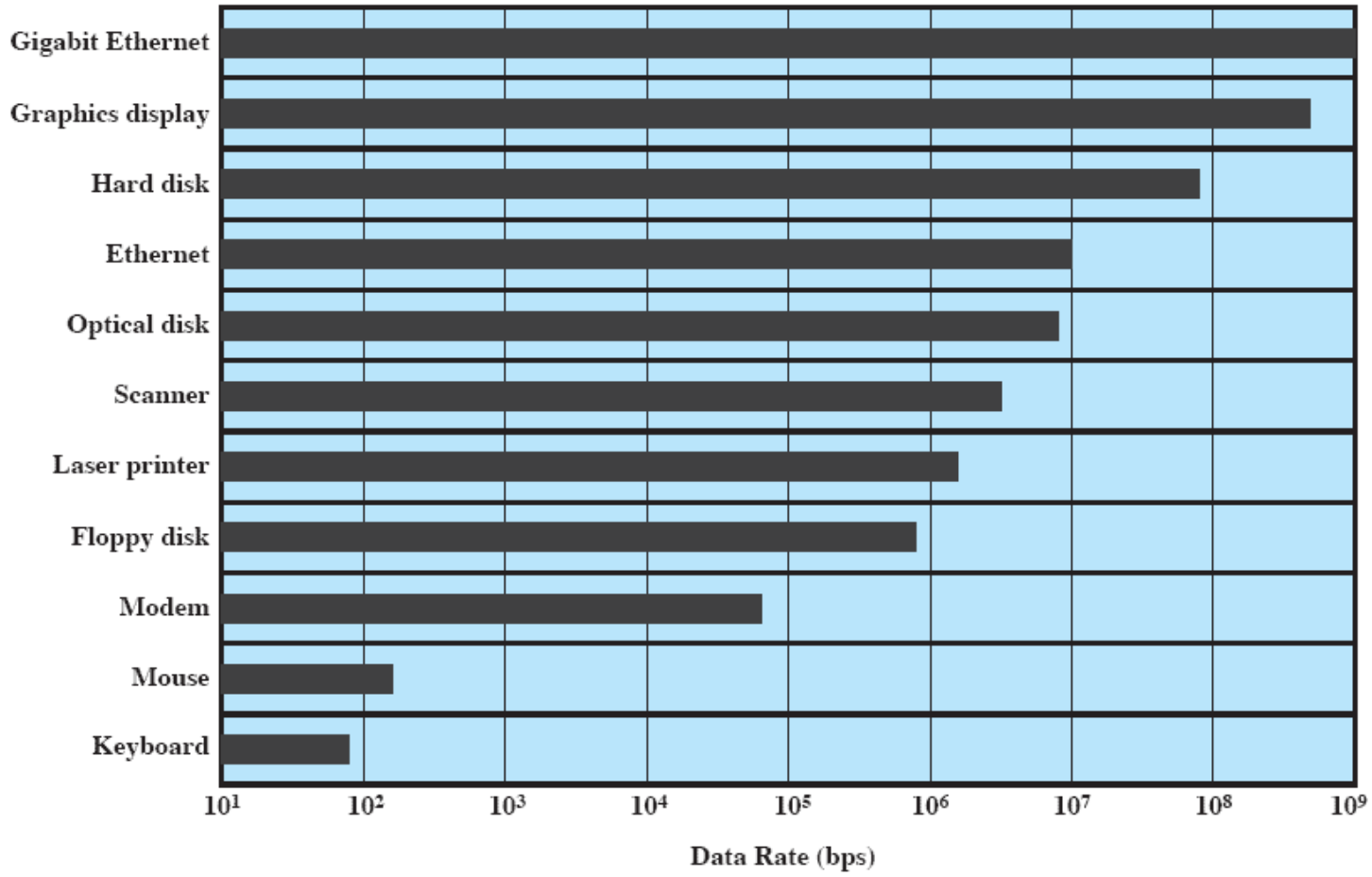


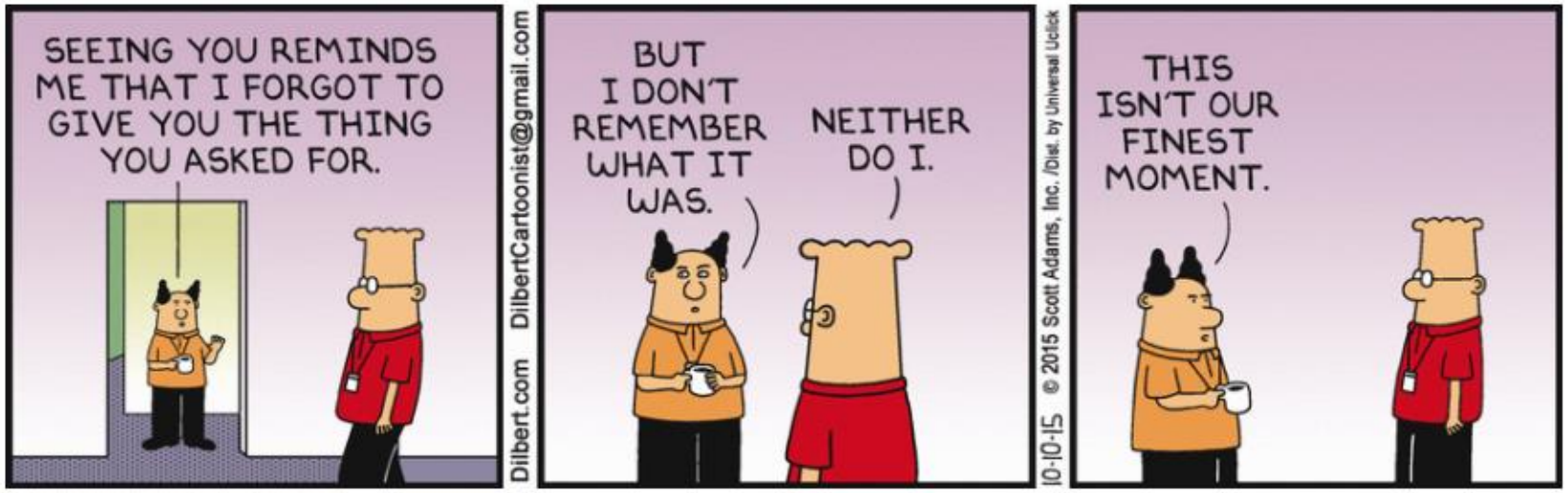
Figure 11.1 Typical I/O Device Data Rates

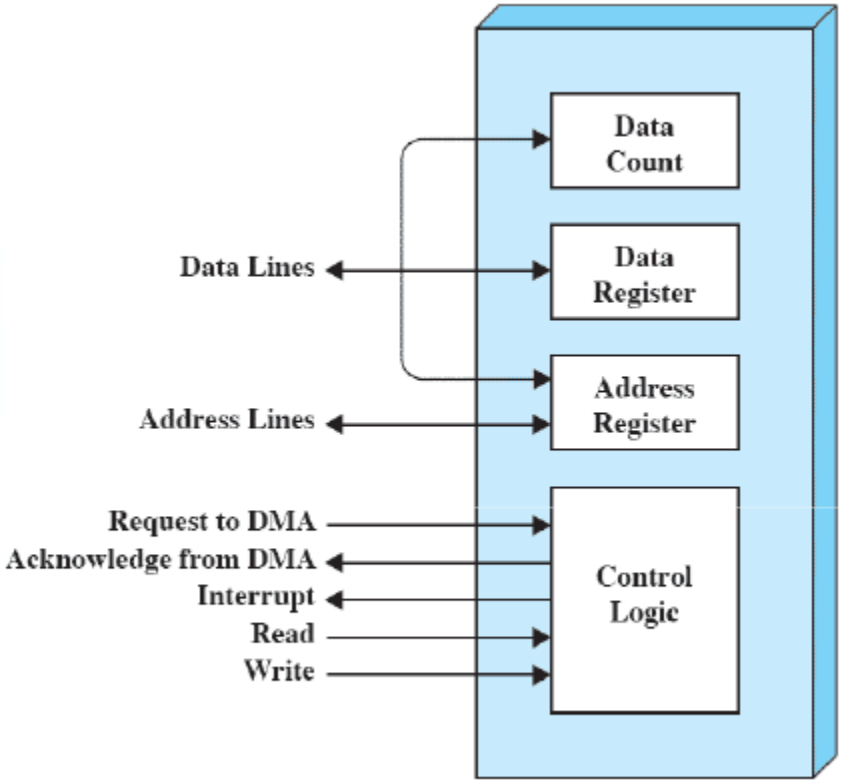
Techniques for Performing I/O

- 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
- Interrupt-driven I/O
  - Efficiency improves as processor does not spend time waiting for an I/O operation to be performed
- Direct Memory Access (DMA)
  - a DMA module controls the exchange of data between main memory and an I/O module

Direct Memory Address

- Processor delegates I/O operation to the DMA module
- DMA module transfers data directly to or from memory
- When complete DMA module sends an interrupt signal to the processor





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

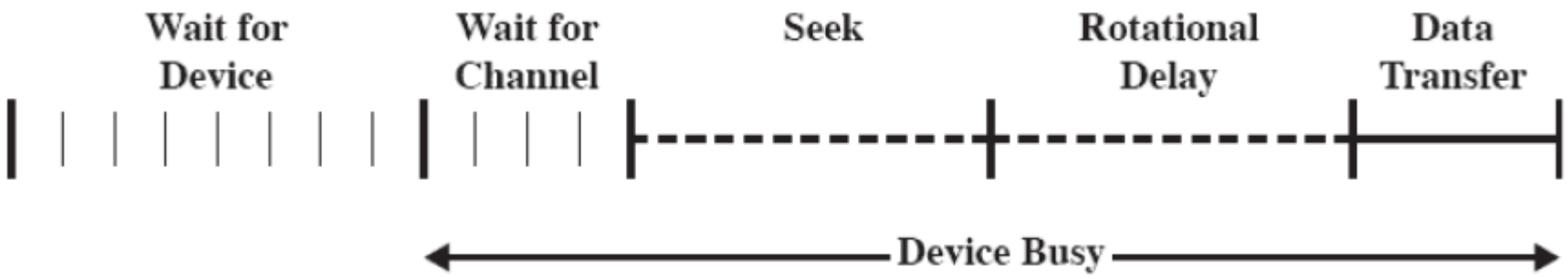
- Desirable to handle all devices in a uniform manner
- Applies to the way processes view I/O devices and the way the operating system manages I/O devices and operations
- Diversity of devices makes it difficult to achieve true generality
- Use a hierarchical, modular approach to the design of the I/O function

I/O Buffering

- Processes must wait for I/O to complete before proceeding, to avoid deadlock certain pages must remain in main memory during I/O
- It may be more efficient to perform input transfers in advance of requests being made and to perform output transfers some time after the request is made.
  - No Buffer
  - Single Buffer
  - Block-Oriented Single Buffer
  - Stream-Oriented Single Buffer
  - Double Buffer
  - Circular-Buffer

Disk Performance Parameters

- **Access Time** is the sum of:
  - **Seek time**: The time it takes to position the head at the desired track
  - **Rotational delay** or **rotational latency**: The time its takes for the beginning of the sector to reach the head
- **Transfer Time** is the time taken to transfer the data.



Disk Scheduling Algorithms

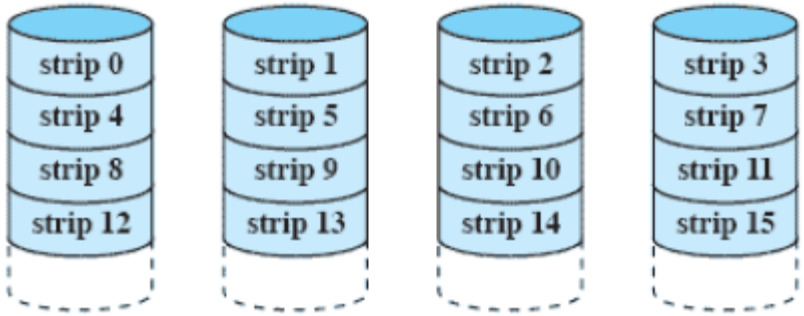
Name	Description	Remarks
Selection according to requestor		
RSS	Random scheduling	For analysis and simulation
FIFO	First in first out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last in first out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of N records at a time	Service guarantee
FSCAN	N-step-SCAN with N = queue size at beginning of SCAN cycle	Load sensitive

RAID

- Redundant Array of Independent Disks
- Set of physical disk drives viewed by the operating system as a single logical drive
- Data are distributed across the physical drives of an array
- Redundant disk capacity is used to store parity information which provides recoverability from disk failure

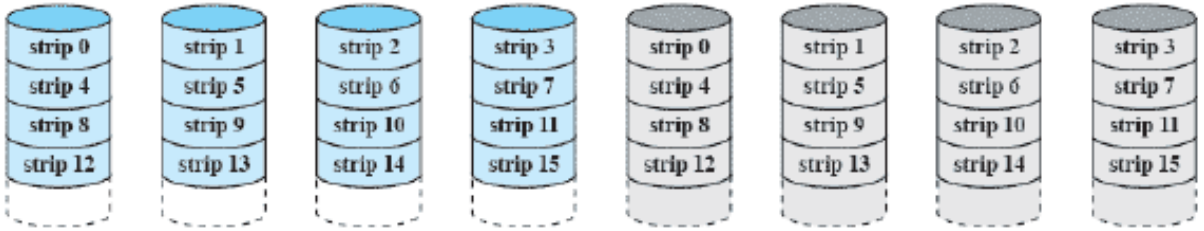
RAID 0 – Stripped

- Not a true RAID – no redundancy
- Disk failure is catastrophic
- Very fast due to parallel read/write



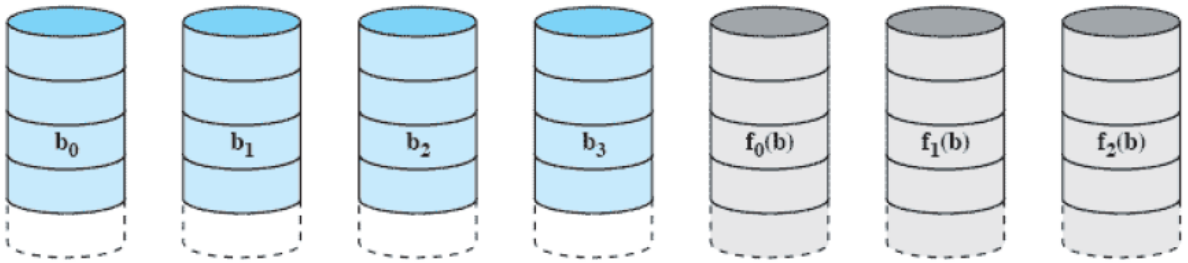
RAID 1 - Mirrored

- Redundancy through duplication instead of parity.
- Read requests can made in parallel.
- Simple recovery from disk failure



RAID 2 – Hamming

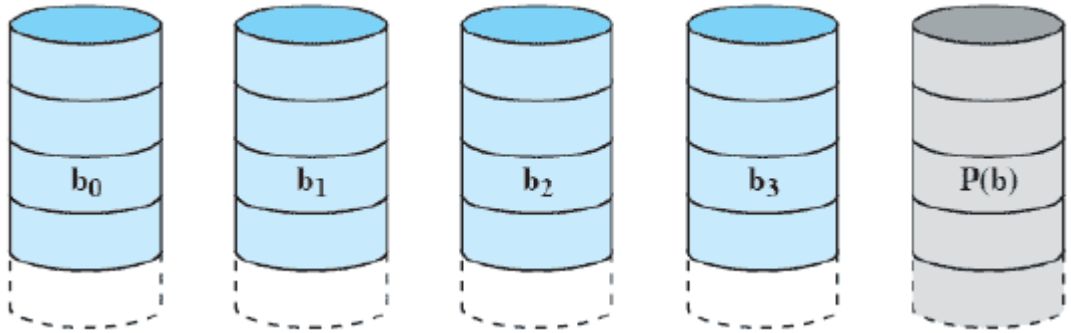
- Synchronised disk rotation
- Data stripping is used (extremely small)
- Hamming code used to correct single bit errors and detect double-bit errors



YouTube Hamming Code - Simply Explained

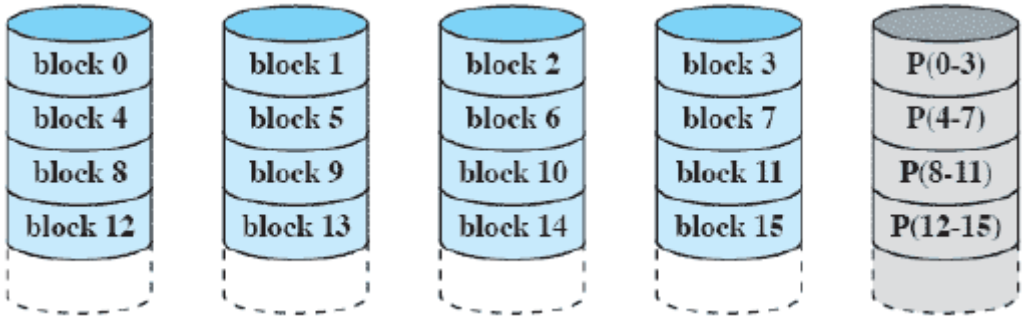
RAID 3 - Bit-interleaved parity

- Similar to RAID-2 but uses all parity bits stored on a single drive



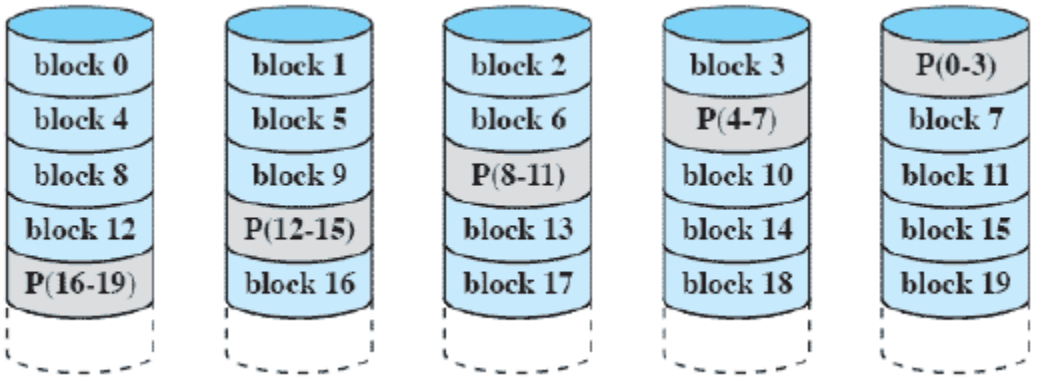
RAID 4

- Block-level parity
- A bit-by-bit parity strip is calculated across corresponding strips on each data disk
- The parity bits are stored in the corresponding strip on the parity disk



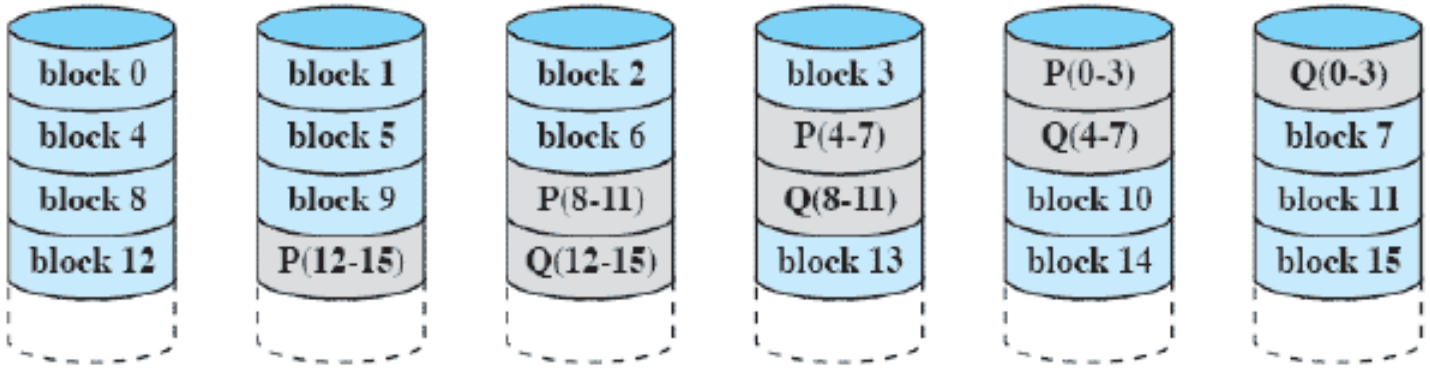
RAID 5

- Block-level Distributed parity
- Similar to RAID-4 but distributing the parity bits across all drives



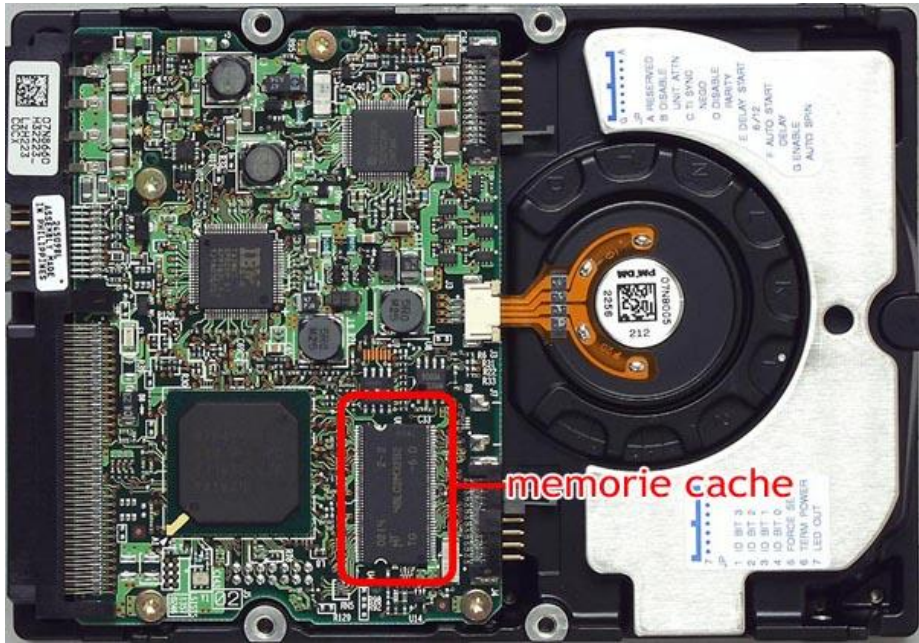
RAID 6 - Dual Redundancy

- Two different parity calculations are carried out
  - stored in separate blocks on different disks.
- Can recover from two disks failing



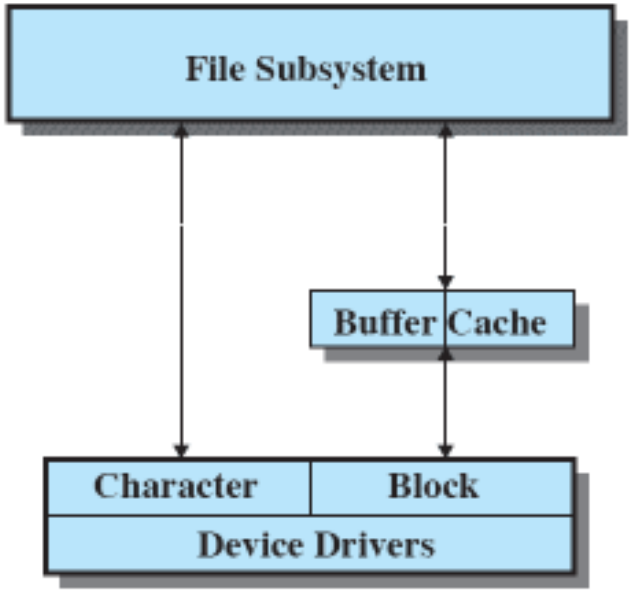
Disk Cache

- Buffer in main memory for disk sectors (ex: HDD with 64mb cache)
- Contains a copy of some of the sectors on the disk
- When an I/O request is made for a particular sector,
  - a check is made to determine if the sector is in the disk cache.
- A number of ways exist to populate the cache
  - Least Recently Used
  - Least Frequently Used



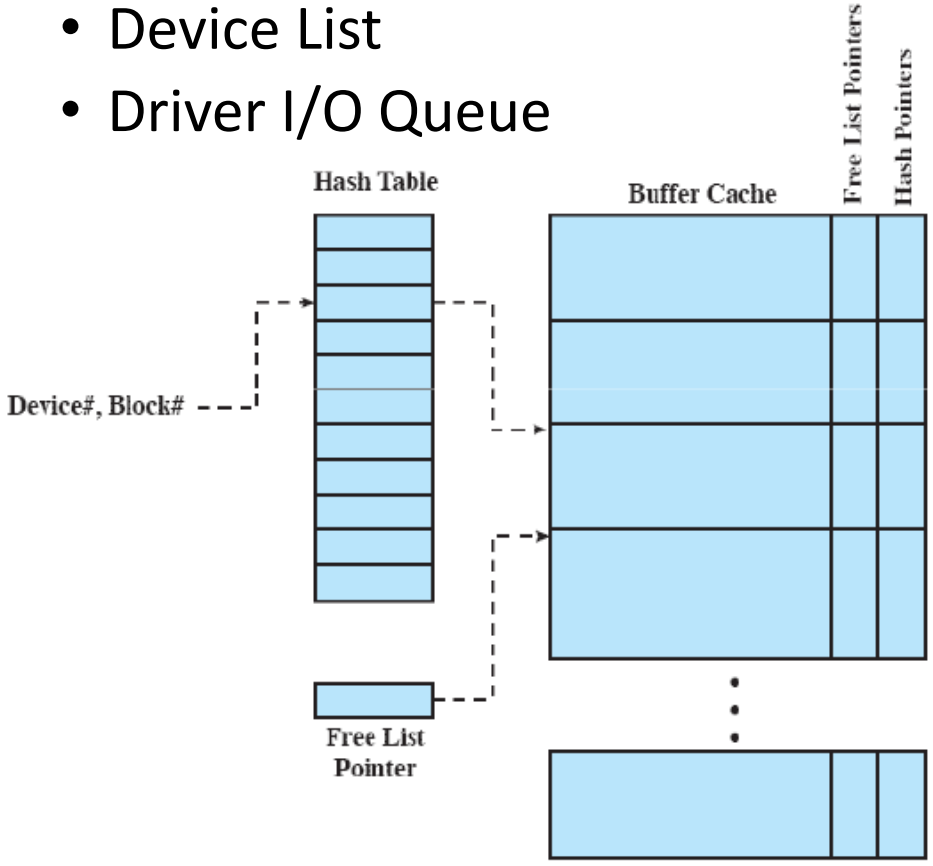
UNIX SVR4 I/O

- Each I/O device is associated with a special file
  - Managed by the file system
  - Provides a clean uniform interface to users and processes.
- To access a device, read and write requests are made for the special file associated with the device.
- Two types of I/O
  - Buffered
    - Buffer Cache
    - Character Queue
  - Unbuffered



Buffer Cache

- Three lists are maintained
  - Free List
  - Device List
  - Driver I/O Queue



Character Queue

- Used by character oriented devices
  - Ex: terminals and printers
- Either written by the I/O device and read by the process or vice versa
  - Producer/consumer model used

Unbuffered I/O

- Unbuffered I/O is simply DMA between device and process
  - Fastest method
  - Process is locked in main memory and can not be swapped out
  - Device is tied to process and unavailable for other processes

	Unbuffered I/O	Buffer Cache	Character Queue
Disk drive	X	X	
Tape drive	X	X	
Terminals			X
Communication lines			X
Printers	X		X

Thank You For Listening

Any Questions?

Resources

1. I/O management - UnixWare 7 Documentation, http://uw714doc.sco.com/en/SEC_admin/_IO_Management.html
2. Sanjiv K. Bhatia, www.cs.umsl.edu/~sanjiv/classes/cs4760/lectures/io.pdf,
3. Indiana Univercity, Knowledge Base, https://kb.iu.edu/d/agjs
4. Dave Bremer Otago Polytechnic, NZ, www.csd.uwo.ca/courses/CS3305a/Chapter11-new.pdf
5. William Stallings, I/O Management and Disk Scheduling, http://cs.nyu.edu/courses/fall12/CSCI-GA.2250-001/slides/Chapter11.pdf
6. Patricia Roy Manatee Community College, Venice, www.dcs.bbk.ac.uk/~szabolcs/CompSys/cs-io.pdf