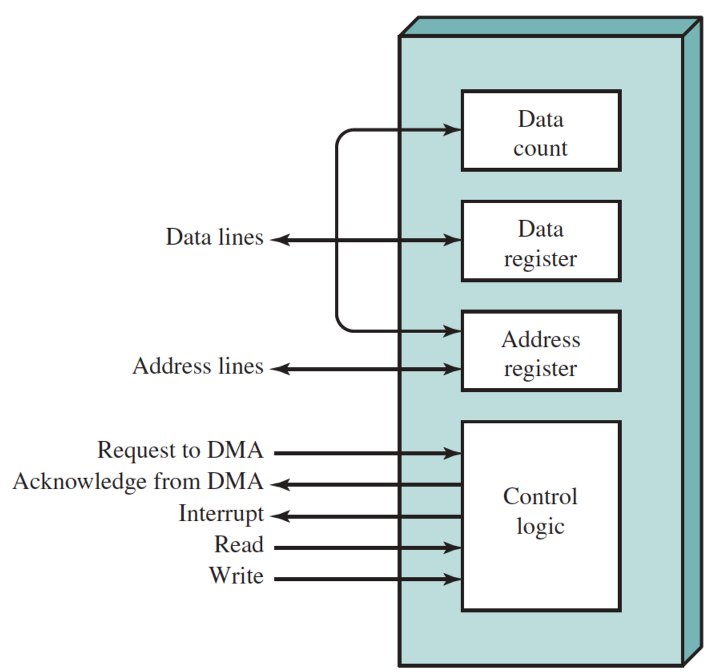
Chapter 11

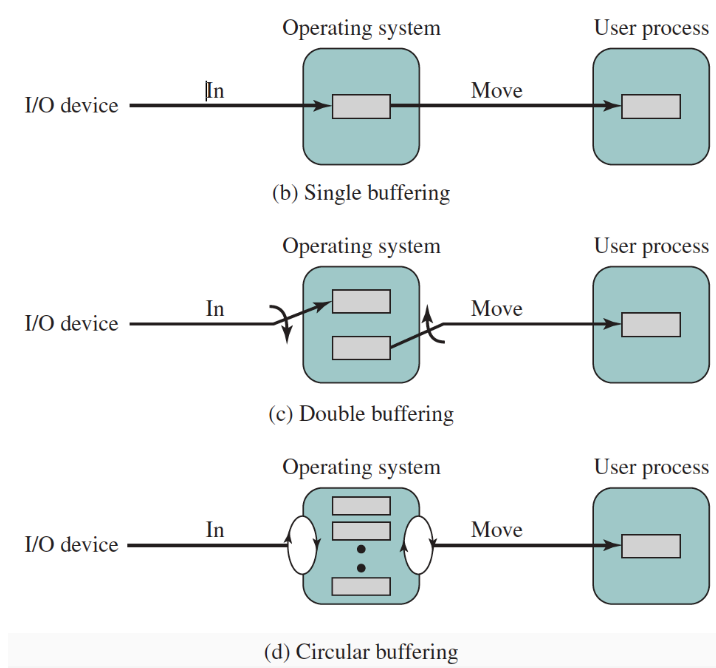
- Categories of I/O devices:
    - Human readable (keyboard, printer)
    - Machine readable (sensors, disk drives)
    - Communication (modems)
- Properties of I/O devices:
    - Data rate
    - Application – the use affects its software
    - Complexity of control
    - Unit of transfer
    - Data representation
    - Error conditions
- Techniques of 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		Direct memory access (DMA)

- Evolution of the I/O function:
    - Processor directly controls peripheral device
    - → programmed I/O module
    - → I/O module w/ interrupts
    - → direct memory access
    - → I/O channel, can execute programs on its own
    - → I/O processor, has its own memory
- **Direct memory access**
    - Processor sends to DMA:
        - Read or write
        - Address of I/O device
        - Starting address
        - # of words
    - DMA transfers data directly between I/O and memory
    - DMA sends interrupt to processor
    - Having DMA share the system bus w/ the processor is inefficient
        - DMA can be directly connected with one or more I/O devices
        - DMA can share I/O bus with all devices



- OS design issues:
    - **Efficiency** – I/O is slow; tend to be bottlenecks
    - **Generality** – desirable to handle all devices in a uniform manner
- Layers of I/O function:
    - Logical I/O – deals with the device as a resource/model
        - Interacts via commands, e.g. open, close, read, write
    - Device I/O – converts operations & data into I/O commands
    - Scheduling & control – queueing & scheduling of I/O operations; interrupts
        - Interacts with actual device hardware
    - In a peripheral device:
        - *User process → logical I/O → device I/O → scheduling & control → hardware*
    - Directory management – converts symbolic file names to identifiers
    - File system – logical organization of files; user operations, e.g. open, close, read, write
    - Physical organization – converts logical file references to physical storage addresses
    - In a file system:
        - *User process → directory mgmt. → file system → physical org. → device I/O → scheduling & control → hardware*
- Block-oriented device – data transfers are made one block at a time
- Stream-oriented device – data is transferred in streams of bytes
- **I/O buffering**
    - Pages of memory involved in I/O need to be locked in MM during I/O
        - E.g. I/O transfer → user memory; user process cannot be swapped out
    - Single buffer – OS assigns a buffer in system memory
        - Block-oriented
            - Input transfer → buffer; buffer block → user space; request another block
            - OS can process one block while next block is being read in
            - Swapping can occur since transfer is in system memory (not user)
        - Stream-oriented
            - Buffer reads in/writes out one line at a time
    - Double buffer
        - Process transfers data to/from one buffer while the OS empties/fills the other buffer
    - Circular buffer
        - Process & OS cycle between >2 buffers

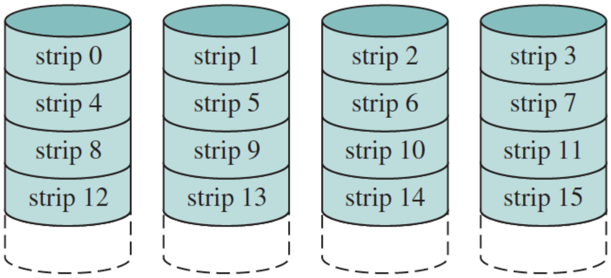


- **Disk scheduling**
- Disk performance
  - Seek time – time taken to position the head of the track
  - Rotational delay – time for beginning of sector to reach the head
  - Access time = seek time + rotational delay
  - Transfer time – time taken to transfer data
  - Sequential access is *much faster* than random access
- Disk scheduling policies
  - FIFO – access disk in the same order the requests were received (fair)
  - Priority – not intended to optimize but to meet OS objectives
    - Starvation possible
  - Last in first out – takes advantage of locality, improves throughput
    - Starvation possible
  - Shortest service time first – select request that requires the least disk arm movement (minimum seek time)
    - Starvation possible
  - SCAN – arm moves in one direction and satisfies all requests along the way; reverse when finished
    - Biased against area most recently visited
  - Circular SCAN – scans in one directory, arm returns to opposite end after finishing
  - N-step SCAN – segments disk request queue into sub-queues of length N
    - Process sub-queues using SCAN
    - N =1 $\rightarrow$ FIFO
- **RAID** – redundant array of independent disks
  - Set of physical disks viewed by OS as a single logical drive
  - Data are distributed across the disks by striping
  - Redundant disk capacity stores parity information, ensures data recoverability

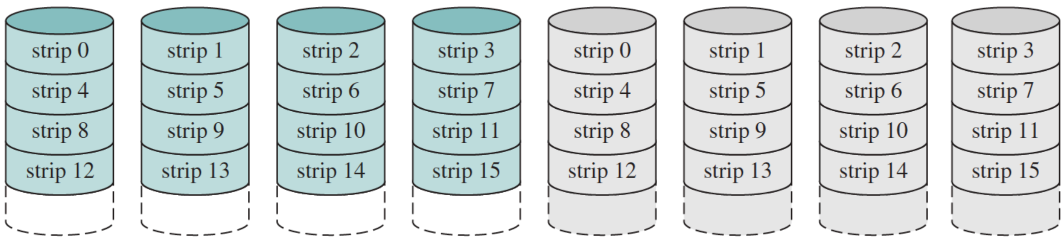
Category	Level	Description	Disks Required	Data Availability	Large I/O Data Transfer Capacity	Small I/O Request Rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

Note: N, number of data disks; m, proportional to log N.

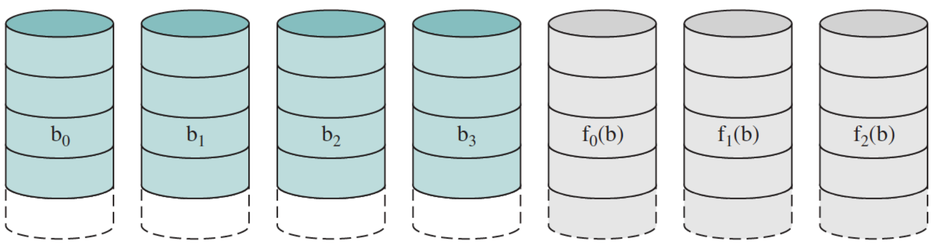
- Striping: (strip 0, 1, 2, 3 form a <u>stripe</u>)



(a) RAID 0 (nonredundant)

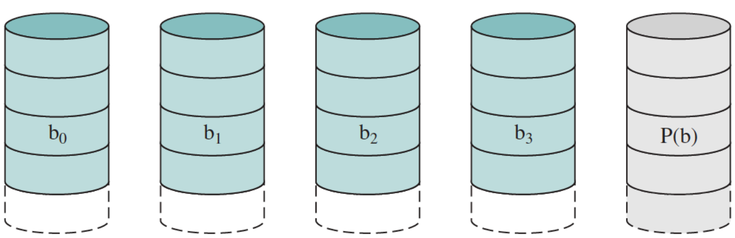


(b) RAID 1 (mirrored)

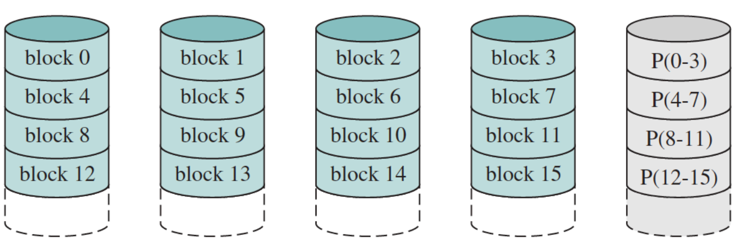


(c) RAID 2 (redundancy through Hamming code)

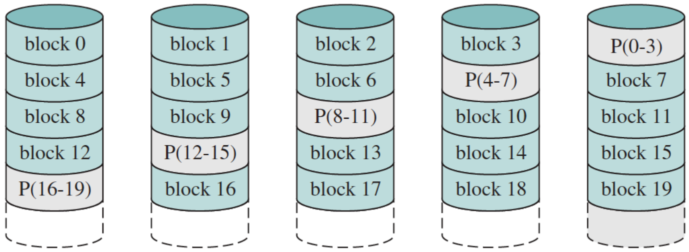
- Redundancy by parity – parity disk stores bits from which lost data can be recovered



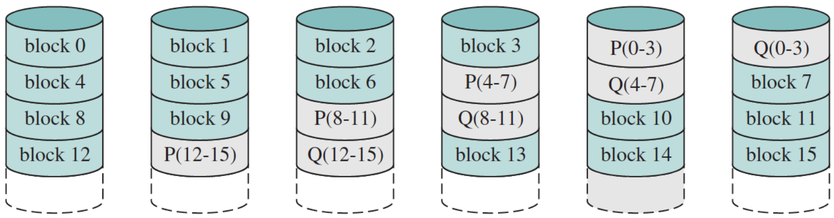
(d) RAID 3 (bit-interleaved parity)



(e) RAID 4 (block-level parity)

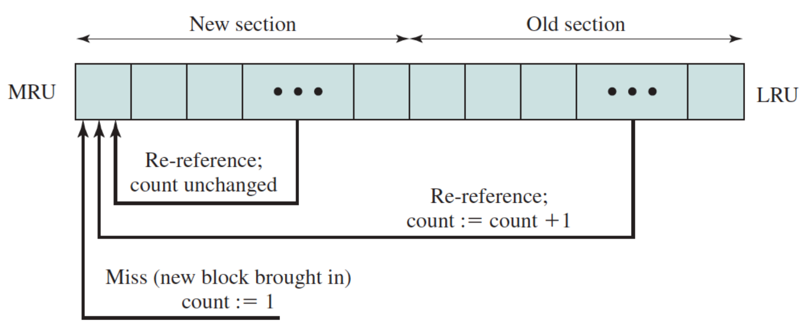


(f) RAID 5 (block-level distributed parity)

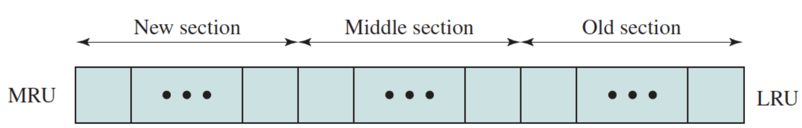


(g) RAID 6 (dual redundancy)

- **Disk cache**
  - Replacement policy
    - Least recently used
    - Least frequently used – block with fewest references
    - Frequency based replacement



(a) FIFO



(b) Use of three sections

- **UNIX SVR4 I/O**
  - Buffered I/O – uses buffer cache (disk cache)
  - Unbuffered I/O – uses DMA
- **Linux I/O**
  - Disk scheduling – uses the elevator scheduler
    - Deadline scheduler
    - Anticipatory I/O scheduler
- **Windows I/O**
  - Synchronous vs. asynchronous I/O

Chapter 12

-