Nanyang Polytechnic

IT1313

Operating Systems and Administration

Chapter 4

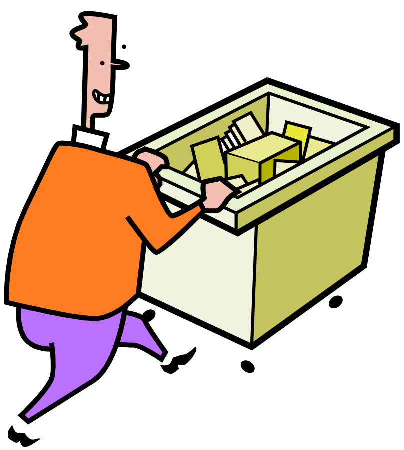
Device Management

Device Management

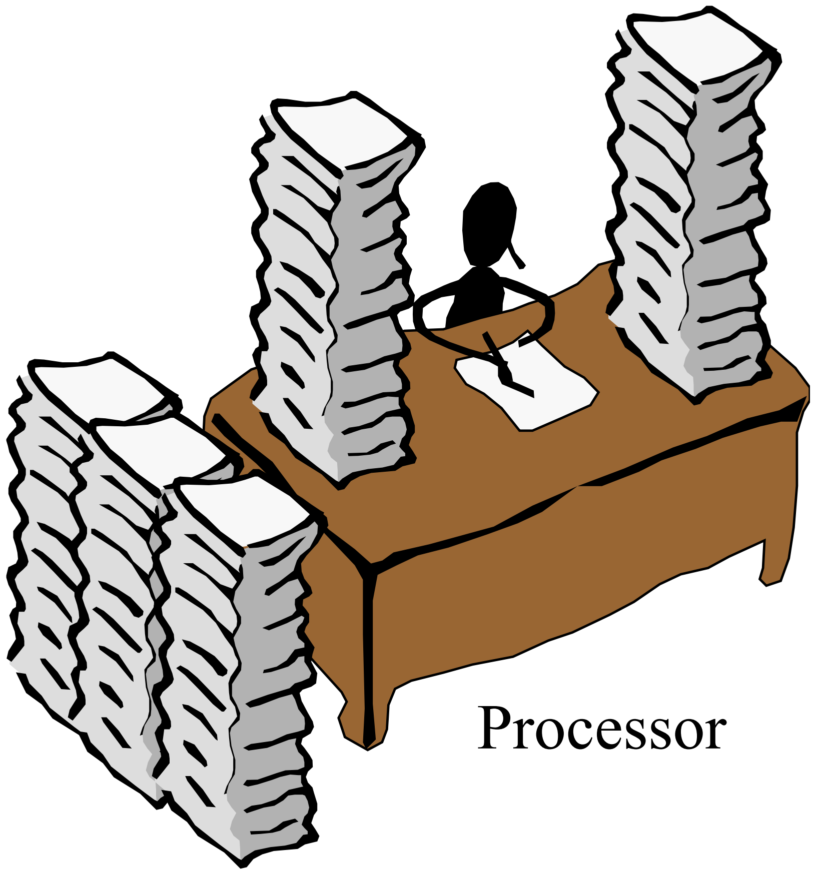
At the end of this chapter, you should be able to

- Understand device management abstraction and organization
- Understand the various I/O strategies and buffering techniques
- Understand the various device class characteristics
- To understand some of the rotating disk optimization techniques

Input/Output Devices



Input Device



Processor

Output Device

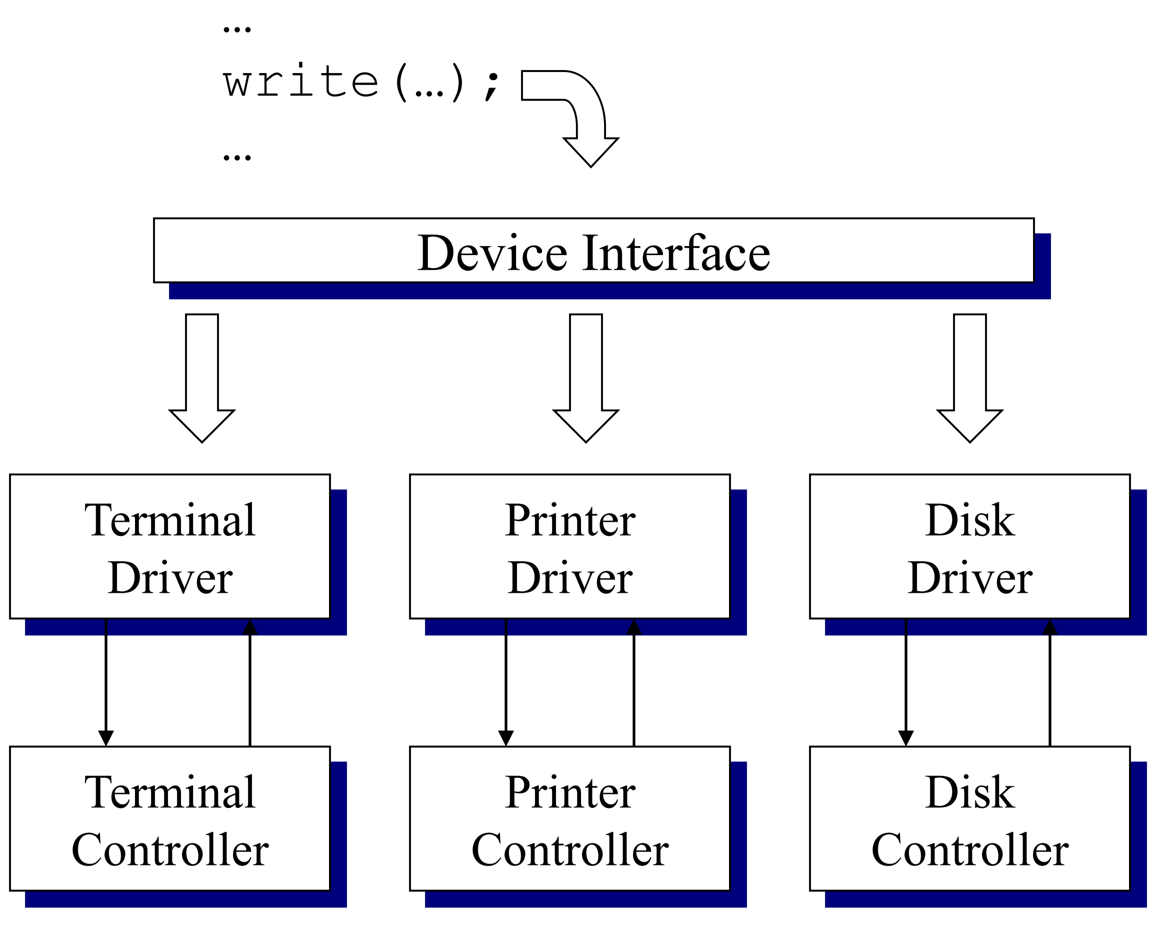
I/O System

- In the early days of computing, input and output is at a speed similar to that of processing.
- When processors became electronic, I/O devices still remain mechanical. (e.g., hard disk vs CPU)
- Today, I/O systems are designed to handle I/O devices which are of many magnitudes slower than the slowest CPU.
  - To provide simple, abstract software interfaces to manage the I/O operations needed
  - Ensure that there is as much overlap as possible between the operation of the I/O devices and the CPU

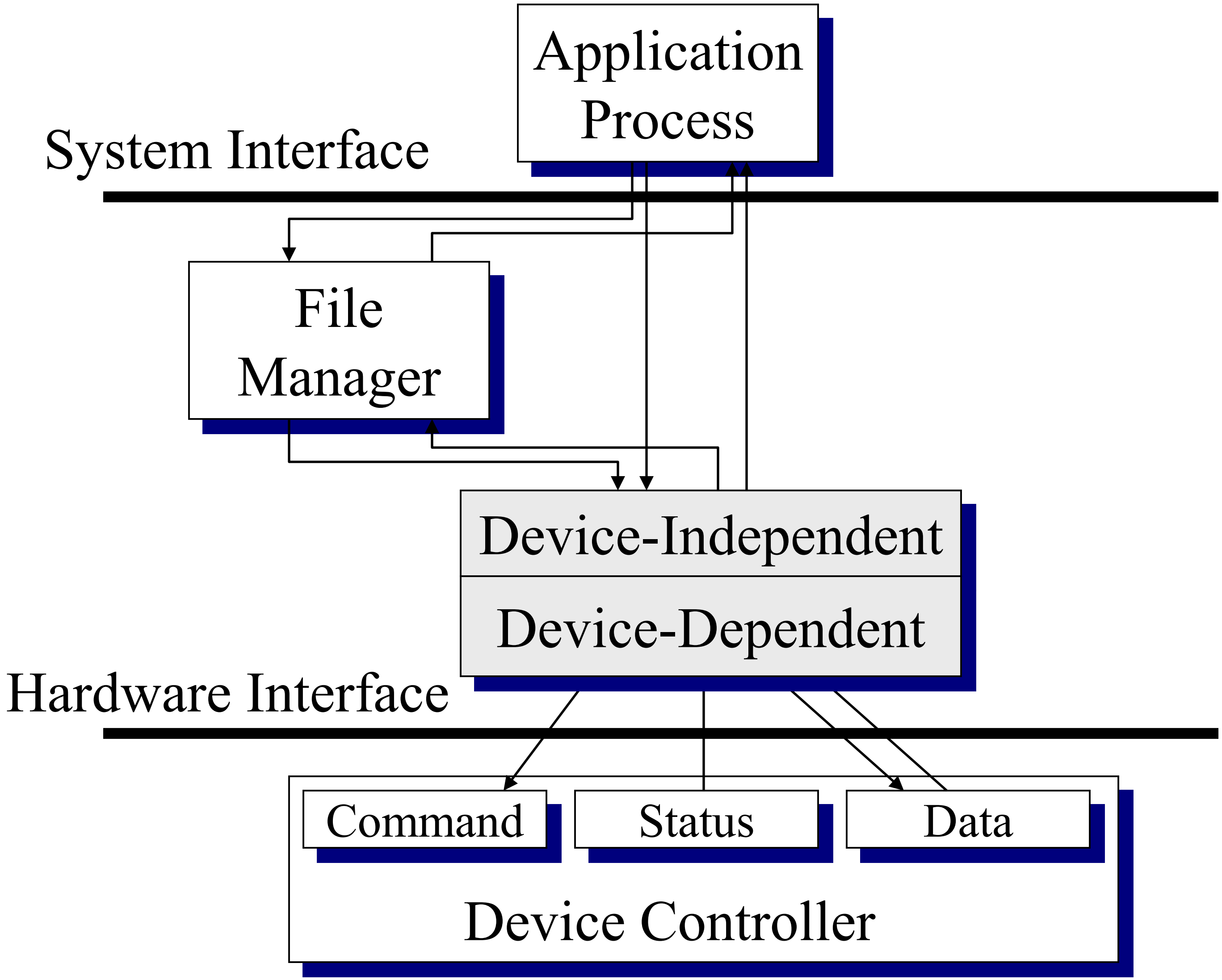
Device Manager Abstraction

- Different devices require different operations in order to work them.
- It is important to create device drivers, which takes on the task of working these devices, but requiring only a standard set of functions to make them (the devices) work.
- The device manager, in turn, manages the collection of device drivers.
- The manager makes it possible for the OS to then provide a standard set of system calls to application programs, which use the devices.

The Device Driver Interface



Device Management Organization



Nanyang Polytechnic

System Call Interface

- Functions available to application programs
- Abstract all devices (and files) to a few interfaces
- Make interfaces as similar as possible
  - Block vs character
  - Sequential vs direct access
- Device driver implements functions (one entry point per API function)

Example: BSD UNIX Driver

`open`	Prepare dev for operation
`close`	No longer using the device
`ioctl`	Character dev specific info
`read`	Character dev input op
`write`	Character dev output op
`strategy`	Block dev input/output ops
`select`	Character dev check for data
`stop`	Discontinue a stream output op

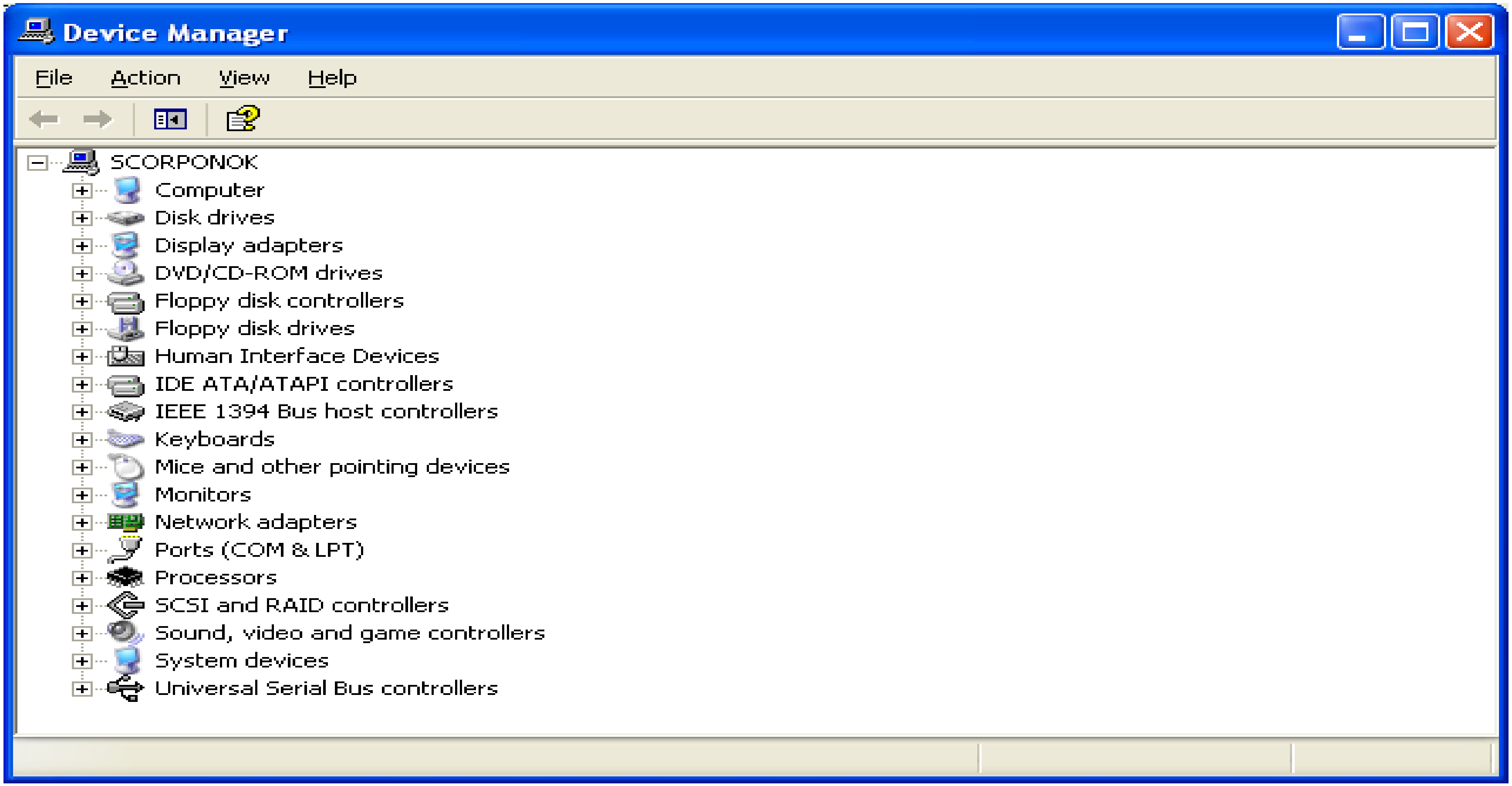
Example: Windows system calls and low-level I/O

_close	Close file
_commit	Flush file to disk
_creat, _wcreat	Create file
_dup	Return next available file descriptor for given file
_dup2	Create second descriptor for given file
_eof	Test for end of file
_lseek, _lseeki64	Reposition file pointer to given location
_open, _wopen	Open file
_read	Read data from file
_sopen, _wsopen, _sopen_s, _wsopen_s	Open file for file sharing
_tell, _telli64	Get current file-pointer position
_umask, _umask_s	Set file-permission mask

Device Status Table

- Device Manager keeps track of the status of the various devices using a **Device Status Table**, which consists of these info:
  - **Device ID**
  - **Device Status (busy, done, idle)**
  - **Queue of processes waiting for the Device**

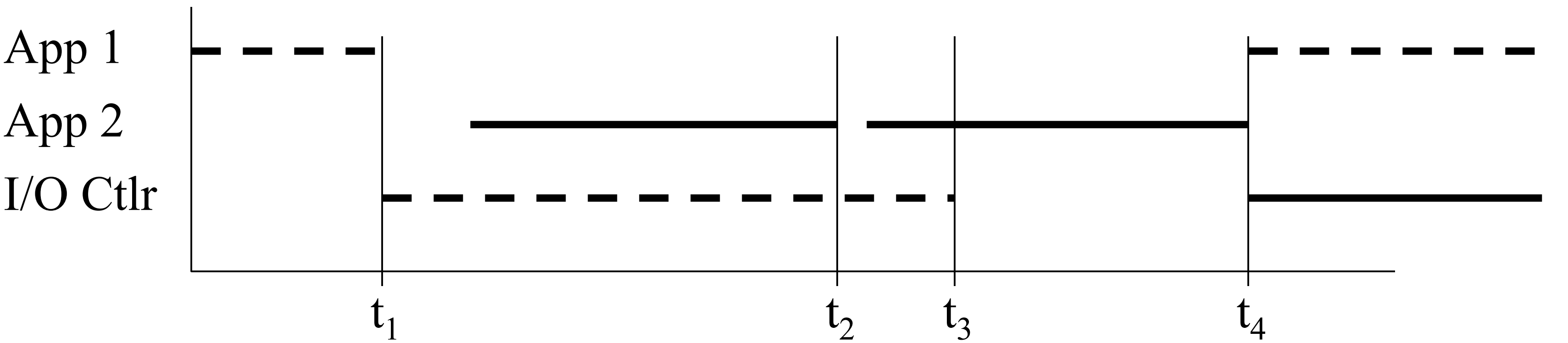
Device Manager in Windows



Overlapping the Operation of a Device and the CPU

- Programs which perform IO expects IO operations to complete before the next statement is executed.
- However, performance gains can be gained if we can make the program execute instructions while the IO is taking place.
- This should be done without violating the serial execution order of the program.

Overlapping Processing and I/O



- To maximize IO, we can have the CPU operate on another process when the IO is busy with the original process.
- This increases the efficiency of the computer and reduces the overall time required to execute all the processes.

Buffering

- The speed of I/O is much slower than that of CPU.
- In order to speed up I/O, it is possible to keep I/O devices busy when the processes do not require I/O operations.
- This increases the overlap between I/O and processing.
- A buffer is a temporary memory-based storage area, that stores the data from an I/O operation.

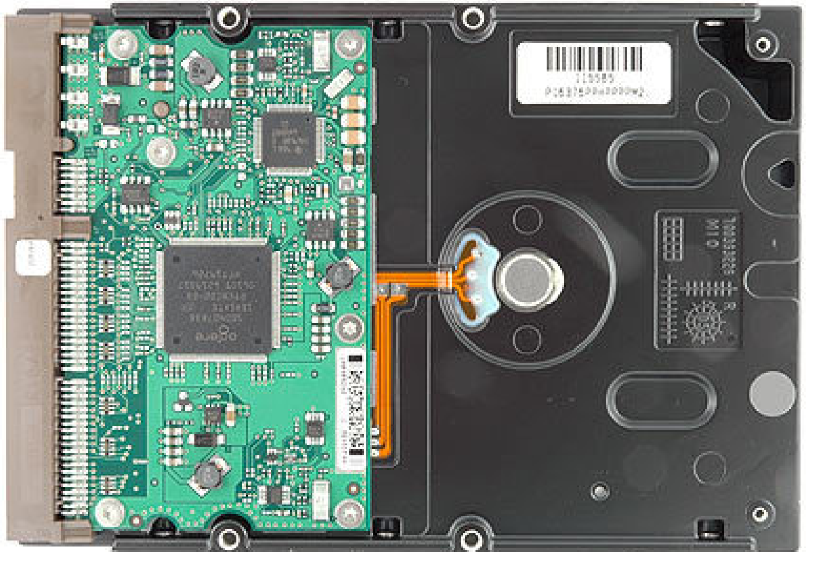
Types of Buffering

- Input buffering
  - Copy the data into memory before the process requests it.
- Output buffering
  - Temporarily stores the data in memory and have it written out to the device when the process resumes execution.
- Hardware buffering
  - Implement buffering in the hardware by having specialized registers to act as buffers. Also known as cache.
- Double buffering
  - Implement hardware buffering and having a separate buffer implemented at the software level, i.e., using the primary memory as a buffer.
- Circular buffering
  - Having a buffer which contains n number of locations. The increased number of locations allow for more data to be stored in the buffer.
  - Circular buffering is the technique of managing these locations so that they can be re-used.

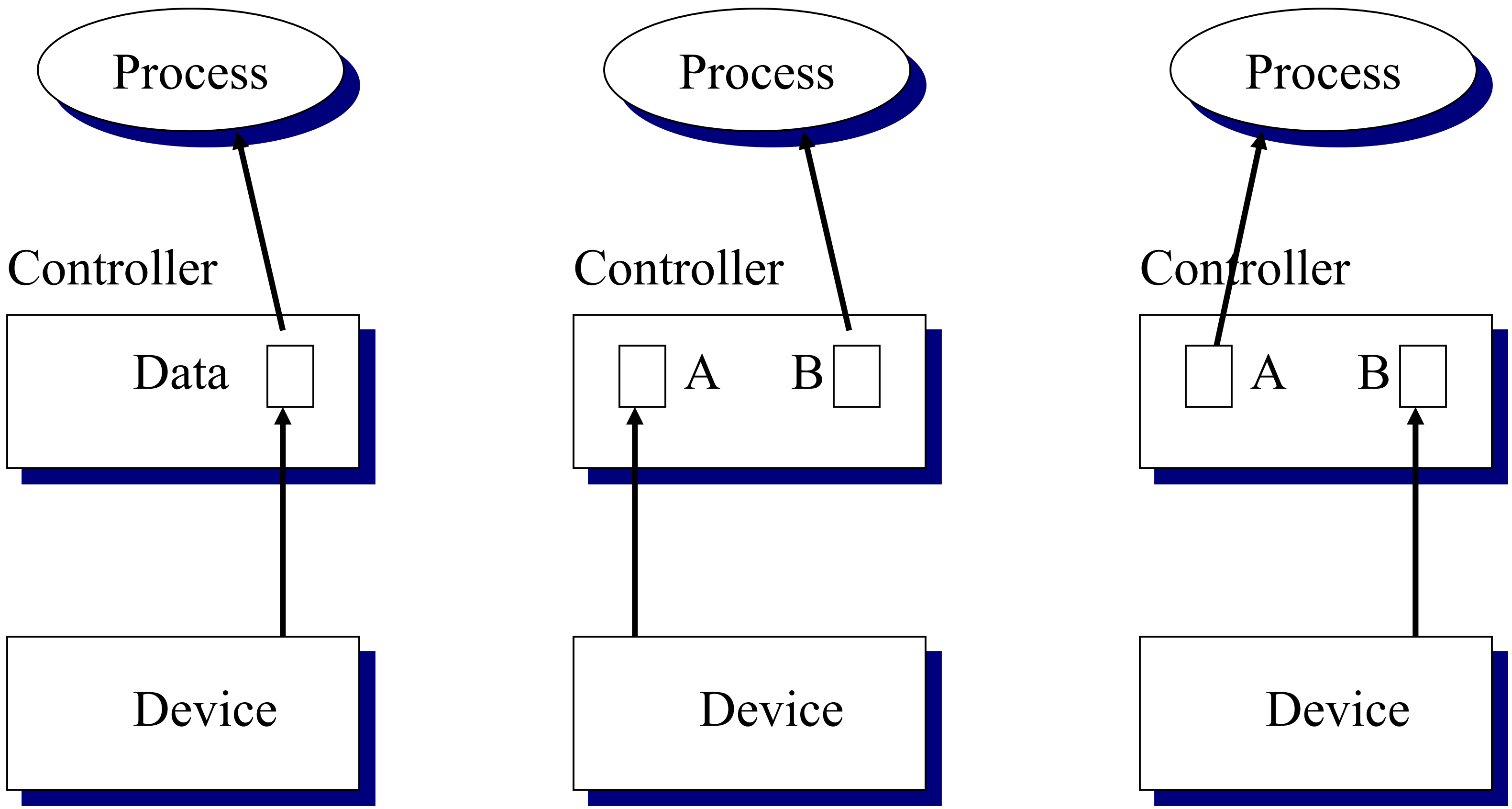
Hardware buffer - cache

Seagate Barracuda 7200.10 500 GB Comparison Table

Manufacturer	Seagate	Seagate	Seagate	Seagate
Product	Barracuda 7200.10	Barracuda 7200.10	Barracuda 7200.10	Barracuda 7200.10
Model	ST3500630A	ST3500630AS	ST3500830A	ST3500830AS
Capacity	500 GB	500 GB	500 GB	500 GB
Spindle Speed	7,200 RPM	7,200 RPM	7,200 RPM	7,200 RPM
Platter	3	3	3	3
Cache	16 MB	16 MB	8 MB	8 MB
Native Command Queuing (NCQ)	No	yes	no	yes
Interface	UltraATA/100	SATA/300	UltraATA/100	SATA/300



Hardware Buffering

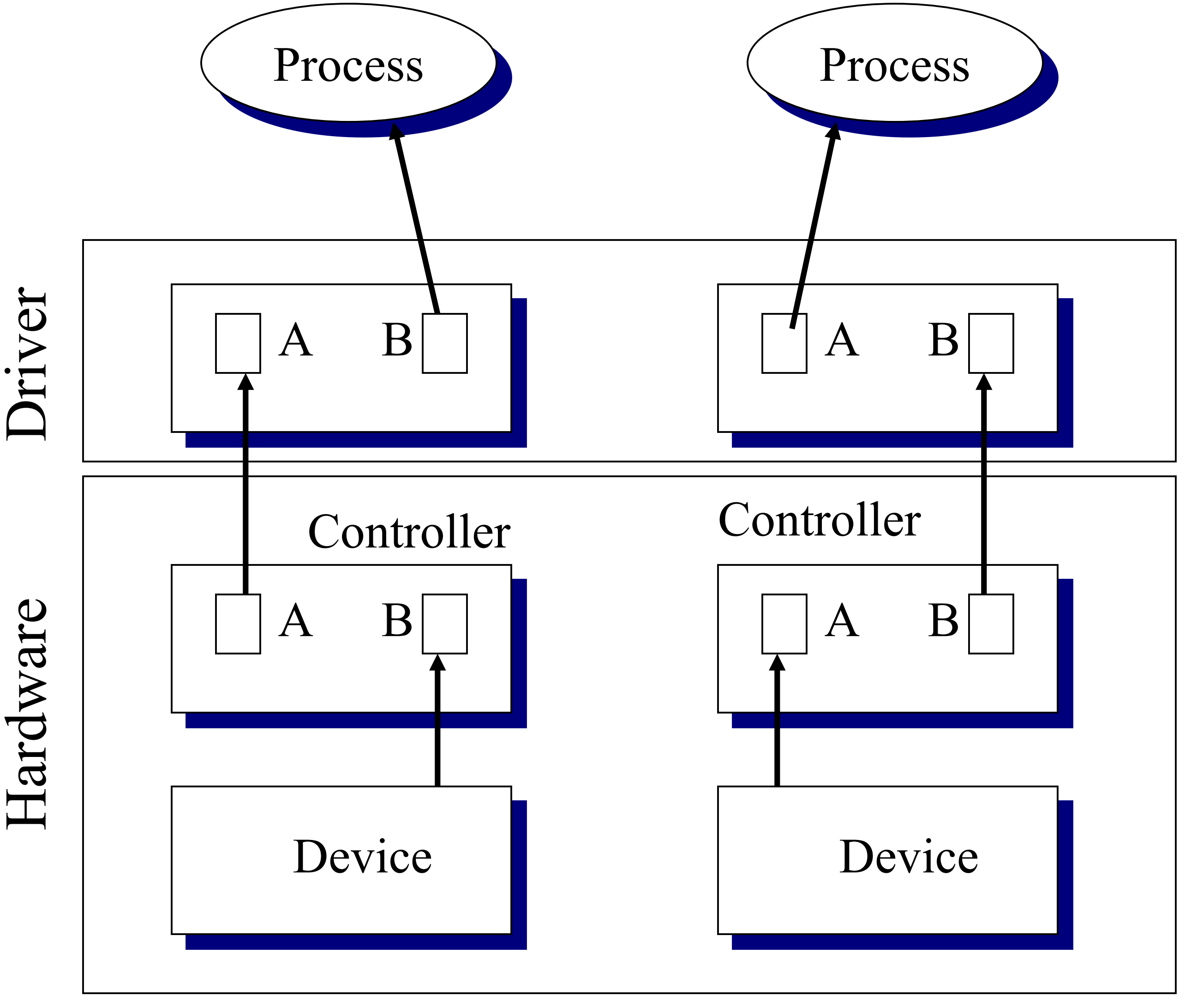


Unbuffered

Process reads b_{i-1}
Controller reads b_i

Process reads b_i
Controller reads b_{i+1}

Double Buffering in the Driver



Device Class Characteristics

- A typical computer system will handle many types of devices, such as
  - Character devices (keyboard, monitor)
  - Block devices (printer, USB)
  - Network devices (network, modems)

- Character based devices such as tty (teletype, or terminal) and serial devices are where data stream is transferred and handled one character or byte at a time.
- Block type devices such as hard drives transfer data in blocks, typically a multiple of 256 bytes.

Linux devices

```
brw-rw----   1 root disk       8,   0 Nov  7 07:06 sda
brw-rw----   1 root disk       8,   1 Nov  7 07:06 sda1
brw-rw----   1 root disk       8,  16 Nov  7 07:06 sdb
brw-rw----   1 root disk       8,  17 Nov  7 07:06 sdb1
brw-rw----   1 root disk       8,  18 Nov  7 07:06 sdb2
crw--w----   1 root tty        4,   0 Nov  7 07:06 tty0
crw--w----   1 root tty        4,   1 Nov  7 07:07 tty1
crw--w----   1 root tty        4,  10 Nov  7 07:06 tty10
crw--w----   1 root tty        4,  11 Nov  7 07:06 tty11
```

- Notice the leftmost character of each line in the output. The ones that have a "b" are block type devices and the ones that begin with "c" are character devices.
- sda, sda1 are naming conventions for disk devices
- tty0, tty1 are naming conventions for terminal devices or virtual consoles

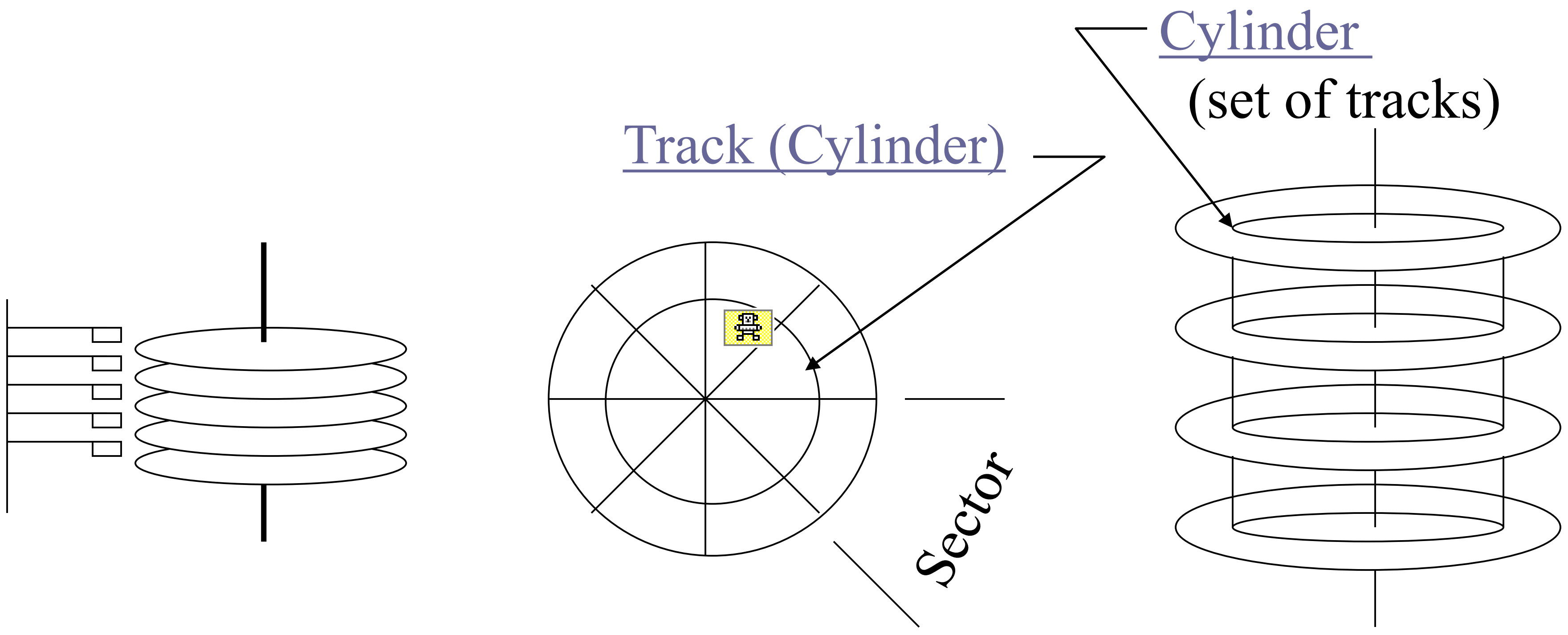
Storage Devices

- Randomly accessed storage
  - Flash memory
  - Optical disks
  - Rotating magnetic disk
  - Floppy disk
- Sequentially accessed storage
  - Magnetic tapes

Randomly Accessed Storage Devices

- Allow a driver to access blocks of data in the device in any order. Need not be sequential.
- Non-volatile memory (such as USB memory devices) also fall in this category.
- Rotating disks are still common used today.
- Both can be easily managed using a common API.

Rotating Disk storage



(a) Multi-surface Disk (b) Disk Surface (b) Cylinders

Rotating Disk Optimizations

- Disk-based devices form the most common randomly accessed storage devices.
- E.g., Hard disks, CD/DVD, Zip disks, MO disks.
- Although they are random, <u>we can improve on their efficiency using disk optimization techniques.</u>
- Their effectiveness is increased in a multi-programming environment, where data blocks can be read from a large range of locations in the disk.

Disk Optimizations

- Transfer Time: Time to copy bits from disk surface to memory
- Disk latency time: Rotational delay waiting for proper sector to rotate under R/W head
- Disk seek time: Delay while R/W head moves to the destination track/cylinder
- Access Time = seek + latency + transfer

Disk Optimizations

- Disk optimization techniques for disk based devices
  - First-Come-First-Serve (FCFS)
  - Shortest Seek Time First (SSTF)
  - SCAN / C-SCAN
  - LOOK / C-LOOK

Conclusion

- OS has to handle many types of devices.
- Handling alone is not enough.
- We need to find ways to optimize the use of these devices.
- Method of optimization depends on device characteristics.
- Common techniques include buffering and disk optimization (for disk-based devices).