

"pthread_mutex_t" should be properly

"pthread_mutex_t" should not be consecutively locked or unlocked

initialized and destroyed

📆 Bug





"std::move" and "std::forward" should not be confused



A call to "wait()" on a "std::condition_variable" should have a condition



A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast



Functions with "noreturn" attribute should not return



RAII objects should not be temporary



"memcmp" should only be called with pointers to trivially copyable types with no padding

📆 Bug

"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types

📆 Bug

"std::auto_ptr" should not be used

📆 Bug

Destructors should be "noexcept"

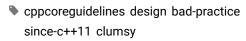
📆 Bug

Function parameters should not be of type "std::unique_ptr<T> const

Analyze your code

Code Smell





If you use std::unique_ptr<T> const & for a function parameter type, it means that the function will not be able to alter the ownership of the pointed-to object by the unique ptr:

- It cannot acquire ownership of the pointed-to object (this would require a parameter of type std::unique_ptr<T>)
- It cannot transfer the object ownership to someone else (this would require a std::unique ptr<T> &).

That means the function can only observe the pointed-to object, and in this case passing a T* (if the unique_ptr can be null) or a T& (if it cannot) provides the same features, while also allowing the function to work with objects that are not handled by a unique_ptr (E.G. objects on the stack, in a vector, or in another kind of smart pointer), thus making the function more general-purpose.

Noncompliant Code Example

```
using namespace std;
void draw(unique_ptr<Shape> const &shape); // Noncompliant
void drawAll(vector<unique_ptr<Shape>> v)
  for (auto &shape : v) {
      if (shape) {
        draw(shape);
      }
 }
}
```

Compliant Solution

```
using namespace std;
void draw(Shape const &shape); // Compliant
void drawAll(vector<unique_ptr<Shape>> v)
  for (auto &shape : v) {
      if (shape) {
        draw(*shape);
 }
}
```

See

• C++ Core Guidelines R.32 - Take a unique_ptr<widget> parameter to express that a function assumes ownership of a widget

Available In:

sonarlint in sonarcloud sonarqube Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. Privacy Policy