Assignment #3
Using APIs to Communicate
80 Points

Problem

Using the code provided as a template, write a program in C or C++ that can communicate with the provided API running in a singularity container on the localhost. The APIs are defined below in the API Definitions section. The code provided will work to send and receive data via PUT and GET on the /initialize API. Your first task is to understand how this code works and then replicate it for the PUT and GET on the /modify API. You must then alter your C/C++ program to perform the following actions (keep in mind this will require editing the functions provided to you):

1) Print your Name to standard out (stdout) on its own line.
2) Print your R# to standard out (stdout) on its own line.
3) Send the integer 3360 to /initialize via PUT.
4) Send the integer 4 to /modify via PUT.
5) Retrieve the data in /initialize via a GET command and store the value as an integer.
    a. Print this value to standard out (stdout) on its own line.
6) Retrieve the data in /modify via a GET command and store the value as an integer.
    a. Print this value to standard out (stdout) on its own line.
7) Send the value received from /modify (step #6) to /initialize via PUT.
8) Send the value received from /initialize (step #5) to /modify via PUT.
9) Retrieve the data in /initialize via a GET command and store the value as an integer.
    a. Print this value to standard out (stdout) on its own line.
10) Retrieve the data in /modify via a GET command and store the value as an integer.
    a. Print this value to standard out (stdout) on its own line.

You are also required to construct a makefile for your source code that produces a binary executable named *assignment_3*.

The code necessary for this assignment can be found at */lustre/work/errees/courses/cs4352/assignment3/code*

- assignment_3.sh
    - Bash script that contains the necessary HPCC submission script configuration, the code to set up the API environment, the call to *make* to compile your code, and a call to your binary executable to run your program.
    - You may edit this file for testing, but I will provide a similar script when testing your code so don't rely on any changes to this script for your application to function.
- client.c
    - C source code to interact with the /initialize API (when running).
    - Use this as your template code if you opt to write the program in C.
- client.cpp
    - C++ source code to interact with the /initialize API (when running).
    - Use this as your template code if you opt to write the program in C++.

API Definitions

/initialize

- PUT
    - Accepts an integer value to initialize the internal data objects.
    - Returns a string that says the data was modified successfully – this can be safely ignored.
- GET
    - Returns the altered form of the integer you supplied to PUT.
    - Returns 0 if you have not yet set the data object using PUT.

/modify

- PUT
    - Accepts an integer value to modify the internal data objects.
    - Returns a string that says the data was modified successfully – this can be safely ignored.
- GET
    - Returns the altered form of the integer you supplied to PUT.
    - Returns 0 if you have not yet modified the data object using PUT.

Additional Information and Hints

1) You may work in C or C++, but this assignment and future assignments will be a bit easier to complete using C++, so I would suggest working in C++ if possible. Keep in mind this is the first assignment in a series of assignments – each one expanding on the one before it – so you will be required to get your code working and switching back and forth between C and C++ would not be advised.
2) Helpful Links:
    a. Makefile Tutorial
    b. HPCC User Guides
        i. Pay particular attention to the "Job Submission Guide" for Slurm.
        ii. **DO NOT** run the API environment or your code from the login nodes, if you wish to test your code interactively then you will need to request an interactive session as described in the "Job Submission Guide".

What to turn in to BlackBoard

A zip archive (.zip) named <FirstName>_<LastName>_R<#>_Assignment3.zip that contains the following files:

- Your C / C++ source code
    - You can select any names you want, your makefile will dictate the executable name.
- `makefile`
    - The makefile required to compile your application – ensure your final executable is called *assignment_3.*