Programming Assignment 1

Instructions

- Due by 11:59pm of September 23, 2024.
- Late penalty: **10%** penalty for **each day late**.
- This is an individual assignment. Although you may work in groups to brainstorm on possible solutions, your code implementation, report, and evaluation must be your own work.
- Upload your assignment on the **Canvas** with the following name: **Section_LastName_FirstName_PA1.zip**
- Please do **NOT** email your **assignment** to the **instructor** or **TA**!
- Choose language from C/Java/Python/Rust. If you insist on using other languages, please get approval from the teaching staffs first.

Overview

This assignment is a subscriber-publisher system. You need to create: a server program that acts as Message Broker, a client API library that can be used to create subscriber & publisher programs, and some client programs that act as subscriber/publisher to prove the correctness and benchmark your system.

Detailed Description of Assignment

- Client API

  Your client API library should implement following libraries:

  Publisher:
  <PID> registerPublisher();
  createTopic (PID, String topic);
  deleteTopic (PID, String topic);
  send (PID, String topic, String message);
  Subscriber:
  <SID> registerSubscriber ();
  subscribe (SID, String topic);
  List <String> pull (SID, String topic);

  Note:

  1. The register APIs returns an unique ID which allows the client to act as a publisher or subscriber.
  2. subscribe (…) register the caller on given topic. To receive messages from publisher, subscriber needs to proactively call the pull (…) function. Pull (…) returns all new or empty messages to the subscriber, not all history messages that the subscriber had pulled before.
  3. A message buffer is garbage collected when it is read by all subscribers.
  4. You need to hide the details of message transporting from users. That is, your client program does not need to take care of connection establishment (with server),

message sending & receiving. To send/receive messages, it simply calls the functions aforementioned.

- Message Broker
  Your server program is responsible for: maintaining message buffers for different topics, and handling requests from clients (publisher/subscriber).

- Clients
  You need to create at least 2 client programs. One acts as publisher, and one acts as subscriber. You need to use them to validate the correctness of your system. Some of your tests will involve multiple instances of publishers and subscribers.

Additional Requirements

- The server should be able to accept multiple client connections simultaneously. You **may** achieve this using threads. You **may** also use other means to achieve this goal.

Evaluation

1. Start by connecting one pair of publisher and subscriber to the server. Ensure you can perform all commands properly and both client and server can support its respective requirements aforementioned.
2. Starting with one client program, use it to benchmark server's throughput for createTopic(…). Keep increasing the number of client programs until you find the maximum throughput.
3. Repeat step 2 for all other APIs.
4. Use 2 client programs and 2 topics to do a message ping-pong test. Find out server's maximum throughput.
5. Increase the number of subscriber & publisher pairs doing ping-pong tests. Find out how server's throughput changes.
6. Graph what you find and put them in a report.

Submission Information

When you have finished implementing the complete assignment as described above, you should submit your solution to the blackboard.

Each program must work correctly and be well documented. You should hand in:

1. **Source Code** : You must hand in all your source code, including with in-line documentation.
2. **Makefile/Ant/requirements**: You must use Makefile or Ant to automate your programming assignment compilation or a requirements file to download any dependencies. If using external libraries (mainly for compiled languages) put them in a

folder marked external. **Note:** The external library cannot do the heavy lifting of the assignment tasks for you. If in doubt reach out to the TA.

3. **Deployment scripts**: You must provide a server deployment and the different number of client deployment scripts.
4. **Readme** : A detailed manual describing how the program works. The manual should be able to instruct users other than the developer to run the program step by step.
5. **Compiles Correctly** : Your code must be compiled in a Linux environment.
6. **Report** : Put your graph, system design, ideas… in the report (See grading policy – Discussion).
7. Please structure your assignment root folder as follows:
    a. Code: for your source code and make files and deployment scripts. Your file structure in here is up to you but please make sure it is clean.
    b. Docs: for all written documents, report, readme, etc
    c. Misc: for other files not mentioned specifically.
8. Submit the screen shoot for each test scenario.
9. Please put all of the above into **one** .zip file, and upload it to Canvas. The name of .zip should follow this **format**: “Section_LastName_FirstName_PA1.zip”

Grading policy:

- Working code: 50%. We have 5 APIs therefore 10% each. The code works if it passes the student's tests.
- Good testing: 25%.
- Documentation 15% including README and Make scripts and manual pages for the APIs
- Discussion: 10%. You should graph your benchmark results in the report. You should also discuss the design of your code. E.g. Do you think the APIs are sufficient? Where are the bottlenecks? You don’t have to answer exactly these questions. We look for any opinions/ideas/discussion that show you are actively thinking about this project is fine.

Submission checklist:

- Source code
- Makefile or equivalent specified above
- Deployment scripts
- Readme
- Your Evaluation report
- Screen shots of tests
- Any additional files