

# Instituto Tecnológico de Costa Rica



## **Proyecto Programado I**

### **Chat**

#### **Carrera:**

Administración de tecnología de información

#### **Curso:**

Lenguajes de Programación

#### **Profesor:**

Andrei Fuentes Leiva

#### **Estudiantes:**

Marianne Cordero

Coraima Fonseca

Arlyn López

**I Semestre, 2014**

## Contenido

1) Descripción del problema .....	4
2) Diseño del programa.....	6
3) Librerías.....	14
4) Análisis de Resultados.....	17
5) Manual de Usuario.....	19
Configuración del usuario .....	21
6) Conclusiones .....	24

# **1) Descripción del problema**

El problema consiste en crear un programa en lenguaje de programación C que sea capaz de permitir la conexión entre dos máquinas con la finalidad de poder enviar y recibir mensajes entre las dos, esto se debe lograr con la creación de dos funciones (servidor y cliente). Además deberá permitir la funcionalidad de agregar contactos, escribiendo y leyendo de un archivo txt, de donde se tendrán que obtener los datos de la IP y puerto de cada contacto. El programe deberá ser lo ms eficiente posible, por lo que deben cerrarse todas las conexiones antes de finalizar el programa.

## **2) Diseño del programa**

El algoritmo principal del sistema es el que establece la comunicación entre dos computadoras, esto se logra por medio de la utilización de un socket. Posteriormente se explicará el algoritmo usada para la funcionalidad de agregar contactos.

- **Envío y Recepción de Mensajes**

El algoritmo se encuentra formado por dos funciones de tipo int, las cuales son: Cliente y Server. Antes de crear dichas funciones se establece la función de error la cual se invoca cuando una llamada al sistema falla.

### **Envío de mensajes**

En el caso de la función Cliente, está recibe dos parámetros, uno de tipo int y el otro de tipo char a puntero, estos parámetros serán:

- **Int puerto Cliente:** Parámetro que contendrá el puerto del cliente, por el cual se establecerá el envío y recepción de datos, dicho puerto se obtendrá de un archivo txt el cual se llamará "Amigos.txt".
- **char \*IpCliente:** Parámetro que contendrá la dirección IP del cliente, la cual le permitirá al servidor saber a qué cliente deberá conectarse.

Una vez que se ha creado la función Cliente en el programa, se inicializan y declaran las variables locales, así como también las struct que se utilizarán y permitirán la realización del programa. Posteriormente se definirá la función que creara el socket, la cual devolverá el identificador de dicho socket, es decir no lo asocia a ninguna IP o puerto. Si el identificador es menor a 0, se invoca a una función de error de lo contrario se establecen los cuatros campos que contiene la struct sockaddr\_in (Definida al inicio de la función Cliente). Después se establece la conexión con el servidor por medio de la función connect (), la cual toma tres

argumentos descriptor del fichero, dirección de la máquina con la que se conectará y el tamaño de dicha dirección. Si `connect ()` es menor a 0, se invoca a la función de error. Cuando se establezca la conexión se creará un ciclo infinito que permitirá el envío de datos de manera simultánea por medio de la función `send ()`, dentro de ese ciclo también se crea una sentencia de control de flujo en donde se comparan dos string (El dato enviado con la palabra chao), si se cumple se envía el dato de lo contrario manda el dato y cierra el socket.

- **Recepción de Mensajes**

En el caso de la función `Server`, está recibe un parámetro de tipo `int` y el otro de tipo `char` a puntero, estos parámetros serán:

- **Int puerto Server:** Parámetro que contendrá el puerto del servidor, por el cual se establecerá el envío y recepción de datos, dicho puerto se obtendrá de un archivo `txt` el cual se llamará “Archivo de Configuración”.

Una vez que se ha creado la función `Server` en el programa, se inicializan y declaran las variables locales, así como también las `struct` que se utilizarán y permitirán la realización del programa. Después se definirá la función que creará el socket, la cual devolverá el identificador de dicho socket, es decir no lo asocia a ninguna IP o puerto. Si el identificador es menor a 0, se invoca a una función de error de lo contrario se establecen los cuatro campos que contiene la `struct sockaddr_in` (Definida al inicio de la función `Cliente`) y si inicializa a cero el `sockaddr_in` servidor `()`.

Luego de obtener el identificador se crea la función `bind ()` la cual enlaza una conexión a una dirección; esta función tendrá tres argumentos: descriptor del fichero, dirección de la máquina con la que se conectará y el tamaño de dicha dirección. Si `bind ()` es menor a 0, se invoca a la función de error. Cuando se haya establecido la conexión se hace la llamada a la función `listen ()` que permite el número de conexiones en que servidor puede esperar mientras tenga una conexión en proceso. Finalmente se creará un ciclo infinito que contendrá la función `accept ()` la cual hace que el proceso bloquee hasta que se logre establecer



conexión con un cliente, dentro de ese ciclo habrá otro ciclo el cual permite que los datos se reciban de manera simultánea gracias a la función `recv()`. En el segundo ciclo se comparan dos string dentro de una sentencia de flujo de control, si se cumple se cerrará la conexión del socket.

- **Agregar amigos**

En esta funcionalidad el usuario podrá agregar a cualquier amigo, en la cual se le solicita que ingrese el nombre de la persona que desea agregar, su dirección IP y el puerto al que desea comunicarse. Para ello se implementó el uso de structs con el propósito de almacenar en ellos los datos necesarios para agregar un contacto, tales como el nombre del mismo, su dirección IP y puerto, mencionados anteriormente.

Primeramente, se crea un struct llamado `NodosLista` el cual contiene tres char (nombre, dirección IP y puerto) y un puntero al siguiente struct. El objetivo fue crear una lista simple enlazada formada por structs, para lo cual se utilizó la función de insertar en donde se inserta cada elemento nuevo al final de la misma. Se utilizó este tipo de estructura de datos ya que se ha tenido experiencia en el manejo de las mismas para lo cual se facilitó su implementación en dicha tarea, además, de que no hay que preocuparse por definir el tamaño de la estructura donde se almacenarán los datos ya es esta crece en tiempo de ejecución.

Cada vez que el usuario agrega un contacto, este se escribe en un archivo txt, es decir, se escribe su nombre, dirección IP y puerto, los cuales están separados por comas dentro del mismo. Cuando se quiera ver la lista de contactos entonces se llamará a la función que lee el archivo, la cual abre el archivo donde se escriben los contactos (`Amigos.txt`), obtiene cada token de la línea que leyó por medio de la función `strtok` (la cual separa cada token cada vez que encuentre el delimitador, en este caso las comas) y copia dichos tokens (con la función

strcpy) en cada dato del struct NodosLista, respectivamente. Una vez copiados estos tokens entonces se cargan en una lista, todo esto se encuentra dentro de un ciclo el cual termina cuando se llega al final del archivo. Por último, la lista se muestra con un método que imprime la misma.

Se creó un struct llamado NodoPuerto el cual almacena los puertos en los cuales el servidor estará escuchando las peticiones de conexión. Dichos puertos se escriben en un archivo txt (Archivo\_Configuración.txt) cada vez que el usuario agrega un amigo. Se implementó la misma lógica que en agregar los datos de los contactos, ya que se cargan los elementos del archivo de configuración en una lista simple enlazada ya sea para mostrar los puertos, leer el archivo o realizar una búsqueda.

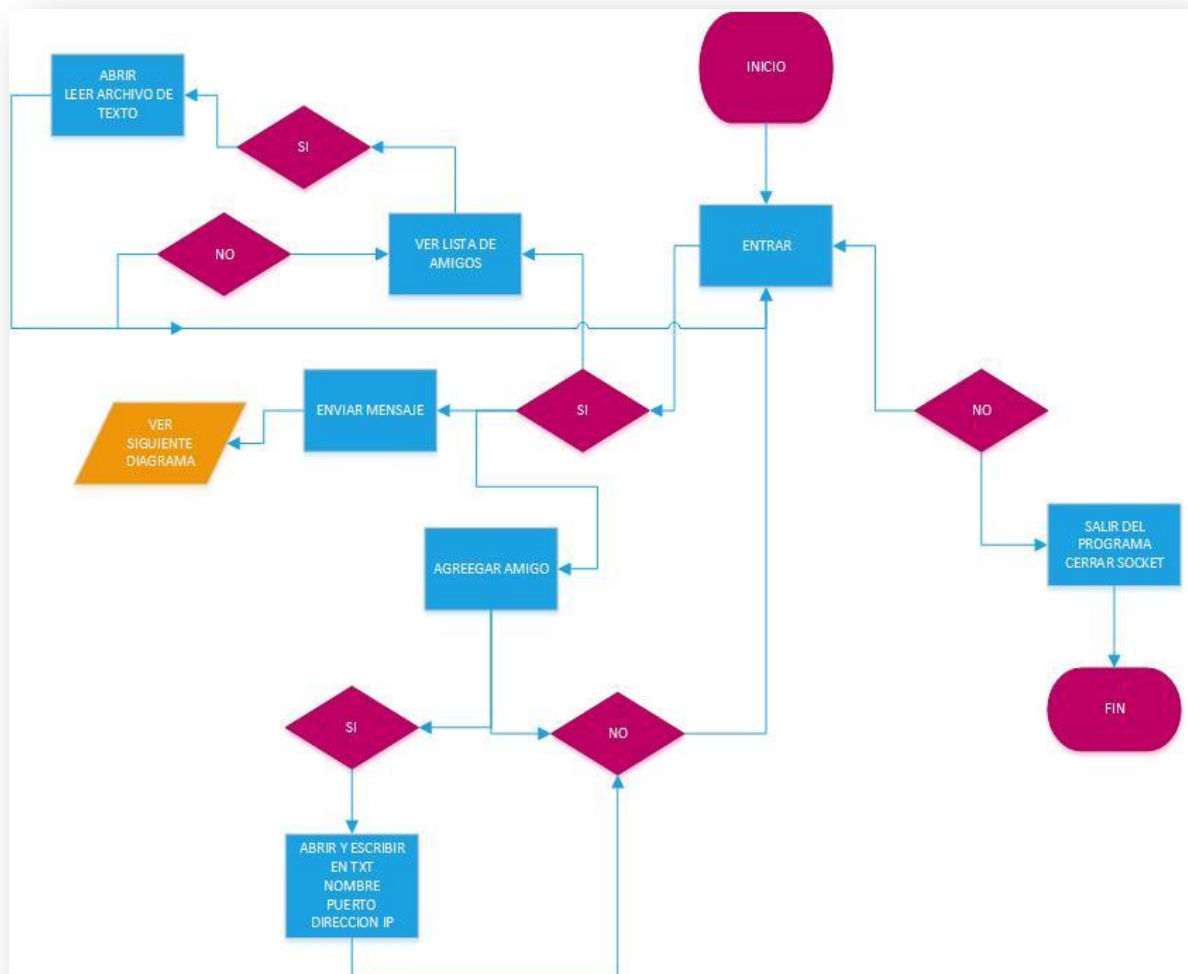
Si el usuario desea buscar un contacto en específico entonces se le solicita que ingrese el nombre del mismo, y en seguida se le mostrará los datos de este, o por el contrario, se le indicará si el contacto que busca no se encuentra en la lista.

En cuanto al envío de mensajes, se le solicita al usuario que ingrese el nombre de la persona a la que desea enviarle un mensaje, el programa automáticamente obtiene su dirección IP y puerto. La dirección IP se obtiene mediante una función llamada busqueda\_IP, la cual recibe el nombre del contacto y una vez que lo encuentra entonces devuelve la dirección IP. En caso contrario, se le indica al usuario que el contacto al cual quiere enviarle un mensaje no se encuentra su lista.

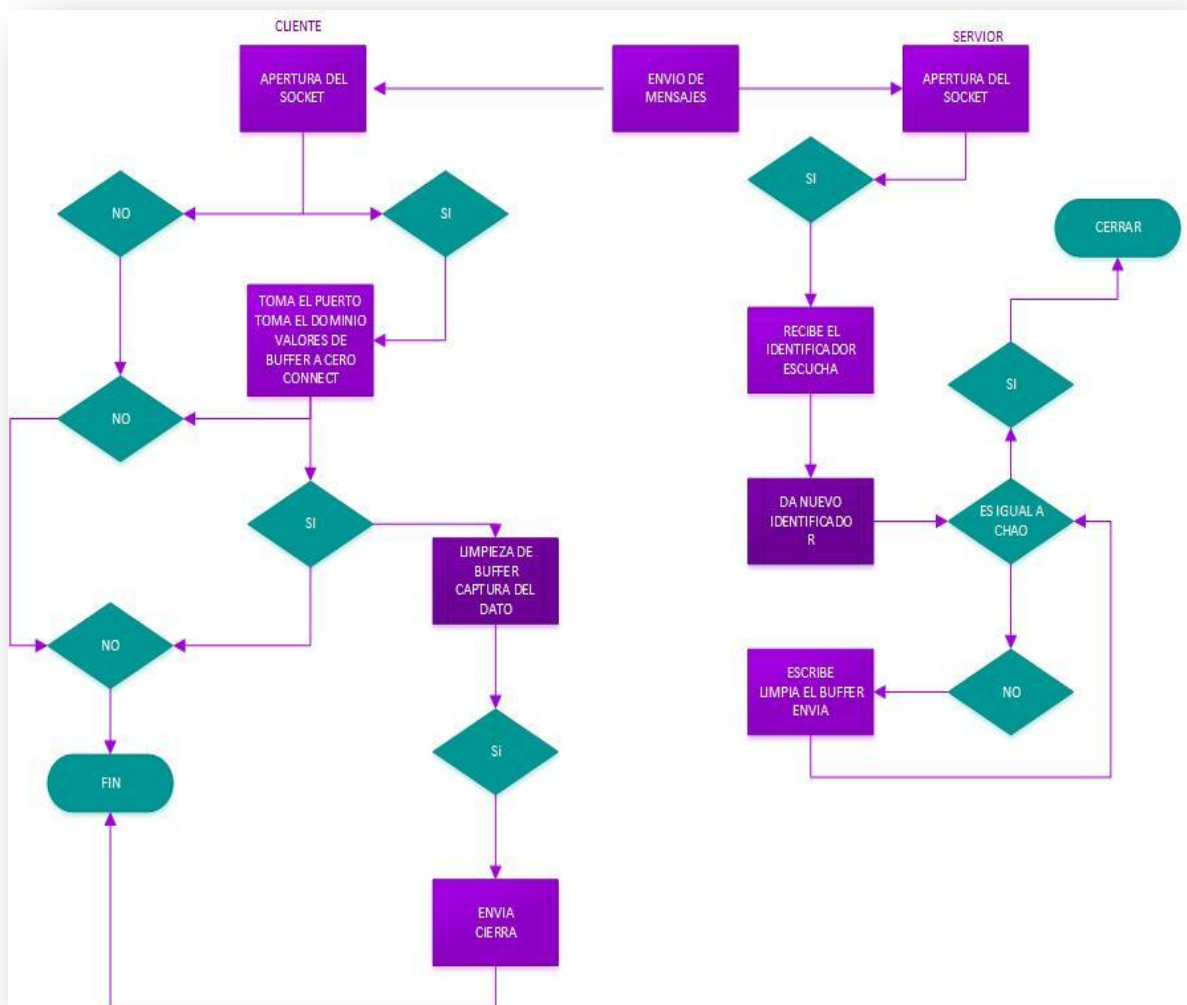
El puerto se obtiene del archivo de configuración, para lo cual, lo primero que se realiza es obtener el puerto del contacto y se verifica que ese puerto esté en el archivo de configuración, si es así, la función devolverá un entero que representa dicho puerto, o por otro lado, devuelve cero, donde se le indica al usuario que ese puerto no está en la lista de configuración. Una vez obtenidos la dirección IP y el puerto se procede a iniciar la conversación, en la cual cada usuario está identificado de distinto color.

## Diagramas Logicos

### Función Main() y Agregar Contactos



## Envío y Recepción de Mensajes(Socket)



## **3) Librerías**

A continuación se explicarán las diferentes librerías utilizadas para la elaboración y ejecución del programa.

Las librerías son las siguientes:

- **stdio.h:** Librería que permite hacer un uso efectivo de las variables en el programa ya sea por medio de scanf, printf o bien size of (x), es la librería universal de C.
- **string.h:** Se utiliza para realizar un manejo de strings en el programa de una manera general, como por ejemplo para realizar comparaciones entre strings y determinar su tamaño.
- **sus/types.h:** Sirve para dar estructura a los sockets con los símbolos e identificados que estos utilizan. ip\_addr\_, hace referencia a la dirección IP, in\_addr\_t, toma las direcciones de la red.
- **stdlib.h:** Esta librería contiene funciones automáticas para realizar conversiones entre tipos de datos (strings, int) y colabora en el manejo de los punteros. Un ejemplo de lo anterior sería:  
Atof convierte un arreglo de chars a float y atoi los convierte a int.
- **sus/socket.h:** Esta librería contiene la estructura base de un socket.
- **netinet/in.h:** Importa todos los protocolos de comunicación que son usados en la red, en este caso TCP/IP.
- **Netdb.h:** Define las descripciones para los sockets del servidor utilizando el hostent y los tipos de puertos en los que se realizara la conexión.

- **Unistd.h:** Proporciona el acceso a la POSIX del sistema operativo en el API.
- **Errno.h:** Es una macro que devuelve los códigos de error al compilar el programa.
- **Arpa/inet.h:** Influye en las operaciones que se llevan a cabo en el internet.
- **Wait.h:** Realiza las operaciones de espera en el proceso de implementación de la conexión en los sockets.



## **4) Análisis de Resultados.**

Al concluir el proyecto se lograron alcanzar ciertos la mayoría de los objetivos dentro de los cuales están:

- **Agregar amigos:** lo cual implica que el programa al solicitarle los datos al usuario sobre el amigo que desea agregar este lo realiza escribiendo dichos datos en un archivo txt por lo que los datos no se pierden al finalizar el programa, ya que cada vez que se desea ver la lista de contactos, esta se podrá acceder desde el archivo txt.
- **El usuario es capaz de comunicarse con un amigo seleccionado:** esto quiere decir que el programa le dará la opción de escoger al amigo con el cual se quiere comunicar y a su vez le habilitará la conexión por medio de los datos guardados en el txt, con lo cual el usuario solo debe digitar el nombre y se podrá entablar la conversación.
- Se pueden enviar y recibir mensajes de forma simultánea.
- Se implementa el uso de los colores para identificar los mensajes que envía cada usuario.

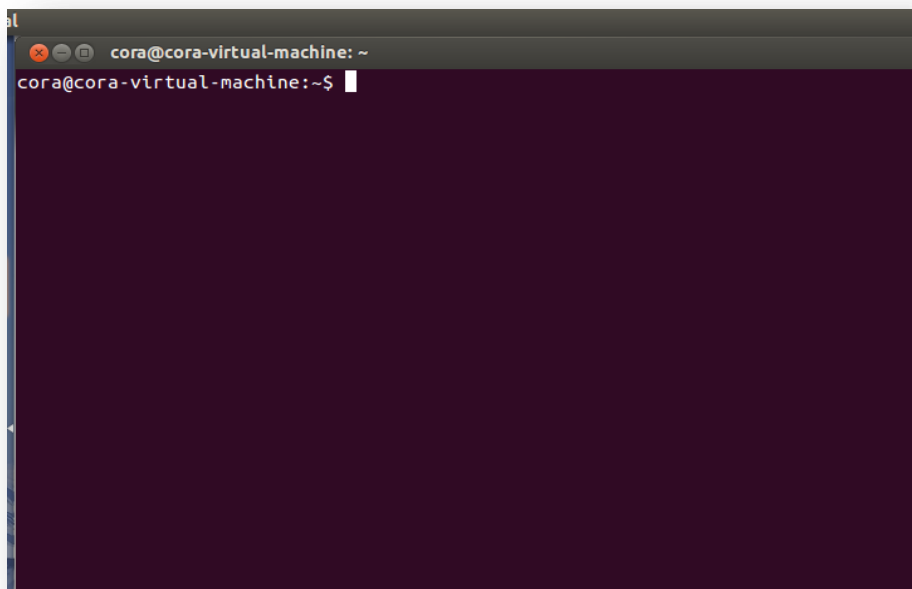
A pesar de haber logrado la mayoría de los objetivos no se pudo cerrar bien la conexión. La idea es que al enviar la palabra chao el programa finalice, es decir, se cierre la conexión en ambos lados, sin embargo, esto no sucede ya que solo se cierra en una computadora.

# **5) Manual de Usuario**

Este programa consiste en establecer una comunicación entre distintas computadoras o bien en una sola. Para la ejecución del programa es indispensable que cuente con un sistema operativo basado en Linux.

Para poder realizar de manera correcta la ejecución del programa se deben seguir los siguientes pasos:

- Ejecute la terminal de Ubuntu.



- Instale GCC en caso de no tenerlo instalado.
- Para la compilación es necesario que descargue los archivos del repositorio y a su vez cuente con GCC, en la terminal escriba:
  - Cd Escritorio
  - ls
  - Gcc (nombre del archivo).c

- Para la ejecución del chat después de ser compilado siga las siguientes instrucciones:
  - ✓ Cd Escritorio
  - ✓ ls.
  - ✓ ./(nombre del archivo) puerto
  - ✓ Seleccione crear un usuario de lo contrario seleccione uno de los que están guardados.

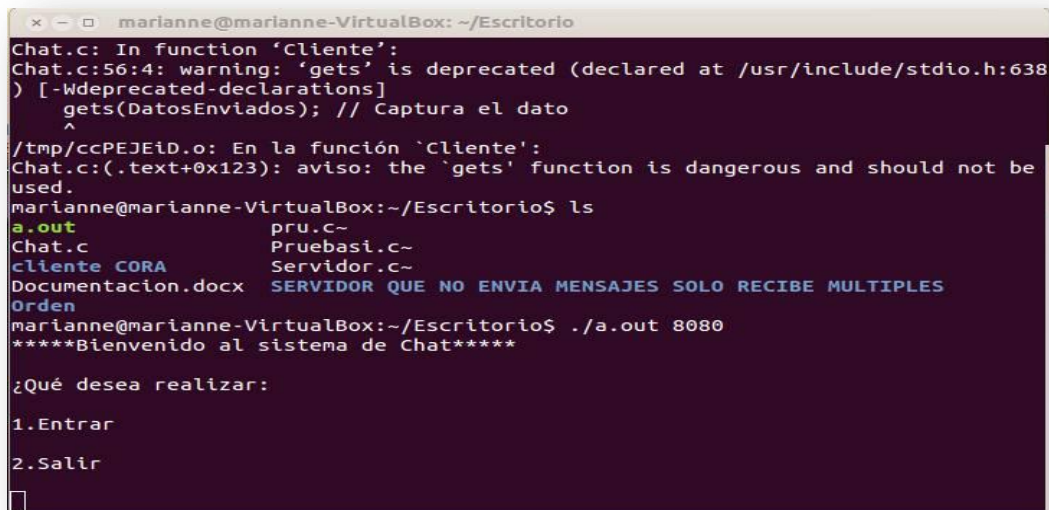
## Instrucciones de uso

### Configuración del usuario

Ejecute el programa como se menciona anteriormente.

Seleccione lo que desea realizar.

1. Entrar
2. Salir



```
marianne@marianne-VirtualBox: ~/Escritorio
Chat.c: In function 'Cliente':
Chat.c:56:4: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638)
) [-Wdeprecated-declarations]
    gets(DatosEnviados); // Captura el dato
    ^
/tmp/ccPEJEiD.o: En la función 'Cliente':
Chat.c:(.text+0x123): aviso: the 'gets' function is dangerous and should not be used.
marianne@marianne-VirtualBox:~/Escritorio$ ls
a.out      prU.c~
Chat.c     PruebasI.c~
cliente CORA  Servidor.c~
Documentacion.docx  SERVIDOR QUE NO ENVIA MENSAJES SOLO RECIBE MULTIPLES
Orden
marianne@marianne-VirtualBox:~/Escritorio$ ./a.out 8080
****Bienvenido al sistema de Chat****

¿Qué desea realizar:

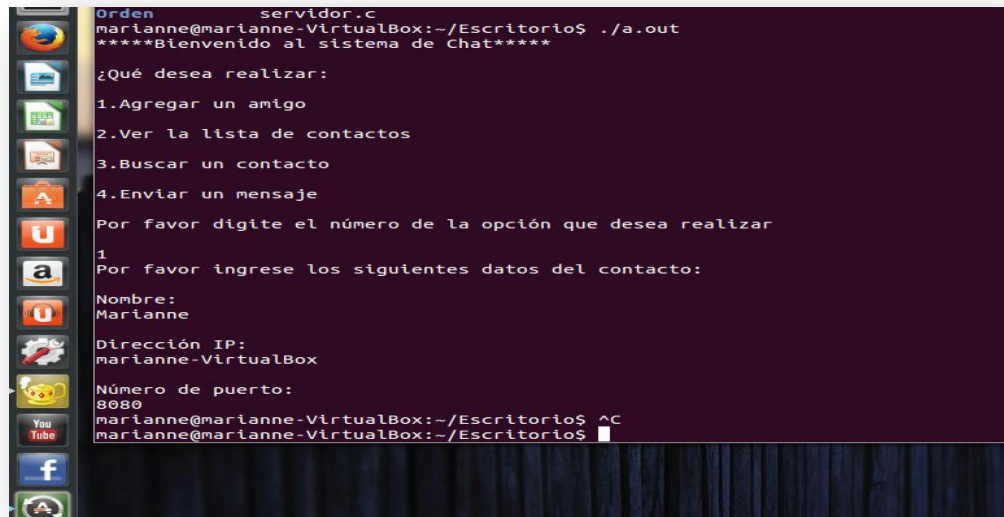
1. Entrar
2. Salir

```

En caso de que seleccione la opción 1 le aparecerán las siguientes cuatro opciones para elegir

**Nota:** en caso de que sea la primera vez que está usando el sistema seleccione la opción número uno para crear un usuario

Proceda a agregar un amigo



```
Orden servidor.C
marianne@marianne-VirtualBox:~/Escritorio$ ./a.out
****Bienvenido al sistema de Chat****

¿Qué desea realizar:
1.Agregar un amigo
2.Ver la lista de contactos
3.Buscar un contacto
4.Envíar un mensaje
Por favor digite el número de la opción que desea realizar
1
Por favor ingrese los siguientes datos del contacto:
Nombre:
Marianne
Dirección IP:
marianne-VirtualBox
Número de puerto:
8080
marianne@marianne-VirtualBox:~/Escritorio$ ^C
marianne@marianne-VirtualBox:~/Escritorio$
```

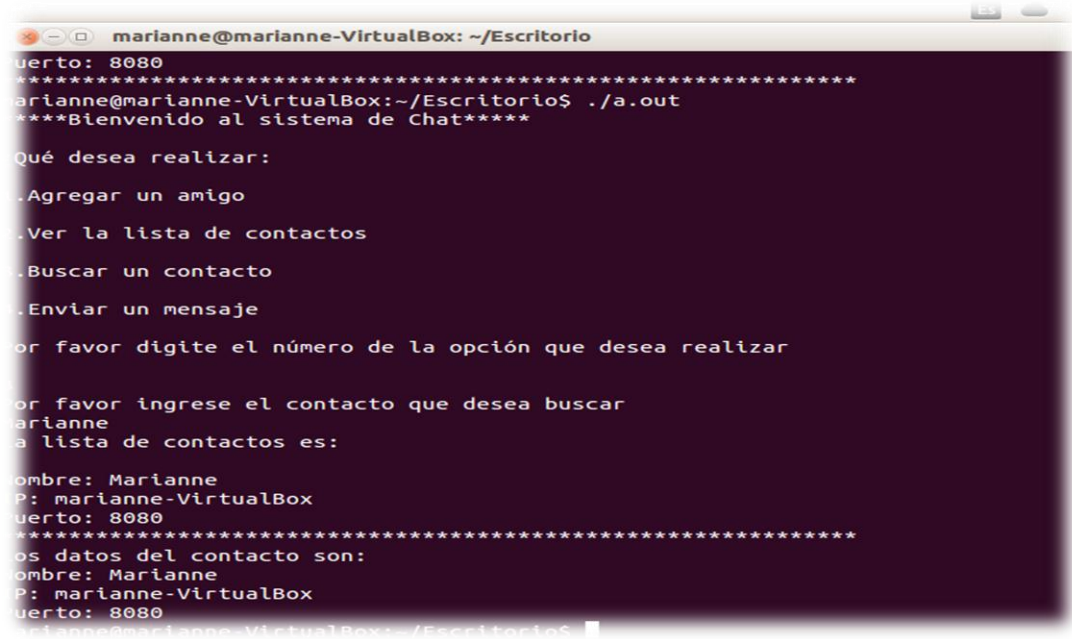
Para ver la lista de contactos presione dos



```
marianne@marianne-VirtualBox: ~/Escritorio
Marianne
Dirección IP:
marianne-VirtualBox
Número de puerto:
8080
marianne@marianne-VirtualBox:~/Escritorio$ ^C
marianne@marianne-VirtualBox:~/Escritorio$ ./a.out
****Bienvenido al sistema de Chat****

¿Qué desea realizar:
1.Agregar un amigo
2.Ver la lista de contactos
```

Para buscar un contacto presione tres, luego acceda el nombre del contacto que desea buscar.



```
marianne@marianne-VirtualBox: ~/Escritorio
uerto: 8080
*****
marianne@marianne-VirtualBox:~/Escritorio$ ./a.out
****Bienvenido al sistema de Chat****

Qué desea realizar:
1.Agregar un amigo
2.Ver la lista de contactos
3.Buscar un contacto
4.Envíar un mensaje

Por favor digite el número de la opción que desea realizar

Por favor ingrese el contacto que desea buscar
marianne
La lista de contactos es:
Nombre: Marianne
P: marianne-VirtualBox
uerto: 8080
*****
Los datos del contacto son:
Nombre: Marianne
P: marianne-VirtualBox
uerto: 8080
marianne@marianne-VirtualBox: ~/Escritorio$
```

Para enviar mensaje presione cuatro y proceda a seleccionar el amigo.



```
marianne@marianne-VirtualBox: ~/Escritorio
****Bienvenido al sistema de Chat****

Qué desea realizar:
1.Agregar un amigo
2.Ver la lista de contactos
3.Buscar un contacto
4.Envíar un mensaje

Por favor digite el número de la opción que desea realizar
4
Digite el nombre del contacto al que desea enviar un mensaje:
marianne

Esperando la conexión en el puerto 8080
Yo (Escriba chao para cerrar la conversación):
Yo (Escriba chao para cerrar la conversación):
```

Puede iniciar el envío de mensajes, para salir escriba “chao” y para cerrar el programa presione la opción de salir.

## **6) Conclusiones**



A través de este proyecto se logró profundizar en el aprendizaje de como utilizar los sockets, sobre el uso del fork lo cual permitió el envío y recepción de mensajes de forma simultánea, también sobre el

manejo de archivos, tanto escritura como lectura de ellos, el uso de structs para almacenar diferentes tipos de datos tales como los del usuario y los del archivo de configuración, entre otros aspectos. Para lo cual en dicho trabajo los objetivos alcanzados por cada una fueron los siguientes:

Coraima Fonseca:

- Envío de mensajes uno a uno desde el servidor al cliente y viceversa.
- Uso de punteros.
- Configuración de los ciclos para la lectura y escritura en los sockets.
- Configuración de la lectura y escritura de mensajes por lotes.
- Investigación de modelo TCP/IP.

Arlyn López:

Objetivos alcanzados:

- Uso de structs para crear amigos.
- Configuración para el envío y recepción de mensajes.
- Creación, lectura y escritura de archivos de texto.
- Manejo de punteros y listas.
- Manejo de switches para la configuración de usuario.
- Investigación sobre ejecución de los programas.

Marianne Cordero:

Objetivos alcanzados:

- Envío de mensajes uno a uno desde el servidor al cliente y viceversa.
- Uso de punteros.
- Configuración de los ciclos para la lectura y escritura en los sockets.
- Configuración de la lectura y escritura de mensajes por lotes.
- Investigación sobre sockets y uso de fork.