63

```cpp
template<typename mutex_type>
struct checked_lock
{
    checked_lock (mutex_type& m)
        : lock (m)
    {
        if (is_real_time_context())
        {
            non_realtime_context nrtc;
            std::cerr << "!!! WARNING: Locking in real-time context\n";
            std::cerr << get_stacktrace();
        }
    }

    ~checked_lock()
    {
        if (is_real_time_context())
        {
            non_realtime_context nrtc;
            std::cerr << "!!! WARNING: Unlocking in real-time context\n";
            std::cerr << get_stacktrace();
        }
    }

private:
    std::unique_lock<mutex_type> lock;
};
```

```
void do_mutex_lock_unlock (std::mutex& m)
{
    checked_lock l (m);
}
```

```
!!! WARNING: Locking in real-time context
./example_external(_Z14get_stacktraceB5cxx11v+0x3c) [0xaaaabdfc80c4]
 ./example_external(_ZN12checked_lockISt5mutexEC2ERS0_+0x6c) [0xaaaabdfc8b30]
  ./example_external(_Z20do_mutex_lock_unlockRSt5mutex+0x2c) [0xaaaabdfc7628]
   ...

!!! WARNING: Unlocking in real-time context
./example_external(_Z14get_stacktraceB5cxx11v+0x3c) [0xaaaabdfc80c4]
 ./example_external(_ZN12checked_lockISt5mutexED1Ev+0x58) [0xaaaabdfc8c0c]
  ./example_external(_Z20do_mutex_lock_unlockRSt5mutex+0x34) [0xaaaabdfc7630]
   ...
```

```
template<typename mutex_type>
struct checked_lock
{
    checked_lock (mutex_type& m)
        : lock (m)
    {
        if (is_real_time_context())
        {
            non_realtime_context nrtc;
            std::cerr << "!!! WARNING: Locking in real-time context\n";
            std::cerr << get_stacktrace();
        }
    }

    ~checked_lock()
    {
        if (is_real_time_context())
        {
            non_realtime_context nrtc;
            std::cerr << "!!! WARNING: Unlocking in real-time context\n";
            std::cerr << get_stacktrace();
        }
    }

private:
    std::unique_lock<mutex_type> lock;
};
```

```
void do_mutex_lock_unlock (std::mutex& m)
{
    checked_lock l (m);
}
```

```
!!! WARNING: Locking in real-time context
./example_external(_Z14get_stacktraceB5cxx11v+0x3c) [0xaaaabdfc80c4
 ./example_external(_ZN12checked_lockISt5mutexEC2ERS0_+0x6c) [0xaaaa
  ./example_external(_Z20do_mutex_lock_unlockRSt5mutex+0x2c) [0
   ...

!!! WARNING: Unlocking in real-time context
./example_external(_Z14get_stacktraceB5cxx11v+0x3c) [0xaaaabdfc80c4
 ./example_external(_ZN12checked_lockISt5mutexED1Ev+0x58) [0xaaaabd
  ./example_external(_Z20do_mutex_lock_unlockRSt5mutex+0x34) [0
   ...
```

```cpp
void do_read_file()
{
    namespace fs = std::filesystem;

    if (fs::path file_path ("/usr/bin/awk");
        fs::exists (file_path))
    {
        // Open the file in binary mode
        const auto file_size = fs::file_size (file_path);
        std::vector<char> file_data (file_size);

        // Read the file into the vector
        std::ifstream file (file_path, std::ios::binary);
        file.read (file_data.data(), file_size);

        // Check if the file was read successfully
        if (file)
            std::cout << "File read successfully." << std::endl;
        else
            std::cout << "Error reading the file." << std::endl;

        file.close();
    }
    else
    {
        std::cout << "File does not exist." << std::endl;
    }
}
```