Assignment 2

(A3B2)

The main goal of this task is to create a simple that supports custom protocol on top of TCP socket with socat utility and bash script. Also client script should be implemented to support communication with server.

The main components of communication protocol: ***custom hanshake procedure*** based on your variant of *Part A*(table Part A below) and ***API for interaction between service and client*** based on your variant of *part B*(table Part B below). Also you will need to provide a ***simple database***(text file with a csv-like format) that will be used by server.

***Submission format:*** The solution should be sent as a ***single archive*** with all the files inside it, without any subfolders. Here is a complete list of requested files:

1. ***README.md*** – this file should contain a brief description of what your solution does, installation instructions that allows to setup the whole solution to a fresh multipass Ubuntu VM, instructions for client script usage.
2. ***ai\_usage.txt*** – this file should describe what parts of your solution were implemented with a help of AI, key prompts should be also provided.
3. ***configureServer.sh*** – this script should install systemd service *apiServer.service* and *apiServer.sh* script. Any packages required for the solution should be also installed. As a result, *apiServer.service* should be running on the system. That service should be started automatically during system boot.
4. ***apiServer.sh*** – this script should handle all the handshake logic and client’s requests based on your variant of the task. This file should be installed to /usr/bin folder during installation. All the requests should be logged to ***/var/log/apiServer.log***  file. On connection close your server should send a special bye message based on your variant. ***Important***: server should immediately drop the connection in case of any issues during handshake procedure e.g. if corresponding client doesn’t follow the protocol.
5. ***apiService.service*** – a systemd service file should cover dependencies of the apiServer and also guarantee that apiServer.sh will be running on TCP port 4242. Service should start automatically on system boot and in case of any errors and crashes it should be automatically restarted by systemd.
6. ***db.txt –*** text file with data which will be used by apiServer based on your variant. Each line of the file corresponds to some data point, columns of the record must be separated by **; (semicolon)** symbol. Your database should contain at least 20 records.
7. ***apiClient.sh*** – this script should allow user to communicate with apiServer. User should be able to provide IP address and port. Client script should perform handshake to that server and in case of success it should allow user to perform commands from apiServer API. ***Important***: client should immediately drop the connection in case of any issues during handshake procedure e.g. if the server doesn’t follow the protocol.

Each student will get a unique variant AnBm based on randomizer shuffle of the list.

To find your variant please check the tables below.

**PART A**

|  |  |  |
| --- | --- | --- |
|  | Handshake Procedure | Bye message |
| A1 | Client: Hello!  Server: Hello! Your IP?  Client: \*.\*.\*.\*  Server: Your name?  Client: Name  Server: Ready. | You shall not pass! |
| A2 | Client:Bonjour!  <no response from server, client waits at least 1 second for the next message>  Client: Hello!  Server: I am here. Your mac?  Client: aa:aa:aa:aa:aa:aa  Server: Ready. | Let it be! |
| A3 | Client: Hi!  Server: Hello! IP of your gateway?  Client: \*.\*.\*.\*  Server: Your favorite city?  Client: someCity  Server: Ready. | Don’t stop me now! |
| A4 | Client: Aloha!  Server: Oh, no.  Client: My name is someName  Server: And broadcast address of your network?  Client: \*.\*.\*.\*  Server: Ready. | May the Force be with you! |
| A5 | Client: Buonjorno!  Server: Buonjorno! Your surname?  Client: someSurname  Server: Your DNS server?  Client: \*.\*.\*.\*  Server: Ok. Ready. | I'll be back! |
| A6 | Client: Hi&Hello!  Server: Hello&Hi! Your Country?  Client: someCountry  Server: Ok. Your ipv6 address?  Client: someIPv6Address  Server: Ready. | Houston, we have a problem! |

**PART B**Be very careful during validation of user input!

Response should be a few lines. First line specifies count of elements N. Next N lines contains the values. If N is 0 then client should print No data

|  |  |  |
| --- | --- | --- |
|  | Commands | Database columns |
| B1 | GetDishOfCountry country  GetCountriesWithPopulationBiggerThan count(millions) | Country;Dish;Population |
| B2 | GetMoviesByYear year  GetMovieDirector movie | Movie;Director;Year |
| B3 | GetMoviesWithMoreThanOscars count  GetYearForMovie movie | Movie;Oscar’ count;Year |
| B4 | GetAlbumBySong song  GetSongsOfPerformer performer | Song;Performer;Album |
| B5 | GetCapitalOfCountry country GetCountriesWithCurrency currency | Country;Capital;Currency |
| B6 | GetPaintingsByYear year | Painting;Author;Year |
| B7 | GetMountsHigherThan height(m) GetMountInfo mount\_name | Mount;Country;Height |
| B8 | GetFootballClubsOfCountry country  GetCityOfFootballClub club | Football club;Country;City |
| B9 | GetRiversLongerThan length(km)  GetCountryForRiver river | River;Country;Length |