I/O Management

Amir H. Payberah
payberah@kth.se

2022

Overview

- I/O management is a major component of OS design and operation.

Overview

- I/O management is a major component of OS design and operation.
- Ports, busses, device controllers connect to various devices.

Overview

- I/O management is a major component of OS design and operation.
- Ports, busses, device controllers connect to various devices.
- Device drivers encapsulate device details.
  - Present uniform device-access interface to I/O subsystem.

I/O Hardware

I/O Hardware

- Variety of I/O devices:
  - Storage, e.g., disks, tapes
  - Transmission, e.g., network connections, bluetooth
  - Human-interface, e.g., screen, keyboard, mouse, audio in and out

I/O Hardware

- Variety of I/O devices:
  - Storage, e.g., disks, tapes
  - Transmission, e.g., network connections, bluetooth
  - Human-interface, e.g., screen, keyboard, mouse, audio in and out

- We only need to understand how the devices are attached and how the software can control the hardware.

Common Concepts in I/O Hardware

- Port: connection point for device.

Common Concepts in I/O Hardware

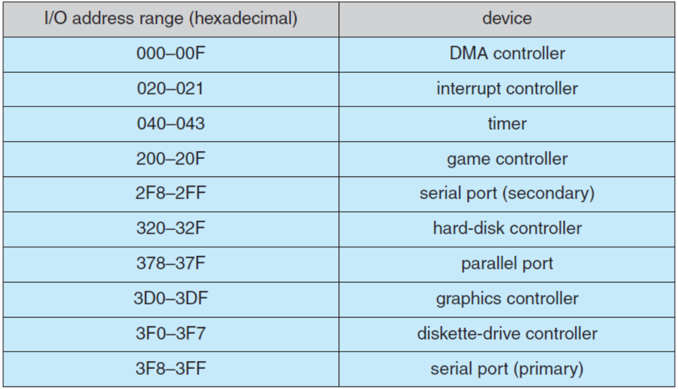
- Port: connection point for device.
- Bus: set of wires and protocols that specify the messages that can be sent on the wires.

Common Concepts in I/O Hardware

- Port: connection point for device.
- Bus: set of wires and protocols that specify the messages that can be sent on the wires.
- Controller: integrated or separate circuit board that operate a port, a bus, or a device.

Port

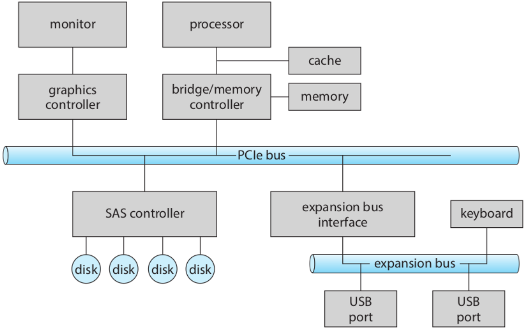
- Device I/O ports addresses on PCs.



I/O address range (hexadecimal)	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	diskette-drive controller
3F8–3FF	serial port (primary)

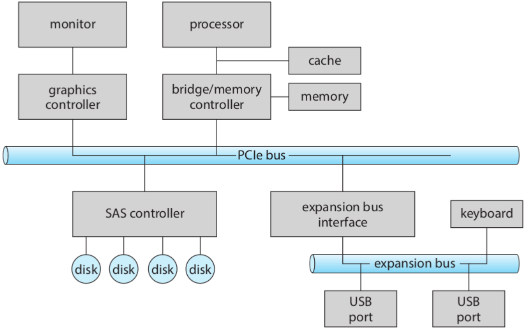
Bus

- PCI bus: connects the processor-memory subsystem to fast devices.



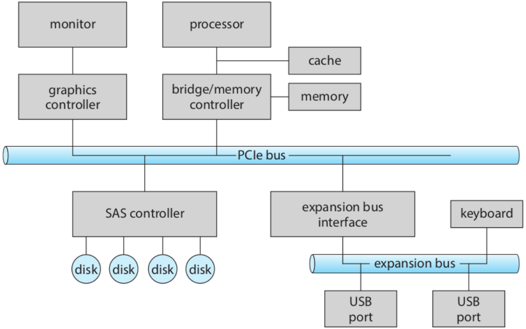
Bus

- PCI bus: connects the processor-memory subsystem to fast devices.
- Expansion bus: connects relatively slow devices.



Bus

- PCI bus: connects the processor-memory subsystem to fast devices.
- Expansion bus: connects relatively slow devices.
- Serial-attached SCSI (SAS)



Host-Device Interaction

Host-Device Interaction

- Polling
- Interrupt
- Direct memory access (DMA)

Polling (1/2)

- A handshake between the host and a controller.

Polling (1/2)

- A handshake between the host and a controller.
- Assume 2 bits for coordination: `busy` and `command-ready` bits.

Polling (1/2)

- A handshake between the host and a controller.
- Assume 2 bits for coordination: `busy` and `command-ready` bits.
- For each byte of I/O:
  1. Host reads the `busy` bit from the status register until 0.

Polling (1/2)

- A handshake between the host and a controller.
- Assume 2 bits for coordination: `busy` and `command-ready` bits.
- For each byte of I/O:
  1. Host reads the `busy` bit from the status register until 0.
  2. Host sets the `write` bit and if write copies data into the data-out register.

Polling (1/2)

- A handshake between the host and a controller.
- Assume 2 bits for coordination: `busy` and `command-ready` bits.
- For each byte of I/O:
  1. Host reads the `busy` bit from the status register until 0.
  2. Host sets the `write` bit and if write copies data into the data-out register.
  3. Host sets the `command-ready` bit.

Polling (1/2)

- A handshake between the host and a controller.
- Assume 2 bits for coordination: `busy` and `command-ready` bits.
- For each byte of I/O:
  1. Host reads the `busy` bit from the status register until 0.
  2. Host sets the `write` bit and if write copies data into the data-out register.
  3. Host sets the `command-ready` bit.
  4. Controller sets the `busy` bit, executes transfer.

Polling (1/2)

- A handshake between the host and a controller.
- Assume 2 bits for coordination: `busy` and `command-ready` bits.
- For each byte of I/O:
  1. Host reads the `busy` bit from the status register until 0.
  2. Host sets the `write` bit and if write copies data into the data-out register.
  3. Host sets the `command-ready` bit.
  4. Controller sets the `busy` bit, executes transfer.
  5. Controller clears the `busy` bit, `error` bit, and `command-ready` bit when transfer done.

Polling (2/2)

- Step 1 is busy-wait cycle (polling) to wait for I/O from device.

Polling (2/2)

- Step 1 is busy-wait cycle (polling) to wait for I/O from device.
- Reasonable if device is fast.

Polling (2/2)

- Step 1 is busy-wait cycle (polling) to wait for I/O from device.
- Reasonable if device is fast.
- But inefficient if device slow.

Interrupts (1/3)

- Polling can happen in 3 instruction cycles.

Interrupts (1/3)

- Polling can happen in 3 instruction cycles.
  - (1) read status, (2) extract status bit, and (3) branch if not zero.

Interrupts (1/3)

- Polling can happen in 3 instruction cycles.
  - (1) read status, (2) extract status bit, and (3) branch if not zero.
  - Inefficient, but more efficient way?

Interrupts (1/3)

- Polling can happen in 3 instruction cycles.
  - (1) read status, (2) extract status bit, and (3) branch if not zero.
  - Inefficient, but more efficient way?
- CPU interrupt-request line is triggered by I/O device.

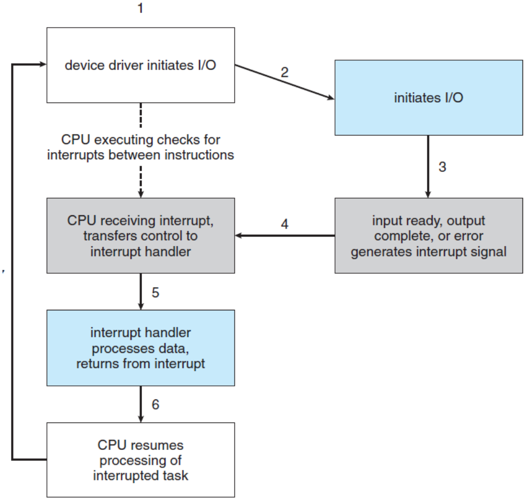
Interrupts (1/3)

- Polling can happen in 3 instruction cycles.
  - (1) read status, (2) extract status bit, and (3) branch if not zero.
  - Inefficient, but more efficient way?
- CPU interrupt-request line is triggered by I/O device.
  - Checked by processor after each instruction.

Interrupts (1/3)

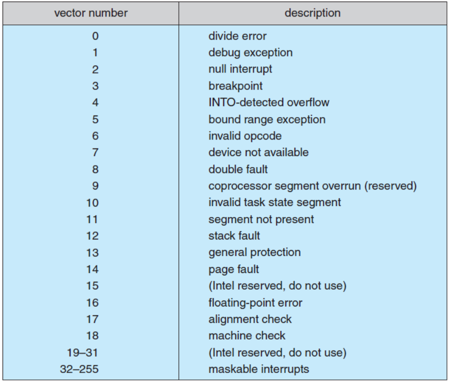
- Polling can happen in 3 instruction cycles.
  - (1) read status, (2) extract status bit, and (3) branch if not zero.
  - Inefficient, but more efficient way?
- CPU interrupt-request line is triggered by I/O device.
  - Checked by processor after each instruction.
  - Saves state and jumps to interrupt-handler routine at a fixed address in memory.

Interrupts (2/3)


1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

Interrupts (3/3)

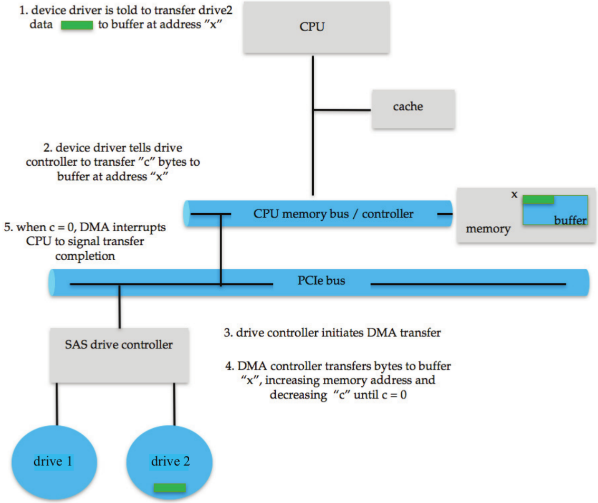
- The interrupt mechanism accepts an address: a number that selects a specific interrupt-handling routine.



vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Direct Memory Access (DMA)

- Bypasses CPU to transfer data directly between I/O device and memory.


1. device driver is told to transfer drive2 data 🟩 to buffer at address "x"
CPU
cache
2. device driver tells drive controller to transfer "c" bytes to buffer at address "x"
CPU memory bus / controller
x
buffer
memory
5. when c = 0, DMA interrupts CPU to signal transfer completion
PCIe bus
SAS drive controller
3. drive controller initiates DMA transfer
4. DMA controller transfers bytes to buffer "x", increasing memory address and decreasing "c" until c = 0
drive 1
drive 2

Application I/O Interface

Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes.

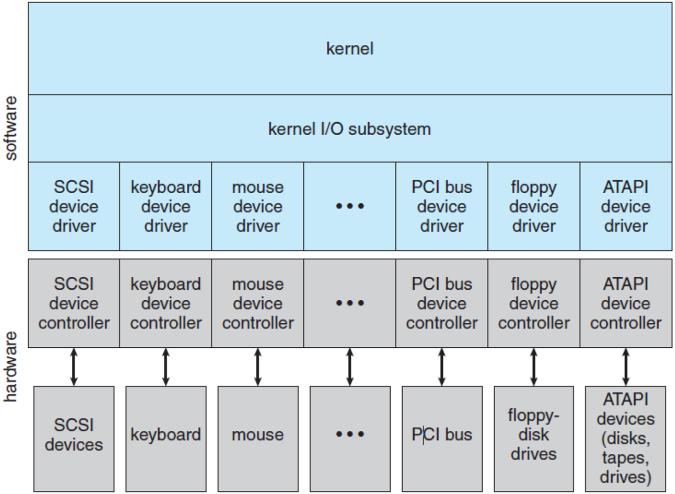
Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes.
- Device-driver layer hides differences among I/O controllers from kernel.

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.
- Each OS has its own I/O subsystem structures and device driver frameworks.

A Kernel I/O Structure



Characteristics of I/O Devices (1/2)

- Devices vary in many dimensions

Characteristics of I/O Devices (1/2)

- Devices vary in many dimensions
    - Data-transfer mode: character or block

Characteristics of I/O Devices (1/2)

- Devices vary in many dimensions
  - Data-transfer mode: character or block
  - Access method: sequential or random-access

Characteristics of I/O Devices (1/2)

- Devices vary in many dimensions
  - Data-transfer mode: character or block
  - Access method: sequential or random-access
  - Transfer schedule: synchronous or asynchronous (or both)

Characteristics of I/O Devices (1/2)

- Devices vary in many dimensions
  - Data-transfer mode: character or block
  - Access method: sequential or random-access
  - Transfer schedule: synchronous or asynchronous (or both)
  - Sharing: sharable or dedicated

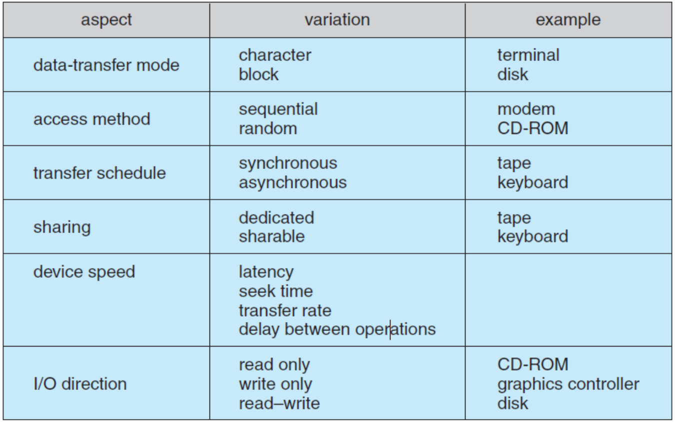
Characteristics of I/O Devices (1/2)

- Devices vary in many dimensions
  - Data-transfer mode: character or block
  - Access method: sequential or random-access
  - Transfer schedule: synchronous or asynchronous (or both)
  - Sharing: sharable or dedicated
  - Device speed: speed of operation

Characteristics of I/O Devices (1/2)

- Devices vary in many dimensions
  - Data-transfer mode: character or block
  - Access method: sequential or random-access
  - Transfer schedule: synchronous or asynchronous (or both)
  - Sharing: sharable or dedicated
  - Device speed: speed of operation
  - I/O direction: read-write, read only, or write only

Characteristics of I/O Devices (2/2)



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	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Character Devices

- Character devices include keyboards, mouse, serial ports.

Character Devices

- Character devices include keyboards, mouse, serial ports.
- A character device transfers bytes one by one.

Character Devices

- Character devices include keyboards, mouse, serial ports.
- A character device transfers bytes one by one.
- Commands include `get()` and `put()`.

Block Devices

- Block devices include disk drives.

Block Devices

- Block devices include disk drives.
- Commands include `read()` and `write()` and `seek()` for random-access devices.

Network Devices

- Varying enough from block and character to have own interface.

Network Devices

- Varying enough from block and character to have own interface.
- Linux, Unix, Windows and many others include socket interface.
  - Separates network protocol from network operation.

Clocks and Timers

- Provide current time, elapsed time, and timer (trigger operation X at time T)

Clocks and Timers

- Provide current time, elapsed time, and timer (trigger operation X at time T)
- Programmable interval timer, the hardware used for timings, and periodic interrupts.

Clocks and Timers

- Provide current time, elapsed time, and timer (trigger operation X at time T)
- Programmable interval timer, the hardware used for timings, and periodic interrupts.
- Normal resolution about $1/60$ second.

Clocks and Timers

- Provide current time, elapsed time, and timer (trigger operation X at time T)
- Programmable interval timer, the hardware used for timings, and periodic interrupts.
- Normal resolution about 1/60 second.
- Some systems provide higher-resolution timers.

Blocking, Nonblocking and Asynchronous I/O

- Blocking: process suspended until I/O completed
  - Insufficient for some needs

Blocking, Nonblocking and Asynchronous I/O

- Blocking: process suspended until I/O completed
  - Insufficient for some needs
- Nonblocking: I/O call returns as much as available
  - User interface, data copy (buffered I/O)

Blocking, Nonblocking and Asynchronous I/O

- Blocking: process suspended until I/O completed
  - Insufficient for some needs
- Nonblocking: I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading

Blocking, Nonblocking and Asynchronous I/O

- Blocking: process suspended until I/O completed
  - Insufficient for some needs
- Nonblocking: I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - `select()` to find if data ready then `read()` or `write()` to transfers.

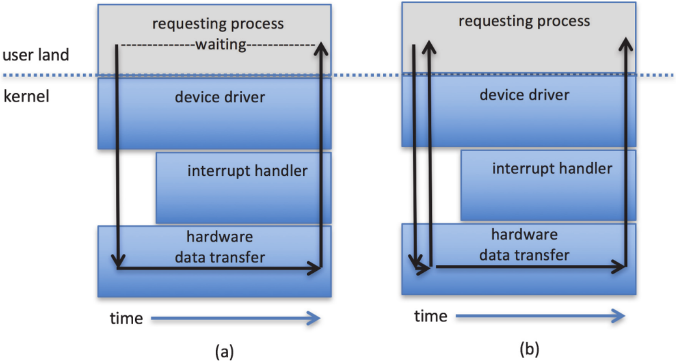
Blocking, Nonblocking and Asynchronous I/O

- Blocking: process suspended until I/O completed
  - Insufficient for some needs

- Nonblocking: I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - `select()` to find if data ready then `read()` or `write()` to transfers.

- Asynchronous: process runs while I/O executes
  - I/O subsystem signals process when I/O completed.

Synchronous vs. Asynchronous I/O Methods



Synchronous

Asynchronous

Kernel I/O Subsystem

Kernel I/O Subsystem

- Kernels provide many services related to I/O:

Kernel I/O Subsystem

- Kernels provide many services related to I/O:
  - Scheduling

Kernel I/O Subsystem

- Kernels provide many services related to I/O:
  - Scheduling
  - Buffering

Kernel I/O Subsystem

- Kernels provide many services related to I/O:
  - Scheduling
  - Buffering
  - Caching

Kernel I/O Subsystem

- Kernels provide many services related to I/O:
  - Scheduling
  - Buffering
  - Caching
  - Spooling

Kernel I/O Subsystem

- Kernels provide many services related to I/O:
  - Scheduling
  - Buffering
  - Caching
  - Spooling
  - Device reservation

Kernel I/O Subsystem

- Kernels provide many services related to I/O:
  - Scheduling
  - Buffering
  - Caching
  - Spooling
  - Device reservation
  - Error handling

Scheduling (1/2)

- Determine a good order in which to execute I/O requests.

Scheduling (1/2)

- Determine a good order in which to execute I/O requests.
- Some I/O request ordering via per-device queue.

Scheduling (1/2)

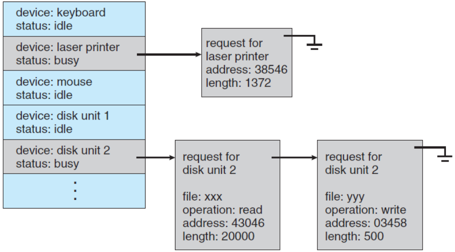
- Determine a good order in which to execute I/O requests.
- Some I/O request ordering via per-device queue.
- Some OSs try fairness.

Scheduling (2/2)

- In asynchronous I/O the kernel must be able to keep track of many I/O requests at the same time.

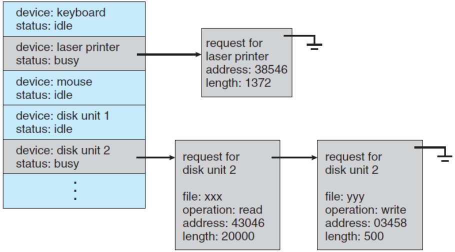
Scheduling (2/2)

- In asynchronous I/O the kernel must be able to keep track of many I/O requests at the same time.
  - The OS attaches the wait queue to a device-status table.



Scheduling (2/2)

- In asynchronous I/O the kernel must be able to keep track of many I/O requests at the same time.
  - The OS attaches the wait queue to a device-status table.
  - The table contains an entry for each I/O device.



device: keyboard
status: idle

device: laser printer
status: busy

device: mouse
status: idle

device: disk unit 1
status: idle

device: disk unit 2
status: busy

⋮

request for
laser printer
address: 38546
length: 1372

request for
disk unit 2

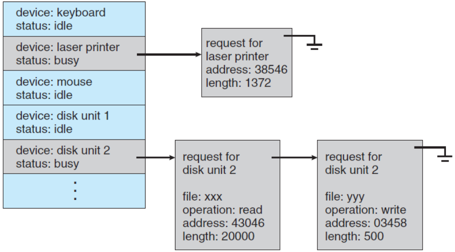
file: xxx
operation: read
address: 43046
length: 20000

request for
disk unit 2

file: yyy
operation: write
address: 03458
length: 500

Scheduling (2/2)

- In asynchronous I/O the kernel must be able to keep track of many I/O requests at the same time.
  - The OS attaches the wait queue to a device-status table.
  - The table contains an entry for each I/O device.
  - If the device is busy with a request, the type of request and other parameters will be stored in the table entry for that device.



Buffering and Caching

- Buffering: stores data in memory while transferring between devices.

Buffering and Caching

- Buffering: stores data in memory while transferring between devices.
  - To cope with device speed mismatch.

Buffering and Caching

- Buffering: stores data in memory while transferring between devices.
  - To cope with device speed mismatch.
  - To cope with device transfer size mismatch, e.g., fragmentation and reassembly of messages.

Buffering and Caching

- Buffering: stores data in memory while transferring between devices.
  - To cope with device speed mismatch.
  - To cope with device transfer size mismatch, e.g., fragmentation and reassembly of messages.
  - To maintain copy semantics.

Buffering and Caching

- Buffering: stores data in memory while transferring between devices.
  - To cope with device speed mismatch.
  - To cope with device transfer size mismatch, e.g., fragmentation and reassembly of messages.
  - To maintain copy semantics.
- Caching: faster device holding copy of data.

Buffering and Caching

- **Buffering**: stores data in memory while transferring between devices.
  - To cope with device speed mismatch.
  - To cope with device transfer size mismatch, e.g., fragmentation and reassembly of messages.
  - To maintain copy semantics.

- **Caching**: faster device holding copy of data.
  - Always just a copy
  - Key to performance

Spooling and Device Reservation

- **Spooling**: a buffer that holds output for a device.
  - If device can serve only one request at a time, i.e., printing

Spooling and Device Reservation

- Spooling: a buffer that holds 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 de-allocation
  - Watch out for deadlock

Error Handling

- OS can recover from disk read, device unavailable, and transient write failures.

Error Handling

- OS can recover from disk read, device unavailable, and transient write failures.
    - Retry a read or write.

Error Handling

- OS can recover from disk read, device unavailable, and transient write failures.
  - Retry a read or write.
  - Track error frequencies, stop using device with increasing frequency of retry-able errors.

Error Handling

- OS can recover from disk read, device unavailable, and transient write failures.
  - Retry a read or write.
  - Track error frequencies, stop using device with increasing frequency of retry-able errors.
- Most return an error number when I/O request fails.

Error Handling

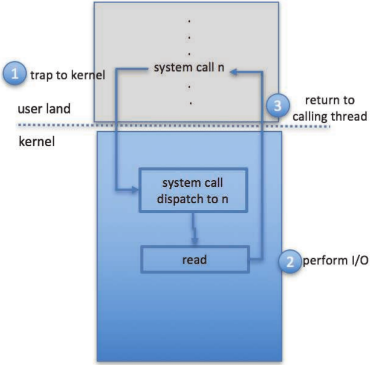
- OS can recover from disk read, device unavailable, and transient write failures.
  - Retry a read or write.
  - Track error frequencies, stop using device with increasing frequency of retry-able errors.
- Most return an error number when I/O request fails.
- System error logs hold problem reports.

I/O Protection

- A user process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions.

I/O Protection

- A user process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions.
- I/O must be performed via system calls.



Summary

Summary

- I/O hardware: port, bus, controller

Summary

- I/O hardware: port, bus, controller
- Host-device interaction: polling, interrupt, DMA

Summary

- I/O hardware: port, bus, controller
- Host-device interaction: polling, interrupt, DMA
- Devices: char, block, network

Summary

- I/O hardware: port, bus, controller
- Host-device interaction: polling, interrupt, DMA
- Devices: char, block, network
- Kernel I/O: schedulling, buffering, caching, spooling, device reservation, error handling

Questions?

Acknowledgements

Some slides were derived from Avi Silberschatz slides.