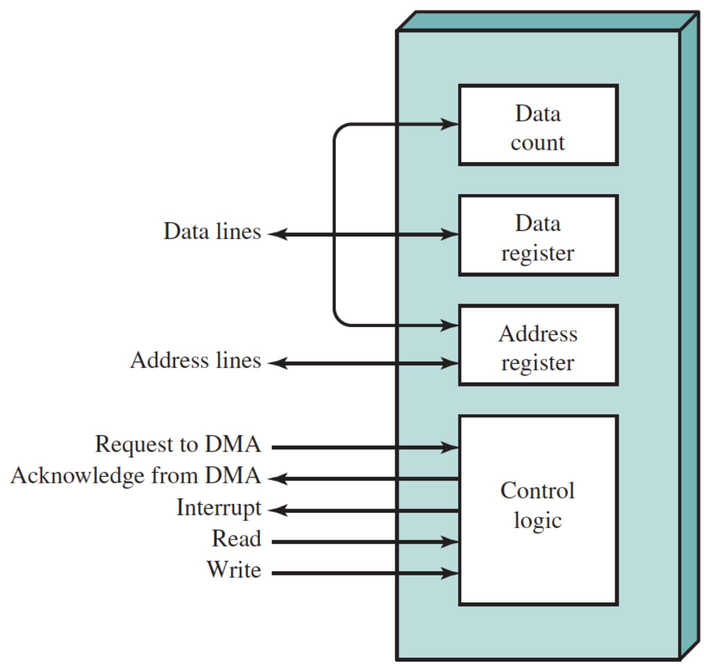
Chapter 11 (11.1 – 11.9)

- 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:

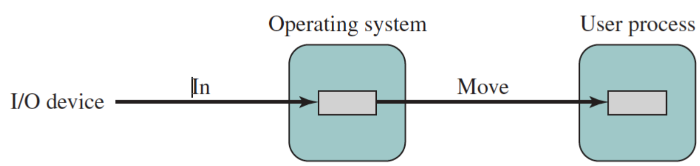
	No Interrupts	**Use of Interrupts**
I/O-to-Memory Transfer through Processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-Memory Transfer		Direct memory access (DMA)

- **Direct memory access**
    - Processor sends to DMA:
        - Read or write
        - Address of I/O device
        - Starting address
        - # of words (length)
    - 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

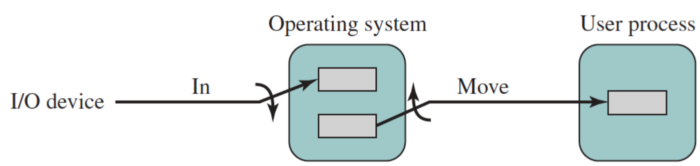


- OS design issues:
    - <u>Efficiency</u> – I/O is slow; tend to be bottlenecks
    - <u>Generality</u> – desirable to handle all devices in a uniform manner
- Layers of I/O function:
    - <u>Logical I/O</u> – deals with the device as a resource/model
        - Interacts via commands, e.g. open, close, read, write
    - <u>Device I/O</u> – converts operations & data into I/O commands
    - <u>Scheduling & control</u> – queueing & scheduling of I/O operations; interrupts

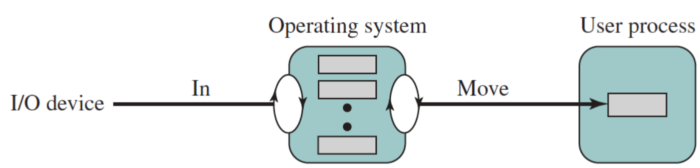
        - Interacts with actual device hardware
    - In a peripheral device:
        - *User process → logical I/O → device I/O → scheduling & control → hardware*
    - Directory management – converts symbolic file names to identifiers
    - File system – logical organization of files; user operations, e.g. 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
        - Solution = buffering
        - Disadvantage: deadlock is possible
    - 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
        - Good for short bursts of I/O



(b) Single buffering



(c) Double buffering



(d) Circular buffering

- **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

Policy	**Description/advantages**	**Disadvantages**
FIFO	• Treats requests fairly	• Performance approaches random access if # of processes is high
Priority	• Not intended to optimize but to meet OS objectives	• Starvation possible
LIFO	• Takes advantage of locality → high throughput	• Starvation possible
Shortest service time first (SSTF)	• Minimizes disk arm movement/seek time	• Starvation possible
SCAN/elevator	• Arm moves in one direction and satisfies all requests along the way; reverse when finished • Deterministic	• Biased against area most recently visited; arm stickiness
LOOK	• Arm reverses at the last request (not the end of track)	
Circular-SCAN	• Only scan in only direction → reduces delay for new requests • Deterministic	• Arm stickiness
N-step SCAN	• Segment request queue into sub-queues of length N • Sub-queues use SCAN; main queue is FIFO • Reduces arm stickiness	
FSCAN	• Two queues for old & new requests • New requests are deferred until old requests are serviced	

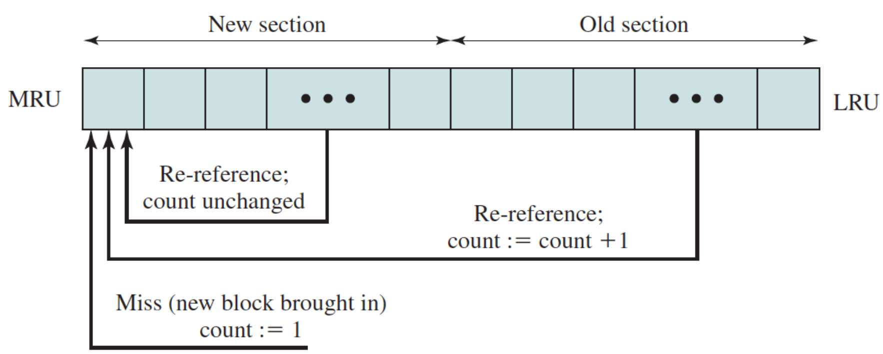
  - Performance:
    - SSTF ≈ SCAN < C-SCAN < FIFO

- **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 <u>striping</u>
    - Logically contiguous strips are spread over multiple disks → can be accessed in parallel, reducing transfer time
  - Redundant disk capacity stores parity information, ensures data recoverability
  - Redundancy by parity – parity disk stores bits from which lost data can be recovered

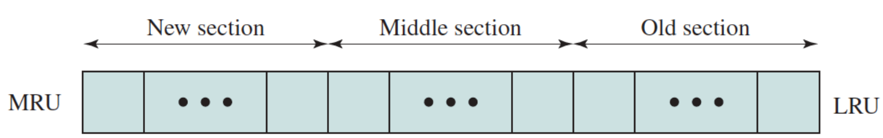
Category	Level	Description	Disks Required	Data Availability	Large I/O Data Transfer Capacity	Small I/O Request Rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single 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.

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



(a) FIFO

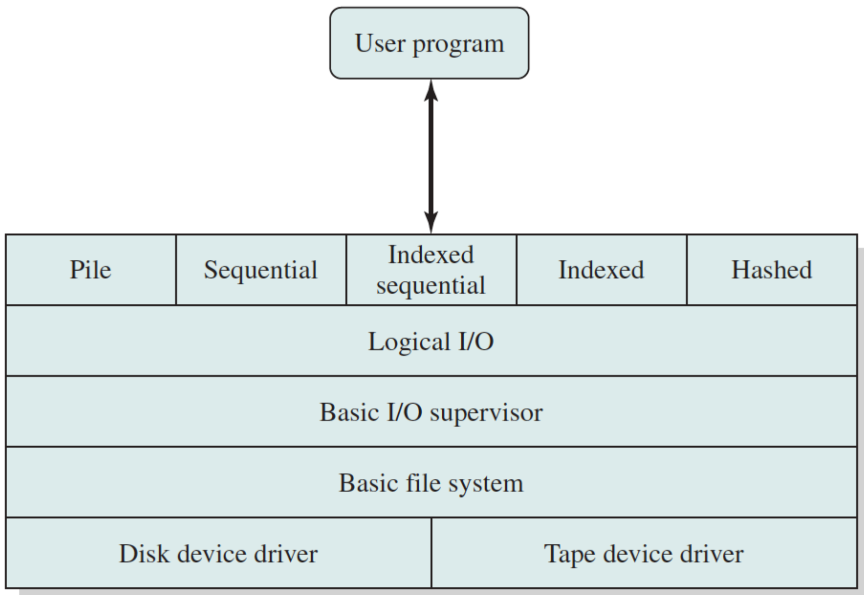


(b) Use of three sections

- **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 (variation of LOOK)
    - Deadline scheduler
    - Anticipatory I/O scheduler

Chapter 12 (12.1 – 12.2, 12.4 – 12.8)

- Properties of files:
  - Long-term existence
  - Sharable between processes
  - Structure
- File structure:
  - Field → record → file → database
- **File system architecture**



- **File organization**
  - Criteria:
    - Short access time
    - Ease of update
    - Economy of storage
    - Simple maintenance
    - Reliability
  - Pile file
    - Chronological list of variable-length records
    - Record access is exhaustive (linear)
  - Sequential file
    - Fixed-length records with fixed fields (i.e. table)
    - Sorted by a key field
    - Record access is exhaustive
  - Indexed sequential file
    - Index – pointer into file that allows faster random access
    - Only key field is indexed
  - Indexed file
    - Exhaustive index – every record is indexed

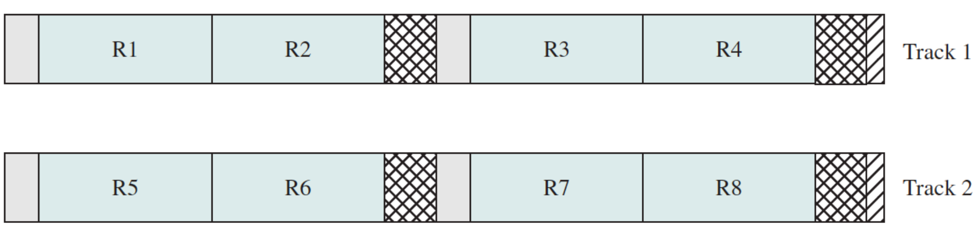
    - Partial index – points to area of interest
- Hashed file
    - No ordering; every key is hashed

	Space Attributes		Update Record Size		Retrieval		
File Method	Variable	Fixed	Equal	Greater	Single record	Subset	Exhaustive
Pile	A	B	A	E	E	D	B
Sequential	F	A	D	F	F	D	A
Indexed sequential	F	B	B	D	B	D	B
Indexed	B	C	C	C	A	B	D
Hashed	F	B	B	F	B	F	E

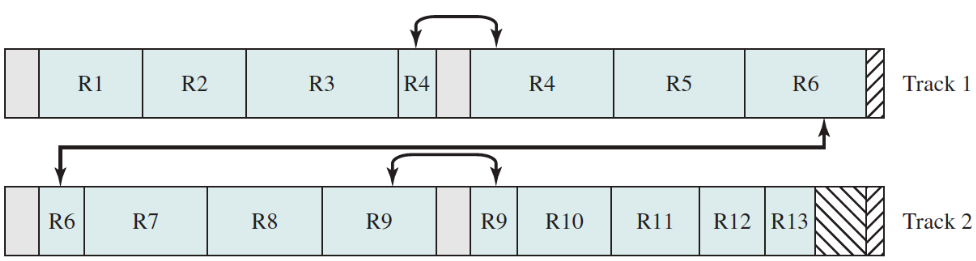
A = Excellent, well suited to this purpose $\approx O(r)$
B = Good $\approx O(o \times r)$
C = Adequate $\approx O(r \log n)$
D = Requires some extra effort $\approx O(n)$
E = Possible with extreme effort $\approx O(r \times n)$
F = Not reasonable for this purpose $\approx O(n^{>1})$

where
r = size of the result
o = number of records that overflow
n = number of records in file

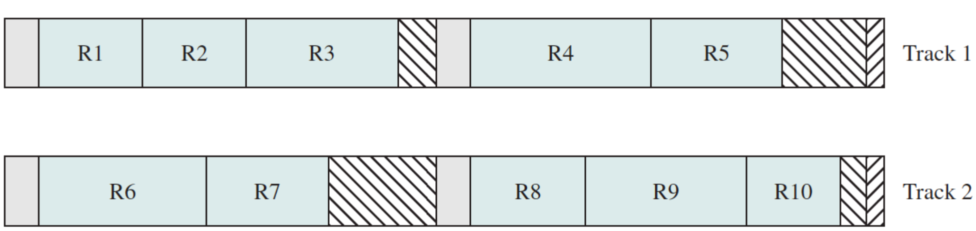
- **Record blocking**



Fixed blocking



Variable blocking: spanned



Variable blocking: unspanned

- File allocation

	Contiguous	Chained	Indexed	
Preallocation?	Necessary	Possible	Possible	
Fixed or Variable Size Portions?	Variable	Fixed blocks	Fixed blocks	Variable
Portion Size	Large	Small	Small	Medium
Allocation Frequency	Once	Low to high	High	Low
Time to Allocate	Medium	Long	Short	Medium
File Allocation Table Size	One entry	One entry	Large	Medium

-