How can I tell C++ that I want to discard a nodiscard value?

devblogs.microsoft.com/oldnewthing/20240329-00

March 29, 2024

Raymond Chen

C++ lets you add the `[[nodiscard]]` attribute to a function return value to indicate that the caller must use the result.

Given the declaration

```
[[nodiscard]] int important();
```

simply calling the function and allow the value to be discarded produces diagnostics.

```
void test()
{
    important();
}
```

```
clang: ignoring return value of 'int important()', declared with attribute 'nodiscard' [-
Wunused-result]

gcc: ignoring return value of 'int important()', declared with attribute 'nodiscard' [-
Wunused-result]

msvc: C4834: discarding return value of function with [[nodiscard]] attribute
```

Explicitly casting to `(void)` works:

```
void test()
{
    (void)important();
}
```

Note that this requires a C-style cast. You cannot `static_cast` or `reinterpret_cast` to `void`.

Another option is to store the result into a variable which is attributed as unused, and then allowing the variable to go out of scope immediately.

```
void test()
{
    { [[maybe_unused]] auto&& unused = important(); }
}
```

There is a proposal for C++26 to express the discard with `std::ignore`:

```
void test()
{
    std::ignore = important();
}
```

Although the ability to assign to `std::ignore` is not formally required, in practice, you have always been able to do it, and the C++ Core Guidelines even recommends it!

The first is tersest, though it suffers from pedagogical issues discussed in the `std::ignore` proposal. The third is fairly brief and has the benefit of clarity, but suffers from technically not being allowed (though everybody allows it in practice, so much so that even the C++ Core Guidelines were fooled). The second is most verbose, and the only things it has going for it are the pedagogical avoidance of the `(void)` cast and the language-lawyer avoidance of undocumented use of `std::ignore`. (In other words, the third option is "technically" the most correct, the best kind of correct.)

There's an alternate C++26 proposal for expressing the discard with a new `[[discard]]` attribute.

```
void test()
{
    [[discard("reason")]] important();
}
```