rules.sonarsource.com

C++ static code analysis: "std::string_view" should be used to pass a read-only string to a function

2 minutes

`std::string_view` is a read-only view over a string, it doesn't hold any data, it only holds a pointer to the first character of the string and its length. `std::string_view` can offer better performance than `std::string` in several cases:

- no memory allocations are required during construction, it is cheap to pass them by value, no need to pass them by reference
- no heap allocation when passing a string literal to a `std::string_view` function argument
- `substr` operations over a `std::string_view` do not require memory allocation

When using `std::string_view` you shouldn't however forget that:

- it's a non-owning range, you should keep into consideration the liveness of the pointed range
- it doesn't guarantee a null-terminated string like `std::string`

This rule flags `const std::string&` function arguments, which can be safely replaced with `std::string_view` ones when not relying on the null-termination character.

Note that, if you are calling `substr` on the parameter, you may have to modify your code to explicitly cast the result to `std::string`.

Noncompliant Code Example

```
void fun(const std::string& name) { // Noncompliant, replace const std::string& by std::string_view
  // ...
}
```

Compliant Solution

```
void fun(std::string_view name) {
  // ...
}
```