Instructors: Tuna Tuğcu & Cem Ersoy
TAs: Ömer Cihan Benzer & Selen Parlar
Contacts: omer.benzer@boun.edu.tr & selen.parlar@boun.edu.tr

Due: **December 16, 2021, Thursday, 23.59**

1 Introduction

In this project, you are expected to create a simple program in $C/C++$ that implements an application using RPC as the layer of communication between the client and the server. The project consists of three separate parts; local procedure, remote procedure, and logging. The local procedure part includes the creation of several processes and executing a binary file. The remote procedure part includes the creation of remote processes and executing a binary file. The logging part, which is a bonus, includes remotely logging of the inputs and outputs.

The project will be evaluated automatically in the Linux environment (Ubuntu Version 20.04.1) with either gcc/g++ or cc compiler (Version 9.3.0). Please follow all the requirements specified below. Your submissions will be compiled and tested via automatic scripts. Therefore, it is crucial that you follow the protocol (i.e. the rules and the specifications) defined in the project document. Failure in the compilation and/or execution is your responsibility. You should use the file names, parameters, etc. as mentioned in the project specifications.

2 Project Description

The project focuses on an unknown binary file, namely *blackbox*. The *blackbox* is an executable file that you should use without knowing the source code. So, you won't be able to know exactly what is happening in the *blackbox*. What you should know is that it takes some input from the console (i.e., by reading stdin, not command line parameters) and it displays some output based on the given input values. The main idea of the project is to automate the process using this *blackbox*. However, the project is to be designed in a flexible manner. So, do not develop your code depending on a specific binary file. There are 3 steps of the project that are built on top of each other:

- In Part A, you will run the *blackbox* with the given inputs, get the output from the *blackbox* and redirect it as requested.
- In Part B, you will upgrade the wrapper designed in Part A to be a Remote Procedural Call (RPC) server that executes the *blackbox* for any client that requests.
- In part C, you will add one more process to the project which will log the client activity in a different machine via a given Port.

You are expected to have three independent submissions; *Part A*, *Part B*, and *Part C* with the properties specified below. Note that, the input and output standards of the *blackbox* are going to be defined and an example *blackbox* program is going to be provided to you but don't forget that your project should work on not only the given binary file but any *blackbox*. We might replace *blackbox* with some other binary while grading.

2.1 Part A

1. File Properties:

   - You should create a **Makefile** to create an executable file (cmake files are not accepted). The name of the executable file should be **part_a.out**.

2. Operations:

   - Your main program should create a child process that executes the given *blackbox*.
   - The parent process should read the input from stdin (e.g., using *scanf()* function) and pass it to *blackbox* as input. Note that *blackbox* takes two integer values from stdin as input, and either produces one integer to stdout or an error text to stderr as output. Redirecting input and output files at the command line by using redirection operators ("<" and ">") will not be accepted; you must capture the input and output of *blackbox*
   - Your main program should capture the output of the *blackbox* and print it to the specified file in the following format:

     ```
     <input1> <input2> <output1>
     ```

     Make sure you append to the end of the file instead of overriding the output file.

2.2 Part B

1. File Properties:

   - You should create a **Makefile** to create the executable files. The name of the executable files should be **part_b_server.out** and **part_b_client.out**.

2. Operations:

   - Your client program should call an RPC function in the Server, passing three parameters: path, number1, and number2.
   - Your server program should be ready for clients to run the RPC function given above. This function should create a child process that executes the given *blackbox* and return the response to the Client.
   - Your client program then should print the response to the specified file in the following format:

     ```
     <input1> <input2> <output1>
     ```

2.3 Part C - BONUS

1. File Properties:

   - You should create a **Makefile** to create the executable files. The name of the executable files should be **part_c_server.out**, **part_c_client.out**, and **part_c_logger.out**.

2. Operations:

   - Your client program is exactly the same as *Part B*; it should call an RPC function in the Server, passing three parameters: path, number1, and number2.
   - Your server program is mostly the same as *Part B* with an addition; it should be ready for clients to run the RPC function given above. This function should create a child process that executes the given *blackbox* and return the response to the Client.
     However, distinct from *Part B*, this time Server process should also send the input and the output values to the logger program through the given Port which then will log these into a .log file.
   - Your client program then should print the response to the given file named to the specified file.

     ```
     <input1> <input2> <output1>
     ```

   - Your logger program should be ready for any incoming data and write them down to the specified log file:

     ```
     <input1> <input2> <output1>
     ```

3 Input & Output

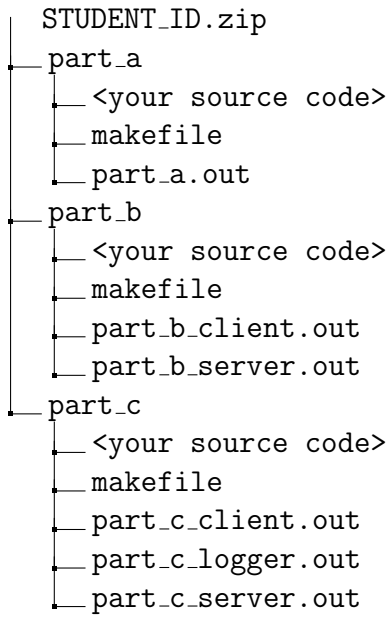
Make sure that your final submission compiles and runs with these commands. We will run your code as follows (Parameters in uppercase are to be substituted with appropriate values.):

```
/* Part A */
make
./part_a.out blackbox part_a_output.txt

/* Part B */
make
./part_b_server.out &
./part_b_client.out blackbox part_b_output.txt SERVER_IP_ADDRESS

/* Part C */
make
./part_c_logger.out part_c.log PORT_NUMBER &
./part_c_server.out LOGGER_IP_ADDRESS PORT_NUMBER &
./part_c_client.out blackbox part_c_output.txt SERVER_IP_ADDRESS
```

3.1 Input

- For Part A;
  1. The *part_a.out* program takes two command line arguments:
     blackbox: Path of the executable file
     part_a_output.txt: Path of the output file
- For Part B;
  1. The *part_b_server.out* program takes no argument and runs in the background.
  2. The *part_b_client.out* program takes three command line arguments:
     blackbox: Path of the executable file
     part_b_output.txt: Path of the output file
     SERVER_IP_ADDRESS: IP address of the server
- For Part C;
  1. The *part_c_logger.out* program takes two arguments and runs in the background.
     log_file.log: Path of the log file
     PORT_NUMBER: Port number of logger
  2. The *part_c_server.out* program takes two arguments and runs in the background.
     LOGGER_IP_ADDRESS: IP address of the logger
     PORT_NUMBER: Port number of logger
  3. The *part_c_client.out* program takes three command line arguments:
     blackbox: Path of the executable file
     part_c_output.txt: Path of the output file
     SERVER_IP_ADDRESS: IP address of the server

The file structure for the evaluation is provided below. Please design your code and zipped file accordingly. We'll provide you absolute paths, so you do not need to create directories.



3.2 Output

The *blackbox* returns an integer if the process executes successfully, and an error message if the process fails for some reason. You should capture the output and write ***SUCCESS:*** or ***FAIL:*** to the output file accordingly.

3.2.1 Example

For simplicity let's say ./blackbox sums up two positive integers. (Note that, ">" stands for console input.)

- For Part A;

  1. If *blackbox* runs successfully with the command below

     ```
     ./part_a.out ./blackbox ./part_a_output.txt
     >10
     >20
     ```

     the content of the part_a_output.txt file should be:

     ```
     SUCCESS:
     30
     ```

  2. For the next run, *blackbox* fails to run with the command below

     ```
     ./part_a.out ./blackbox ./part_a_output.txt
     >42
     >-20
     ```

     the content of the part_a_output.txt file should be:

     ```
     SUCCESS:
     30
     FAIL:
     Negative numbers are not accepted for this blackbox
     ```

- For Part B;

  1. If *blackbox* runs successfully with the command below

     ```
     ./part_b_server.out &
     ./part_b_client.out ./blackbox ./part_b_output.txt localhost
     >5
     >12
     ```

     the content of the part_b_output.txt file should be:

```
SUCCESS:
17
```

2. For the next run, *blackbox* fails to run with the command below

```
./part_b_server.out &
./part_b_client.out ./blackbox ./part_b_output.txt localhost
>-55
>12
```

the content of the part_b_output.txt file should be:

```
SUCCESS:
17
FAIL:
Negative numbers are not accepted for this blackbox
```

- For Part C;

1. If *blackbox* runs successfully with the command below

```
./part_c_logger.out ./logger.log 8080 &
./part_b_server.out localhost 8080 &
./part_b_client.out ./blackbox ./part_c_output.txt localhost
>3
>5
```

the content of the part_c_output.txt file should be:

```
SUCCESS:
8
```

and the content of the ./logger.log file should be:

```
3 5 8
```

2. For the next run, if *blackbox* fails to run the same command,

```
./part_c_logger.out ./logger.log 8080 &
./part_b_server.out localhost 8080 &
./part_b_client.out ./blackbox ./part_c_output.txt localhost
>3
>-2
```

the content of the part_c_output.txt file should be:

```
SUCCESS:
3 5 8
FAIL:
Negative numbers are not accepted for this blackbox
```

and the content of the ./logger.log file should be:

```
3 5 8
3 -2 _
```

4 Report & Grading

You are expected to submit a well-documented code. You need to explain the parameters, variables, and functions used in the code. You also need to provide author information, the main idea of the project, and a conclusion. Corresponding points are **tentatively** specified in parentheses.

1. **Code&Comment:** *20*
2. **Part A:** *(30 pts)*
3. **Part B:** *(40 pts)*
4. **Part C:** *(+20 pts)*
5. **Auto-runnable submission:** Submit your code that runs smoothly with the specified commands on the Linux environment. Each re-submission deduces 5 points. *(10 pts)*

5 Submission

Submissions will be through Moodle. Submit a single .zip file named with your student ID (e.g. 2019400XXX.zip) that matches the specified structure. Your zipped project should directly contain the required file structure, not the folder that contains the files structure. You do not need to send the *blackbox* file. Please pay attention to the file naming.

Note that your project will be inspected for plagiarism with previous years' materials as well as this year's. Any sign of **plagiarism** will be **penalized**.