C++ 11

std :: lock_guard

TOPICS

1. What is std::lock_guard?
2. How to use std::lock_guard?
3. Example of std::lock_guard

What is std::lock_guard?

- A class in C++ standard library from C++11
- Used to wrap an std::mutex object
- Class designed based on RAII idiom
  - Objects are created in stack
  - Accquires mutex in Constructor
  - Releases mutex in Destructor

Note: The lock_guard class is non-copyable.

How to use std::lock_guard?

Step 1: Include header file <mutex>

```
#include <mutex>
```

Step 2: Create lock_guard as stack object

```
std::mutex mutexObject;
{
    const std::lock_guard<std::mutex> lg( mutexObject );
    // Start of Critical code section
    // ...
    // ...
    // End of Critical code section
} // Scope of lg  Ends here
```

- Destructor is invoked when scope of lock_guard object 'lg' ends.
- Destructor of lock_guard **releases** the mutex lock.

Example of std::lock_guard

code snippet without comments

```cpp
#include <thread>
#include <mutex>
#include <iostream>

int nThreadSharedVaraible = 0;
std::mutex g_mutexObject;

void safe_increment() {
    const std::lock_guard<std::mutex> lockGaurdObject( g_mutexObject );
    ++nThreadSharedVaraible;
    std::cout << std::this_thread::get_id() << ": " << nThreadSharedVaraible << '\n';
}

int main() {
    std::cout << "main: " << nThreadSharedVaraible << '\n';
    std::thread Thread1( safe_increment );
    std::thread Thread2( safe_increment );
    Thread1.join();
    Thread2.join();
    std::cout << "main: " << nThreadSharedVaraible << '\n';
    return 0;
}
```

code snippet with comments

```cpp
#include <thread>
#include <mutex>
#include <iostream>

int nThreadSharedVaraible = 0;
std::mutex g_mutexObject; // mutex object useed for thread syncronization

void safe_increment() {
    // Create lock_gaurd object using its initialization construcor
    // with the mutex object as parameter.
    // mutex is locked in constructor of lock_gaurd
    const std::lock_guard<std::mutex> lockGaurdObject( g_mutexObject ); // calls g_mutexObject.lock()

    // Start of critical code section
    ++nThreadSharedVaraible;
    std::cout << std::this_thread::get_id() << ": " << nThreadSharedVaraible << '\n';
    // End of critical code section
} // calls g_mutexObject.unlock()
  // Scope of 'lockGaurdObject' ended, so lockGaurdObject destructor is invoked
  // which unlocks the g_mutexObject

int main() {
    std::cout << "main: " << nThreadSharedVaraible << '\n';
    std::thread Thread1( safe_increment );
    std::thread Thread2( safe_increment );
    Thread1.join();
    Thread2.join();
    std::cout << "main: " << nThreadSharedVaraible << '\n';
    return 0;
}
```

Thank You