Name: Kevin Nammour, Omar Al Rafei, Theresia Tawk, and Elie Irani.
ID: 201807260, 201900211, 202004509 and 201801617

1 Introduction

The objective of this project is to develop a client-server model. This model includes a communication scheme between the client and the server. To do so, we developed an online assistant called Bamby that can communicate with several clients.

2 Client/Server

Most inter-process communication uses the client server model that will be communicating with each other. The client connects to the server in order to make a request for information or service. Prior to the connection being established, the client needs to know the address of the server, but the server does not need to know the address of the client. Once this connection is established, both sides can send and receive information. The system call for establishing a connection between client and server is socket. Where socket is an inter-process communication channel. The client and server need to establish their own socket in order to communicate.

The steps involved in establishing a socket on the client side are as follows:

1. Create a socket with the socket() system call
2. Connect the socket to the address of the server using the connect() system call
3. Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls.

The steps involved in establishing a socket on the server side are as follows:

1. Create a socket with the socket() system call
2. Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
3. Listen for connections with the listen() system call
4. Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
5. Send and receive data.

3 Parent/Child Processes

After building this connection between the client and server through the socket system call, the parent process will fork a child. The parent will receive the input from the clients and send it to the child process. The child process analyzes this input, and predicts the answer. During this time, the parent will be waiting for the child process to finish execution. Once done, the child process will send the should-be answer to the parent and then the parent will send it to the client with the send system call.

4 What Does Bamby Offer and How?

4.1 Commands

Bamby has a list of commands already defined in a data structure. Every command has a possible answer. When the user enters the commands, Bamby calls a functino that will search for the value of the command between the key/value pairs. Commands can range between greetings, personal preferences of Bamby, and some chatting.

4.2 Online Help

Bamby can actually help users to open many websites. This will be very beneficial for them instead of going to the icon of the actual app and clicking on it. Bamby can open youtube, github, google drive, LAU portal, stackoverflow, the weather, etc... This is implemented using the xdg-open function that takes as a parameter the link to be opened. This function is invoked through a system call in the server.

4.3 Sending Emails

This feature requires a command of the following format: "email receiver@example.com message". This command is divided into 3 parts. The first part is the email keyword which is used to recognize that the user wants to send an email. The second part of this command is the email that the user wants to send the email to. Last, the third part is the content of the email that was sent. Once Bamby recognizes that this command is an email, it will write the actual email and the content on the Node.js file. Then, through the xdg-open system call, we were able to execute the Node.js file.

5 Interfaces

Two interfaces were made in this project:

1. One is for the user and one is for the administrator. The interface made for the user is the documentation. The documentation is a list of all the commands available that the user can enter to communicate with Bamby.
2. The second interface is made for the administrator. The administrator can see all the communication between Bamby and all the clients, acting as "History". This can also be used to make Bamby more advanced by seeing all the commands that clients entered that are not recognized by our bot so that it can later add the commands requested in upcoming updates.

6 Languages Used

The languages used for this project are C, Node.js, Html, and Css. C was used for the Client/Server communication. Node.js was used to make it possible for the user to send emails. And finally, Html and Css are used for the interfaces.

7 How to run the code

In order to start the server, you have to be using ubuntu or Linux since in the server file we have used many functionalities offered only by Ubuntu. First, run this command on your terminal:

```
gcc -pthread -o server server.c
./server
```

In order to create a client, run this on another terminal tab,

```
gcc client.c -o client
./client
```

Try creating many clients to see the magic of client/server communication.

However, in order to open administrator mode, you will have to run the seeHistory.c file by typing in your terminal the following commands:

gcc seeHistory.c -o seeHistory
./seeHistory