std::filesystem::remove, std::filesystem::remove_all

Defined in header `<filesystem>`

`bool remove(const std::filesystem::path& p);`
`bool remove(const std::filesystem::path& p, std::error_code& ec) noexcept;`
	(1)	(since C++17)

`std::uintmax_t remove_all(const std::filesystem::path& p);`
`std::uintmax_t remove_all(const std::filesystem::path& p, std::error_code& ec);`
	(2)	(since C++17)

1) The file or empty directory identified by the path p is deleted as if by the POSIX remove (http://pubs.opengroup.org/onlinepubs/9699919799/functions/remove.html) . Symlinks are not followed (symlink is removed, not its target)

2) Deletes the contents of p (if it is a directory) and the contents of all its subdirectories, recursively, then deletes p itself as if by repeatedly applying the POSIX remove (http://pubs.opengroup.org/onlinepubs/9699919799/functions/remove.html) . Symlinks are not followed (symlink is removed, not its target)

Parameters

p	-	path to delete
ec	-	out-parameter for error reporting in the non-throwing overload

Return value

1) `true` if the file was deleted, `false` if it did not exist. The overload that takes `error_code&` argument returns `false` on errors.

2) Returns the number of files and directories that were deleted (which may be zero if p did not exist to begin with). The overload that takes `error_code&` argument returns `static_cast<std::uintmax_t>(-1)` on error.

Exceptions

The overload that does not take a `std::error_code&` parameter throws `filesystem_error` on underlying OS API errors, constructed with p as the first path argument and the OS error code as the error code argument. The overload taking a `std::error_code&` parameter sets it to the OS API error code if an OS API call fails, and executes `ec.clear()` if no errors occur. Any overload not marked `noexcept` may throw `std::bad_alloc` if memory allocation fails.

Notes

On POSIX systems, this function typically calls `unlink` and `rmdir` as needed, on Windows `RemoveDirectoryW` and `DeleteFileW`.

Defect reports

The following behavior-changing defect reports were applied retroactively to previously published C++ standards.

DR	Applied to	Behavior as published	Correct behavior
LWG 3014 (https://cplusplus.github.io/LWG/issue3014)	C++17	`error_code` overload of `remove_all` marked noexcept but can allocate memory	noexcept removed

Example

Run this code

```
#include <iostream>
#include <cstdint>
#include <filesystem>
namespace fs = std::filesystem;
int main()
{
    fs::path dir = fs::temp_directory_path();
    fs::create_directories(dir / "abcdef/example");
    std::uintmax_t n = fs::remove_all(dir / "abcdef");
    std::cout << "Deleted " << n << " files or directories\n";
}
```

Possible output:

```
Deleted 2 files or directories
```

See also

`remove`	erases a file (function)