I/O Management and Disk Scheduling

Click to add subtitle

Roadmap

- I/O Devices
- Organization of the I/O Function
- Operating System Design Issues
- I/O Buffering
- Disk Scheduling
- Raid
- Disk Cache
- UNIX SVR4 I/O
- LINUX I/O
- Windows I/O

Categories of I/O Devices

- Difficult area of OS design
  - Difficult to develop a consistent solution due to a wide variety of devices and applications

- Three Categories:
  - Human readable
  - Machine readable
  - Communications

Human readable

- Devices used to communicate with the user
- Printers and terminals
  - Video display
  - Keyboard
  - Mouse etc

Machine readable

- Used to communicate with electronic equipment
  - Disk drives
  - USB keys
  - Sensors
  - Controllers
  - Actuators

Communication

- Used to communicate with remote devices
  - Digital line drivers
  - Modems

Differences in I/O Devices

- Devices differ in a number of areas
  - Data Rate
  - Application
  - Complexity of Control
  - Unit of Transfer
  - Data Representation
  - Error Conditions

Data Rate

- May be massive difference between the data transfer rates of devices

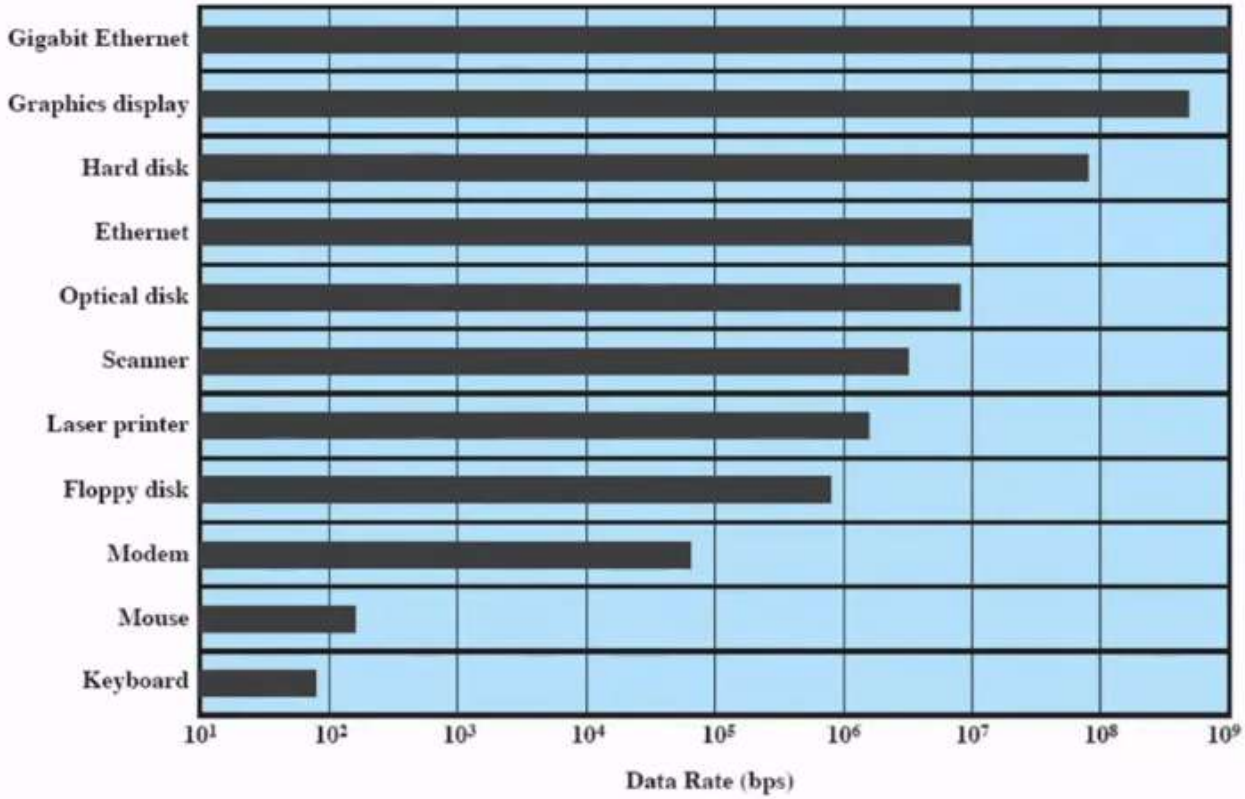


Figure 11.1 Typical I/O Device Data Rates

Application

- Disk used to store files requires file management software
- Disk used to store virtual memory pages needs special hardware and software to support it
- Terminal used by system administrator may have a higher priority

Complexity of control

- A printer requires a relatively simple control interface.
- A disk is much more complex.
- This complexity is filtered to some extent by the complexity of the I/O module that controls the device.

Unit of transfer

- Data may be transferred as
  - a stream of bytes or characters (e.g., terminal I/O)
  - or in larger blocks (e.g., disk I/O).

Data representation

- Different data encoding schemes are used by different devices,
  - including differences in character code and parity conventions.

Error Conditions

- The nature of errors differ widely from one device to another.
- Aspects include:
  - the way in which they are reported,
  - their consequences,
  - the available range of responses

Roadmap

- I/O Devices
- Organization of the I/O Function
- Operating System Design Issues
- I/O Buffering
- Disk Scheduling
- Raid
- Disk Cache
- UNIX SVR4 I/O
- LINUX I/O
- Windows I/O

Techniques for performing I/O

- Programmed I/O
- Interrupt-driven I/O
- Direct memory access (DMA)

Table 11.1 I/O Techniques

	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

1. Processor directly controls a peripheral device
2. Controller or I/O module is added
   - Processor uses programmed I/O without interrupts
   - Processor does not need to handle details of external devices

Evolution of the I/O Function cont…

3. Controller or I/O module with interrupts
   - Efficiency improves as processor does not spend time waiting for an I/O operation to be performed
4. Direct Memory Access
   - Blocks of data are moved into memory without involving the processor
   - Processor involved at beginning and end only

Evolution of the I/O Function cont…

5. I/O module is a separate processor
   - CPU directs the I/O processor to execute an I/O program in main memory.
6. I/O processor
   - I/O module has its own local memory
   - Commonly used to control communications with interactive terminals

Direct Memory Address

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

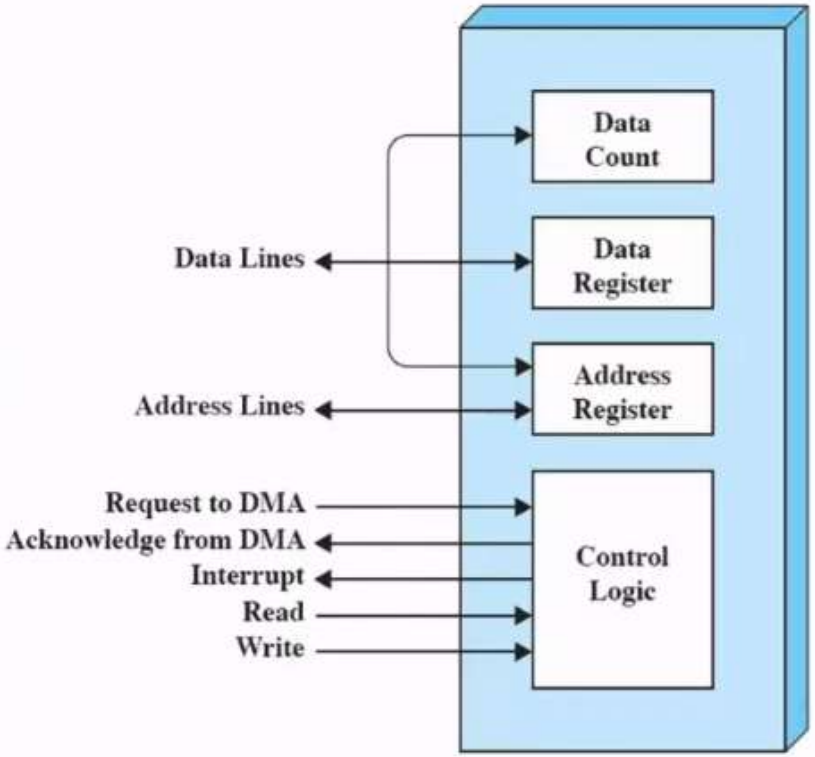
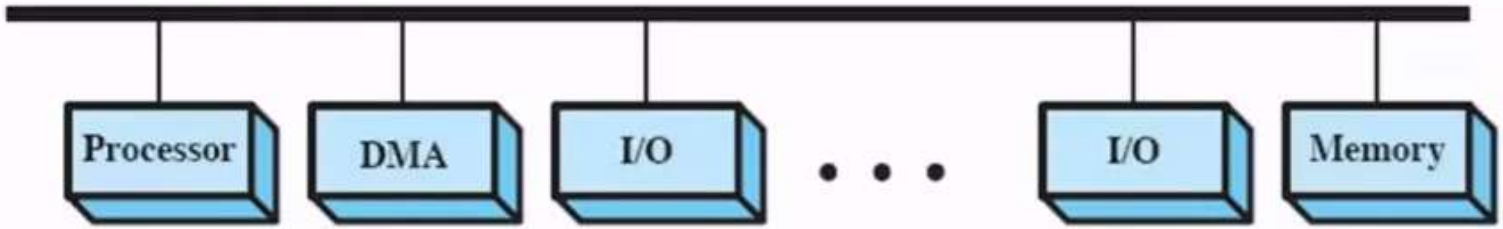


Figure 11.2 Typical DMA Block Diagram

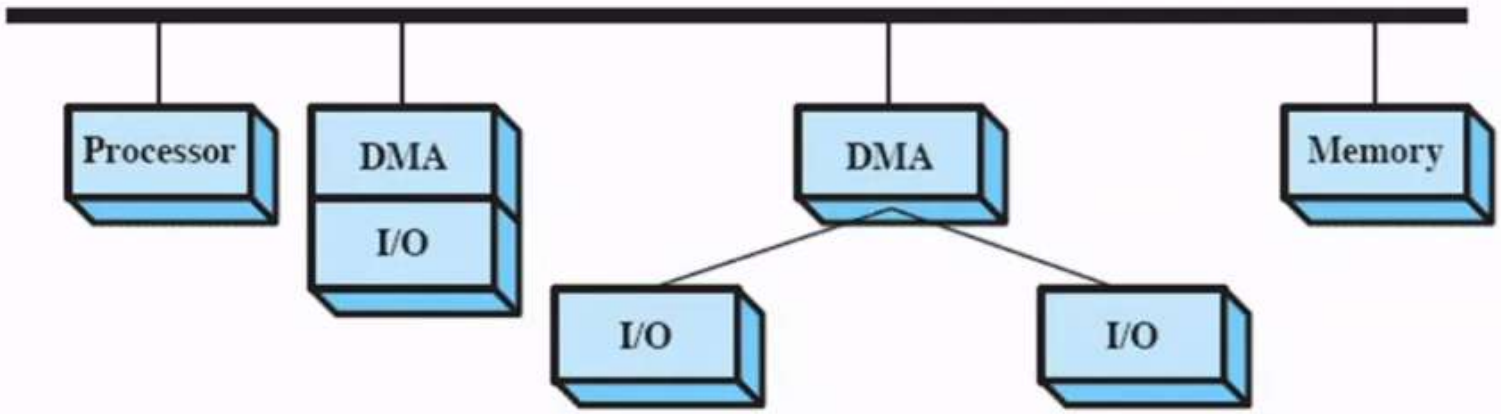
DMA Configurations: Single Bus



(a) Single-bus, detached DMA

- DMA can be configured in several ways
- Shown here, all modules share the same system bus

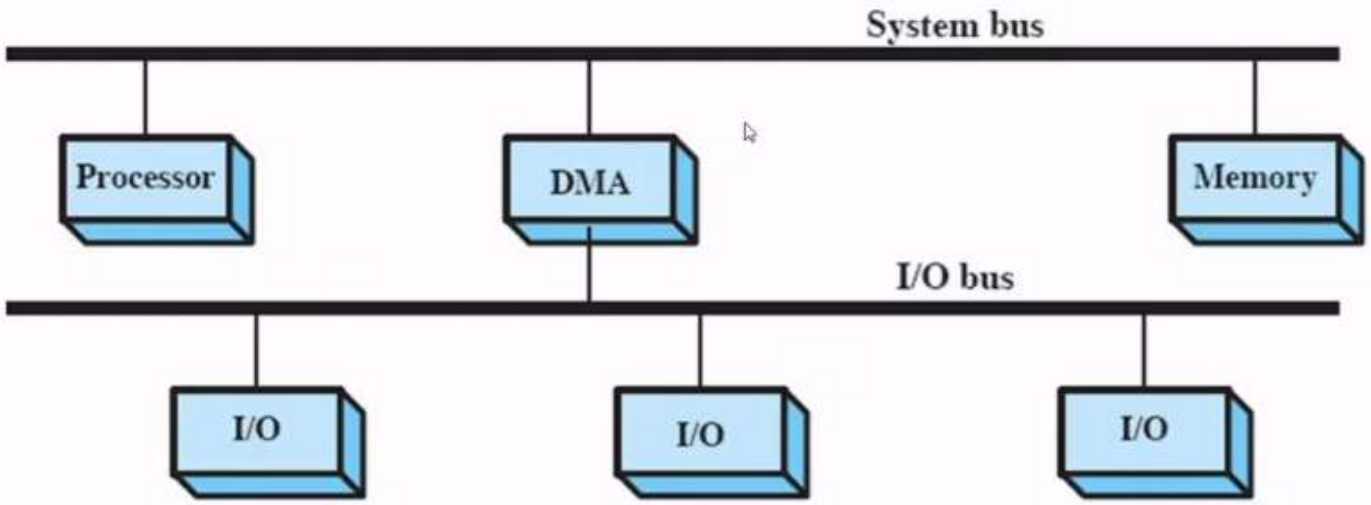
DMA Configurations: Integrated DMA & I/O



(b) Single-bus, Integrated DMA-I/O

- Direct Path between DMA and I/O modules
- This substantially cuts the required bus cycles

DMA Configurations: I/O Bus



(c) I/O bus

- Reduces the number of I/O interfaces in the DMA module

Roadmap

- I/O Devices
- Organization of the I/O Function
- Operating System Design Issues
- I/O Buffering
- Disk Scheduling
- Raid
- Disk Cache
- UNIX SVR4 I/O
- LINUX I/O
- Windows I/O

Goals: Efficiency

- Most I/O devices extremely slow compared to main memory
- Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
- I/O cannot keep up with processor speed
  - Swapping used to bring in ready processes
  - But this is an I/O operation itself

Generality

- For simplicity and freedom from error it is desirable to handle all I/O devices in a uniform manner
- Hide most of the details of device I/O in lower-level routines
- Difficult to completely generalize, but can use a hierarchical modular design of I/O functions

Roadmap

- I/O Devices
- Organization of the I/O Function
- Operating System Design Issues
- I/O Buffering
- Disk Scheduling
- Raid
- Disk Cache
- UNIX SVR4 I/O
- LINUX I/O
- Windows I/O

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.

Block-oriented Buffering

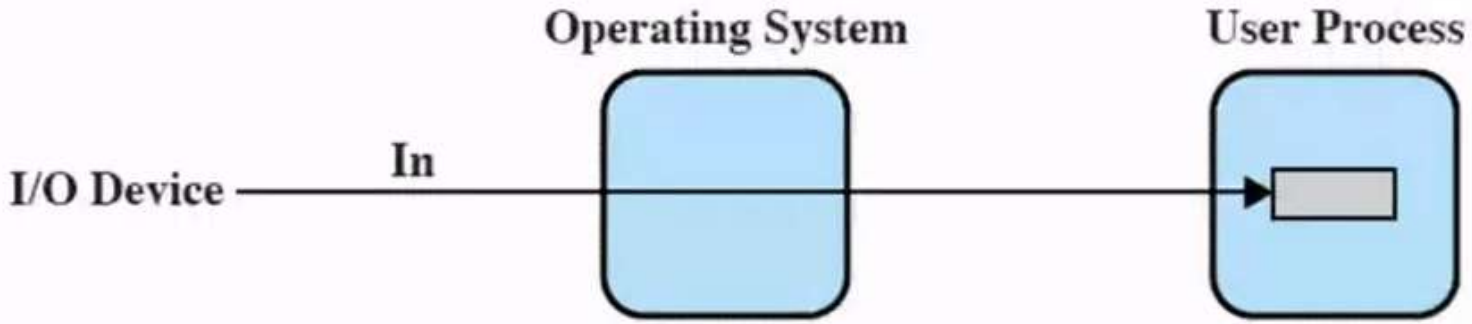
- Information is stored in fixed sized blocks
- Transfers are made a block at a time
  - Can reference data b block number
- Used for disks and USB keys

Stream-Oriented Buffering

- Transfer information as a stream of bytes
- Used for terminals, printers, communication ports, mouse and other pointing devices, and most other devices that are not secondary storage

No Buffer

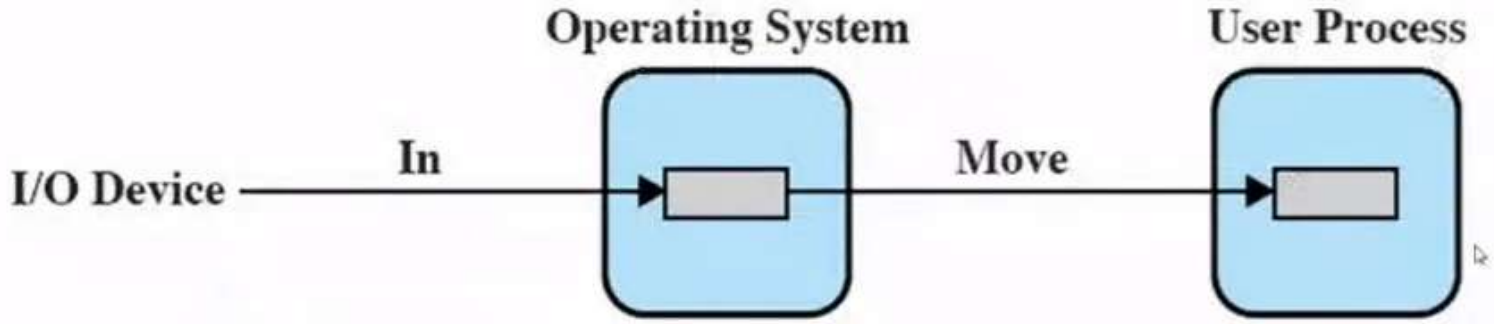
- Without a buffer, the OS directly access the device as and when it needs



(a) No buffering

Single Buffer

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



(b) Single buffering

Block Oriented Single Buffer

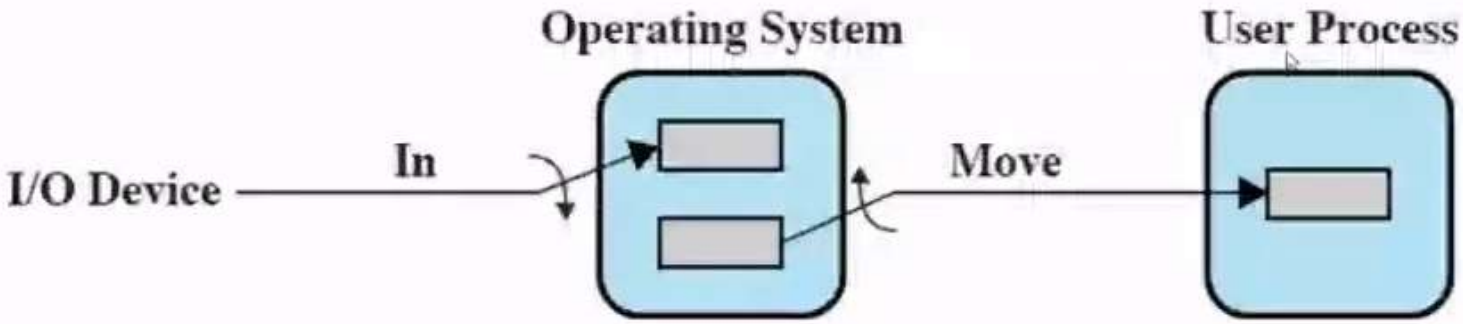
- Input transfers made to buffer
- Block moved to user space when needed
- The next block is moved into the buffer
  - *Read ahead* or *Anticipated Input*
- Often a reasonable assumption as data is usually accessed sequentially

Stream-oriented Single Buffer

- Line-at-time or Byte-at-a-time
- Terminals often deal with one line at a time with carriage return signaling the end of the line
- Byte-at-a-time suites devices where a single keystroke may be significant
  - Also sensors and controllers

Double Buffer

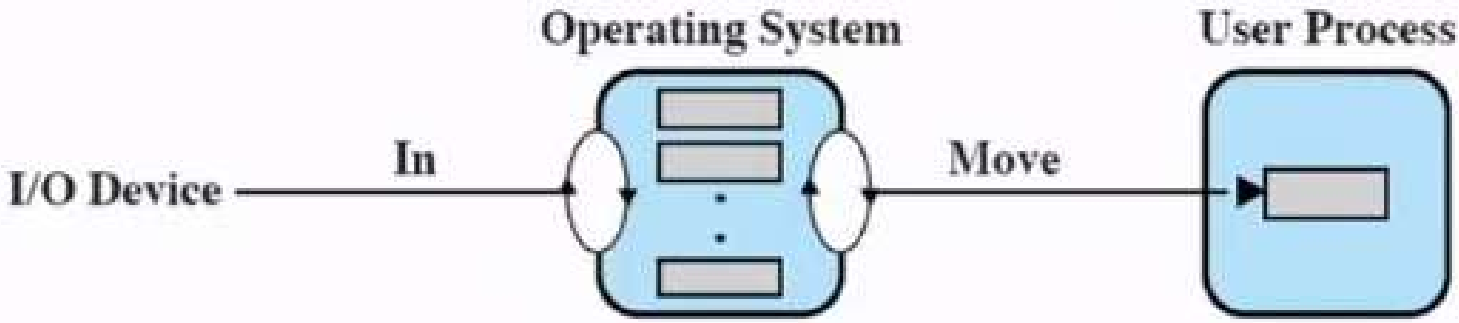
- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer



(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

Buffer Limitations

- Buffering smoothes out peaks in I/O demand.
  - But with enough demand eventually all buffers become full and their advantage is lost
- However, when there is a variety of I/O and process activities to service, buffering can increase the efficiency of the OS and the performance of individual processes.