Computer architecture

Gigabyte Z170 Gaming

- 2 PCIe x16/x4
- 4 PCIe x1
- 2 USB 3.1
- 6 USB 3.0
- 4 USB 2.0
- 6 SATA-III
- 2 SATA Express
- 1 M.2
- 1 gigabit Ethernet
- 4 DDR4 SDRAM

Why?

Give me two reasons why we would like to have secondary storage?
System architecture

- application
- POSIX API
- I/O library
- generic block interface
- device
- HDD
- SSD

how to interact with a device

- A register to read the status of the device.
- A register to instruct the device to read or write.
- A register that holds the data.
- I/O-bus could be separate from memory bus (or the same).
- The driver will use either special I/O instructions or regular load/store instructions.

70 percent of the code of an operating system is code for device drivers.

if you have the time

```c
char read_from_device() {
    while (STATUS == BUSY) {} // do nothing, just wait
    COMMAND = READ;
    while (STATUS == BUSY) {} // do nothing, just wait
    return DATA;
}
```

asynchronous I/O and interrupts

```c
int read_request(int pid, char *buffer) {
    while (STATUS == BUSY) {} // do nothing, just wait
    COMMAND = READ;
    interrupt->process = pid;
    interrupt->buffer = buffer;
    block_process(pid);
    scheduler();
}
```
The kernel is interrupt driven.

Each physical device is controlled by a *device driver* that provides the abstraction of a *character device* or *block device*.

Block devices used as interface to disk drives that provide persistent storage.

All though all storage devices are presented using the same abstraction, they have very different characteristics.

To understand the challenges and options of the operating system, you should know the basics of how storage devices work.
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides

Only one head at a time is used (no parallel read).

Sector addressing

- Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:
  - cylinder: 1024 (10-bits)
  - heads: 16 (4-bits)
  - sectors per cylinder: 63 (6-bits)
  - number of sectors: 1 Mi
  - largest disk assuming 512 Byte sectors: 512 MiByte

- Today, sectors are addressed linearly 0..n, Linear Block Addressing (LBA):
  - 28-bit or 48-bit address
  - up to 256 Ti sectors
  - largest disk assuming 4 KiByte sectors: 1 PiByte

```
> sudo hdparm -I /dev/sda
> dmesg | grep ata2
```
access time

- seek time: time to move arm to the right cylinder
- rotation time: time to rotate the disk
- read time: read one or more sectors

HDD - shoot out

- Seagate Desktop
  - rotation speed: 7200 rpm
  - average seek time: < 10 ms
  - average rotation time: 4 ms
  - average time to read a sector: < 14 ms
  - capacity: 2 TiByte
  - aprx. price: 900:-
  - cost capacity: 0.44 SEK/GiByte

- Seagate Cheeta 15K
  - rotation speed: 15000 rpm
  - average seek time: < 4 ms
  - average rotation time: 2 ms
  - average time to read a sector: < 6 ms
  - capacity: 600 GiByte
  - aprx. price: 2.200:-
  - cost capacity: 3.70 SEK/GiByte

read/write performance

If a sector is 512 bytes, it takes 10 ms to find and read a sector, and we want to read 512 MiBytes then .....?

- Time to find first sector is less relevant.
- If sectors that belong to the same file are close to each other we minimize movement of arm.
- Rotational speed should be high.
- The density i.e. how many sectors in each track is important.
- The communication with the drive should be fast.
- Typical read and write performance is between 150 MiByte/s to 250 MiByte/s.

who's in control

Historically, the Operating System was in complete control:

- it knew the layout cylinder-head-sector (CHS),
- could order data in segments that were close to each other and,
- would schedule disk operations to minimize arm movement.

There is a reason why MS-DOS is called MS-DOS.

Today, the drive can often make a better decision:

- it knows, but might not reveal, the layout.
- The operating system can help in grouping operations together, allowing the drive to decide in what order they should be done (Native Command Queuing).
SSD - Solid State Drive

Samsung 850 EVO

- total capacity: 250 GiByte
- form factor: 2.5"
- connection: SATA III
- cache size: 64 MiByte
- random access: 30 µs
- read throughput: 540 MiByte/s

MZ-75E250B/EU, aprx price, October 2016, 1000:-

SD cards - flash memory

SanDisk Ultra SDXC

- form factor: SDXC
- capacity: 64 GiByte
- read performance: 80 MiByte/s

aprx price, October 2016, 300:-

NAND - flash storage

You have constant time access to any page.
You can only write to (or program) an erased page.
You can only erase a block.

price performance

<table>
<thead>
<tr>
<th>Drive</th>
<th>Capacity</th>
<th>Price</th>
<th>SEK/GiByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD Desktop</td>
<td>2 TiByte</td>
<td>900:-</td>
<td>44 öre</td>
</tr>
<tr>
<td>HDD Performance</td>
<td>600 GiByte</td>
<td>2.200:-</td>
<td>3.70:-</td>
</tr>
<tr>
<td>SSD Desktop</td>
<td>250 GiByte</td>
<td>1000:-</td>
<td>4:-</td>
</tr>
</tbody>
</table>
Bus limitations

- SATA-III - 6 Gb/s, most internal HDD and SSD today
- SAS-3 - 12 Gb/s, enterprise RAID HDD
- USB3.1 - 10 Gb/s, everything
- PCI Express 3.0 x16 - 128 Gb/s, what is it used for?

An SSD has a read throughput of 500 MiByte/s which is a ... b/s?

SSD on the PCIe bus

Intel SSD 750 Series

- total capacity: 400 GiByte
- connection: PCI Express 3.0 x4
- read performance: 2200 MByte/s
- write performance: 900 MByte/s

aprx price, October 2016, 4.599:-

The M.2 connector

Samsung 960 EVO 500GB

- total capacity: 500 GiByte
- form factor: M.2-
- connection: PCI Express 3.0 x4
- read performance: 3.200 MByte/s
- write performance: 1.800 MByte/s

aprx price, November 2017, 2.400:-

SSD on the memory bus

HP NVDIMM 8GB

- regular DRAM backed up by Flash
- total capacity: 8 GiByte
- form factor: DDR4 SDIM
- bus speed: 2133 MHz

aprx price, October 2016, ???
Next year?

Intel Optane - 3D XPoint NVDIMM

- in the pipe line
- total capacity: 512 GiByte

Increase capacity, performance and/or reliability

Redundant Array of Independent Disks (RAID)

- Multiple disks that can provide:
  - capacity: looks like a 20 TiByte disk but is actually 10 2TiByte disks
  - performance: spread a file across ten drives, read and write in parallel
  - reliability: write the same file to several disks, if one crashes - not a problem

the abstraction layer

Alternatives:

- The cabinet that holds the disks present itself as one drive.
- A device driver in the kernel knows that we have several disks but the kernel presents it as one disk to the application layer.
- The application layer knows that we have several disks but provides a API to other applications that looks a single drive.

RAID levels

- RAID 0: stripe files across several drives.
- RAID 1: keep a complete mirror copy of each file.
- RAID 2-6: spread a file plus parity information across several drives.
application layer, simple to understand

system calls: open, read, write, lseek ...

all devices have a generic API
device drivers that know what they are doing

now it’s a bit structured
I/O and memory buses, protocols such as SATA, SCSI, USB etc

hardware - a complete mess