File Permissions

In the last lesson you must have noticed functions related to file permissions. This lesson will talk about them.

WE'LL COVER THE FOLLOWING

Permission Functions

We have two major functions related to file permissions:

- `std::filesystem::status()` and
- `std::filesystem::permissions()`

The first one returns `file_status` which contains information about the file type and also its permissions.

And you can use the second function to modify the file permissions. For example, to change a file to be read-only.

std::filesystem::perms

File permissions - `std::filesystem::perms` - it's an enum class that represents the following values:

Name	Value (octal)	POSIX macro	Notes
none	0000		There are no

			permissions set for the file
owner_read	0400	`S_IRUSR`	Read permission, owner
owner_write	0200	`S_IWUSR`	Write permission, owner
owner_exec	0100	`S_IXUSR`	Execute/search permission, owner
owner_all	0700	`S_IRWXU`	Read, write, execute/search for owner
group_read	0040	`S_IRGRP`	Read permission, group
group_write	0020	`S_IWGRP`	Write permission, group
group_exec	0010	`S_IXGRP`	Execute/search permission, group
group_all	0070	`S_IRWXG`	Read, write, execute/search by group
others_read	0004	`S_IROTH`	Read permission,

			others
others_write	0002	`S_IWOTH`	Write permission, others
others_exec	0001	`S_IXOTH`	Execute/search permission, others
others_all	0007	`S_IRWXO`	Read, write, execute/search for others
all	0777		`owner_all | group_all | others_all`
set_uid	04000	`S_ISUID`	Set-user-ID on execution
set_gid	02000	`S_ISGID`	Set-group-ID on execution
sticky_bit	01000	`S_ISVTX`	Operating system dependent
mask	07777		`all | set_uid | set_gid | sticky_bit`
unknown	0xFFFF		The permissions are not known

Demo

Demo

Here’s a short code that demonstrates how to print file permissions:

```
#include <filesystem>
#include <fstream>
#include <iostream>
#include <string>

namespace fs = std::filesystem;

std::ostream& operator<< (std::ostream& stream, fs::perms p) {
    stream << "owner: "
       << ((p & fs::perms::owner_read)  != fs::perms::none ? "r" : "-")
       << ((p & fs::perms::owner_write) != fs::perms::none ? "w" : "-")
       << ((p & fs::perms::owner_exec)  != fs::perms::none ? "x" : "-");
    stream << " group: "
       << ((p & fs::perms::group_read)  != fs::perms::none ? "r" : "-")
       << ((p & fs::perms::group_write) != fs::perms::none ? "w" : "-")
       << ((p & fs::perms::group_exec)  != fs::perms::none ? "x" : "-");
    stream << " others: "
       << ((p & fs::perms::others_read)  != fs::perms::none ? "r" : "-")
       << ((p & fs::perms::others_write) != fs::perms::none ? "w" : "-")
       << ((p & fs::perms::others_exec)  != fs::perms::none ? "x" : "-");
    return stream;
}

int main(int argc, char* argv[]) {
    const std::string sTempName { "hello.txt" };
    {
        std::ofstream sample(sTempName);
        sample << "Hello World!\n";
    }

    try {
        std::cout << "after creation: " << fs::status(sTempName).permissions() << '\n';
        fs::permissions(sTempName, fs::perms::owner_read, fs::perm_options::remove);
        std::cout << "after change: " << fs::status(sTempName).permissions() << '\n';

        if (fs::remove(sTempName))
            std::cout << "temp file removed...\n";
    }
    catch (const fs::filesystem_error& err) {
        std::cerr << "filesystem error! " << err.what() << '\n';
    }
    catch (const std::exception& ex) {
        std::cerr << "general exception: " << ex.what() << '\n';
    }
    catch (...) {
        std::cerr << "general exception!\n";
    }
}
```

You can use the above `operator<<` implementation as follows:

You can use the above `operator<<` implementation as follows:

```
std::cout << "perms: " << fs::status("myFile.txt").permissions() << '\n';
```

Setting Permissions

To change the permissions you can use the following code:

```
std::cout << "after creation: " << fs::status(sTempName).permissions() <<
'\n';
fs::permissions(sTempName, fs::perms::owner_read, fs::perm_options::remove
);
std::cout << "after change: " << fs::status(sTempName).permissions() <<
'\n';
```

`std::filesystem::permissions` is a function that takes a path and then a flag and the “action” parameter.

`fs::perm_options` has three modes:

- `replace` - The permissions flag you pass will replace the existing state. It’s the default value for this parameter.
- `add` - The permission flag will be bitwise OR-ed with the existing state.
- `remove` - The permissions will be replaced by the bitwise AND of the negated argument and current permissions.
- `nofollow` - The permissions will be changed on the symlink itself, rather than on the file it resolves to.

For example:

```
// remove "owner_read"
fs::permissions(myPath, fs::perms::owner_read, fs::perm_options::remove);

// add "owner_read"
fs::permissions(myPath, fs::perms::owner_read, fs::perm_options::add);

// replace and set "owner_all":
fs::permissions(myPath, fs::perms::owner_all); // replace is default param
```

Note for Windows

Windows is not a POSIX system, and it doesn’t map POSIX file permissions to its scheme. For `std::filesystem` it only supports two modes: read-only and all its scheme. For `std::filesystem` it only supports two modes: read-only and all.

From Microsoft Docs filesystem documentation:

> The supported values are essentially “readonly” and all. For a readonly file, none of the *_write bits are set. Otherwise, the `all` bit (0777) is set.

Thus, unfortunately, you have limited options if you want to change file permissions on Windows.

Now let’s see if the library also offers any methods for error handling.