

***C++ Project***

Introduction

In this project, you are responsible for writing a C++ memory card emulator. The emulator is necessary to test real devices in advance. The system does not require a real payload; we are only interested in the commands presentation.

Requirements:

Following are the bold requirements to build the system:

* The system should be written in C++ standard 11 as a console application.
* You must consider the clean code and object-oriented principles in your implementation.
* The system has to maintain a priority queue with a configurable size (initial 10 commands).
* Design a storage class, with a configurable capacity. The class should store the command metadata like opcode and size. Then, you have to create a suitable structure to define the following memory commands:
  + **Read:** read a block of memory, each block has a logical block address (LBA), and size.
  + **Write:** write a block of memory. Each block has a logical block address (LBA), and size. Please note the actual payload is not important.
  + **Delete:** delete a range of logical block addresses.
* Following is a suggested structure layout for both command and response. Please feel free to modify based on your design:
  + **Commands:**

|  |  |  |
| --- | --- | --- |
| Command ID | Command Type (**Write**) | Command priority (0 or 1) |
| Data length (uint8\_t) | Data address (uint8\_t) | Data[0] (uint8\_t) |
| Data[1] (uint8\_t) | Data[2] (uint8\_t) | Data[3] (uint8\_t) |
| Data… |  |  |

|  |  |  |
| --- | --- | --- |
| Command ID | Command Type (**Read**) | Command priority (0 or 1) |
| Data length (uint8\_t) | Data address (uint8\_t) | Not used |
| Not used | Not used | Not used |
| Not used | Not used | Not used |

|  |  |  |
| --- | --- | --- |
| Command ID | Command Type (**Delete**) | Command priority (0 or 1) |
| Data length (uint8\_t) | Data address (uint8\_t) | Not used |
| Not used | Not used | Not used |
| Not used | Not used | Not used |

* **Responses for write / delete**

|  |  |  |
| --- | --- | --- |
| Command ID | Command Type (Response) | Command priority (0 or 1) |
| Status (Succeeded or Failed) | Not used | Not used |
| Not used | Not used | Not used |
| Not used | Not used | Not used |

* **Response for Read**

|  |  |  |
| --- | --- | --- |
| Command ID | Command Type (Response) | Command priority (0 or 1) |
| Status (Succeeded or Failed) | Data[0] (uint8\_t) | Data[1] (uint8\_t) |
| Data[2] (uint8\_t) | Data[3] (uint8\_t) | Data[4] (uint8\_t) |
| Data… | Data… | Data… |

* The system provides the following API interfaces:
  + **Add**: to add a command to the system queue. Considering the following priorities:
    - Simple command: should be present in the place.
    - Quick command: should be placed on the head of the queue.
  + **Remove:** to remove a command from the system queue head unless an id is defined.
  + **Execute:** pull the commands from the queue, and prepare a successful response as a callback for each command. For simplicity, implement a sequential execution.
  + **Abort:** stop all the execution, and mark the unexecuted commands found in the queue as failed.
* The system must handle exceptions.
* You must use the git repository (GitHub, GitLab, bitbucket, ...) to submit your code. And attach the repo link.
* Comment your code. We recommend you to use Doxygen convention.
* Optional: It would be great to have a logger subsystem to track the command traffic.

Constrains:

* Don’t copy/paste from the Internet.
* The task should be working (bug-free) and submitted to git.
* **You must use polymorphism in the task**
* The task is individual work. So, you should code it yourself.
* The task has some open-ended questions. So, it's your job to make a decision and justifications.

***– The End –***