

# TAREA PROGRAMADA I

## TI-3404

Integrantes:

- Daniel Mora Chacón
- Gerardo Calderón Badilla
- Edgar Solórzano Pacheco

Profesor:

Andrei Fuentes Leiva

*Documentación  
Externa*

## Contenido

Descripción del Problema .....	2
Librerías Usadas.....	3
Manual de Usuario .....	5
Diseño del Programa .....	13
Análisis de Resultados .....	14
Conclusión Personal .....	15
Bibliografía.....	16

## Descripción del Problema

El objetivo principal de este proyecto programado es la creación de un programa Messenger el cual sea capaz de enviar y recibir mensajes, así como el registro de usuarios para la implementación de este. Dicho programa se desarrolló en el lenguaje de Programación C e implementado en el sistema operativo Ubuntu para su posterior revisión. Para realizar la comunicación se utilizaron los sockets, los cuales permiten la conexión entre dos computadoras, solicitando el nombre del usuario, el cual contiene los puertos y la IP necesarias para la compilación.

En el registro de usuarios cada usuario deberá agregar el puerto y la IP de la persona con la que desea conectarse.

La mayoría de las funciones en C relacionadas con sockets requieren un puntero a alguna estructura Internet socket address como argumento. Se utiliza la estructura `in_addr` que sirve para albergar una dirección de internet en formato de entero largo. Las estructuras socket son siempre pasadas por referencia en los argumentos de cualquier función que tenga que ver con sockets. Pero esas funciones socket que toman esos punteros deberán lidiar con estructuras socket provenientes de diferentes familias de protocolos soportados. Se deberá a. Crear un socket TCP usando la llamada al sistema `socket()`. Asignar un número de puerto al socket mediante la llamada a `bind()`. Decirle al sistema que permita conexiones del puerto que se mencionó anteriormente, esto lo haremos con `listen()`. Y repetidamente llamar a `accept()` para obtener un nuevo socket para cada conexión que haga el cliente, comunicar con el cliente vía el nuevo socket usando `send()` y `recv()` y cerrar la conexión cuando se deje de usar.

## **Librerías Usadas**

### **Librería para entrada y salida de datos**

`#include <stdio.h>`

### **Librería para uso de sockets**

`#include <sys/socket.h>`

Permite la conexión entre dos computadoras

### **Salida del programa en caso de error**

`#include <stdlib.h>`

`#include <sys/types.h>`

### **Librería para el uso de base de datos en red**

`#include <netdb.h>`

`#include <signal.h>`

Esta última librería es la encargada de matar el proceso del servidor y usuario

Contiene la función `Kill()`

### **Librería para uso de `fork()`**

`#include <unistd.h>`

Encargada de la bifurcación de dos procesos idénticos

### **Librería que posee los headers para resolución de DNS**

`#include <netinet/in.h>`

### **Librería para el uso de bzero y bcopy**

`#include <string.h>`

Bzero: Llena de ceros el array de caracteres y limpiar la cadena de caracteres.

Bcopy: Copia de bytes para capturar el mensaje del puerto.

### **Define la palabra de cierre de la conversación: Chao**

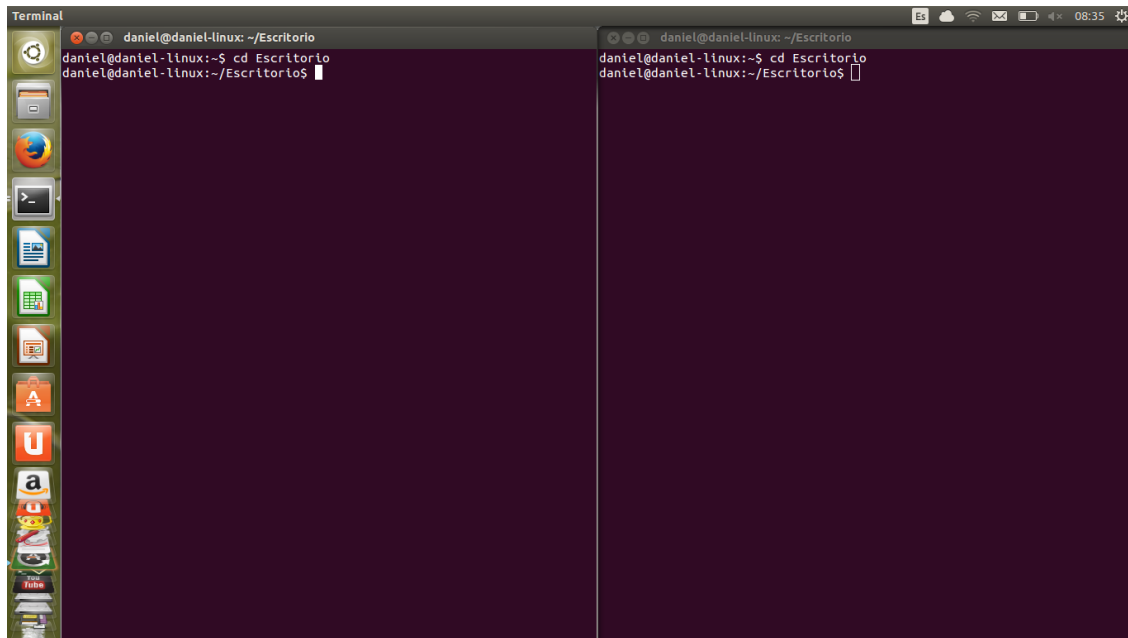
`#define escape "Chao\n"`

## Manual de Usuario

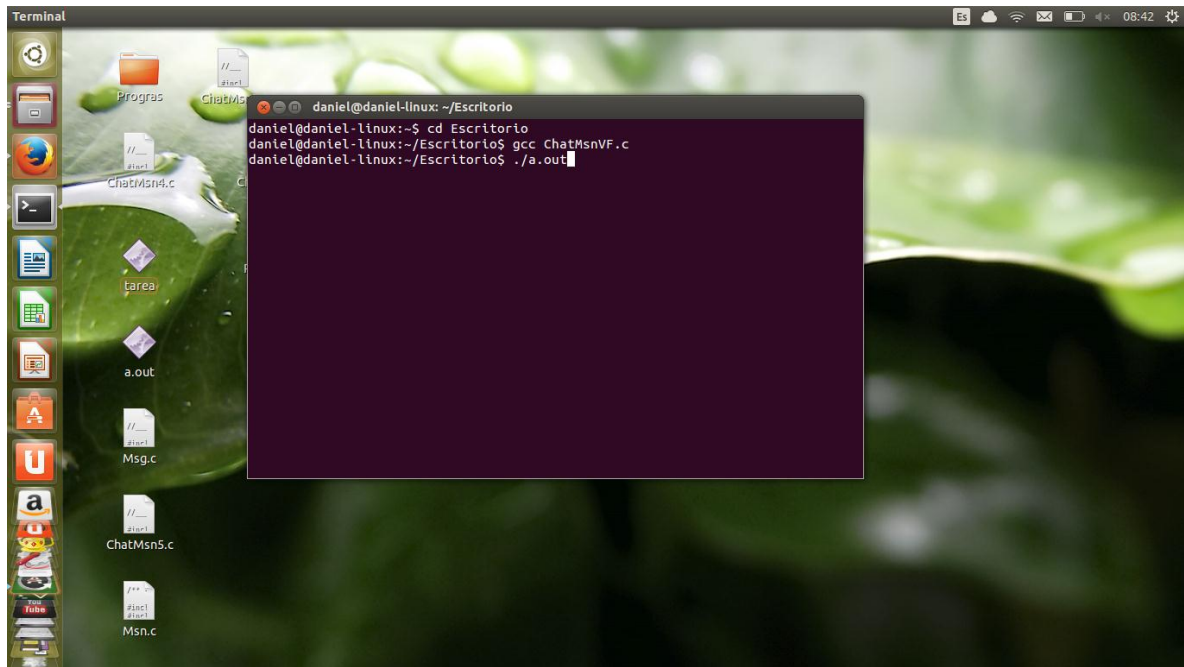
- 1- Para correr el programa ingrese a la terminal que posee Ubuntu.



- 2- Para este caso se correrá en dos terminales para probarlo. Debe digitar la dirección donde se encuentra el archivo.c, en ese caso se encuentra en el escritorio.

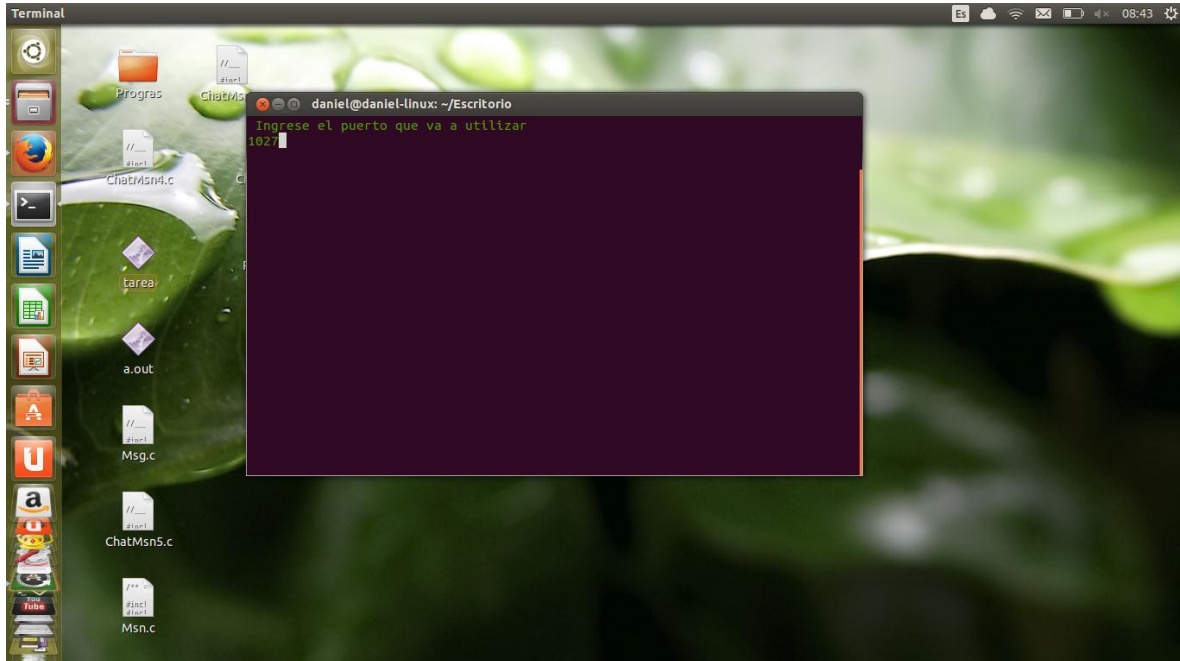


- 3- Debe digitar el nombre del ejecutable donde tiene su archivo, utilizándolo así. /Nombre, en este caso se utilizará el que se asigna por defecto.

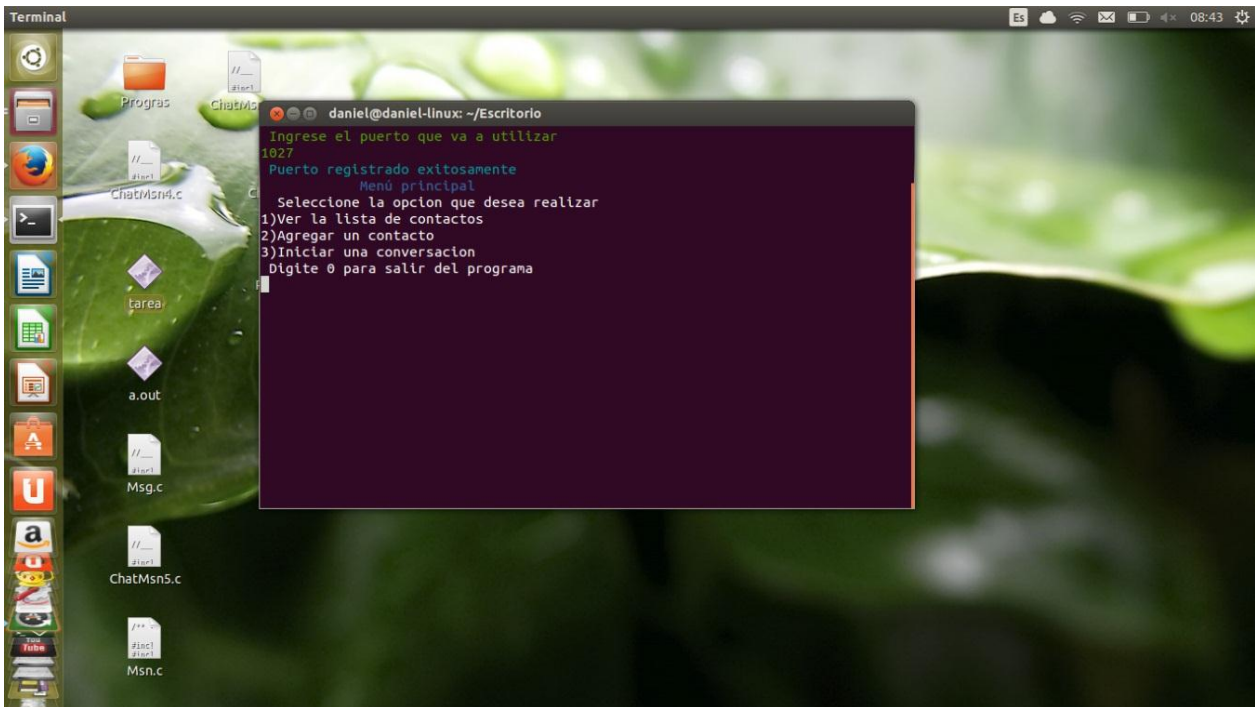




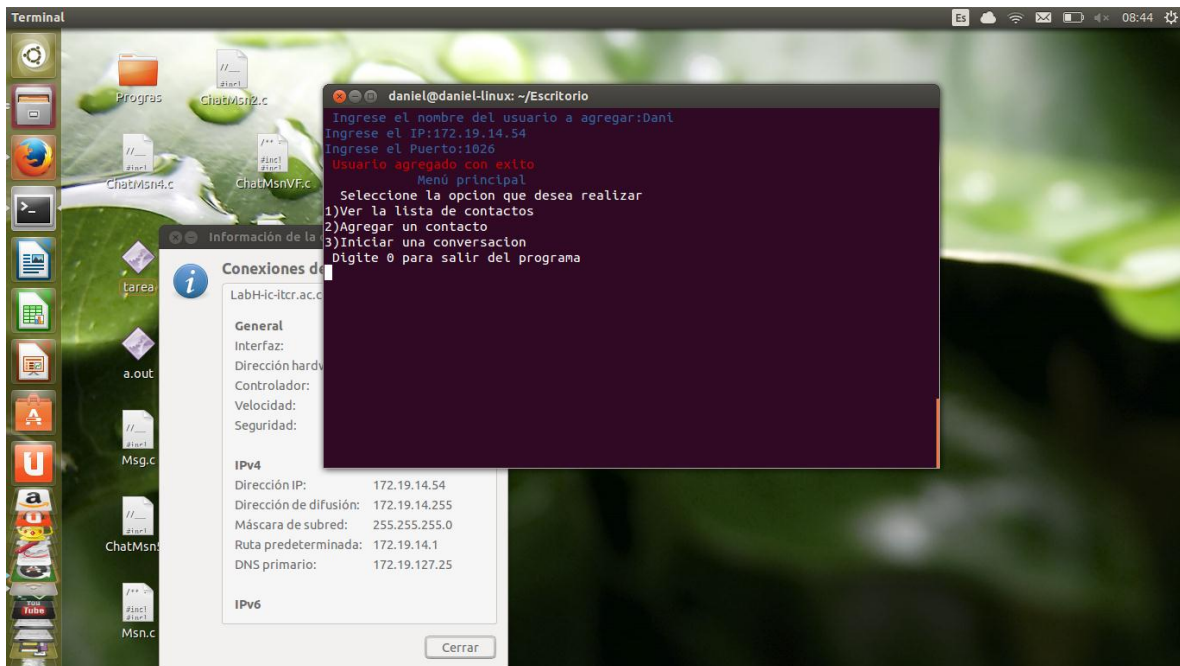
- 4- Una vez hecho esto el programa le preguntará el puerto a utilizar. Debe digitarlo.



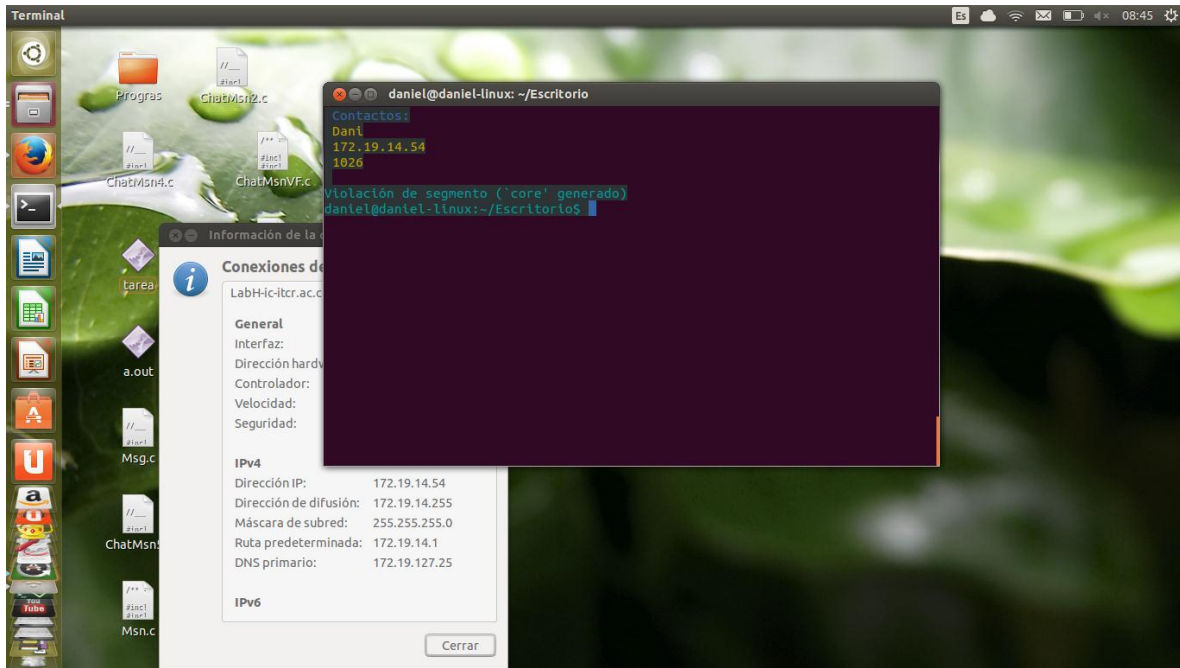
5- Con el puerto registrado, le aparecerá el menú al que puede utilizar.



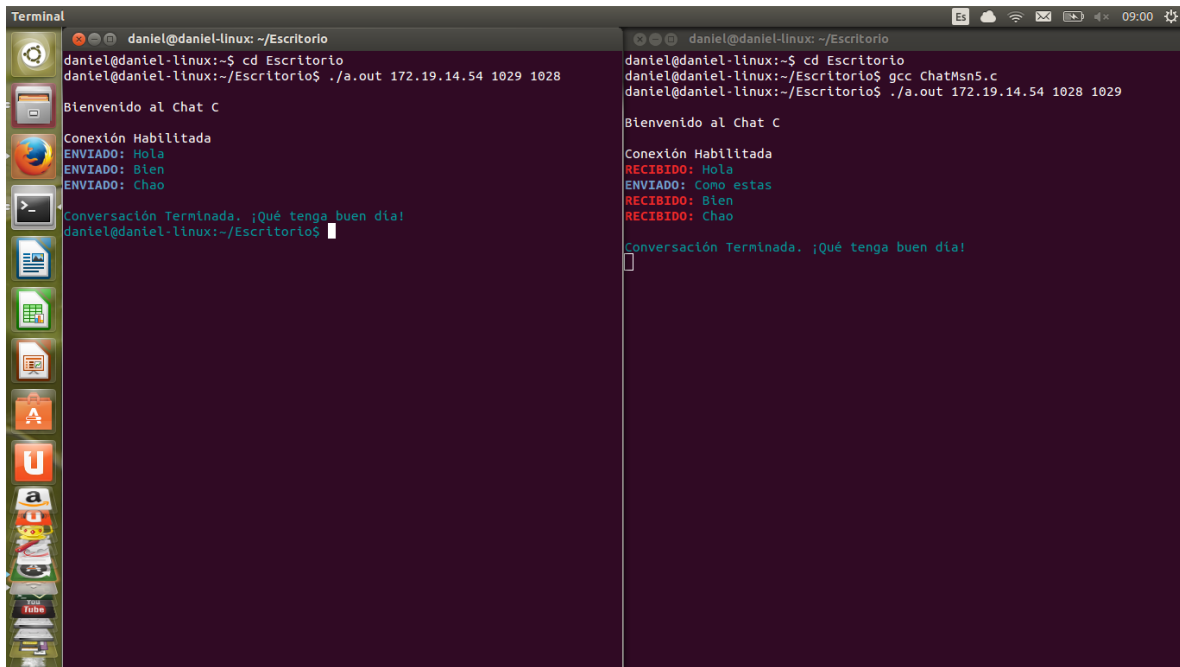
6- Para agregar un usuario solo debe digitar la opción 2 e introducir los datos.



7- Para poder ver los contactos digite el 1



- 8- Para iniciar una conversación digite el 3, y el usuario con quien desea hablar y así puede comenzar la conversación.



```
daniel@daniel-linux: ~/Escritorio
daniel@daniel-linux:~$ cd Escritorio
daniel@daniel-linux:~/Escritorio$ ./a.out 172.19.14.54 1029 1028

Bienvenido al Chat C

Conexión Habilitada
ENVIADO: Hola
ENVIADO: Bien
ENVIADO: Chao

Conversación Terminada. ¡Qué tenga buen día!
daniel@daniel-linux:~/Escritorio$
```

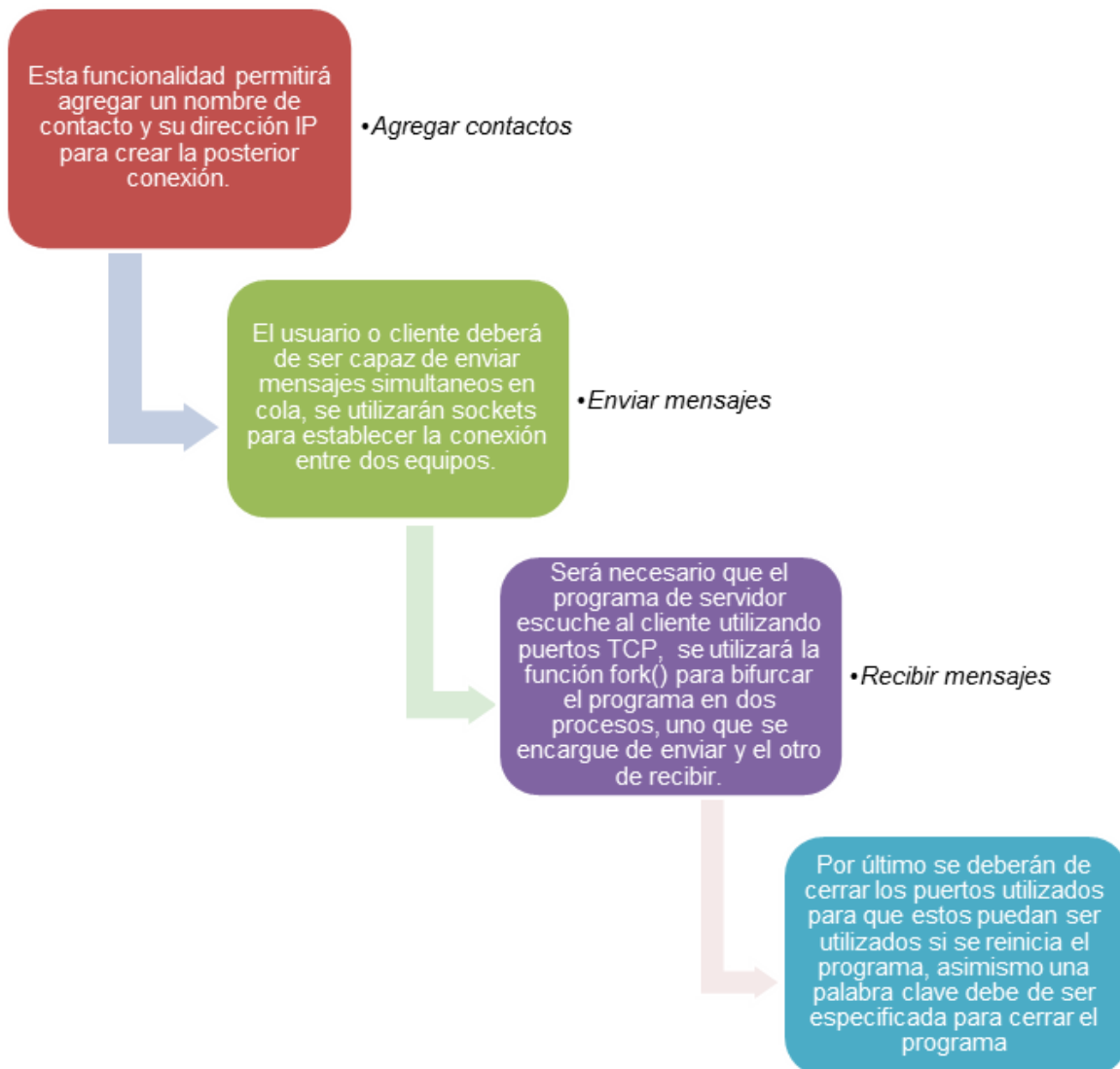
```
daniel@daniel-linux: ~/Escritorio
daniel@daniel-linux:~$ cd Escritorio
daniel@daniel-linux:~/Escritorio$ gcc ChatMsn5.c
daniel@daniel-linux:~/Escritorio$ ./a.out 172.19.14.54 1028 1029

Bienvenido al Chat C

Conexión Habilitada
RECIBIDO: Hola
ENVIADO: Como estas
RECIBIDO: Bien
RECIBIDO: Chao

Conversación Terminada. ¡Qué tenga buen día!
█
```

## Diseño del Programa



## Análisis de Resultados

En esta tarea programada se logró implementar la funcionalidad del uso de sockets y sus funciones, ya que es posible enviar y recibir mensajes. La conexión entre dos computadoras logra cerrar los sockets de servidor y cliente en el momento que alguno de los usuarios digita la palabra de despedida “Chao”.

El usuario puede enviar mensajes los cuales cuentan con un color al desplegarse en la pantalla, estos son el azul para enviados y rojo para recibidos según lo solicitado por el profesor. Los puertos a utilizar en el programa son los utilizados por otros sistemas de mensajería, aunque si se desea, se pueden definir nuevos puertos.

La funcionalidad de agregar amigos funciona correctamente creando un archivo.txt al cual se le pide ingresar ciertos datos para el registro.

Se presenta el problema de que la funcionalidad de usuarios afecta a la conexión entre dos computadoras, ya que en ocasiones falla la comunicación. No es posible correrlo entre dos terminales al mismo tiempo.

## Conclusión Personal

Esta tarea programada fue una gran experiencia, ya que esta permitió extender los conocimientos adquiridos a lo largo de la carrera. Además de la implementación de un nuevo paradigma, la utilización de un nuevo lenguaje de programación como lo es C fue interesante y desafiante, lo que ayudo a fomentar la investigación y el aprendizaje.

Por otro lado se aprendió a manejar el uso de sockets para una conexión entre dos computadoras, a utilizar la IP y los puertos.

Además, el uso de Ubuntu como nuevo sistema operativo para realizar la tarea programada, fue un poco difícil adaptarse al cambio, ya que por lo general se esta acostumbrado a Windows. Una vez adaptados al sistema operativo se aprendió mucho en el manejo archivos, la terminal, etc.

Se puede decir que esta tarea fue de gran provecho, ya que los conocimientos adquiridos valen mucho.

## Lecciones aprendidas

- Se presentó problemas con la manipulación de los puertos debido que estos se “ensuciaban” en la memoria. Por lo que este problema no se pudo resolver del todo, ya que era necesario utilizar otros puertos o reiniciar la computadora para poder volver a correr el programa.
- También se presentó el problema para poder terminar el proceso de la conexión de los sockets, para esto se implemento la función Kill lo que logró, si así se puede decir, matar el proceso y pudiera terminar dicho proceso.
- Se logró observar la dificultad del manejo de los puertos, ya que al ser utilizados quedaban abiertos, para esto se usó la función close() lo que permite cerrar esos puertos para seguir usándolos luego.



## Bibliografía

Programación Básica de Sockets en Unix para Novatos (s. f.). Recuperado de <http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html>

Color Bash Prompt - ArchWiki (s. f.). Recuperado de [https://wiki.archlinux.org/index.php/Color\\_Bash\\_Prompt](https://wiki.archlinux.org/index.php/Color_Bash_Prompt)

Bonet, E. (s. f.). Introducción a la programación con sockets en C Recuperado de <http://informatica.uv.es/iiguia/R/apuntes/laboratorio/Uso.pdf>

CREACION DE PROCESOS. FORK. (s. f.). Recuperado de [http://sopa.dis.ulpgc.es/ii-dso/leclinux/procesos/fork/LEC7\\_FORK.pdf](http://sopa.dis.ulpgc.es/ii-dso/leclinux/procesos/fork/LEC7_FORK.pdf)

La Función fork (s. f.). Recuperado de <http://nereida.deioc.ull.es/~pp2/perlexamples/node66.html>

Multitarea en C usando Fork para OS tipo Unix (s. f.). Recuperado de <https://www.lastdragon.net/?p=180>

Procesos en C. Función kill (s. f.). Recuperado de <http://www.mailxmail.com/curso-informatica-sincronizacion/procesos-c-funcion-kill>

Programación de sockets en C de Unix/Linux (s. f.). Recuperado de [http://www.chuidiang.com/clinux/sockets/sockets\\_simp.php](http://www.chuidiang.com/clinux/sockets/sockets_simp.php)

MANUAL DE SOCKETS EN C (s. f.). Recuperado de <http://www.fic.udc.es/files/asignaturas/56XR/files/ManualSocketsC-2009.pdf>