

File Permissions Solutions

File Status

- Briefly describe the status function in `std::filename`
 - `status()` takes a path as argument
 - It returns a `file_status` object
 - This can be used to find out whether the file is a regular file, a directory, or a special file
 - It can also be used to find the file's permissions
- Write a simple program which uses `status()` to display information about a file

Getting File Permissions

- What is meant by a file's "permissions"?
 - The permissions on a file determine which operations users are allowed to perform on the file (read, write and execute)
- How can we get the file's permissions using `std::filesystem`?
 - The `file_status` class has a member function which returns a permissions bitmask
 - Each bit corresponds to a different user/operation combination
 - We check the relevant bit for the user and operation we are interested in
- Write a simple program which displays the permissions of a file

Setting File Permissions

- How can we set the file's permissions using `std::filesystem`?
 - `std::filesystem::permissions` sets the permissions on a file
 - The first argument is the path of the file
 - The second argument is a permissions bitmask
- Write a simple program which sets the permissions of a file and displays the result

Setting File Permissions

- What effect does this have by default?
 - The permissions in the bitmask overwrite the existing permissions
- How can we change this behaviour?
 - We can pass a third argument
 - `perm_options::replace` overwrites the current permissions (the default)
 - `perm_options::add` sets all bits in the file that are set in the mask
 - `perm_options::remove` clears all bits in the file that are clear in the mask