Non-volatile storage

So far, we've taken a closer look at the primary memory. Now, it's time to discuss the most important kind of non-volatile memory, which is **mass storage** (including: hard drives, SSD drives, but also external memory carriers such as CD/DVD discs or USB memory). Although theoretically a computer can operate without any memory of such type (e.g. in the mode "receive the program to be executed from the network and print the output on a printer"), this is very rare in practice. A typical (personal or office) computer is usually equipped with a built-in mass storage, popularly called a **drive** (or a *disk*), which should be regarded as one of the basic computer components.

Storage drives

Below, we'll discuss two main kinds of such large devices: the HDD and SSD drives.

To begin with: a note on the language

Before getting into the details, let us make a language-related digression: there is some confusion about the phrases *drive*, *disk*, *hard drive* and *hard disk*. Although they differ in meaning, in common language all of them happen to be used interchangeably.

A **drive** is defined by the Cambridge Dictionary as "a *device* for storing computer information", while a **disk** refers to the proper data container, assuming (at least in theory) that it's flat and round. That's why, back in the 90's, we would insert a *floppy disk* (a data container) into a *floppy (disk) drive* (a device), or store our files on a *hard disk* (a metal plate) inside a *hard disk drive* (a device for storing data there). At that time, however, "*hard disk drive*" — or any sub-phrase of that — became commonly treated as a synonym of "large secondary memory", just because the latter was rarely implemented in any other way.

Later, when the *solid-state drives* built on flash memory appeared, they kept being commonly called *disks*, despite not containing anything of that shape. Years are passing — but still, whenever abstractly speaking about the secondary memory layer in the computer architecture, we may often hear it called "the *disk*" or "the *hard drive*", even though the actual implementation may be SSD-based (and then, strictly speaking, neither *hard* nor a *disk*).

To add another bit of chaos, the optical data containers have been traditionally described with the British orthography: we'll more often see a *DVD disc* than a *DVD disk*.

Hard drives

The **hard disk drives** (**HDD**, also known as *hard drives* or *hard disks*) are a technology of long history (the first such drive was manufactured by IBM in 1956). They're based on an even older idea of **magnetic storage**, in which information is represented in the local magnetic field of the data container. In every cell, we can create a permanent magnetic field in one of two possible ways (directed in one direction or another), which allows encoding binary digits. (However, the actual encoding is a bit more complex than "encode every bit in the magnetic orientation" — instead, it enforces frequent changes of the orientation, so that any larger area of the disk observes a negligible total magnetic field). In order to magnetize the cells, a head element is used that is capable of creating a targeted (and frequently changing) electromagnetic field. The head also allows testing magnetization of the disk, and hence reading the data.

Physical structure

Due to the necessity of placing the moving head above an arbitrary cell of the data storing material, the shape of that material becomes essential. In the HDD drives, these are flat and round carriers, called **platters** (and originally *hard disks*). A single HDD drive may consist of several such platters: typically 1 or 2 in laptops, but up to 4 in desktop computers. The platters keep rotating, typically with a speed of 7,200 rpm (*revolutions per minute*), i.e. 120 revolutions per second, though there still exist cheaper disks with 5,400 rpm. Consecutive data are stored on along concentric arc paths, called **tracks**.

The following figures show a real look of an HDD drive, and a diagram of its most important components:

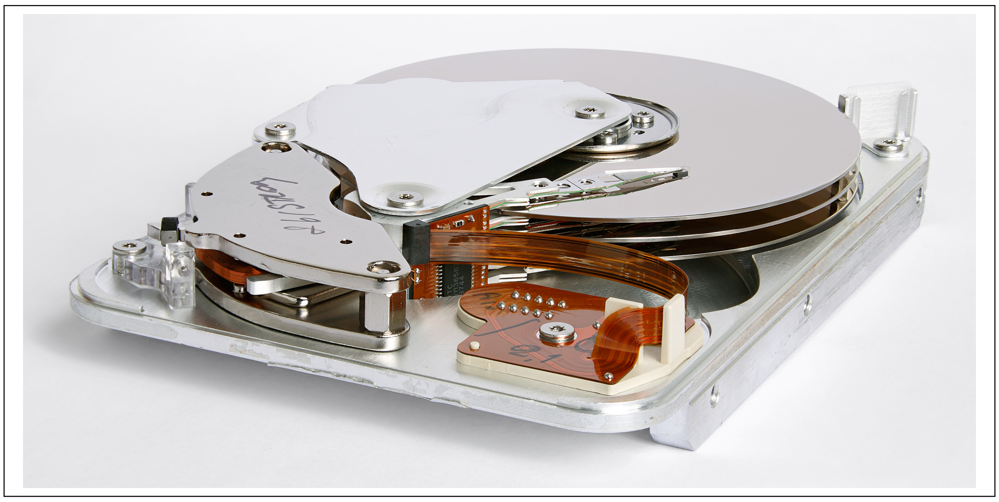


Figure 1. An internal view of a HDD drive from the late 1990's.
(Source: `http://en.wikipedia.org`)

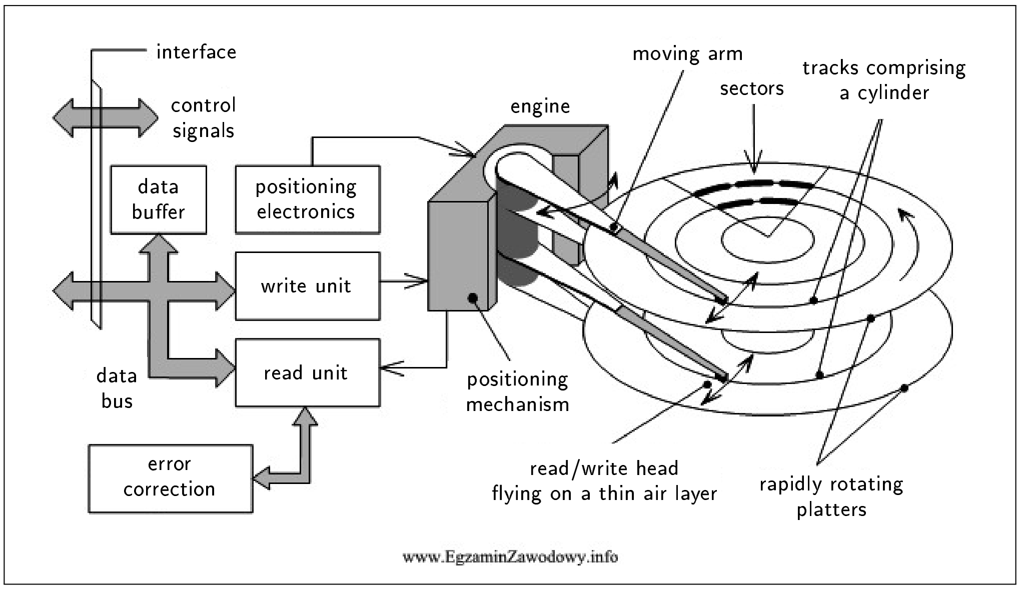


Figure 2. A diagram of a HDD drive structure and its communication with the computer.

The capacity of a hard drive depends on the number of tracks and on the density of information on a single track. The main obstacles for increasing that density are magnetic effects: if single cells are too small and placed too close to each other, they can sometimes flip their magnetic orientation to the opposite one — under the influence of neighbours, or even just by themselves. Until the 2000's, this forced the industry to use the so-called **longitudal recording** (in which the magnetic poles were directed parallelly to the platter surface), which in turn restricted HDD capacity to ca. 1 TB. Currently — thanks to several technological novelties (including a new magnetic material, and a new type of write head) we can employ **perpendicular recording**, which means smaller cells and hence greater capacity.

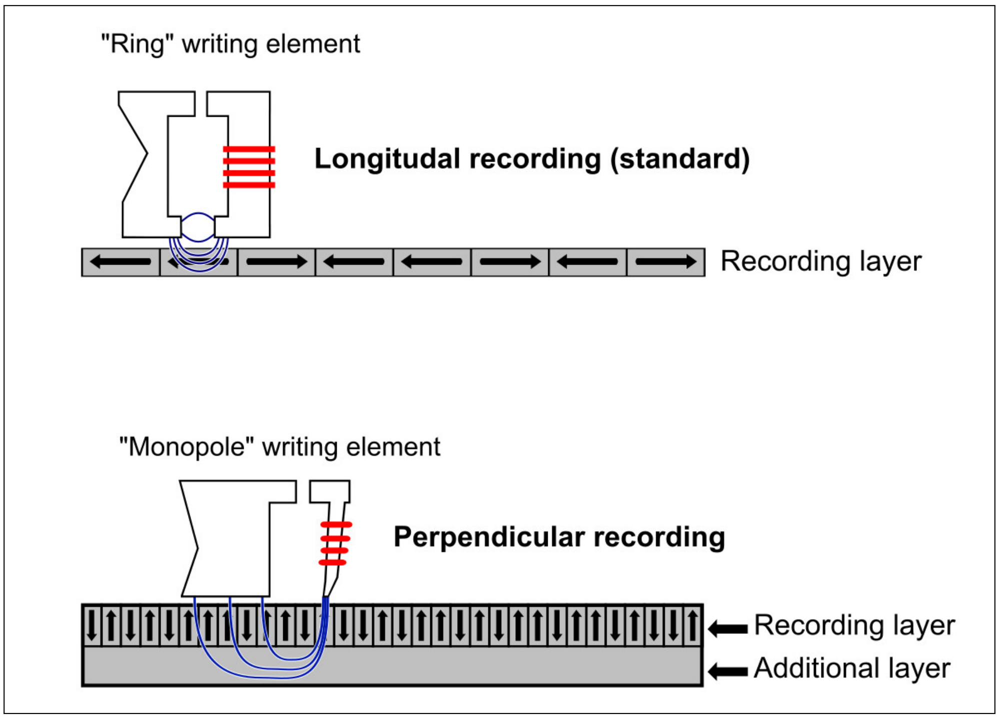


Figure 3. Porównanie zapisu równoległego (u góry) i prostopadłego (u dołu). (Source: `http://en.wikipedia.org`)

One of the technologies that give hopes for further overcoming the abovementioned barriers is *thermally assisted recroding*, in which the platter surface is temporarily heated up by a laser beam. In 2021, this method allowed Seagate to produce a hard drive of 20 TB capacity — however, it is not yet used commercially. Another promising method (but not yet deployed to the market) is *bit-pattern media*, whose aim is — somewhat roughly — to achieve a decrease of memory cell sizes by ensuring their more regular shapes.

Organization of the data

Due to the structure of the disk (accessing the data with read/write heads placed on a single moving arm), the physical placement of the data becomes important for its access time. The most efficient access will happen when all the relevant data is placed together: ideally on a single track, or at least on a set of nearby tracks (to minimize the arm movement). Also, to minimize the number of read/write operations, data are always read or written in chunks. The smallest such chunk, corresponding to a fragment of a track, is called a **sector**; modern computers use sectors of size 512 B or 4 kiB. Every sector also stores additional metadata for controlling the disk operation: these include information on the deviation of the rotational speed, as well as a checksum which allows detecting (and sometimes even fixing) cases of single bit corruption.

The problem of placing related data (e.g. single files) in nearby disk sectors is on one hand crucial for the efficiency of the whole computer, but on the other hand not trivial as the existing files are constantly modified (and, in particular, can keep growing). This can lead to disk **fragmentation**, which can mean either dispersing files throughout distant disk sectors (leading to slower read and write operations) or appearance of unused sectors which are not worth filling with small portions of files (which in turn decreases the effective capacity of a disk). Managing the placement of data on the disk and fighting the fragmentation issue is a responsibility of the software layer — and, specifically, is done by a particular **file system** used by the operating system — which we will describe in more detail below.

SSD drives

As we mentioned before, HDD drives are one of the two main types of internal mass storage that is currently broadly used. The other one are **solid-state drives** (known as *SSD drives*), based on the semiconductor **flash** memory.

The flash memory, proposed in 1980 and manufactured industrially since 1989, is built of special **floating-gate transistors** (*FGT*), equipped with an additional "floating gate" which is surrounded by an isolated material which allows charging it with electric change, with a *long-lasting effect.* (That's a substantial difference in comparison with flip-flops discussed on previous lectures, in which holding any information requires a constant electric powering). This makes FGT transistors suitable as non-volatile memory cells. Still, it's worth realizing that building mass storage out FGTs involves multiple technical complications — for example, in the early versions of this approach, writing to memory was only possible after a prior erasure (called a *flash* operation; hence the name of this technology). Later developments led to the state which we know today, in which SSD drives offer (from the user's perspective) almost unrestricted read and write capabilities.

In terms of the internal structure, SSD drives resemble the already discussed ROM memory dies; in particular, they contain no rotating elements. This allows significantly higher data access times in comparison with the HDD technology. Also, SSD drives are more resistant against heat, physical shocks etc. Their most significant disadvantage is the price, which remains several times higher than for HDD drives.

The SSD drives in home computers typically employ the *multi-level cell* (**MLC**) technology, in which a single memory cell can take more than 2 states: we no longer classify the voltage as just high or low, but instead distinguish between more levels of voltage. These will be typically 4 levels, though it can be also 8 or 16. This allows a single memory cell to store respectively 2, 3 or even 4 bits of information. (Works are in progress to increase that number to 5). Reducing SSD prices to a widely acceptable level was achievable only thanks to using MLC. The disadvantage of this technique is an increased chance for bit errors, and reduction of the lifetime of the drive. Therefore, MLC is not used in BIOS memory (which we discussed in the lecture "Motherboard").

External drives

Personal computer users also widely employ **external drives** — which are made with the HDD or SSD technology just as the drives discussed above, but stand out in that they're placed outside the computer case, and can be plugged into it as needed, typically via USB ports. Their obvious advantage is an increase of the total available secondary memory; another clear gain is portability.

Such drives can be also used as data backups for the internal HDD or SSD drives (as both these technologies involve a risk of data corruption or loss) — here, though, one needs to be careful about flash memory: even though essentially labeled as non-volatile, it can lose its content (due to discharging of its internal capacitors) after a few years of not being used.

Other storage means

Let us briefly mention some other input-output devices purposed for storing user data:

- **USB memory** — devices known as *USB sticks* or *pen-drives* (though the latter name is registered by a Taiwanese company Add On Technology, so it officially applies only to their products) — these store data in the semiconductor flash memory, just as SSD drives. The difference here is that they're designed to be external, small and portable. As a result, they're significantly slower (the limit set by the standard is 4.6 Gb/s, though in practice it's usually far from being achieved). Also, they're more error-prone and more expensive (per byte) than SSD drives.
- **SD memory cards** — another type of semiconductor flash memory. Used mainly in photo and video cameras, or in mobile phones.
- **Optical memory** (*CD / DVD / Blu-ray discs*) — here reading and writing is done with a laser beam. The data carrier is a surface (typically of a disk shape) covered with a material whose transparency can be easily changed, thus introducing two types of points (reflecting and non-reflecting) which can represent bit values. Similarly as in HDD drives, reading and writing data employs disk rotation to help in positioning the read/write head above the appropriate sector. The data-carrying materials can support just a single write or multiple writes; they can also differ by total capacity:
  - For the CD standard, this will be typically ca. 1 GB.
  - For DVD, the typical capacity is 4.7 GB; by using double-layered and/or double-sided discs, we can reach up to 17.1 GB, though DVDs of such a large size are rarely used.
  - In the Blu-ray technology, the typical capacity of a disc is 25 GB. Here, even more than two layers can be supported, though it's also a rare case. The increase of information density in this technology is achieved by using a laser beam with a shorter wavelength (a blue light).

Drive interfaces

Besides the manufacturing technology, an important factor for the practical efficiency of a storage drive — and its usefulness in combination with a particular computer — is the way in which it is connected, that is: the type of the bus and the plug used.

Bus types

Currently, in home applications, three main types of buses are used to connect storage drives to the processor:

- **SATA** (*Serial ATA*, where *ATA* stands for *Advanced Technology Attachment*) — practically the only choice for HDD drives; also used for many SSD models;
- **PCIe** (*PCI Express*, where *PCI* stands for *Peripheral Component Interconnect*) — a standard which we already met in the lecture "Motherboard" (due to its role in connecting the CPU to the RAM memory); used also for faster SSD models;

- **USB** (*Universal Serial Bus*) — used mostly for attaching external drives.

The SATA bus is derived from an older ATA model, designed specifically for communication with hard drives. For example, its cooperation with the drive controller enables the so-called *native command queuing*, which allows altering the order of executed read and write orders to minimize their total execution time. (Recall that this time depends on the physical placement of data on the disk, so it's beneficial to rearrange the commands so that those involving neighbouring sectors are executed next to each other). This makes SATA a proper choice for HDD drives; this in turn makes it beneficial for SSD manufacturers to support that standard as well, as it allows the end users to replace an older HDD with a newer SSD in a typical computer.

On the other hand, limitations for transmission speed in SATA standard (3 Gb/s in SATA 2 and 6 Gb/s in SATA 3) cause it to be too slow for the fastest currently available SSD drives. Therefore, full utilization of such devices requires connecting them to a faster PCIe bus (where the analogous limit in the fastest version is nearly 1 Tb/s as of 2022, and doubles every few years).

USB stands out from the other two standards in that it's been designed from the start for external devices (and hence we will find USB ports on the outside of a computer case).

It's also worth mentioning here that all the three standards involve, besides transmitting data, supplying the connected devices with an appropriate amount of electric power.

Communication mode

All the above bus types are described as **serial**, which *essentially* means that transmission happens sequentially, bit by bit.

It could seem that a relatively easy and effective idea for increasing the transmission rate could be switching this approach to a **parallel** one, in which multiple bits would be sent at once (or more precisely: in each bus cycle, 1 bit would be sent along every wire). In practice, however, since around 2000 we observe transitioning from older parallel solutions (PCI, ATA) to serial ones (USB, SATA, PCIe). This has happened because mutual interference between the wires, as well as slight differences in transmission times along them, complicate synchronization on the receiving side so much that it led to essential limitations on the frequency of cycles in parallel buses. By migrating to a serial transmission, we achieved significantly higher frequencies.

On the other hand, in a contrast to the word "serial", modern buses also consist of several wires.

In the SATA standard — which is simplest in this regard — we have 7 wires, out of which 3 transmit no data (serving just as isolation layers for the other ones), and the remaining 4 wires are split into 2 pairs, each responsible for transmission in one direction. In each pair, one of the wires is a strict "negation" the other one, always transmitting the opposite bit values. (In exchange for such "waste" of bandwidth, we gain: a more clear signal interpretation on the receiving side, a chance to detect errors in communication, as well as a constant level of the electric charge on both sides of the bus).

The PCIe buses, as already mentioned in the lecture "Motherboard", go much further, introducing **multiple lanes**, each of which allows transmitting 1 bit in each direction. (In practice, each such lane consists of 4 wires, organized into two negation pairs, similarly as in SATA). In version 6.0, using 16 such lanes of transmission rate 60 Gb/s each allows achieving nearly 1 Tb/s in total. (A similar solution — introducing 2 lanes of communication — has been also introduced in the newest versions of USB).

If so, one could ask: why is PCIe still called *serial*? The reason, at least partially, is that each PCIe lane operates as an independent serial connection, with its own timing for receiving subsequent bits, without synchronization known from parallel buses. (Instead, if needed, PCIe controllers perform buffering of data incoming via various lanes, and join them into complete data afterwards. This means that, to ensure no exhaustion of those buffers, the protocol must sill involve some *rough* synchronization between the lanes, by means of dedicated control messages — that technique, however, has a much smaller impact on the overall transmission rate).

Transmission tax

Thanks to applying the serial approach, the modern buses do not require providing a separate clock signal: the timing of arriving bits is *essentially* known upfront, and potential divergences can be detected by just observing the data stream. However, such solution would cause troubles in case of a long series of bits of same value (e.g. having received 1000 "1"s in a row, we wouldn't have certainty if, by some chance, there wasn't one more or less of them). To address that, all buses discussed here employ a kind of **encoding** which guarantees periodic changes of the stream content.

The oldest variant of that, used in SATA and older version of PCIe and USB, is the **8b/10b encoding**, transforming sequences of 8 bits into 10-bit representations in such a way that the resulting stream of bits can never contain more than 5 identical values in a row. The price to be paid, however, is "wasting" as much as 20% of the total throughput. In case of SATA, the typically described maximal rate of 6 Gb/s does not take that effect into account; therefore, the actual maximal throughput of the interesting data is smaller by 20%).

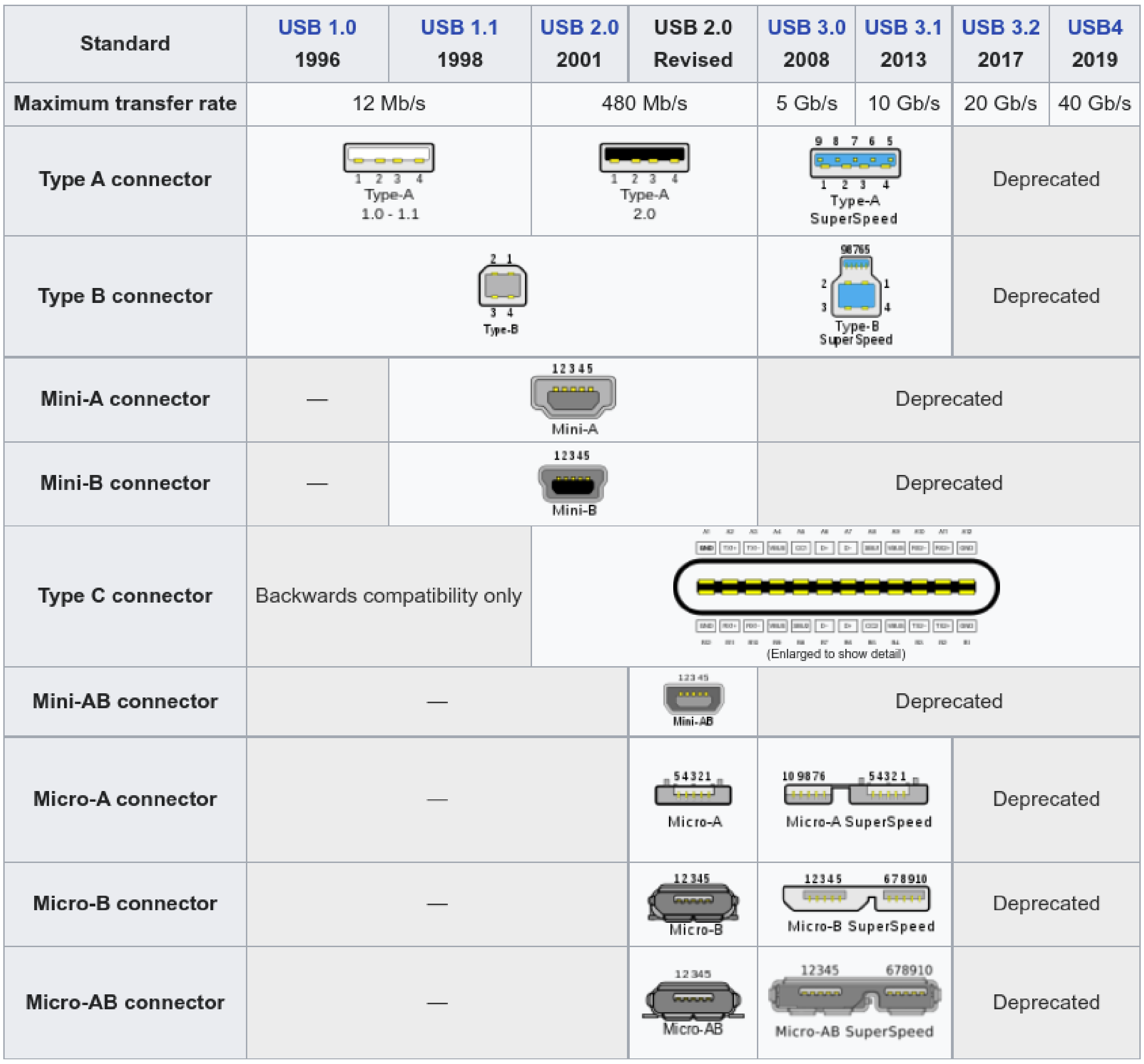
An additional benefit from the 8b/10b encoding is that it acts as a **checksum**, allowing to detect a case when a single bit has been corrupted.

In newer PCIe and USB buses, we can meet newer versions of the above encoding, with a lower transmission tax, e.g. 128b/130b — using just 2 bits per every 130 bits, though requiring a better control of time on the controller side. In the PCIe 6.0 standard, yet another one encoding is used: 242B/256B — that one comes again with a higher tax, but in exchange it contains an **error correction code**, that is, a more developed system of checksums which allows not only detecting that a bit has been corrupted but also fixing problems of such kind.

Compatibility

All the abovementioned types of buses have evolved over years, introducing new versions with always growing transmission rate (altogether — measuring in Gb/s — we saw an increase from 1.5 to 6 between SATA 1 and SATA 3, from 2 to 970 between PCIe 1.0x1 and PCIe 6.0x16, and from 0.0015 to 20 between USB 1.0 and USB 3.2). The multitude of changes would lead to a nightmare for the end users, if the standards did not apply **backward compatibility** between consecutive versions. For example, a USB 2.0 device can be attached to USB 3.0 port; also conversely, a USB 3.0 can be plugged into a USB 2.0 port; in both these cases, the protocol will lead to agreeing on the older of the two versions (in our examples — 2.0), resulting in communication with acoordingly lower efficiency (yet, importantly, the connection will still work). An analogous compatibility will be usually observed between PCIe version, as well as between SATA versions. There are some exceptions, e.g. SATA 2 and SATA 3 are not fully compatible with SATA 1 (though the latter is now rarely used).

Simultaneously, the corresponding standards of **connectors** have been also evolving, with the general direction of decreasing size, but also introducing necessary adjustments for cooperating with the newer bus standards. For example, the broadly known USB-A ports were soon accompanied by so called micro USB (or more strictly: USB micro-B) plugs, which due to their smaller size became widespread in mobile devices, and then by USB-C (which, besides the new centre-symmetric and hence more convenient plug shape, also introduced additional pins that unlocked larger throughput for both data and power).



Standard	USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 2.0 Revised	USB 3.0 2008	USB 3.1 2013	USB 3.2 2017	USB4 2019
Maximum transfer rate	12 Mb/s		480 Mb/s		5 Gb/s	10 Gb/s	20 Gb/s	40 Gb/s
Type A connector	Type-A 1.0 - 1.1		Type-A 2.0		Type-A SuperSpeed		Deprecated	
Type B connector	Type-B				Type-B SuperSpeed		Deprecated	
Mini-A connector	—	Mini-A			Deprecated			
Mini-B connector	—	Mini-B			Deprecated			
Type C connector	Backwards compatibility only		(Enlarged to show detail)					
Mini-AB connector	—			Mini-AB	Deprecated			
Micro-A connector	—			Micro-A	Micro-A SuperSpeed		Deprecated	
Micro-B connector	—			Micro-B	Micro-B SuperSpeed		Deprecated	
Micro-AB connector	—			Micro-AB	Micro-AB SuperSpeed		Deprecated	

Figure 4. Evolution of connectors for the USB buses.
(Source: `http://en.wikipedia.org`)

Last, let us mention that the evolution of connectors can also lead to increasing compatibility, even between different bus standards. While initially SATA and PCIe were simply terminated with SATA and PCIe ports, differing in both shape and size, since 2013 there is an **M.2** standard, specifying the shape of the port (and the corresponding connector in a storage drive) together with a controller allowing communication according to any of the SATA / PCIe / USB protocols — depending on the requirements of the connected device, and potential limitations of the computer motherboard. However, while M.2 can "fit" the whole SATA 3 protocol (thus allowing its top throughput of 6 Gb/s), in case of PCIe it can only handle up to 4 transmission lanes (so for a bus of x16 type, we can only get 25% of the maximal bus throughput). Therefore, while (from the end user's viewpoint) M.2 drives can surpass the SATA barrier of efficiency, they are still behind the drives with the PCIe interface.

Logical organization of the data

From the physical viewpoint, the memory cells form just a sequence of bits (0 and 1 values). However, in case of storage drives, we also want to store them (and then interpret correctly) not just the proper data, but also: the structure of files and folders containing them, metadata of these objects, and finally additional data structures of the operating system. This means that we need an appropriate format of storing information on the drive.

File systems

Organizing data on a storage drive (e.g. choosing the location for saving new data, or encoding the relations between files and folders) is a reponsibility of the operating system, which does this following a specific set of rules, called a **file system**.

For example, for HDD drives, the Windows and Linux systems employ by default, respectively, the **NTFS** and **ext4** file systems. Until a few years ago, this meant that a HDD formatted (with default settings) and modified under Linux became non-readable under Windows; fortunately, the newest versions of Windows introduced support for ext4 (while Linux has been for long supporting NTFS, at least in read-only mode).

A disadvantage of NTFS is the relatively high fragmentation of data, which means that logically related data (e.g. from a single file) can be placed in a distant locations on the disk. As we already mentioned, that is a suboptimal situation, as it increases the total time spent on accessing a file. To address this problem, Windows offers a procedure called **defragmenting** (also called *optimizing* or in short *defragging*), which restores some order of the stored data. In the older versions of Windows, that procedure could be triggered only manually, due to how engaging it was for overall PC efficiency. Newer versions introduced an option for automatic defragmenting, though the process can be still cumbersome.

As for SSD drives, one can either use one of the leading file systems mentioned above, or choose among new file systems dedicated for SSD drives: examples include exFAT in Windows or F2FS in Linux.

Partitions

Partitions are explicit sections of the drive storage space, which — from the viewpoint of the operating system and all the running programs — are fully indepedent storage units, just as if they were physically separate drives. (For example, whenever on Windows we see several "drives" — C:, D:, E: etc. — it may happen that these are subsequent partitions of a single physical drive, just acting as different *logical drives*). In particular, each partition may be accessed with another file system.

If a partition hosts an operating system, its initial sectors are reserved for the boot loader (as we already described on the lecture "Motherboard"). Similarly, if the partition has been formatted in the NTFS file system, then NTFS reserves its initial part for a file called MFT (*Master File Table*), containing metadata of every file (e.g.: its name, location on the drive, size, modification date, and attributes — such as "read-only" or "hidden"). MFT also contains a bitmap informing about free and occupied clusters on the drive. To avoid fragmentation of the MFT file, the file system reserves for it upfront a larger area (called *MFT zone*), in which it avoids placing other files. To be able to recover from a corruption of MFT content, the file system also keeps its copy (called *MFT mirror*) on the drive.

Encryption

File systems also offer an option of **encryption**, which means that the drive stores an encrypted version of the actual information. Thanks to that, an unpriviledged person will not be able to view the file contents on the drive, even if they get into the possession of our computer, unmount the drive and plug it in elsewhere, or use an alternative operating system.

A disadvantage of encryption is some slowdown of computer operation. It also shouldn't be expected that encryption will ensure "full safety". For example, even if file contents are encrypted, the structure of and names of files and folders typically are not.

It is also possible to encrypt a drive on the hardware level. One of methods for this is to use the TPM (*Trusted Platform Module*)[1], which started appearing in computers in the last decade. Every instance of TPM contains unique encryption and decryption keys, and the whole system is designed to prevent retrieving these keys from the device, as well as to prevent removing TPM from the motherboard (without destroying the hardware). Encrypting the whole drive with TPM can be done from the level of BIOS / UEFI. In this case, the booting process involves a need to decrypt the boot loader, which requires the user to provide a password and initiate storage decryption even before the main operating system has been started up. To make this happen, another operating system is introduced — a smaller one, with substantially limited functionality, and focused around security. An example of such system is BitLocker.

Generally, hardware-level encryption will be typically faster and more secure than software solutions. On the other hand, it may prevent the user from accessing their data in case when the computer is somehow damaged.

[1]Microsoft has placed TPM 2.0 among hardware requirements for MS Windows 11.