# I/O Management

Chapter 7

1

1

## Objectives

- Explore the structure of an operating system's I/O subsystem
- Discuss the principles of I/O hardware and its complexity
- Provide details of the performance aspects of I/O hardware and software

2

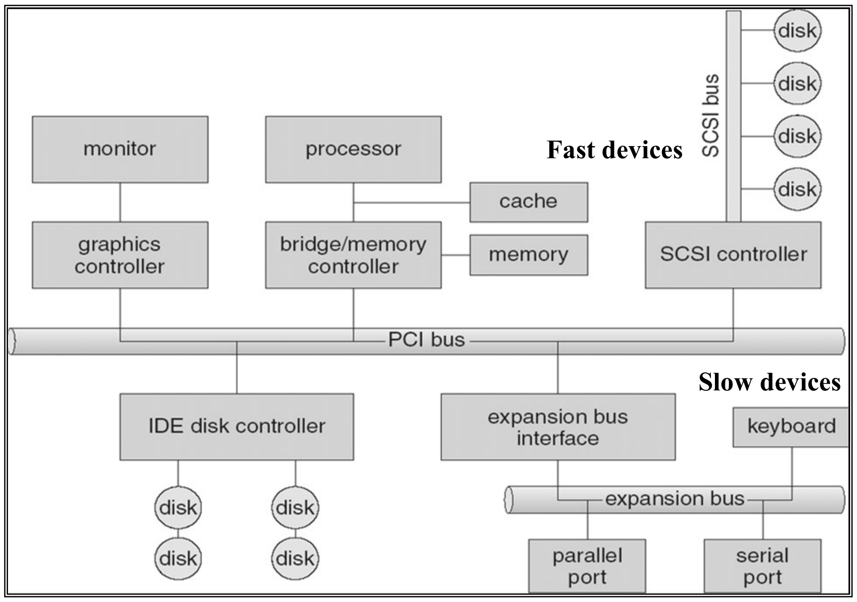
## I/O Hardware

- Incredible variety of I/O devices
- Common concepts
  - **Port**
    - connection point, for device to communicate with machine
  - **Bus** (**daisy chain** or shared direct access)
    - common set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.
    - PCI bus (the common PC system bus) connects the processor-memory subsystem to the fast devices.
      - Peripheral Component Interconnect (PCI)
    - Expansion bus connects slow devices.
    - Small Computer Systems Interface (SCSI) bus

3

3

## A Typical PC Bus Structure



4

## I/O Hardware

- Common concepts
  - **Controller** (**host adapter**)
    - Collection of electronics that can operate a port, a bus, or a device.
    - Serial-port controller – simple; a single chip that controls the signal on the wires of a serial port.
    - SCSI port controller – not as simple; often implemented as a separate circuit board (or a host adapter) that plugs into the computer. Typically contains a processor, microcode, and some private memory.
    - Some devices have their own built-in controller, e.g. disk drives (has a circuit board attached to one side)

## I/O Hardware

- I/O instructions control devices
  - The controller has 1 or more registers for data and control signals, where the processor reads/writes bit patterns from/into
- Devices have addresses, used by
  - Direct I/O instructions
    - Special I/O instructions that specify the transfer of a byte/word to an I/O port address
  - Memory-mapped I/O
    - Device-control registers are mapped into the address space of the processor

## I/O Hardware

- Memory-mapped I/O
  - Device-control registers are mapped into the address space of the processor
  - CPU executes I/O requests using the standard data-transfer instructions to read/write the device-control registers.
  - E.g. Graphics controller has a large memory-mapped region to hold screen contents – sending output to screen by writing data into the memory-mapped region. Controller generates the screen image based on the contents of this memory.
  - Simpler and faster to write millions of bytes to the graphics memory compared to issuing millions of I/O instructions.

7

7

## Categories of I/O Devices

- Human readable
  - Used to communicate with the user
  - Printers
  - Video display terminals
    - Display
    - Keyboard
    - Mouse
- Machine readable
  - Used to communicate with electronic equipment
  - Disk and tape drives
  - Sensors
  - Controllers
  - Actuators

8

8

## Categories of I/O Devices

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

9

9

## Differences in I/O Devices

- Data rate
  - May be differences of several orders of magnitude between the data transfer rates

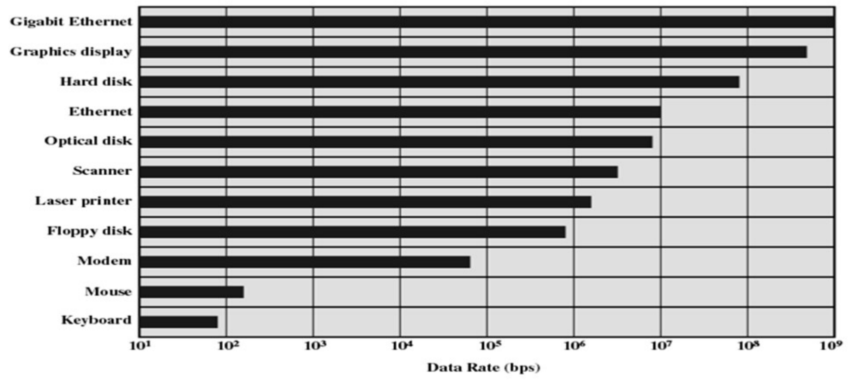


**Figure 11.1 Typical I/O Device Data Rates**

10

10

## Differences in I/O Devices

- 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

11

11

## Differences in I/O Devices

- Complexity of control
  - A printer requires a simpler control interface, a disk much complex.
- Unit of transfer
  - Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk
- Data representation
  - Different data encoding schemes are used by different devices
- Error conditions
  - Devices respond to errors differently, the nature of errors, the way they are reported, their consequences…

12

12

## Performing I/O

- Programmed I/O
  - The simplest form of I/O – the CPU does all the work
  - Processor issues I/O command (on behalf of a process) to an I/O module. Processor has to monitor the status bits and to feed data into a controller register one byte at a time.
  - Process is busy-waiting for the operation to complete
- Interrupt-driven I/O
  - I/O command is issued, there are two possibilities:
    - Nonblocking: Processor continues executing instructions from the current process. I/O module sends an interrupt when done.
    - Blocking: processor executes instruction from the OS, which will put the current process in a blocked state, and schedule another process.
- Direct Memory Access (DMA)
  - DMA module controls exchange of data between main memory and the I/O device
  - Processor is interrupted only after entire block has been transferred

13

13

## Evolution of the I/O Function

- Processor directly controls a peripheral device
  - In simple microprocessor-controlled devices
- Controller or I/O module is added
  - Processor uses programmed I/O without interrupts
  - Processor does not need to handle details of external devices
  - Involves waiting
- Controller or I/O module with interrupts
  - Processor does not spend time waiting for an I/O operation to be performed
  - More efficient, no waiting

14

14

## Evolution of the I/O Function

- Direct Memory Access
  - Blocks of data are moved into memory without involving the processor (I/O <---> MM)
  - Processor involved at beginning and end only
- I/O module is a separate processor
  - With specialized instruction set tailored for I/O
- I/O processor
  - I/O module has its own local memory
  - It is a computer in its own right
  - Large set of I/O devices can be controlled, with minimal CPU involvement
  - Usually used to control comm. with interactive terminals.

15

15

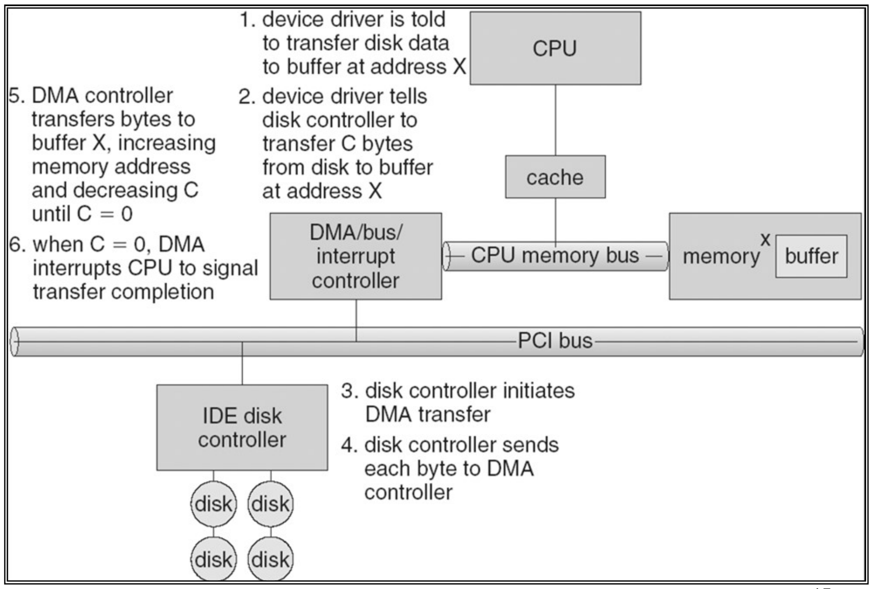
## Direct Memory Access

- Used to avoid **programmed I/O** for large data movement
- Requires **DMA** controller
- Processor delegates I/O operation to the DMA module
- Bypasses CPU to transfer data directly between I/O device and memory -- DMA module transfers data directly to or from memory
- When complete DMA module sends an interrupt signal to the processor

16

16

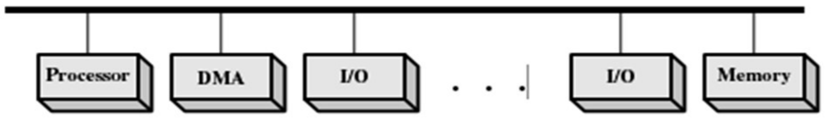
## Six Step Process to Perform DMA Transfer


1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion
CPU
cache
DMA/bus/interrupt controller
CPU memory bus
memory
X
buffer
PCI bus
IDE disk controller
disk
disk
disk
disk

17

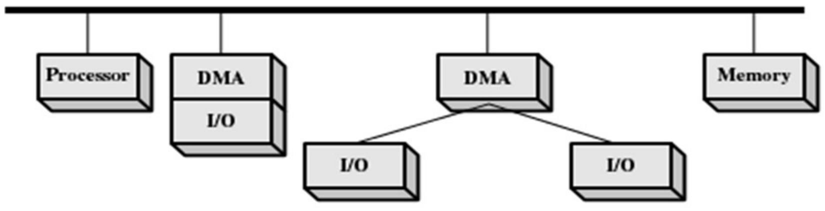
17

## DMA Configurations



(a) Single-bus, detached DMA

**- Share system bus**
**- inefficient**



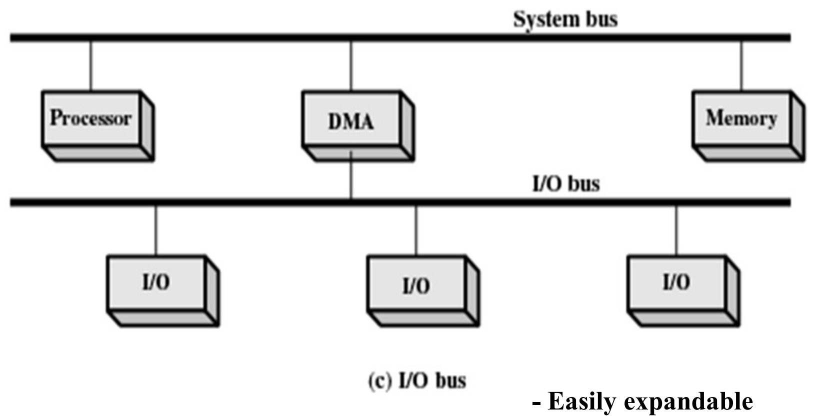
(b) Single-bus, Integrated DMA-I/O

**-DMA logic may be a part of I/O module**
**-Path between DMA module and I/O module does not include system bus**

18

18

## DMA Configurations



- Easily expandable

**Figure 11.3 Alternative DMA Configurations**

19

19

## Operating System Design Issues

- 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 is used to bring in additional Ready processes which is an I/O operation
- Generality
  - Desirable to handle all I/O devices in a uniform manner
  - Hide most of the details of device I/O in lower-level routines so that processes and upper levels see devices in general terms such as read, write, open, close, lock, unlock

20

20

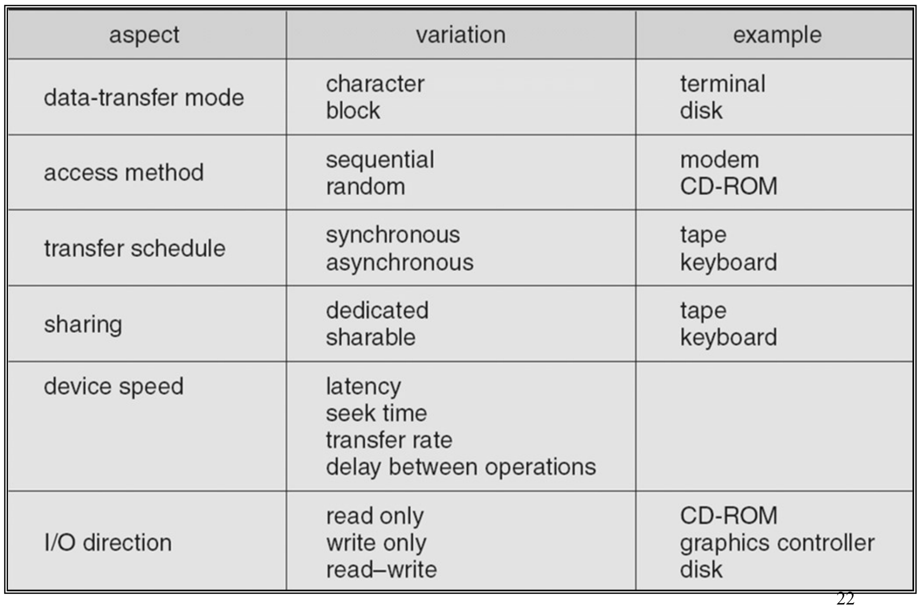
## Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential or random-access**
  - **Sharable or dedicated**
  - **Speed of operation**
  - **read-write, read only,** or **write only**

21

21

## Characteristics of I/O Devices

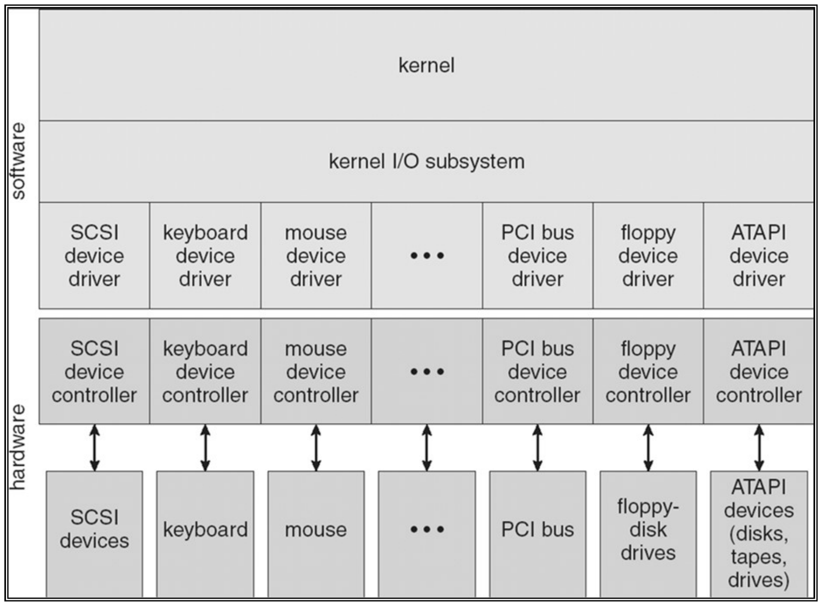


| aspect | variation | example |
|---|---|---|
| data-transfer mode | character<br>block | terminal<br>disk |
| access method | sequential<br>random | modem<br>CD-ROM |
| transfer schedule | synchronous<br>asynchronous | tape<br>keyboard |
| sharing | dedicated<br>sharable | tape<br>keyboard |
| device speed | latency<br>seek time<br>transfer rate<br>delay between operations | |
| I/O direction | read only<br>write only<br>read–write | CD-ROM<br>graphics controller<br>disk |

22

22

## A Kernel I/O Structure



23

## Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible
- Character devices include keyboards, mice, serial ports
  - Commands include `get, put`
  - Libraries layered on top allow line editing

24

24

## Network Devices

- Varying enough from block and character to have own interface
- Unix and Windows NT/9*x*/2000 include socket interface
  - Separates network protocol from network operation
  - Includes `select` functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

25

25

## Clocks and Timers

- Provide current time, elapsed time, timer
- **Programmable interval timer** used for timings, periodic interrupts
- `ioctl` (on UNIX) covers odd aspects of I/O such as clocks and timers

26

26

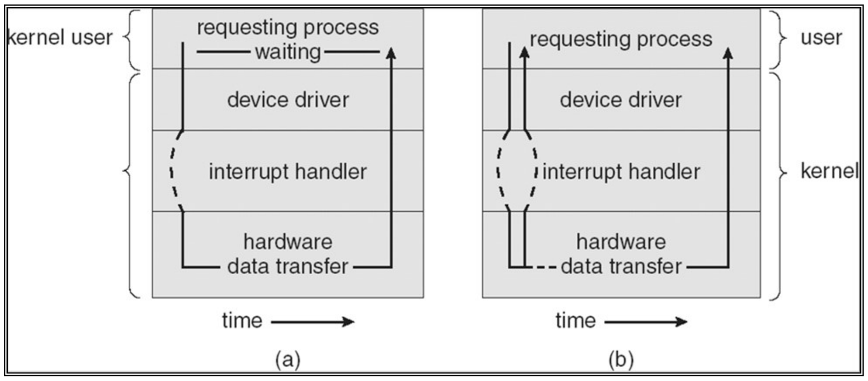
## Blocking and Nonblocking I/O

- **Blocking** - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- **Nonblocking** - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed

27

27

## Two I/O Methods



Synchronous Asynchronous

28

28

## Kernel I/O Subsystem

- **Scheduling**
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness
- **Buffering** - store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
  - To maintain “copy semantics”

29

29

## Kernel I/O Subsystem

- **Caching** - fast memory holding copy of data
  - Always just a copy
  - Key to performance
- **Spooling** - hold output for a device
  - If device can serve only one request at a time
  - i.e., Printing
- **Device reservation** - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

30

30

## Error Handling

- OS can recover from disk read, device unavailable, transient write failures
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

31

31

## I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
  - All I/O instructions defined to be privileged
  - I/O must be performed via system calls
    - Memory-mapped and I/O port memory locations must be protected too

32

32

## I/O Requests to Hardware Operations

- Consider reading a file from disk for a process:
  - Determine device holding file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process

33

33

## 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

1. To cope with speed mismatch between producer and consumer of a data stream
   - E.g: A file is being received via modem to be stored on hard disk
   - Modem is 1000x slower than HD, so a buffer is created in MM to accumulate the bytes received from the modem
   - When the entire buffer of data has arrived, the buffer can be written on disk in a single operation.
   - In the meanwhile, the modem still needs a place to store additional incoming data → two buffers are used (double buffer).
   - By the time the modem has filled the 2nd buffer, the disk write from the 1st buffer should have completed (emptied), so the modem can switched back to it while the disk writes the 2nd one.

34

34

## I/O Buffering

2. To provide adaptations for devices that have different data-transfer sizes
   - Common situation in computer networking
   - At sending site, a large message is fragmented into small network packets to be sent across
   - The receiving site places these fragments in a reassembly buffer to form an image of the source data.
3. To support copy semantics for application I/O
   - E.g: An application wishes to write a buffer to disk, it calls the **`write()`** system call.
   - After the system call returns, what happens if the application changes the contents of the buffer?
   - With copy semantics, the data written to disk is guaranteed to be the version at the time of the system call.
   - The **`write()`** system call copies the application data into a kernel buffer before returning control to the application.

35

35

## I/O Buffering

- Block-oriented
  - Used for disks and tapes
  - Information is stored in fixed sized blocks
  - Input transfers are made a block at a time
  - When transfer is complete, the block is moved to user space when needed
  - Another block is moved into the buffer – a.k.a Read ahead
  - User process can process one block of data while next block is read in
  - Swapping can occur since input is taking place in system memory, not user memory
  - Operating system keeps track of assignment of system buffers to user processes

36

36

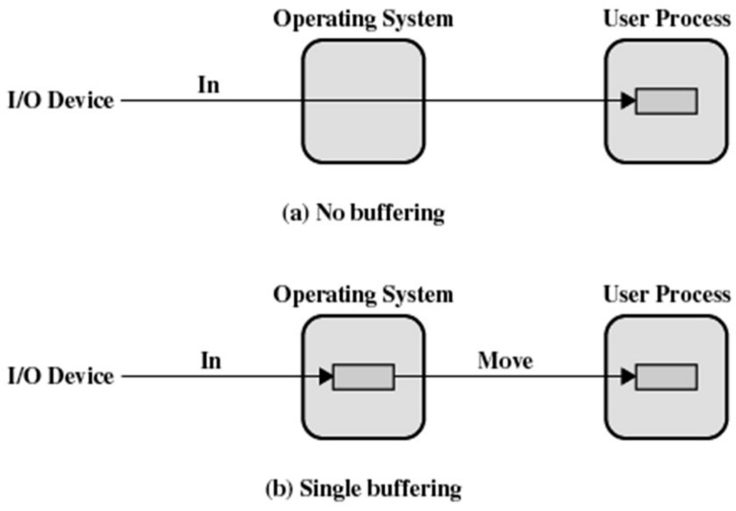
## I/O Buffering

- Stream-oriented
  - Used for terminals, printers, communication ports, mouse and other pointing devices, and most other devices that are not secondary storage
  - Transfer information as a stream of bytes
  - Line-at-a-time or byte-at-a-time
  - Line-at-a-time
    - For scroll-mode terminals, (a.k.a dumb terminals); line printer
    - Buffer can hold a single line
    - Process is suspended during input, waiting for the arrival of the entire line.
  - Byte-at-a-time
    - Appropriate for forms-mode terminals (when each keystroke is significant), and for other peripherals, such as sensors and controllers

37

37

## Single Buffer

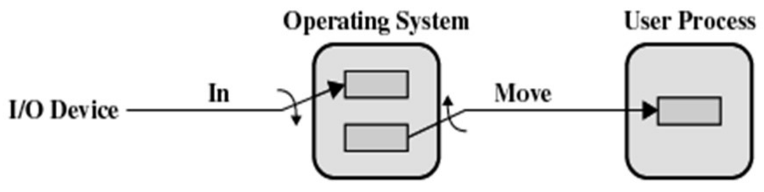


38

38

## 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



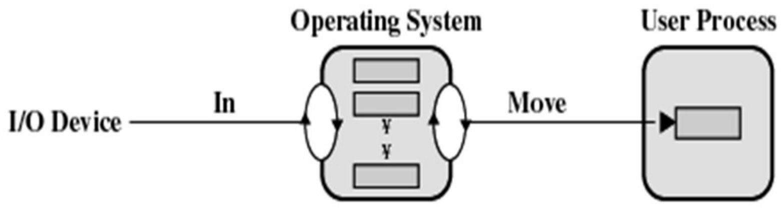
(c) Double buffering

39

39

## 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

40

40

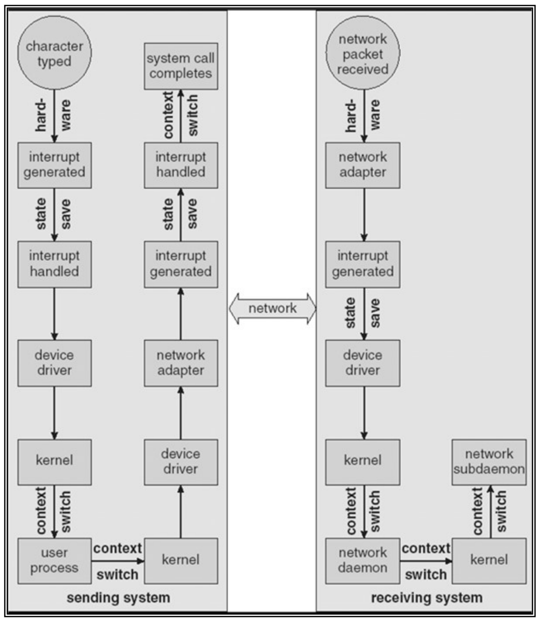
## Performance

- I/O a major factor in system performance:
  - It demands CPU to execute device driver, kernel I/O code and to schedule processes fairly and efficiently
  - The resulting context switches due to interrupts stress the CPU
  - Data copying loads the memory bus during copying between controllers and MM, and again during data copies between kernel buffers and application data space.
  - Network traffic especially stressful, can also cause high context-switch rate. E.g. remote login – a character typed on local machine must be transported to remote machine (see Figure on next slide).

41

41

## Intercomputer Communications – remote login



42

42

## Improving Performance

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers, polling
- Use DMA
- Balance CPU, memory, bus, and I/O performance for highest throughput

43

43

## Epilogue:
## A+ I/O
## A+ SCSI

44

44