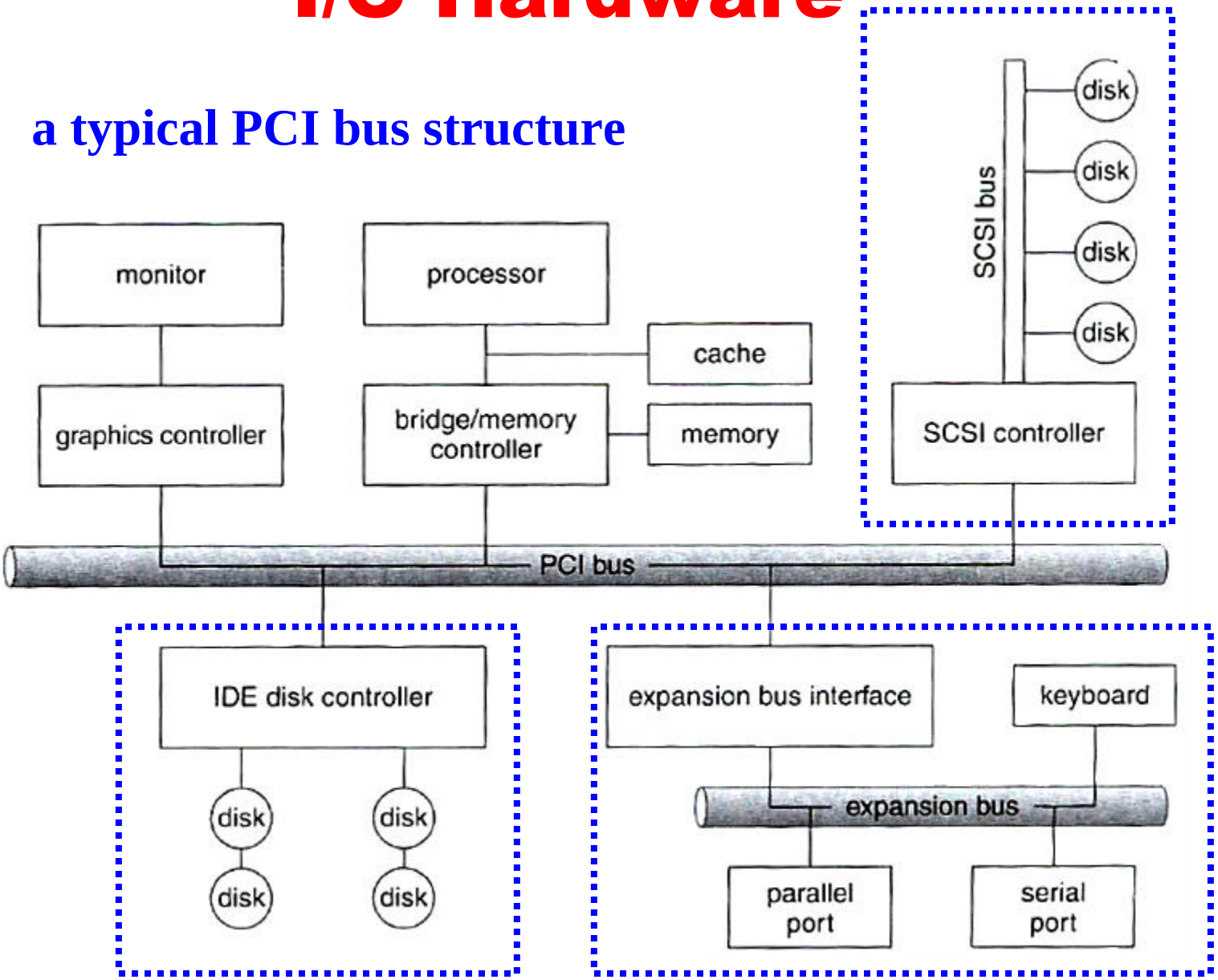
Part IV I/O Systems

Chapter 13: I/O Systems

I/O Hardware

a typical PCI bus structure



How do the processor and controller communicate?

- **Use the controller: a controller usually has a few registers (*e.g.*, status, control, data-in and data-out).**
- **Use memory-mapped I/O.**
- **Or, a combination of both.**

Memory-Mapped I/O

I/O address	Device
000-00F	DMA controller
020-021	Interrupt controller
040-043	timer
200-20F	Game controller
2F8-2FF	Serial port (secondary)
320-32F	Hard-disk controller
378-37F	Parallel port
3D0-3DF	Graphics controller
3F0-3F7	Floppy-disk controller
3F8-3FF	Serial port (primary)

- Each controller has a few registers that are used for communicating with the CPU.
- If these registers are part of the regular memory address space, it is called **memory-mapped I/O**.

Three Commonly Seen Protocols

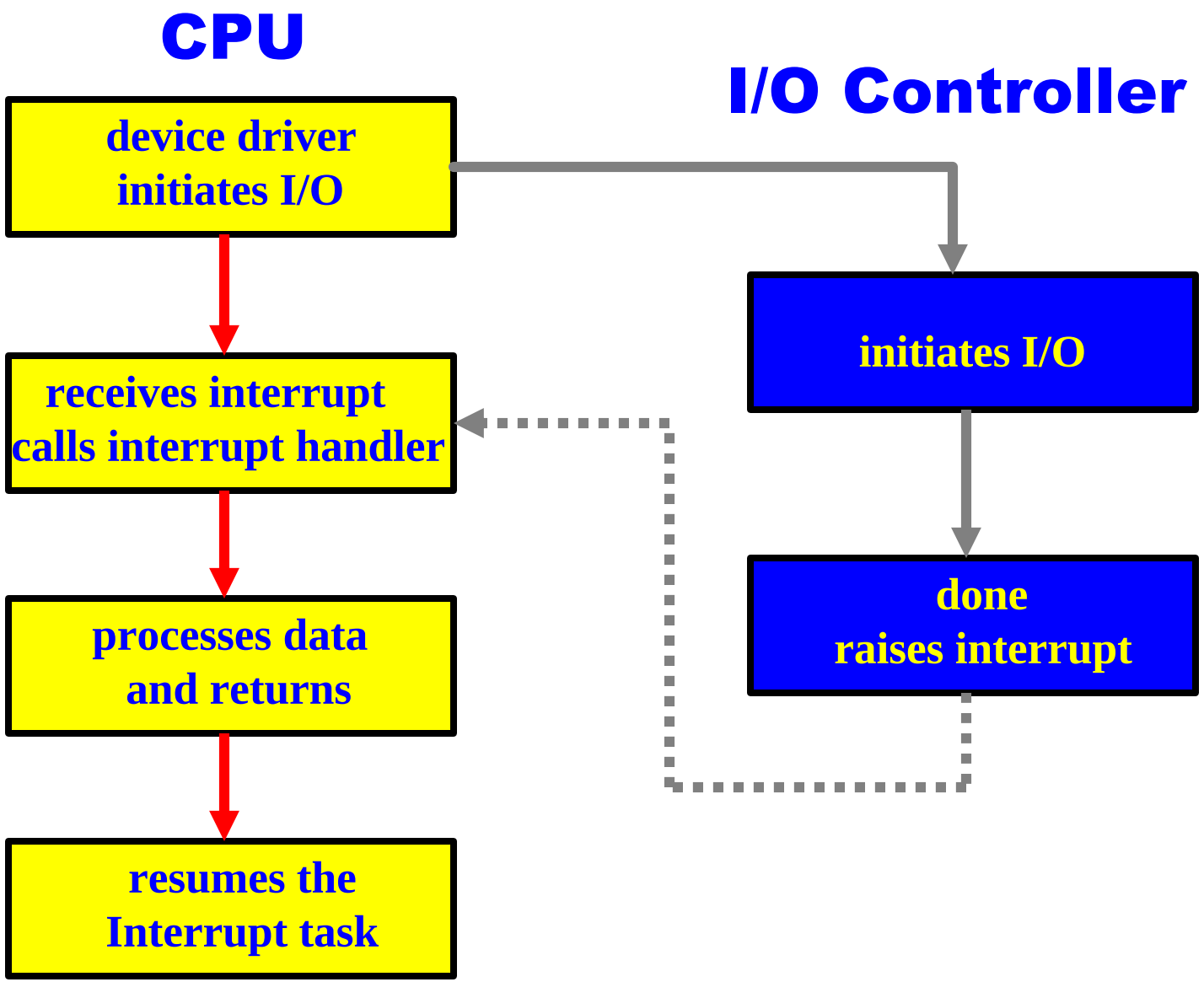
- Pooling
- Interrupts
- Direct Memory Access (DMA)

Polling

- The status register has two bits, *busy* and *command-ready*.

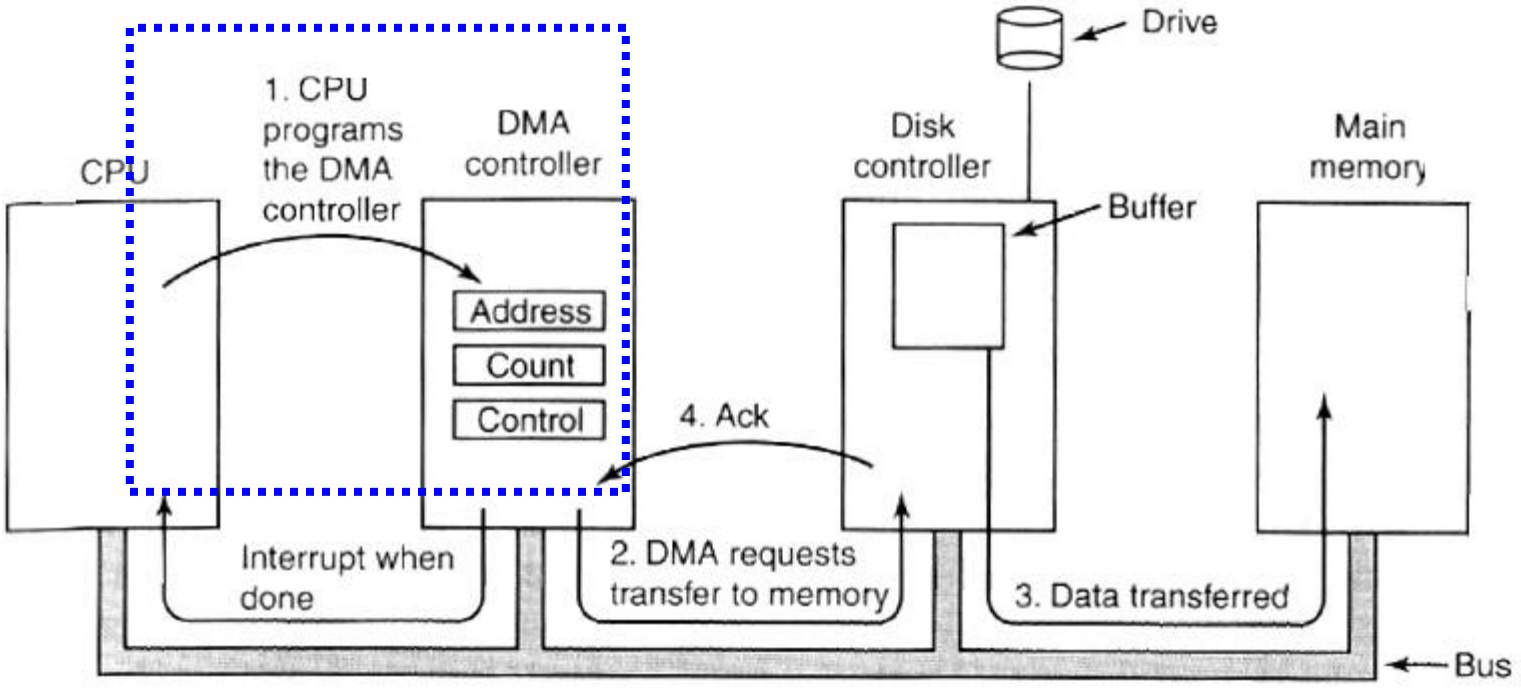
Processor	Controller
wait until the busy bit is not set	
set the write bit in command	
set command-ready bit	
	if command-ready is set, set busy
	do input/output transfer
	clear the command-ready and busy

Interrupt


CPU
I/O Controller
device driver initiates I/O
receives interrupt calls interrupt handler
processes data and returns
resumes the Interrupt task
initiates I/O
done raises interrupt

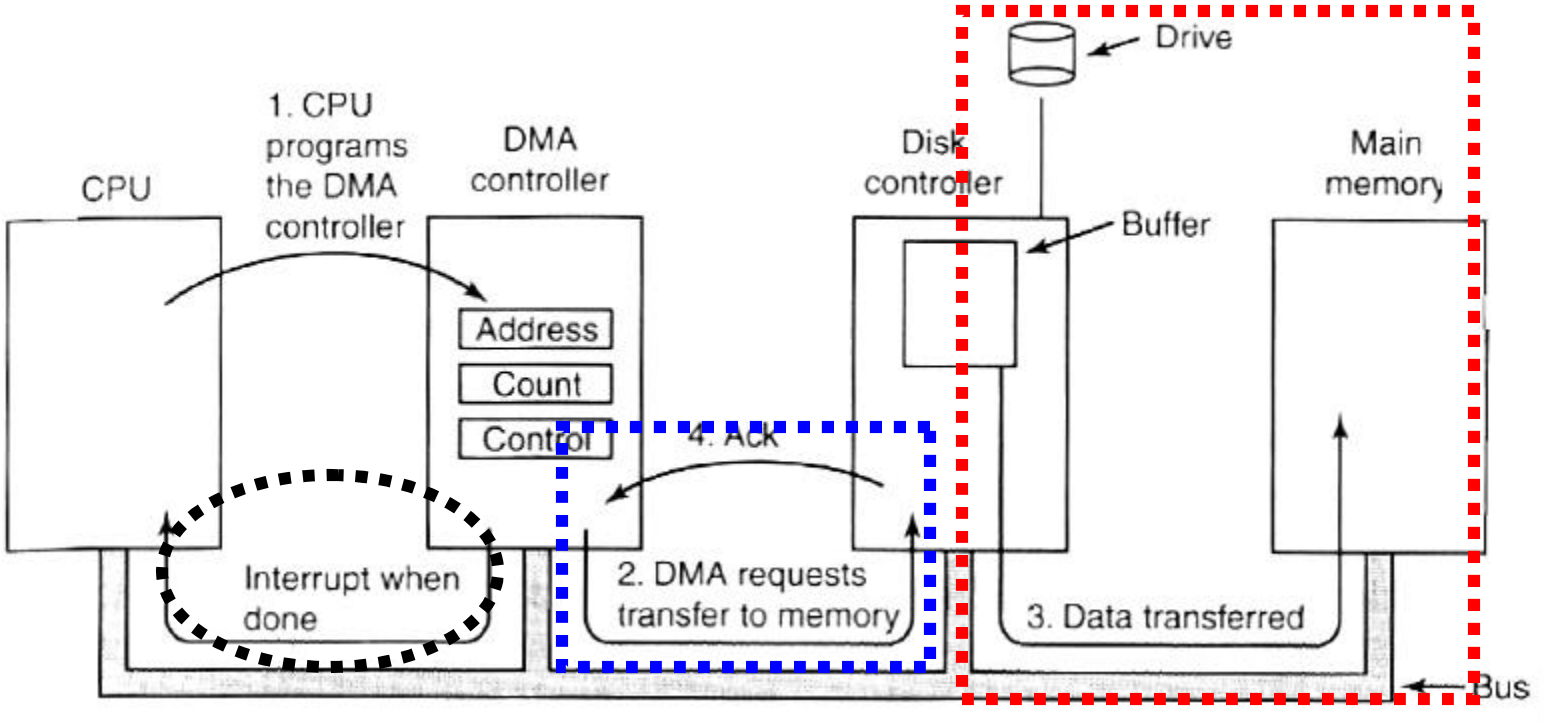
Direct Memory Access: 1/2

- For large volume data transfer, most systems use direct memory access to avoid burdening the CPU.
- The CPU gives the controller (1) disk address, (2) memory address for storing the block, and (3) a byte count. Then, the CPU goes back to work.

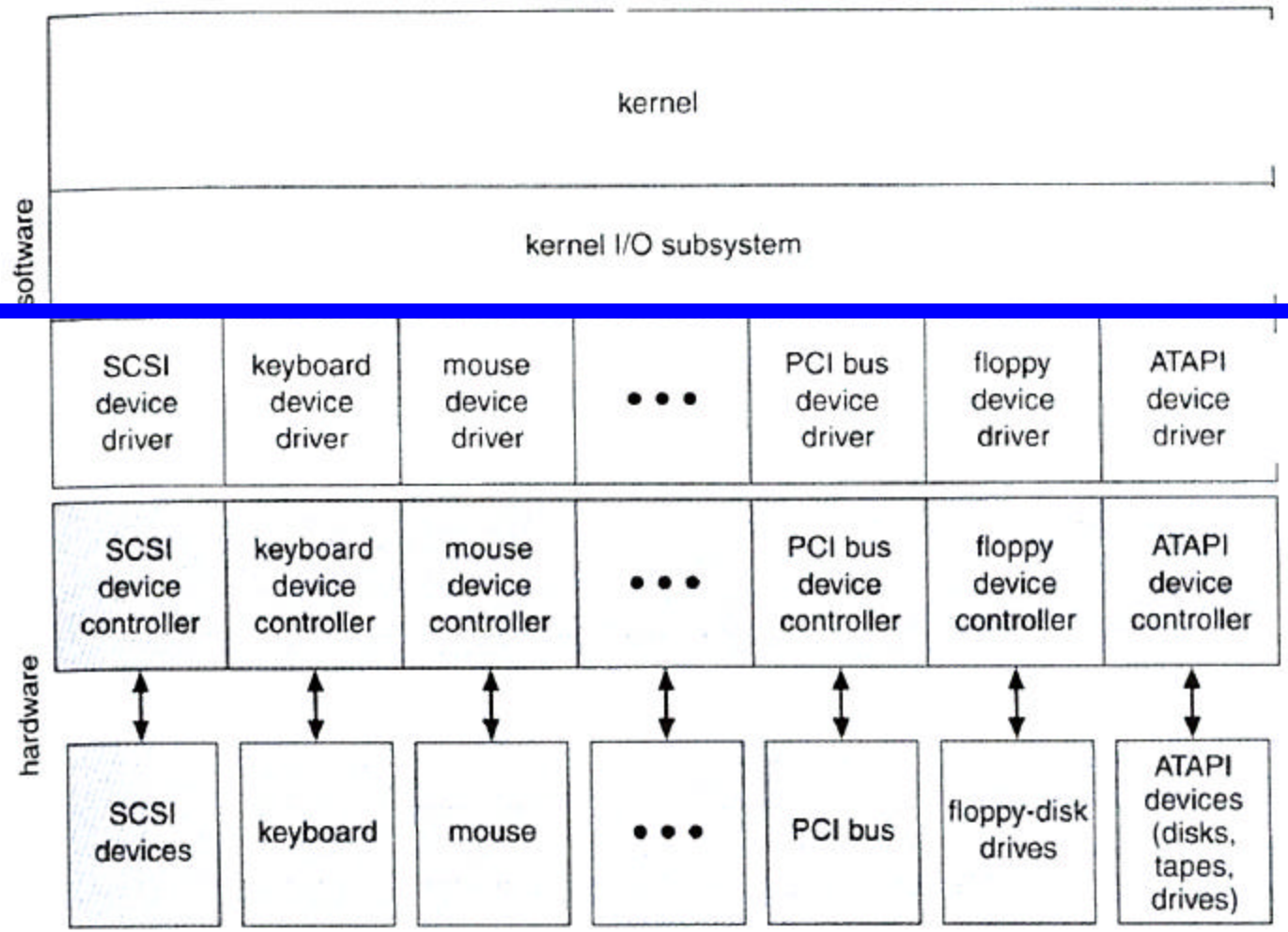


Direct Memory Access: 2/2

- DMA requests data transfer to memory
- The disk controller copies the information into the address provided by the CPU, byte-by-byte, until the counter becomes 0, at which time an interrupt is generated.



Application I/O Interface



I/O Devices

- **Character stream: a character stream device transfers byte one by one (*e.g.*, modem)**
- **Block: a block device transfers a block of bytes as a unit (*e.g.*, disk)**
- **Others: clocks, memory-mapped screens and so on.**
- **Not all devices may be recognized by an OS. Thus, device drivers are needed.**

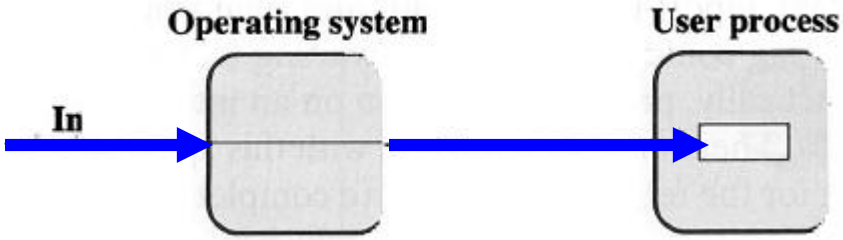
Kernel I/O System

- Build on top of hardware and device drivers, the kernel usually provide many I/O services:
  - I/O scheduling (*e.g.*, disk head scheduling)
  - I/O Buffering (see below)
  - Caching (see below)
  - Spooling
  - Error handling

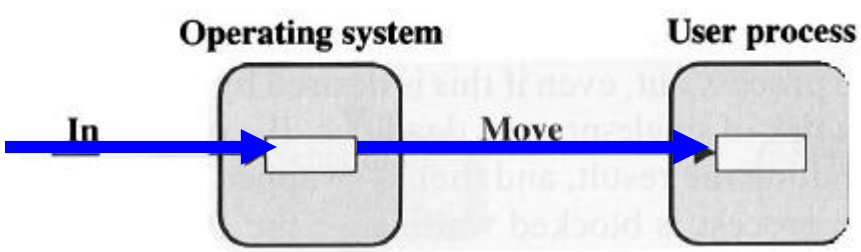
Buffering: 1/2

- A buffer is a memory area that stores data while they are transferred between two devices or between a device and an application.
- Major reasons of using buffers
  - Efficiency (see below)
  - Copy semantics. What if there is no buffer and a process runs so fast that overwrites its previous write? The content on the disk becomes incorrect. The use of buffers overcomes this problem.

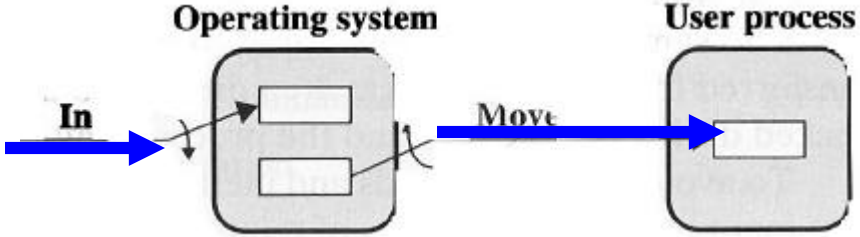
Buffering: 2/2



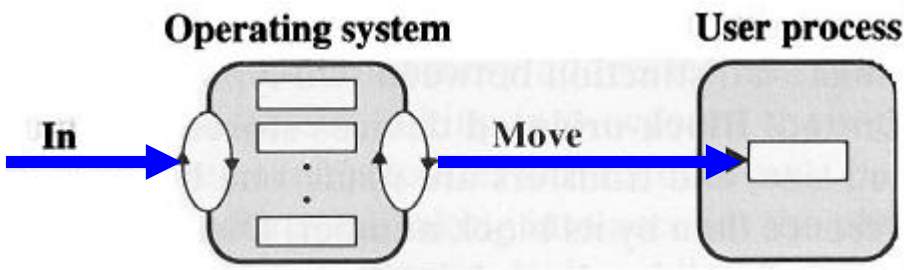
No buffer. The user process must wait until data transfer completes.



One buffer: While the user process is running, next data transfer may begin



Double buffer: while the user process is processing the first buffer, data transfer can be performed on the second.



Multiple buffers: very efficient

(figures taken from W. Stallings' OS text)

Caching

- Just like a cache memory between the faster CPU and slower physical memory, a cache (*i.e.*, disk cache) may be used between the faster physical memory and slower I/O devices.
- Note that buffering and caching are different things.