As you may already know, it is convenient to store data in files. In this topic, we'll figure out what else we can do with the files, how to distinguish them, and what their main types are.

Below we provide a list of file types in the Unix system. In other operating systems, the types may slightly differ, but the basic ones, which we will talk about in more detail, can be found almost everywhere.

**Unix file types**

In general, a file is a container for some information. We can put almost anything in this container: a photo, a text, a link to another file, or just another container with other nested files, etc. All this data is different and therefore the files in which it's stored are of various types.

The Unix filesystem components are:

* **Regular files** where you usually keep your personal data such as texts, pictures, and so on.
* **Directories***,*or folders that make it easier to organize other files in the system.
* **Named pipes***,*that pass the output of one process as input to another one and have a name.
* **Symbolic links***,*which contain links to other files.
* **Device files***,*that contain data required by the operating system to interact with physical devices.
* **Sockets***,*which allow to exchange the data between processes in both directions.

We will study the most commonly used file types in more detail. These are the ones that can be found in every operating system: regular files, directories, and symboliclinks.

**Regular Files**



Regular files are common because users usually store their data in them: documents, videos, photos, music, etc. In general, any data in such files can be presented in two forms:

* textual data with some encoding, then the file will be called a **text file**
* any other sequences of bytes without constraints to encoding, then such a file is called a **binary file**

To make it easier to figure out which file is text and which is binary, let's study a couple of examples. Text files usually contain texts in plain format, tabular data, configuration files, and data formats like CSV or JSON. Binary files often contain data such as video, audio, databases, and archives.

Binary and text files have different characteristics and thus require different tools to work with them. It is worth knowing these differences because it will definitely save you time when reading or writing your data to these files.

The main difference between text and binary files is that the latter has no inherent constraints. It means they can contain any bytes sequence, and they are to be opened in an appropriate program that knows the specific file format such as Media Player, Photoshop, Office, etc. On the other hand, text files can be edited in any text editor program. Also, they must correspond to several constraints such as human-readable content, line-oriented data format, universal reading of newline sequences, and so on.

**Directory**



Directories, which sometimes are also called **folders**, are containers for the files. For example, you can put your music files into one directory titled “*Music*”. Most file systems also allow a directory to contain other directories. It’s called a **parent** directory containing **child**directories or **subdirectories**. This way the organizing data is stored on a medium into tree-like hierarchical structures, where directories that have no parents are called **root** directories and serve as the base of this structure. This hierarchy provides clear links between files and makes it a lot more convenient to search for data on a disk. We just need to know the full path to a file, that's all. To get this full path, we need a full filename, i.e. we add the parent folder to the file, and for the parent folder, we also add its parent and so on till we get to the root.

Directory names in a full filename are usually separated by slash / or backslash \. So if there is a root directory named “*root\_directory*” that contains a subdirectory named “*sub\_directory*”, and in this subdirectory, there is a file named “*my\_file*”, the file system will assign it a full filename like “*root\_directory/sub\_directory/my\_file*”.

Moreover, tree-like hierarchical structures allow us to select file groups and manage them. Also, using a hierarchically organized structure, we can transfer it to another computer.

**Symbolic links**

Lastly, there is one more special sort of file worth considering. These files are called **symbolic links**in the Unix system. If you use Windows, you may know them as *shortcuts*. A symbolic link contains a reference to another file or directory in the form of an absolute or relative path. Let’s say you want some files to be accessed from several directories. If you copy this file and put its copies into each directory, each time you modify one of the copies you’ll have to go and modify all the others, and this feels like a waste of time. If you put symbolic links referencing this file into each directory, they would actually open the original file they are referencing.

**Conclusion**

To sum up, in this topic we've learned that:

* personal data is stored in regular files,
* the data in regular files can be either in binary or text form,
* in addition to regular files, there are also directories, symbolic links, sockets, named pipes, and device files,
* directories are containers for other files,
* you can store links to files in special files called symbolic links.

Now, let's turn to exercises to see how well you understand the different file types.

Match different types of files with their respective descriptions.

 **Regular file**  
 → *Can be of 2 types: text or binary file*

 **Symbolic link**  
 → *Is a file containing a pointer to another file or directory*

 **Directory**  
 → *Can contain many other files and directories*

 **Named pipe (FIFO)**  
 → *Passes the output of one process as input to another one and has a name*

 **Socket**  
 → *Allows the exchange of data between processes in both directions*

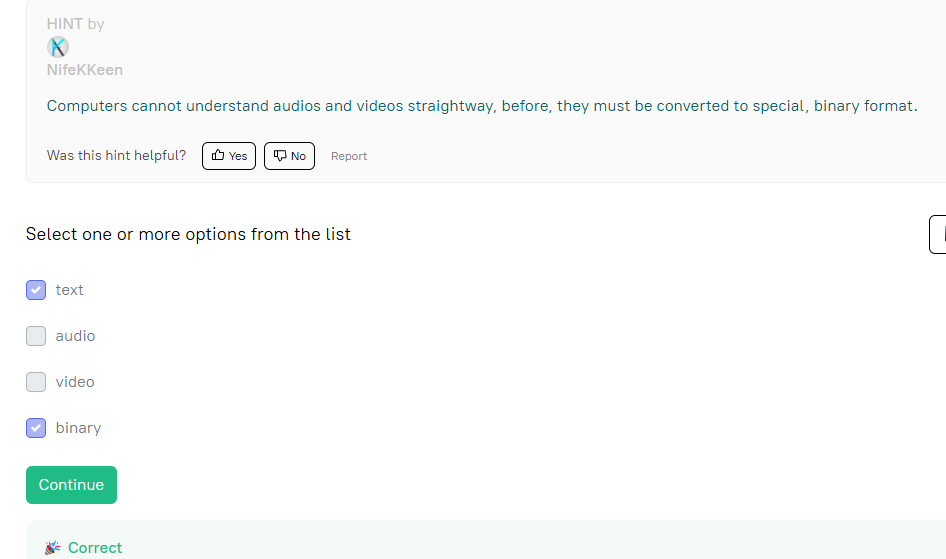
 **Device file**  
 → *File which defines the device in file system*

**File access**

Mary has a datatable.csv file with some calculations. She keeps this file in /my\_project directory and makes some changes to it from time to time. But she also wants the file to be accessed from the root directory with all the changes she already made in the first directory. What should Mary do?

create a symbolic link

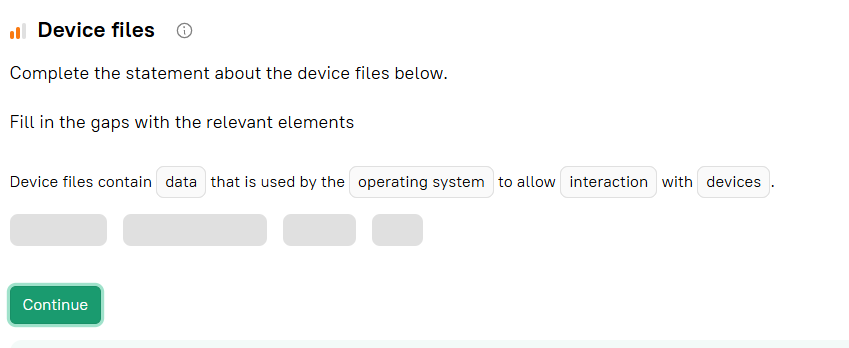
In what form can data be presented in regular files?



**Symbolic links**

Why are symbolic links needed?

 You can store links to files in symbolic links. They make it possible to access a file from multiple locations without making copies of it.



Lina wants to write the full path to her work\_report.docx file, which is in the /work directory. The /work directory is contained in the root folder. Which of the following path variants is the correct one?

root/work/work\_report.docx

What do CSV tables, directories, hard drives, keyboards, printers, some inter-process and network communications in Unix systems?

They are represented as files.

The directory that has no parents is called a root directory

**Keen eye**

Look at the Unix terminal commands below and try to guess what types of files are present.

mkdir my\_documents

touch my\_documents/notes.txt my\_documents/table.csv

ln -s /home/me/TODO.txt todo

### Command Analysis:

1. **mkdir my\_documents**  
   → Creates a **directory** named my\_documents.
2. **touch my\_documents/notes.txt my\_documents/table.csv**  
   → Creates two empty **regular files**: notes.txt and table.csv.
3. **ln -s /home/me/TODO.txt todo**  
   → Creates a **symbolic link** named todo pointing to TODO.txt.

### ✅ Correct answers from the list:

* ✅ **regular files**
* ✅ **directory**
* ✅ **symbolic links**

Why do we need file hierarchy?

it helps organize the data

**Binary file content**

Which data formats are usually stored in binary files?

The following are usually stored in **binary files**:

* ✅ **video**
* ✅ **archives**
* ✅ **audio**

**Not binary**:

* ❌ **plain text** → Stored in text files
* ❌ **JSON** → Also stored in text files (human-readable)