Interview Exam: ByteVectorLogger Code

Interviewer

March 13, 2025

Instructions

Answer the following questions to the best of your ability. Provide clear and concise explanations. You may refer to the code provided below.

Code Reference

```
#include <thread>
#include <vector>
#include <condition_variable>
#include <memory>
#include <iostream>
#include <chrono>
#include <iomanip>
#include <sstream>

class ByteVectorLogger {
public:
    ByteVectorLogger() : isRunning_(false) {
        std::cout << "[ByteVectorLogger] Constructor." << std
    ::endl;
    }

    ~ByteVectorLogger() {
        std::cout << "[ByteVectorLogger] Destructor." << std
    ::endl;
    }

    void start() {
        isRunning_ = true;
```

```
22          ByteVectorLoggerThread = std::thread([this]() {
    waitForDataAndProcess(); });
23      }
24
25      void stop() {
26          if (!isRunning_) {
27              std::cout << "[ByteVectorLogger] Already stopped.
    " << std::endl;
28              return;
29          }
30
31          std::cout << "[ByteVectorLogger] Stopping..." << std
    ::endl;
32          isRunning_ = false;
33          logDataAvailable.notify_one();
34
35          if (ByteVectorLoggerThread.joinable()) {
36              ByteVectorLoggerThread.join();
37              std::cout << "[ByteVectorLogger] Thread joined."
    << std::endl;
38          } else {
39              std::cerr << "[ERROR][ByteVectorLogger] Thread
    not joinable!" << std::endl;
40          }
41      }
42
43      void receiveData(std::vector<uint8_t>& data) {
44          if (byteDataStorage.size() < MAX_STORAGE_SIZE) {
45              std::lock_guard<std::mutex> lock(queueLogMutex);
46              std::pair<std::vector<uint8_t>, std::chrono::
    system_clock::time_point> record;
47              record.first = data;
48              record.second = std::chrono::system_clock::now();
49              byteDataStorage.push_back(record);
50              logDataAvailable.notify_one();
51          }
52      }
53
54      void transitionToNextModule(std::shared_ptr<IModule>
    nextModule) {
55          // Implementation not provided in the code snippets.
56      }
57
58      void waitForDataAndProcess() {
59          while (isRunning_) {
```

```
60              std::unique_lock<std::mutex> lock(queueLogMutex);
61              logDataAvailable.wait(lock);
62              if (!isRunning_) break;
63              printRecords(byteDataStorage);
64              std::this_thread::sleep_for(std::chrono::
    milliseconds(100));
65          }
66      }
67
68      void printRecords(
69          std::vector<std::pair<std::vector<uint8_t>, std::
    chrono::system_clock::time_point>>& records) {
70          for (const auto& record : records) {
71              std::cout << formatTime(record.second) << " | ";
72              for (const auto& byte : record.first) {
73                  std::cout << static_cast<int>(byte) << " ";
74              }
75              std::cout << std::endl;
76          }
77      }
78
79      std::string formatTime(const std::chrono::system_clock::
    time_point& tp) const {
80          auto time_t = std::chrono::system_clock::to_time_t(tp
    );
81          std::tm tm = *std::gmtime(&time_t);
82          std::ostringstream oss;
83          oss << std::put_time(&tm, "%Y-%m-%d %H:%M:%S UTC");
84          return oss.str();
85      }
86
87  private:
88      std::thread ByteVectorLoggerThread;
89      bool isRunning_;
90      std::mutex queueLogMutex;
91      std::condition_variable logDataAvailable;
92      std::vector<std::pair<std::vector<uint8_t>, std::chrono::
    system_clock::time_point>> byteDataStorage;
93      static const size_t MAX_STORAGE_SIZE = 1000;
94  };
```

Questions

1. **Class Design and Purpose**
   (a) What is the purpose of the `ByteVectorLogger` class? Explain its main functionality.
   (b) Why is the constructor initialized with `isRunning_` set to `false`?
2. **Thread Management**
   (a) How does the `start()` method work? What happens when it is called?
   (b) What is the role of the `isRunning_` variable in the `waitForDataAndProcess()` method?
   (c) Explain the purpose of the `stop()` method. What happens if the thread is not joinable?
   (d) Why is `std::thread` used, and why is a lambda function passed as an argument to it?
3. **Data Processing**
   (a) How does the `receiveData()` method work? What is the purpose of the `logDataAvailable` condition variable?
   (b) What is the role of the `queueLogMutex` in the `receiveData()` method?
   (c) What happens if the `byteDataStorage` reaches its maximum size (`MAX_STORAGE_SIZE`)?
4. **Logging and Output**
   (a) How does the `printRecords()` method work? What is the purpose of the `formatTime()` method?
   (b) Why is the `std::chrono::system_clock::time_point` used in the `byteDataStorage`?
5. **Error Handling**
   (a) How does the code handle errors in the `stop()` method? What happens if the thread is not joinable?

(b) What happens if the `logDataAvailable` condition variable is notified but no data is available?

6. **Code Improvements**

   (a) Are there any potential issues with the current implementation of `waitForDataAndProcess()`? How would you improve it?

   (b) How would you modify the code to allow for configurable sleep durations between data processing?

Scoring

Each question is worth 5 points. The total score is out of 35 points.

Good Luck!