



2014

***Paola Solano.  
Veronica R. Vargas.  
Gabriela Reyes.***

## DOCUMENTACIÓN

Primera Tarea Programada | Lenguajes de Programación

## Tabla de contenido

Descripción del Problema.....	2
Diseño del programa .....	3
Estructura de Cliente y Servidor del programa. ....	3
Estructura de la lectura de contactos. ....	4
Diagrama de Flujo. ....	6
Librerías Usadas .....	7
Análisis de Resultados .....	8
Manual de Usuario .....	9
Conclusión Personal .....	12

## Descripción del Problema

---

Se desea crear un programa que simule una mensajería instantánea similar a Google Chat o Facebook Messenger. Este programa deberá permitir la comunicación entre computadoras, para ello se usarán sockets; mecanismos que permiten la comunicación entre dispositivos por medio de comunicaciones orientadas a conexión (TCP/IP) o no orientadas a conexión (UDP).

El programa está deberá tener varias funciones como lo son:

**Agregar contactos:** Para que las computadoras se puedan comunicar, se necesitará el nombre del contacto que se desea agregar, así como la IP de la computadora y el puerto.

**Enviar mensajes:** El programa tendrá la opción de enviar mensajes a sus contactos.

**Recibir mensajes:** Se tendrá la opción de recibir mensajes, para lo cual se deberá tener un puerto que debe ser fijo, para escuchar las comunicaciones entrantes.

## Diseño del programa

---

Durante la elaboración del programa se realizaron varios cambios. El trabajo está dividido en dos programas: `Ultima_Union.c` y `Agregar_Buscar_Usuario.c`, en el primero se encuentran las funciones de cliente-servidor y en el otro se ejecuta la lectura y escritura de archivos. El Main se encuentra en este último.

### Estructura de Cliente y Servidor del programa.

**Colores:** se utiliza el `#define` para la definición de colores que son utilizados para distinguir entre los mensajes enviados y los recibidos durante la ejecución del programa.

**Código del archivo config:** dentro del método de `Servidor()`, exista la llamada a una función, `Obtener_Puerto()`, que retorna un entero que se almacena en la variable `puerto` esta información es enviada al segmento del struct correspondiente al puerto. El método `Obtener_Puerto()`, abre el archivo de texto que contiene el número de puerto, "`Archivo_Config`". Obtiene el número por medio de la función `fgets`, que almacena la información en un arreglo de caracteres. Posteriormente el dato es leído, y se convierte a un entero, por medio de la función `atoi`, la cual se encuentra incluida en la librería `stdlib.h`, el método retornara el entero correspondiente.

**Código del Servidor:** Contiene el puerto asignado al servidor, comienza con la declaración de las variables que van a ser utilizadas durante esta función, así como el arreglo para recibir los mensajes, está formado por:

- **Estructura de socket:** Inicia con la creación del `socket` del servidor (SKT), si el `socket` no funciona envía un mensaje de error, seguidamente se definen los protocolos del `socket` en donde se asigna el puerto y la dirección IP. Luego se crea el `bind` que es el encargado de abrir la conexión y permitir que se encuentre el puerto que está activo, sino funciona envía un mensaje de error. Se crea el `listen` para escuchar, y finalmente la función del `accept` para la aceptación de la conexión.
- **Recibimiento de mensajes:** para que el servidor reciba mensajes se implementó la función de `recv()`, después, se implementa un `while(1)` que indica que mientras, va a recibir mensajes del cliente, este mismo `while` contiene un `if` que realiza una comparación del mensaje que se recibe; si el mensaje es igual a 1, el `socket` se cierra para la cual se ejecuta la función



`kill()` que ayuda a terminar el procedimiento del `fork()` “matando” el proceso hijo y el proceso padre.

**Código del cliente:** Es similar al del servidor, con el cambio de que en vez de recibir mensajes, este código envía mensajes, esto se realiza mediante un `fgets()` que ayuda a obtener el texto que el usuario desea enviar, luego se ejecuta la función `send()` que es la encargada de enviar el mensaje.

### Estructura de la lectura de contactos.

**Struct:** se realiza una definición de un struct llamado *Lista\_Contactos*, esta estructura contiene tres punteros a arreglos de tipo *char* que hacen referencia al nombre, dirección IP y puerto del contacto.

**Es\_Usuario:** este método recibe un puntero a un arreglo, el cual contiene la información del contacto que se encuentra almacenada en el archivo de texto y un puntero al arreglo que contiene el nombre del contacto con el cual el usuario desea establecer comunicación. La función compara cada uno de los elementos de los dos arreglos para determinar si el nombre es el mismo. Si el nombre del contacto es el mismo se retorna 1 que significa verdad, si el nombre es diferente se retorna 0 que indica falso.

**Manejo\_Contactos:** la función separa la información que se encuentra almacenada en el arreglo. Se declaran tres arreglos que van a almacenar la información respectiva. La separación se realiza por medio de ciclos while. Primero se coloca el nombre del contacto en un arreglo llamado *NOM*, luego la IP en el arreglo *I* y el puerto en el arreglo *P*. Cuando se terminan los ciclos se procede a colocar en el struct la dirección de memoria de los arreglos. Para poder acceder a la información correspondiente. Asimismo el arreglo de *P* que almacena el puerto se convierte en un entero por medio de la función `atoi`, que retorna el valor en un entero. Finalmente se imprime la información del contacto en pantalla y se invoca al método `fork`, el cual permite ejecutar el método del *Servidor* y el método del *Cliente*, según sea el caso.

**Consulta\_Contactos:** este método recibe un puntero a una dirección de archivo de texto, el cual posee toda la información de los contactos del usuario. Lo que hace este método es leer cada línea del archivo, e imprimir el nombre del contacto.

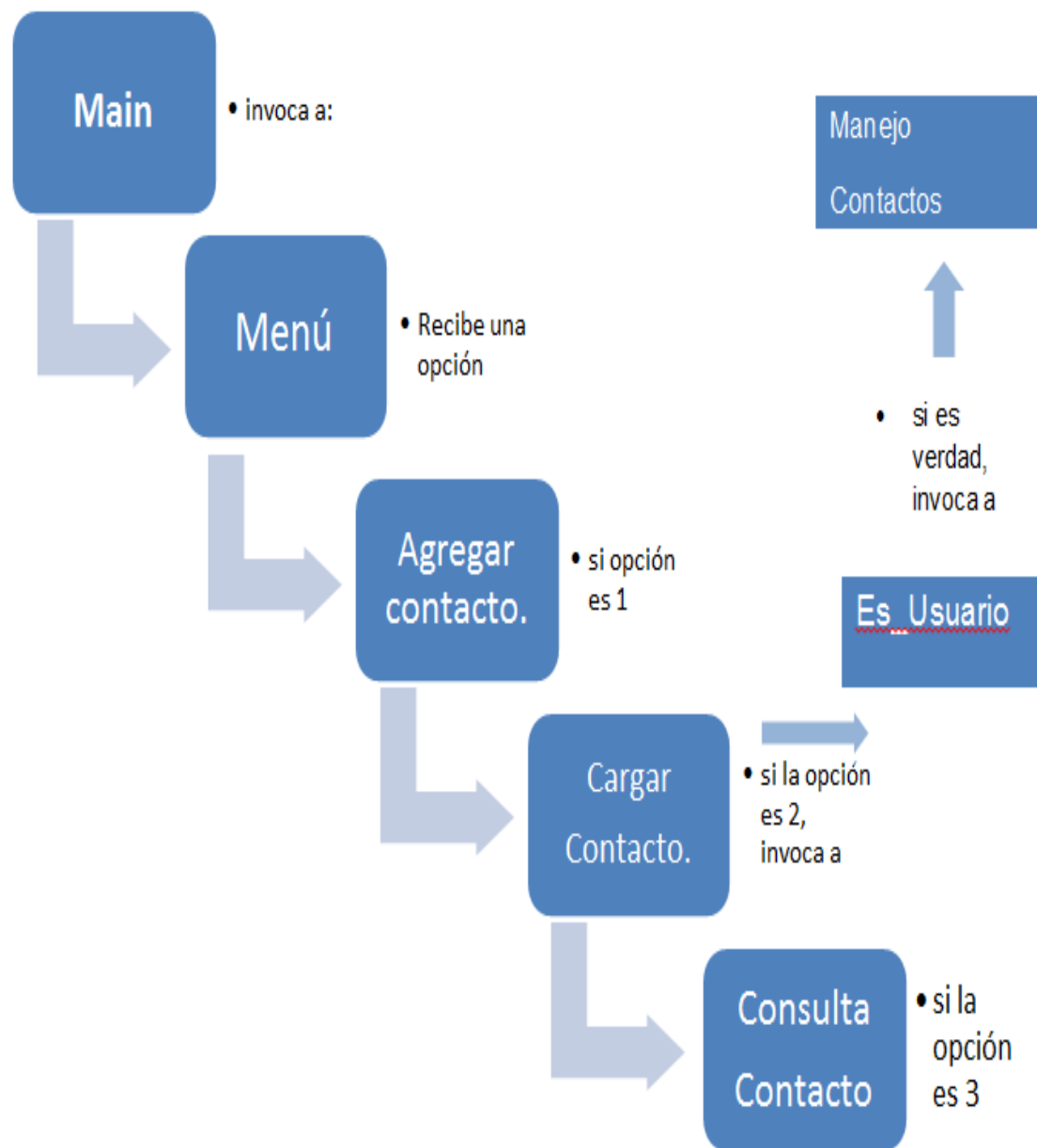
**Cargar\_Contactos:** se recibe un puntero que almacena la dirección del archivo de texto. El programa le indicará al usuario que ingrese el nombre del contacto con el cual desea iniciar una conversación. El nombre del contacto se almacenará en el arreglo *N*, el archivo se recorrerá hasta que se encuentre el nombre del contacto,

la función que indica que se encontró el nombre del contacto es la de *Es\_Usuario*, la cual se describió anteriormente. Si se encontró al amigo, el sistema procederá a invocar al método de *Manejo\_Contactos*, el cual almacenará la información en el struct. Si no se encuentra el contacto el método retornara 0.

**Agregar\_Contacto:** este método recibe el apuntador al archivo de texto. Esta función abre el archivo de texto anteriormente mencionado, y almacena los datos nuevos. El método “fseek(F,4-sizeof(int),SEEK\_END)”, coloca la información nueva al final del archivo. “F” es el puntero al archivo de texto. A continuación se presentan, tres variables de tipo arreglo de char, que contendrán la información ingresada en los “scanf”. El guión (“-”), es el que separa la información de los archivos. Se utiliza la función “fputs”, para almacenar los datos e el archivo de texto correspondiente.

**Main():** se imprime en la consola un menú con las opciones que puede digitar el usuario. Se obtiene la información del usuario por medio del *scanf*. Posteriormente se crea un puntero a un directorio, el cual permite abrir el archivo de texto por medio de la invocación a la función *fopen*, la cual tiene dos parámetros nombre del archivo y el *opentype*. El documento en el que se encuentra almacenada la información de los contactos se llama “ArchivoDeAlmacenamiento”. El *opentype* elegido fue el “r+”, este comando permite abrir el archivo para lectura y escritura de datos. Si el archivo no es encontrado, el apuntador del fichero es nulo, por lo que en la pantalla se mostrará un mensaje indicando que existe un error en el archivo. Si no se presenta ningún problema, se procede a ejecutar la opción ingresada por el usuario. Si la opción es la número 1 se le envía el puntero al fichero al método *Agregar\_Contacto*, descrito anteriormente. Si la opción es 2, se envía el puntero al fichero a *Cargar\_Contactos* y si la opción es 3 se invoca a la función *Consulta\_Contactos*. Si la opción no es válida se pedirá que ingrese nuevamente los datos. Luego se cierra el archivo por medio del método “fclose”.

## Diagrama de Flujo.



## Librerías Usadas

---

Las librerías son archivos de la biblioteca estándar en C que contienen un conjunto de funciones, tipos relacionados y macros, que son proporcionados para facilitar la implementación de los programas. Todas las librerías son declaradas como archivos cabeceras. Para la creación de este programa de mensajería se utilizaron las siguientes:

**Stdio.h:** Contiene las definiciones de macros, constantes, declaraciones de funciones y definición de tipos, usados por operaciones estándar de entrada y salida. Las funciones utilizadas de esta librería en el programa son: perror, scanf, fclose, fopen, printf, fgets, fseek, fputs.

**String.h:** Permite trabajar con cadenas de caracteres y algunas operaciones de manipulación de memoria. De esta librería se utilizó la función bzero para la limpieza de arreglos y su uso posterior.

**Sys/Sockets.h:** Librería que permite la comunicación entre dos programas o procesos. Contiene las funciones de sockets, y sus correspondientes constantes. Funciones utilizadas: socket, connect, bind, accept, listen, recv, send, close, entre otras.

**netinet/in.h:** Nos permite usar la variable errno.

**sys/types.h:** Contiene definiciones para permitir la portabilidad de los programas de BDS.

**stdlib.h:** Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras. Funciones utilizadas: atoi,...

**signal.h:** Especifica como un programa maneja las señales mientras se ejecuta. Funciones utilizadas: Kill, para dar por finalizado el proceso de I socket

**netdb.h:** Librerías para el uso de primitivas Unix.

Estas librerías son utilizadas tanto para el manejo de archivos como para la conexión entre el cliente y el servidor y la creación de los sockets. Además de funciones propias del programador.



## Análisis de Resultados

---

En el método de `Agregar_Usuarios`, no se logró que se almacenara automáticamente la información nueva en el arreglo de `struct`. Se tiene que invocar nuevamente al método de `Leer_Usuarios`, que abre el archivo de texto, para almacenar la información.

Se creó el archivo `Makefile`, pero al compilar el `make`, el archivo presenta unos errores, los cuales indican que los directorios no se encuentran, se buscó en internet como solucionar el problema pero no se encontró la solución.

Al finalizar la conversación el usuario debería de digitar 1 si quiere salirse, sin embargo el cierre de sockets funciona solo por una parte. Sin embargo, separados si funcionan correctamente con la función `kill`.

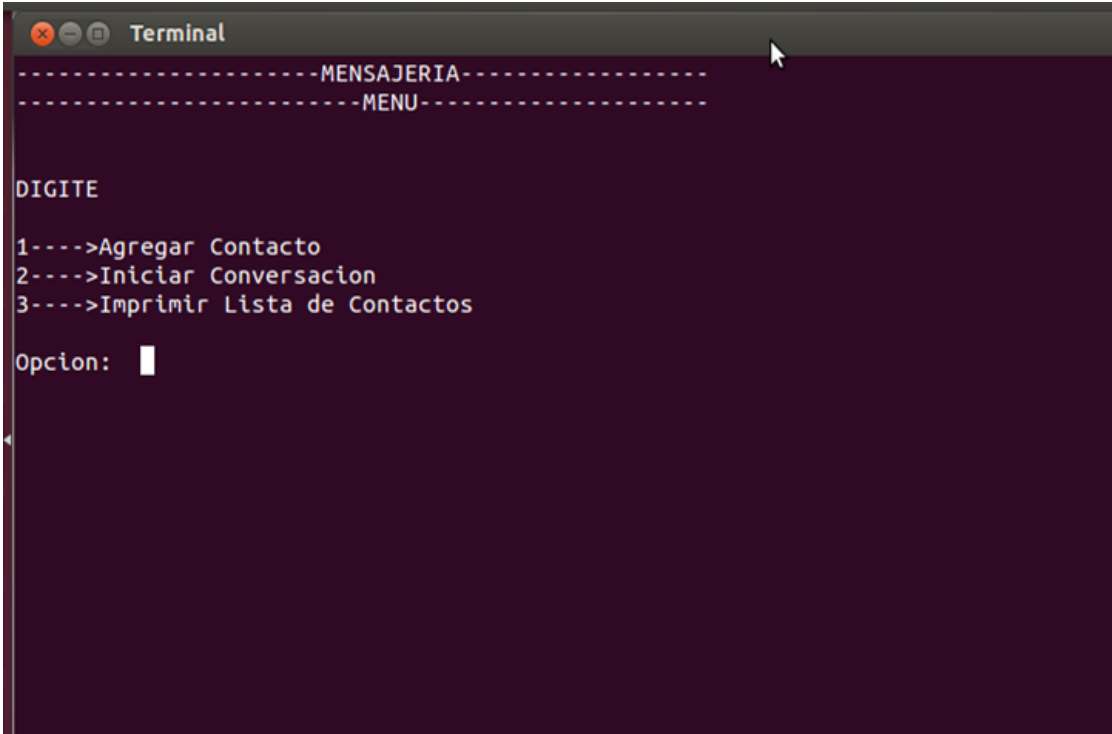
# Manual de Usuario

---

Este manual permitirá aprender a utilizar el programa de Mensajería, en él se presentan las funciones de dicha programa y las instrucciones de uso de las mismas.

Para compilar el programa es necesario acceder a la terminal, y ubicarse a la carpeta donde se ha guardado el archivo del programa y desde ahí dar el comando “make”. Este comando genera el un ejecutable, gracias al archivo makefile del programa.

Una vez realizados los pasos del programa nos aparecerá el menú del programa, el cual nos presenta tres opciones, las cuales se eligen presionando el número de la acción que desea realizar, como se muestra en la siguiente imagen.



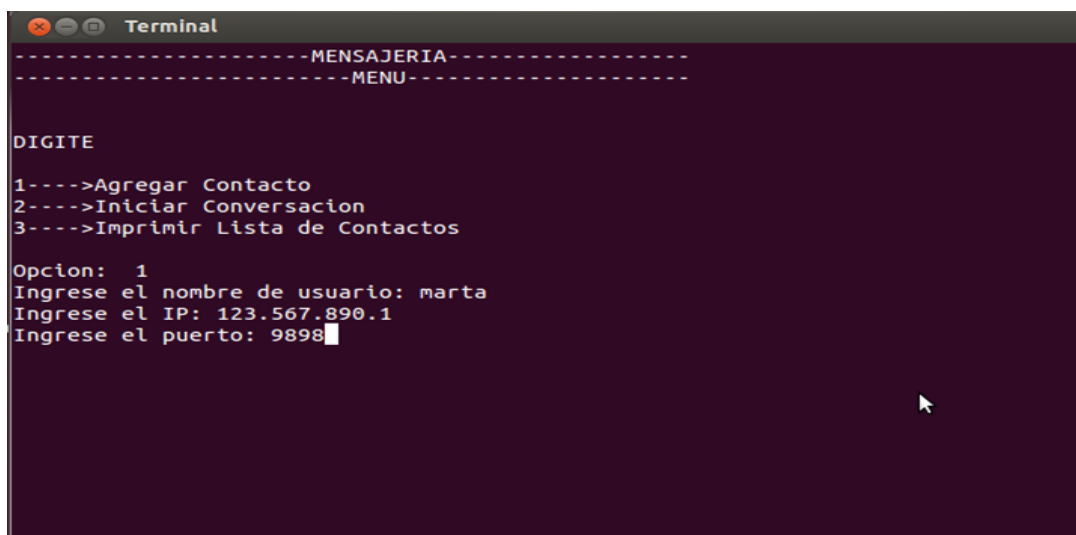
```
Terminal
-----MENSAJERIA-----
-----MENU-----

DIGITE
1---->Agregar Contacto
2---->Iniciar Conversacion
3---->Imprimir Lista de Contactos

Opcion: 
```

Al seleccionar la primera opción podrá agregar usuarios al archivo de Contactos, para ello deberá ingresar el nombre de usuario, seguido del IP del nuevo contacto

y su respectivo puerto, por último al presionar ENTER se guardara el nuevo contacto y aparecera nuevamente el menú. Como se puede observar en la siguiente imagen.



```
Terminal
-----MENSAJERIA-----
-----MENU-----

DIGITE

1---->Agregar Contacto
2---->Iniciar Conversacion
3---->Imprimir Lista de Contactos

Opcion: 1
Ingrese el nombre de usuario: marta
Ingrese el IP: 123.567.890.1
Ingrese el puerto: 9898
```

Al seleccionar la segunda opción podrá iniciar la conversación, para ello es necesario Ingresar el nombre del contacto con el cual desea conversar, el programa buscara el usuario en la lista de contactos en caso de no estar, imprime que no está y deberá seleccionar la agregar para luego iniciar conversación.

Para terminar con la conversación presiona el número 1 y está finalizara e imprimirá nuevamente el menú.



```
Terminal

1---->Agregar Contacto
2---->Iniciar Conversacion
3---->Imprimir Lista de Contactos

Opcion:
2
Ingrese nombre usuario Gaby

Usuario encontrado

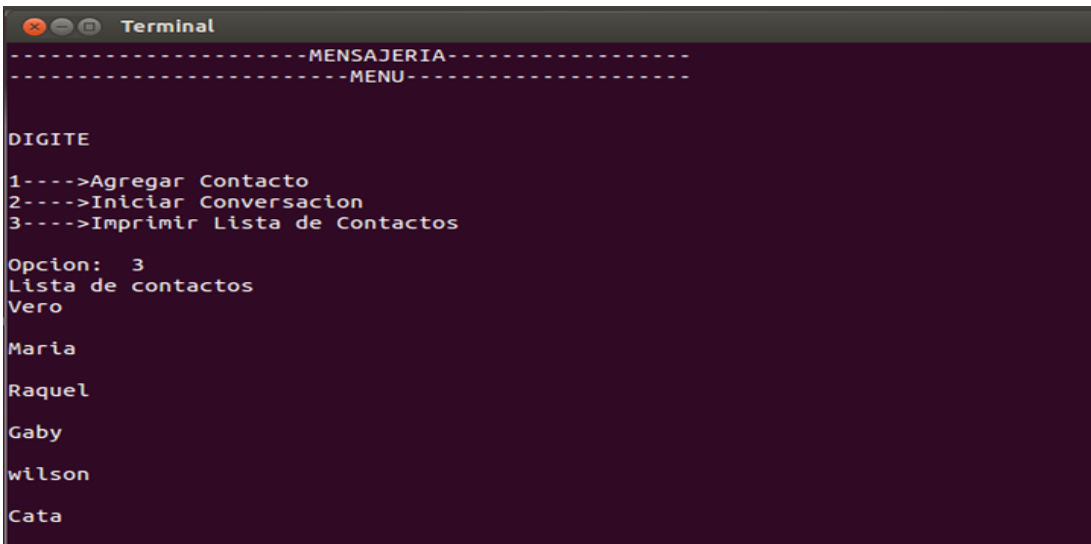
Nombre: Gaby
IP: 192.168.153.140
Puerto: 8888

-----Conectado-----

Ingrese mensaje . . .
Mensaje recibido:kde

Hola
```

Como tercera opción se encuentra la impresión de la lista de contactos, la cual dará a conocer todos los contactos que han sido guardados en el Messenger.



```
Terminal
-----MENSAJERIA-----
-----MENU-----

DIGITE
1---->Agregar Contacto
2---->Iniciar Conversacion
3---->Imprimir Lista de Contactos

Opcion: 3
Lista de contactos
Vero
Maria
Raquel
Gaby
wilson
Cata
```

Y por último, si se presiona alguna opción no especificada anteriormente, el programa la tomara como invalida y responderá con un mensaje de error, y seguidamente mostrara nuevamente el menú.

## Conclusión Personal

---

La elaboración de esta tarea programada ayudó a que el equipo de trabajo conociera más acerca de las herramientas de programación que posee Linux, así como el entendimiento de código en lenguaje C.

Los temas principales para búsqueda de información fueron el uso de sockets así como el manejo de archivos en C.

**Uso de sockets:** Se buscó información del uso de sockets, entendiendo con esta, el funcionamiento de la comunicación por medio de puertos e IP de las computadoras. Además, se aprendió del uso de los fork () y los procedimientos y funciones de este.

**El manejo de archivos en C:** el grupo de trabajo se informó, entendió y aplicó la forma en que C gestiona la información de los archivos de texto. Asimismo se complementó lo visto en clase con la realización de la tarea. Ya que se hizo uso de punteros, definición de “struct”, arreglos, función de entrada y salida de datos.