PRINCIPLES OF OPERATING SYSTEMS

LECTURE 33

APPLICATION I/O INTERFACE

Application I/O Interface

- **The OS software interface to the I/O devices (an API to the programmer)**
- **Attempts to abstract the characteristics of the many I/o devices into a few general classes.**
- 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
    - **units for data transfer bytes vs blocks**
  - Sequential or random-access **- access methods**
  - **Synchronous (predictable response times) vs asynchronous (unpredictable response times)**
  - Sharable or dedicated **- implications on deadlock**
  - Speed of operation **- device/software issue**
  - read-write, read only, or write only **- permissions**

A Kernel I/O Structure

System calls ==>
… "user" API
==>

Example: ioctl(…)
generic call
(roll your own)
in UNIX (p. 468),
and other more
specific
commands or calls
open, read, ...

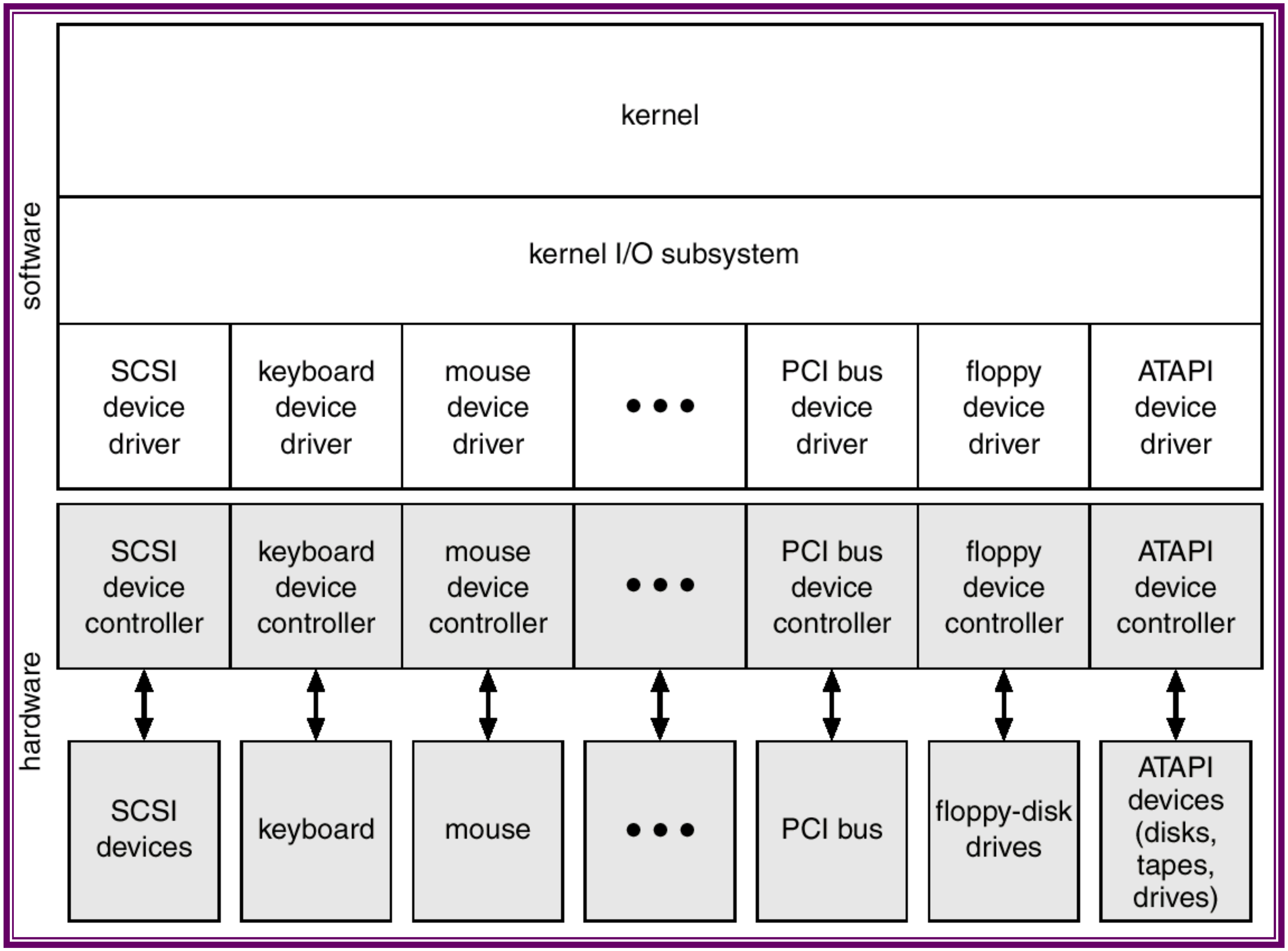
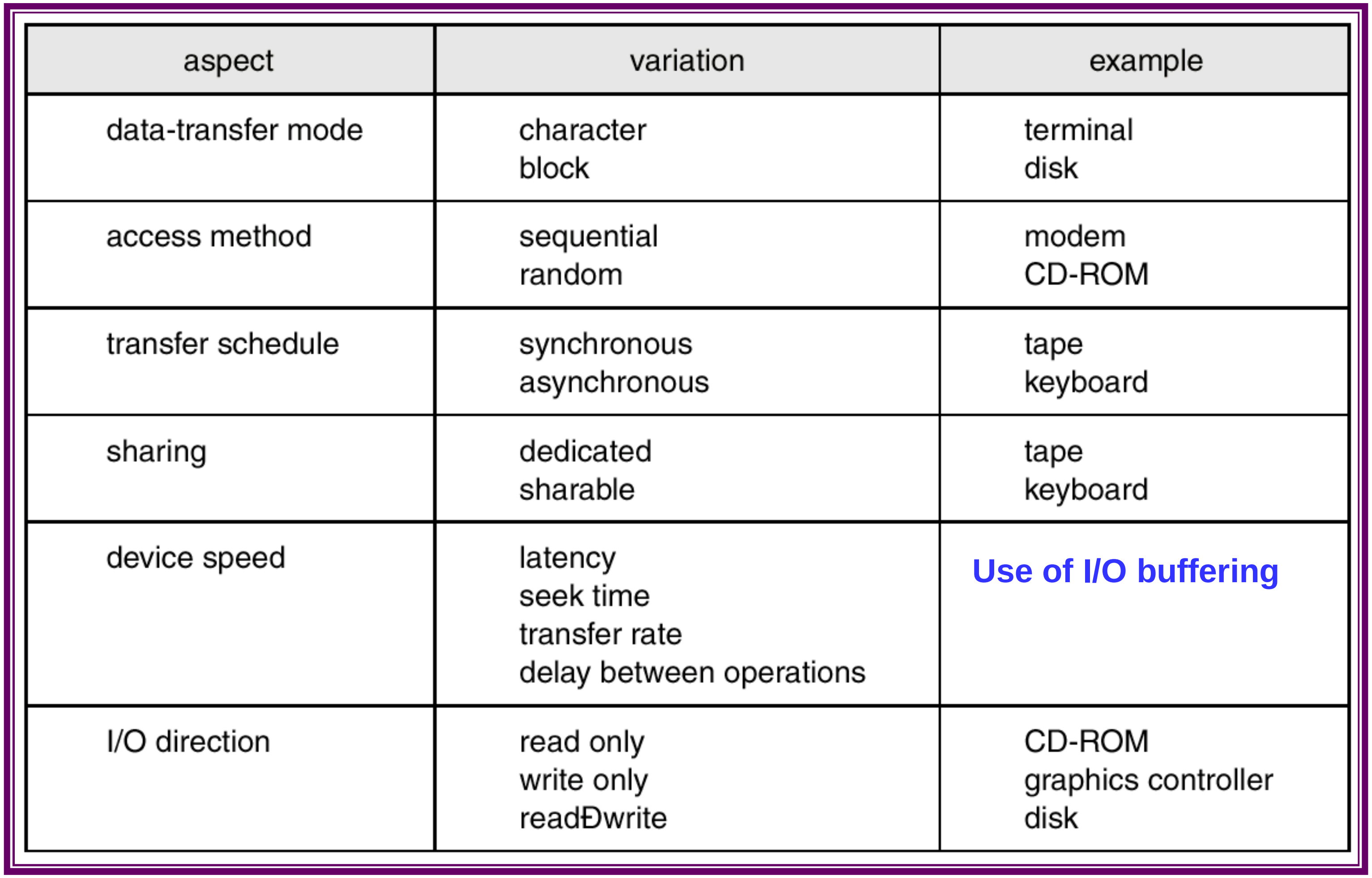


Fig. 13.6

Characteristics of I/O Devices

Device driver must deal with these at a low level



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

Block and Character Devices

- Block devices include disk drives
  - **example sectors or sector clusters on a disk**
  - Commands**/calls** include **read, write, seek**
  - **Access is typically through a file-system interface**
  - Raw I/O or file-system access **- “binary xfr” of file data - interpretation is in application (personality of file lost)**
  - Memory-mapped **(to VM)** file access possible **- use memory instructions rather than I/O instructions - very efficient (ex: swap space for disk).**
  - **Device driver xfr’s blocks at a time - as in paging**
  - **DMA transfer is block oriented**
- Character devices include keyboards, mice, serial ports
  - **Device driver xfr’s byte at a time**
  - Commands include **get, put - character at a time**
  - Libraries layered on top allow *line editing* **- ex: keyboard input**
  - **could be beefed up to use a line at a time (buffering)**
- **Block & character devices also determine the two general device driver catagories**

Network Devices

- Varying enough from block and character to have own interface **- OS makes network device interface distinct from disk interface - due to significant differences between the two**
- Unix and Windows NT/9*i*/2000 include socket interface
  - ☞ Separates network protocol from network operation
  - ☞ ***Encapsulates*** **details of various network devices for application … analogous to a file and the disk???**
  - ☞ Includes `select` functionality - used to manage and access sockets - returns info on packets waiting or ability to accept packets - avoids polling
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes) **… you saw some of these!**

Clocks and Timers

- Provide current time, elapsed time, timer
- If programmable, interval time used for timings, periodic interrupts
- `ioctl` (on UNIX) covers odd aspects of I/O such as clocks and timers **- a back door for device driver writers (roll your own). Can implement “secret” calls which may not be documented in a users or programming manual**

Blocking and Nonblocking I/O

- Blocking - process **(making the request blocks - lets other process execute)** suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
  - multi-threading **- depends on role of OS in thread management**
- 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 **- ex: read a “small” portion of a file very quickly, use it, and go back for more, ex: displaying video “continuously from a disk”**
  - **Asynchronous** - process **(making the asynch request)** *runs while I/O executes*
  - Difficult to use **- can it continue without the results of the I/O?**
  - I/O subsystem signals process when I/O completed **- via interrupt (soft), or setting of shared variable which is periodically tasted.**