AC3: RPC-Based Chat

RPC Protocol

The goal of this practice is to implement a chat based on the RPC protocol. This protocol allows the communication between two processes, one considered the client and the other the server. It is used over a network, in this case over localhost, to serve as a communication between these two processes, in order to have a request and an update of information (in this case).

`rpcgen` command

This practice has been developed in C along the `rpgen` command. This tool produce one or more C language source models, which can then be compiled by a compiler. From a .x file, it generates a stub program for the server, a stub program for the client, and a header file of definitions common to the server and the client.

The command has been the following:

```
rpcgen -C chat.x
```

Program Functionality

The program makes use of two functions for its functionality, write and getChat. In our case the **msg_client** file creates two separate threads, where each one is responsible for one of these functions. These two functionalities must be called in the client, while the behavior is defined in the **msg_server**.

The writing thread prints the username in a specific line, where the above lines are reserved for the chat history. This is done in an infinite loop, where after the program reads any keyboard inputs. The username and new string is concatenated to be sent to the server, who stores this information in a separate text file. We then print the username with an empty string to clear the previous input from the screen.

On the reading thread, every second we send a request to get the chat history from the server. This server reads the file where all chat logs are stores and returns a string. This result that we get from the server is printed on the first line each time. So that new logs will appear below the previous messages.

Thread Management

Since there had to be two concurrent functionalities, we decided to opt for threads for doing both in the same case. We generate two threads, one for each second to log the chat and one to update the chat per input.

```
if (pthread_create(&logThreadId, NULL, logThread, NULL) != 0) exit(1);

if (pthread_create(&updateThreadId, NULL, updateThread, username) != 0)  exit(1);
```

We use the mutual exclusion (mutex) to avoid any possible issues while the log thread prints on screen, using the `ncurses` library, as we have discovered it is not the most reliable and optimized library.

```
pthread_mutex_lock(&logMutex);
int y, x;
getyx(stdscr, y, x);
printf("%d", y);
move(1,0);

char ** chat = getchat_1((void*)&read_1_arg, cl);
addstr(*chat);
refresh();

move(50,x);
pthread_mutex_unlock(&logMutex);
```

Moreover, as it will be explained in the following section, we encountered that the write would take too much time, so, to avoid any error while connecting to the server, we used mutual exclusion again.

```
pthread_mutex_lock(&logMutex);
result_1 = write_1(&total_message, cl);
if (result_1 == (int *) NULL) {
    clnt_perror (cl, "call failed");
}
mvprintw(50, 1, "%s:
pthread_mutex_unlock(&logMutex);
free(total_message);
```

Observed Problems

The use of rpcgen has some peculiar functionalities. We notice how the chat.x file has to define the remote procedure calls characteristics. This is later generates the corresponding files with the appropriate command. These files are done to reflect this

definition and so any change in return types and so needs to be modified on all other files. This is one of the first obstacles we encountered and overcame. Then, we also have the use of string for this chat.x file, which does not exist as a c data type. This inconsistency did confuse us a bit at first.

Lastly, our writing functionality on the server had a serious issue in which each call to store a new message took a very long time. This affected the other functionalities as they needed to work in tandem. We did not find the main cause for this, but as each of us developed a different proposal for this exercise, we managed to concentrate our efforts on the one that didn't have this issue.