Outline

- Diversity of I/O devices
  - block and character devices
- Organization of I/O subsystem of kernel
  - device drivers
- Common hardware characteristics of device
- I/O subsystem tasks
I/O subsystem

- Lowest layer of the OS that separates rest of kernel from complexity of managing I/O devices
  - storage
  - transmission (nic/modem)
  - interface (kbd, mouse, screen, printer)
  - special purpose devices
# Diversity of devices

## Table of device characteristics

<table>
<thead>
<tr>
<th>aspect</th>
<th>variation</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-transfer mode</td>
<td>character</td>
<td>terminal</td>
</tr>
<tr>
<td></td>
<td>block</td>
<td>disk</td>
</tr>
<tr>
<td>access method</td>
<td>sequential</td>
<td>modem</td>
</tr>
<tr>
<td></td>
<td>random</td>
<td>CD-ROM</td>
</tr>
<tr>
<td>transfer schedule</td>
<td>synchronous</td>
<td>tape</td>
</tr>
<tr>
<td></td>
<td>asynchronous</td>
<td>keyboard</td>
</tr>
<tr>
<td>sharing</td>
<td>dedicated</td>
<td>tape</td>
</tr>
<tr>
<td></td>
<td>sharable</td>
<td>keyboard</td>
</tr>
<tr>
<td>device speed</td>
<td>latency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seek time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>transfer rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>delay between operations</td>
<td></td>
</tr>
<tr>
<td>I/O direction</td>
<td>read only</td>
<td>CD-ROM</td>
</tr>
<tr>
<td></td>
<td>write only</td>
<td>graphics controller</td>
</tr>
<tr>
<td></td>
<td>read/write</td>
<td>disk</td>
</tr>
</tbody>
</table>
I/O devices are grouped into some conventional types
- each group is accessed via a standardized interface (set of functions)

I/O subsystem of kernel uses **device drivers** to manage individual devices in each group

Device driver routines:
- handle hardware-specific details of operation of devices
- present uniform interface through which devices are accessed

`ioctl(fd, OPCODE, buf)` – general I/O control system call
- enables any application to access any functionality implemented by any device driver

**NOTE:** hardware manufacturers design devices to be compatible with existing controller interface, or write drivers to provide required interface to the OS.
Organization

The diagram illustrates the organization of kernel I/O subsystems and device drivers. The kernel sits at the top, with the kernel I/O subsystems below it. This subsystem includes various device drivers such as SCSI device driver, keyboard device driver, mouse device driver, PCI bus device driver, floppy device driver, and ATAPI device driver. Below the kernel I/O subsystems, there are device controllers and devices, including SCSI devices, keyboard, mouse, PCI bus, floppy disk drives, and ATAPI devices (disks, tapes, drives). The diagram shows the hierarchical relationship between software and hardware components.
Block and character devices

Block devices:
- Unit of data transfer: a block of bytes
- Example: disk drives
- Operations: read, write, seek
- Usually accessed via a filesystem interface
  - raw I/O: OS, DBMS may directly access device as a linear array of blocks

Character devices:
- Unit of data transfer: one byte
- Example: keyboard, terminal, mouse, modem, printers
- Operations:
  - getting / putting one character
  - line-at-a-time access with buffering and editing services
Network devices

- Use a **socket** interface

- Operations:
  - connecting to a remote socket
  - accepting connections to a local socket
  - sending / receiving packets
  - **select** from a set of sockets
Clocks and timers

- Operations: give current time, give elapsed time, set timers / alarms

- Hardware support:
  - clock may send interrupt at every “tick” (coarse)
  - clock may be high-frequency counter (high resolution)
    - value of counter may be read from device register

- Uses of programmable interval timer:
  - time-slicing
  - periodic flushing of buffers to disk
  - network timeouts
Blocking / nonblocking I/O

- Blocking I/O: calling process suspended till I/O completes
- Non-blocking I/O: process gets whatever input is available + status code
- Asynchronous I/O:
  - I/O request is scheduled, process returns immediately
  - completion of I/O communicated via
    - setting a variable
    - calling a callback procedure
    - sending a signal
Hardware components

- **Port**: connection point between device and computer (host)
- **Bus**: set of wires connecting device(s) to host
  - communication between host and device (for control instructions / data) uses well-defined protocol
  - arrangement: daisy chain or shared direct access
- **Controller**: circuit that operates a port / bus / device
  Examples:
  - serial port controller – controls signals on serial port wires
  - SCSI bus controller – circuit that controls SCSI bus
    - contains processor, microcode, private memory to handle SCSI protocol
    - usually implemented on a separate circuit board
  - Disk controller – circuit that implements the disk side of the protocol for the connection type (IDE, SCSI, etc.)
    - contains processor, microcode to handle prefetching, buffering, caching, etc.
Controllers

- Contain registers for data / control signals
  - processor communicates with controller by writing / reading bit patterns into / from these registers
  - **data-in, data-out** – read/written to get / send data
    - 1–4 bytes long
  - **control** – commands
  - **status** – whether current command has completed, whether a byte is available for reading, error conditions, etc.

- May contain FIFO chips
  - extension of data register
  - can hold several bytes of input / output data until device / host is able to receive the data
Communication methods:
- using special I/O instructions
  - triggers bus lines to select proper device and move bits into/ out of DRs
- using memory mapped I/O
  - DRs are mapped into CPU’s address space
  - standard LOAD, STORE instructions used to read / write DRs
- combination of I/O instructions + memory-mapped I/O
  Example: some graphics controllers
  - I/O ports used for basic control operations
  - large memory-mapped region used to hold screen contents
    (processor sends I/O to screen by writing data into memory-mapped region)
**I/O subsystem tasks: scheduling**

**Aim:** to determine a “good” order of execution for a set of I/O requests

- improve overall system performance
- reduce average waiting time for I/O completion
- ensure better service to higher priority / delay-sensitive requests (e.g. from virtual memory sub-system)
- ensure fairness (?)
I/O subsystem tasks: buffering/caching

- **Aim:**
  - to cope with speed mismatch between producer and consumer of data
    - (i) data arriving from slow device is accumulated in a buffer until full
    - (ii) full buffer written to fast device in single operation
    - (iii) 2nd buffer used while 1st buffer is being written to disk
  - to adapt between devices with different data transfer sizes (e.g. network packet fragmentation and reassembly)
  - to support “copy semantics” for I/O
    - write operations write single version of data
      - (no mixing of old and new data can happen)
  - Same space may be used for both caching and buffering

NOTE: *Buffer* may hold only existing copy of data (e.g. network I/O buffer) *Cache* necessarily holds a copy of a data item that is originally stored elsewhere.
I/O subsystem tasks: spooling

- Spool: buffer that holds output for a device that cannot accept interleaved data streams from multiple concurrent applications (e.g. printer, tape drive)

- I/O subsystem serializes concurrent output from multiple processes

- Implementation:
  1. output from separate processes is spooled to separate files
  2. spooling system queues spool files for output to device

- Spooling system interface typically provides control interface for viewing, pausing, removing jobs
I/O subsystem tasks: error handling

- Transient failures handled by kernel
  e.g. disk-read / network-send failure ⇒ automatic retry

- SCSI devices error reporting
  - sense key: specifies general nature of failure
    (e.g. h/w error, illegal request, etc.)
  - additional sense code: category of failure
    (e.g. bad command parameter, self-test failure, etc.)
  - additional sense code qualifier: more detailed information
    (e.g. which command parameter, which subsystem failed, etc.)
Disk structure

- Sector 0: 1st sector of 1st track of outermost cylinder
- Sectors numbered sequentially by sector, track, cylinder (outer → inner)
- Zones: groups of cylinders for which # sectors per track is constant (outer zones have \( \sim 40\% \) more sectors per track than innermost zones)
- \# sectors/track: \( \sim 100 \); \# cylinders/disk: \( \sim 10^3 \)

**I/O time:** seek time + rotational latency + transfer time
Disk scheduling

- **FCFS**
  - fair algorithm (no starvation)
  - poor performance

- **Shortest seek-time first**: choose pending request that is closest to current head position
  - starvation possible
  - better than FCFS, but non-optimal

- **SCAN/Elevator**: disk arm moves from one end to the other and back, servicing requests

- **C-SCAN**: no requests serviced on return trip
  - more uniform wait time

- **LOOK, C-LOOK**: disk arm moves till farthest request in each direction (instead of end of disk)

**OS issues:**

- Physical layout of directories and files
- Scheduling algorithms may be implemented in hardware / software