Overview

This section deals with the filesystem library in C++17. We'll be looking at the new features of the filesystem one by one.

WE'LL COVER THE FOLLOWING

Filesystem Overview

While the Standard Library lacked some important features, you could always use **Boost** with its dozens of sub-libraries and do the work. The C++ Committee decided that the **Boost** libraries are so important that some of the systems were merged into the **Standard**. For example, smart pointers (although improved with the move semantics in C++11), regular expressions, `std::optional`, `std::any` and much more.

A similar story happened with `std::filesystem`.

The filesystem library is modelled directly from **Boost** filesystem, which has been available since 2003 (with the version 1.30). In C++ implementation, the committee also extended the component with non-POSIX systems. The library was firstly available as TS (Technical Specification) and later, after a long time of improvements and feedback, merged into the **C++17** Standard.

The library is located in the `<filesystem>` header, and it uses namespace `std::filesystem`.

Core Parts of The Library

The filesystem library is a rather significant part of the Standard Library. It defines many types with dozens of methods, and also gives us many free functions.

Below we can define the core elements of this module:

- The `std::filesystem::path` object - that allows you to manipulate paths that represent existing or not existing files and directories in the system.
- `std::filesystem::directory_entry` - represents an existing path with additional status information like last write time, file size, or other attributes.
- Directory iterators allow you to iterate through a given directory. The library provides a recursive and non-recursive version.
- Many supporting functions like getting information about the path, file manipulation, permissions, creating directories, and many more.

In the next section, you'll see a demo of all the parts that compose `std::filesystem`.