6. Consideration of I/O handling

a. Device independence means that it should be possible to write programs that can access any I/O device without having to specify the device in advance.

b. Uniform naming means that the name of a file or device should be a string or an integer and not depend on the device in any way

c. Error handling should be handled as close to the hardware as possible.
   i. If the controller discovers a read error, it should try to correct the error itself if it can.
   ii. If the controller discover a read error and cannot correct the error, then the device driver should handle it.

d. Many error are transients (ie: read error due dust on the read head) such errors go away if the operation is repeated

e. Only if lower layer are not able to deal with the problem should the upper layers be told about it.

f. In many cases error recovery can be done transparently at a low level without the upper level even knowing about the error

g. Synchronous (blocking) versus Asynchronous (interrupt-driven) transfers
   i. Most physical I/O is asynchronous: the CPU starts the transfer and goes off to do something else until the interrupt arrives
   i. if the I/O operation are blocking, user programs are easier to write
      ie: when a program user the system read, the program automatically suspended until the data is available in the buffer
   IT is up to the OS to make operations that are actually interrupt-driven look blocking to the user program

[redacted] final review

h. Buffering: Often data that come off a device cannot be stored directly in its final destination.
Buffering involves considerable copying and often has a major impact on I/O performance.

i. Sharable versus dedicated devices

i. Some I/O devices, such as disks, can be used by many users at the same time. No problem if multiple users having open files on the same disk at the same time

ii Other devices, such as tape drives, has to be dedicated to a single user until that user is finished. Intermixed sectors with two or more users at random will not work
Unshared devices have other problems such as deadlocks

j. Printer Driver interact with the printer spooler

k. Keyboard and mouses are special kind of devices because regular keys goes to the OS while special keys goes to the process.
For example: Disk can have many read and write from any process since is shared while the mouse and keyboard are dedicated so they are own by one process at the time

l. A keylogger will record all keystrokes typed in and periodically sends them to some machine or sequence of machines (including zombies) & ultimately deliver to criminal

7. Programmed I/O versus Interrupt-driven I/O versus DMA I/O

a. Programmed I/O:

i. user program acquires the printer for writing by making a system call to open it.

I. If the printer is currently in use by another process, this call will fail and return error or will block until printer is available.

ii. Once the user program has the printer, the user process makes a system call telling the OS to print the string on the printer

iii the OS copies the buffer with the string to an array in kernel space

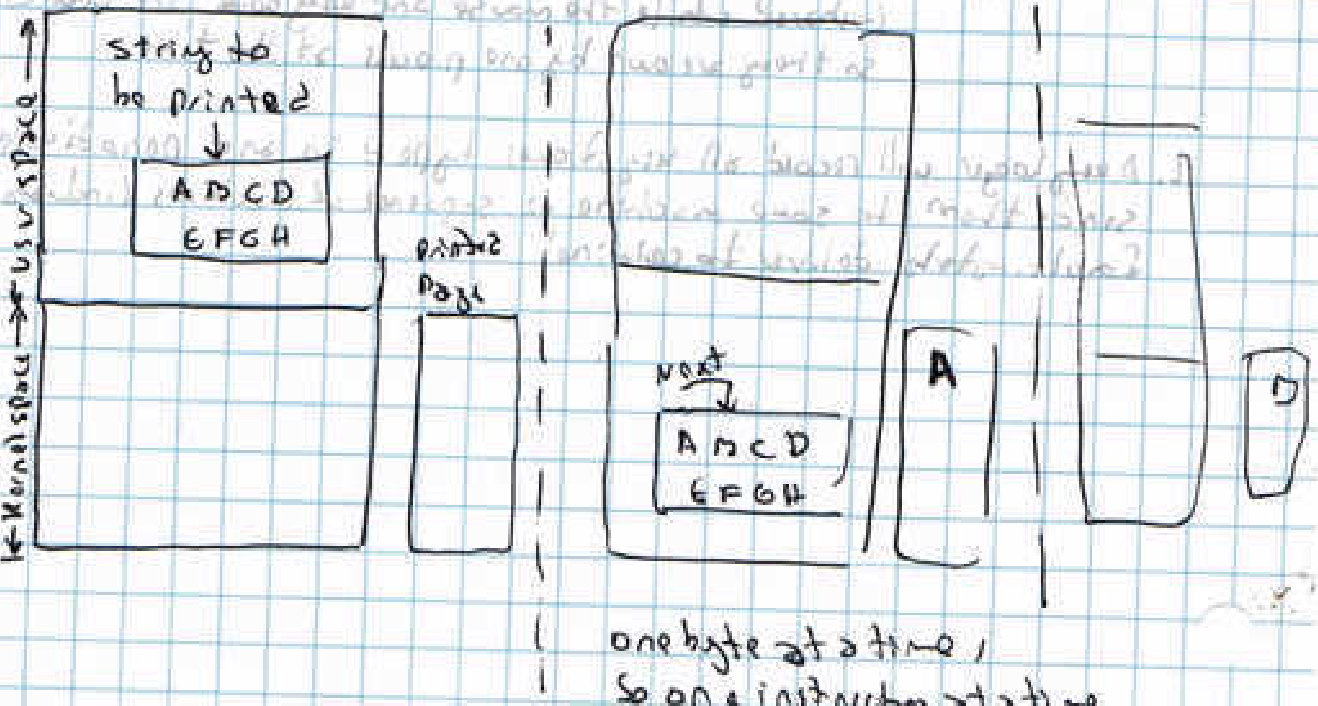
iv Then the OS enters a loop outputting the characters one at a time.

v. After outputting a character, the CPU continuously polls the device to see if it is ready to accept another one. this behavior is often called polling or busy waiting

vi Advantage: simple to implement

vii Disadvantage: tying up the CPU full time until all the I/O is done

viii.



one byte at a time, so one instruction at a time

[redacted] final review

b. Interrupt-Driven I/O

i. to allow the CPU to do something else while waiting for the printer to become ready is to use interrupts

ii. when the system call to print the string is made, the buffer is copied to the printer as soon as it is willing to accept a character

iii. At that point the CPU calls the scheduler, and some other process is run

iv. The process that asked for the string to be printed is blocked until entire string has printed

v. When the printer has printed the character and is prepared to accept the next one, it generates an interrupt

I. If there are no more characters to print, the interrupt handler takes some action to unblock the user, or else it outputs the next character, acknowledges the interrupt, and returns to the process that was running just before the interrupt, which continues where it left off

vi. so: writing a string to the printer using interrupt-driven I/O

I. Code executed when print system call is made (starts off by printing the first character)

II. Interrupt service procedure (called every time the printer completes printing one character and generates an interrupt)

178 C. I/O using DMA:

[redacted]

i. Let the DMA controller feed the characters to the printer one at time, without the CPU being bothered

ii. DMA is programmed I/O but the DMA controller is doing all the work instead of the CPU

iii. this require special Hardware, but freed up the CPU during the I/O to do other work

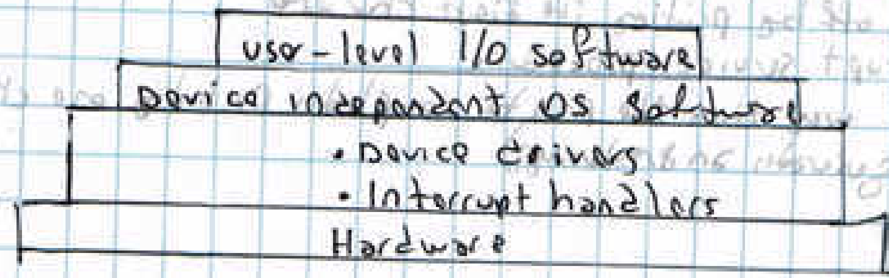
iv. using the DMA reduce the number of interrupts from one per character to one per buffer printed

v. DMA is slower than CPU so if the CPU has nothing to do while waiting for the DMA, then interrupted-driven IO or programme I/O may better, However, DMA is worth

→ vi. When ask in which context is the interrupt done we say interrupt context!

vii. Interrupts have to be serviced very fast and they don't use OS process.

B. I/O Software Layers



[redacted] final review

9. Device Drivers

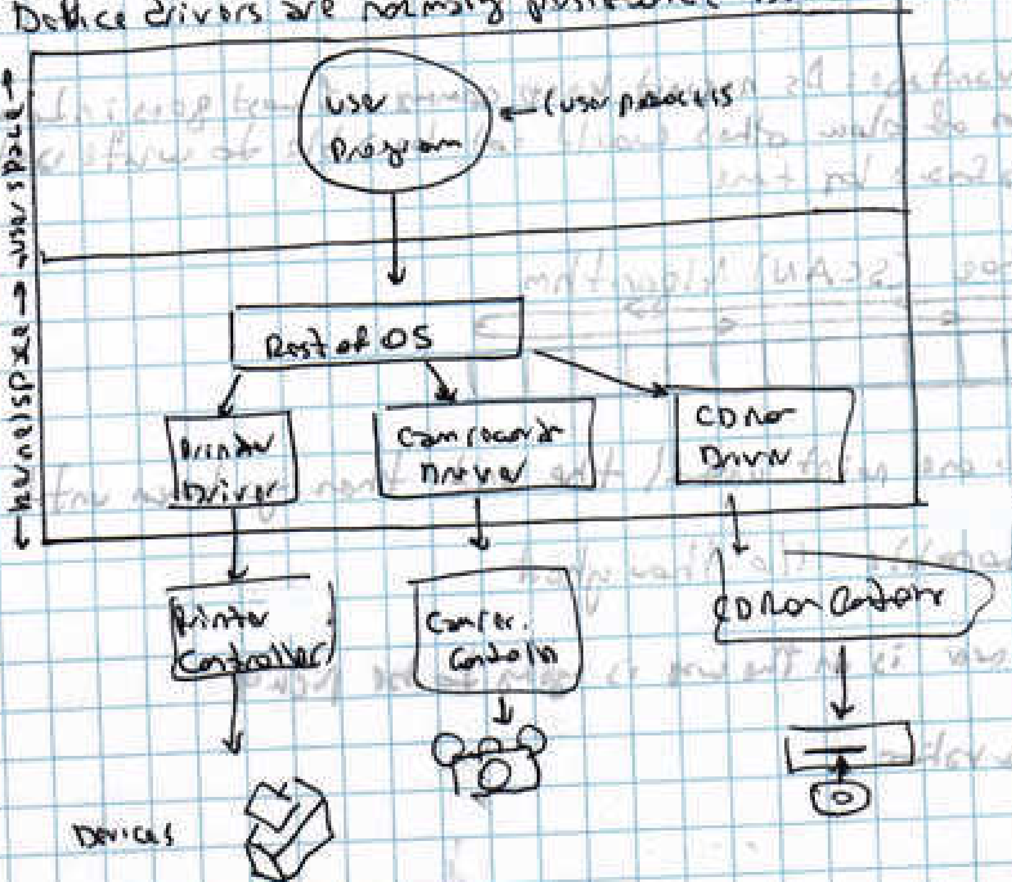
a. A device driver is device-specific code for controlling a I/O device attached to a computer

b. the device driver is written by the device's manufacturer and delivered along with the device

c. Each device driver normally handles one device type, or at most one class of closely related devices. (such as a disk) On the other hand, a mouse and joystick, for example, are so different that different drivers are required.

d. Since a device driver is written by outsiders of the OS architecture, a well-defined model of what a driver does and how interacts with the rest of the OS is needed.

e. Device drivers are normally positioned below the rest of the OS.



(Logical positions of device drivers. In reality all communications between drivers and device controllers goes over bus

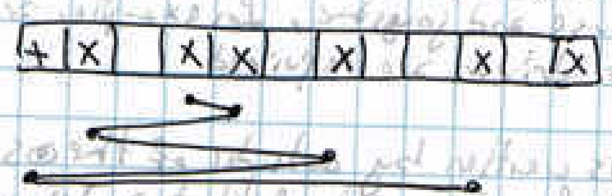
10. Disk Arm Scheduling Algorithms [redacted]

a. Time required to read or write a disk is determined by:
   i. Seek time
   ii. Rotational delay
   iii. Actual transfer time

b. Seek time dominates

c. Error checking is done by controllers

d. Shortest Seek First (SSF) disk scheduling algorithm

i. 

ii. Next request closest to the previous

iii. High I/O throughput

iv. Disadvantage: As request keep coming if most goes in one section of blocks others would not be able to write until waiting for a long time

e. Elevator (SCAN) Algorithm

i. 

ii. Go to one point until the end then go back until the start

iii. Reasonable I/O throughput

iv. Whatever is on the way is going to be picked

v. No starvation