

# DOCUMENTACIÓN EXTERNA TP1.

Brandon Carter 201114390

Andrés Maroto 201128337

David Murillo 201108648

Adrián Siles 201136841



*Curso:  
Lenguajes de  
Programación*

*Profesor:  
Andrei Fuentes.*

# Tabla de Contenidos

• Descripción del problema.....	2
• Diseño del programa .....	3
• Librerías usadas.....	6
• Análisis de resultados.....	7
• Manual de usuario.....	8
• Conclusión personal.....	13
• Bibliografía Consultada.....	14

# Descripción del Problema

El problema que se presentó en esta tarea programada, fue el de realizar un clon de la aplicación WhatsApp utilizando el lenguaje C dentro del sistema operativo Linux.

Se notaron dos problemas principales para desarrollar este programa. El primero era la comunicación en red de dos computadoras para enviarse mensajes utilizando sockets. Y el segundo era la división de los procesos de lectura y escritura para que estos se ejecuten simultáneamente mediante el uso del `fork()`.

Estos son los subproblemas que se relacionan con los problemas principales.

- Agregar contactos de tal manera que se pueda escribir una serie de datos relacionados al mismo necesarios para realizar la conexión y el envío de mensajes como los son el nombre del contacto, la dirección ip y el puerto.
- Que el programa tenga la capacidad de ser cliente y servidor a la vez. Es decir que pueda recibir(estar escuchando en algún puerto) y realizar mensajes de manera simultánea, gracias a la bifurcación de procesos(`fork()`)
- .
- Utilizar colores distintos en las letras de recibido y enviado de mensajes para diferenciarlos.

# Diseño del Programa

El programa está diseñado de tal manera de que el usuario interactúe directamente con un menú que le ofrece las opciones de agregar un nuevo contacto(donde se ingresará el nombre, puerto e ip), ver la agenda de los contactos existentes, guardar el puerto de escucha local y obtener los datos del contacto con el que se desea comunicar para que se realice el proceso de conexión y envío de mensajes.

En el momento que inicie el programa se realiza la lectura del archivo por medio de la función `fopen()` donde se van a guardar los contactos. Lee línea por línea el archivo guardando cada línea temporalmente en la variable de arreglo llamada “buffer”, que además también es utilizada para salvar temporalmente el nombre que el usuario digita cuando quiere captar sus datos para enviar mensajes, y mete en el arreglo de tipo estructura los contactos que ya están almacenados en el archivo hasta que encuentre el final del archivo(`feof`), todo esto para que a la hora de realizar la búsqueda de los datos del contacto con el que se desea comunicar se realice la comparación por medio de índices utilizando la función `strcmp()`. En caso de que sean iguales guarda el nombre, la dirección ip y el puerto del contacto en variables globales usando `strcpy()`(específicamente en nombre y puerto).

Si se elige la opción de nuevo contacto, agrega la estructura del contacto nuevo al arreglo, en caso de que inmediatamente desee establecer conexión con el mismo, y por último realiza el guardado en el archivo por medio de la función `fprintf`.

Al elegir la opción ver la lista de contactos se recorre el arreglo comparando el nombre que el usuario ingresa con el nombre que se encuentra en el índice del arreglo imprimiendo los nombres de cada contacto del arreglo.

Para guardar o actualizar el puerto de escucha local se realiza el mismo proceso que al ingresar un nuevo amigo, solo que el dato se guarda en un archivo distinto al de la agenda, con la particularidad de que solo va a existir un sólo dato o puerto, todo esto gracias a que para escribir en el archivo se

utiliza “w+”, que quiere decir que va a sobrescribir sobre los datos ya existentes en el archivo.

Al escoger la alternativa de continuar envío y recepción de mensajes se inicia el proceso de intercomunicación de los sockets.

Cuando terminamos de guardar los contactos y escoger el contacto para establecer comunicación empieza el proceso del fork() que devuelve un número para identificar el proceso hijo del proceso padre. Dividimos los procesos en el main de esta manera:

- Usamos un solo fork en el main.
- El proceso padre del fork (servidor) se encarga de recibir mensajes del socket, y los escribe a la pantalla.
- El proceso hijo del fork (cliente) se encarga de leer lo que escriba el usuario del stdin, y manda esos mensajes por sockets al destinatario correspondiente.

De esta manera cada computadora que ejecute el programa va a ser cliente y servidor a la vez mediante la bifurcación de procesos.

A la función servidor le enviamos de parámetro el puerto local y a la función cliente le enviamos de parámetros la ip y el puerto del cliente obtenidas anteriormente.

- **Función Servidor:**

Inicializamos la variables que contienen los identificadores de cada proceso y las variables tipo struct sockaddr\_in que van a contener los datos del puerto e ip del cliente y del servidor.

Hacemos un bind para unir el socket idSocket = socket (AF\_INET, SOCK\_STREAM, 0) con la estructura que contiene los datos del servidor.

bind(idSocket,(struct sockaddr \*)&DireccionSocketServidor,sizeof (DireccionSocketServidor).

Luego le indicamos con el listen que empiece a escuchar solicitudes de clientes: listen(idSocket,8) El 8 es el numero maximo de clientes en cola.

Después de esto aceptamos a un determinado cliente con la función accept: idConexionCS = accept (idSocket,(struct sockaddr \*)&DireccionSocketCliente, &largodircliente); Nos devuelve un id que identifica la conexión.

Por último ya estamos listos para leer mensajes, esto se hace con la función `read` que lee de un char pointer buffer que contiene lo escrito por el cliente: `read(idConexionCS,buffer1,255);` Este `read` dentro de un ciclo infinito para que lea todos los mensajes posibles.

Por último imprimimos el mensaje formateado en color rojo: `printf("\033[2K\r\033[01;34m""Mensaje : \033[00;34m %s",buffer1); // \033[00;37m azul`

- **Función Cliente:**

La función cliente también necesita de una estructura tipo `sockaddr_in` para unirla al identificador del socket `idSocket = socket (AF_INET, SOCK_STREAM, 0);`

Adicionalmente se usa una estructura tipo `hostent` para manejar el ip y mandarselo como parámetro a la estructura `sockaddr_in`.

El cliente usa la función `connect()` para realizar el intento de conexión, en ese momento la función `accept()` del servidor retorna con un parámetro que es un nuevo descriptor de socket, el cual se utiliza para realizar la transferencia de datos por la red con el cliente.

`connect(idSocket,(structsockaddr*)&DireccionSocketServidor,sizeof(DireccionSocketServidor)).`

Por último pasamos al `write` para escribir mensajes en el buffer del `stdin`: `idwrite = write(idSocket,buffer1,strlen(buffer1));`

La función `bzero` limpia el buffer y la función `fgets` es la encargada de agarrar los datos del `std in`.

Luego imprime en la pantalla lo escrito en formato color azul: `printf("\033[A\033[2K\033[01;31m""Mensaje Enviado: \033[31m %s",buffer1); // \033[31m color rojo.`

Utilizamos un archivo `Makefile` para facilitar la compilación de los archivos hechos en C. Por medio del llamado “`make TP1`”, buscará dentro de la carpeta donde esté localizado la terminal, y compilara el archivo `TP1.c` y generar un ejecutador con el mismo nombre. Además, si existieran cambios en el archivo `TP1.c`, y se quisiera compilar de nuevo, se debe borrar el ejecutable. Para borrar el ejecutable simplemente se debe llamar por medio del “`make clean`”, el cual borrara el ejecutable del `TP1`.

# Librerías Usadas

1. `<stdio.h>`: Librería Standard para el manejo de entrada y salida.
2. `<unistd.h>`: Esta librería que contiene la función `fork()`, para bifurcar los procesos, en proceso padre y proceso hijo.
3. `<string.h>`: Librería utilizada para el manejo y comparación de “strings” (hileras de caracteres). Algunas de las funciones que se usaron fueron `bzero`, `bcopy` para limpiar los mensajes y capturar los nuevos mensajes, además de `strcpy()`, `strcmp()` y `puts()` para imprimir cadena de caracteres, comparar y copiar a alguna variable.
4. `<stdlib.h>`: Librería utilizada para la función `atoi()`, que se utilizó para pasar caracteres a enteros los puertos de un contacto determinado que se recupere de la agenda.
5. `<sys/types.h>` - `<sys/socket.h>`: Librería para la creación y el manejo de los Sockets. Algunas de las funciones que se utilizaron: `open`, `accept`, `bind`, `connect` entre otras.
6. `<netinet/in.h>`: Librería usada para el manejo de los protocolos de Internet, de manera que almacenan direcciones de cada familia de protocolos. Para luego ser usados por las instancias de los sockets.
7. `<netdb.h>`: Librería para el uso de BD en red, se utilizó para la función `gethostbyname()`, para obtener la dirección IP del servidor.
8. `<signal.h>`: Librería para manejar señales de terminación del programa en caso de error. En nuestro caso utilizamos `exit(1)` siempre que tuviéramos un error en la creación de los sockets.

# Análisis de Resultados

Se cumplieron con los objetivos establecidos en el documento de la tarea:

- Almacenar contactos
- Buscar contacto deseado
- Guardar el puerto de escucha local en un solo archivo
- Identificación de mensajes de envío y recibido.
- Capacidad de enviar mensajes de forma continúa.
- Capacidad de recibir mensajes de forma continúa.

El objetivo no alcanzado que tuvimos es que a la hora de ejecutar el proceso fork para la bifurcación de procesos servidor y cliente, no se logra establecer comunicación. Si lo ejecutamos primero con mensajes fijos, o sea, mandarle desde un cliente un mensaje a un servidor simple si sirve. Pero no funciona correctamente cuando utilizamos el fork. Tampoco nos da un error de sockets por lo que fue difícil identificar donde está la causa del problema.

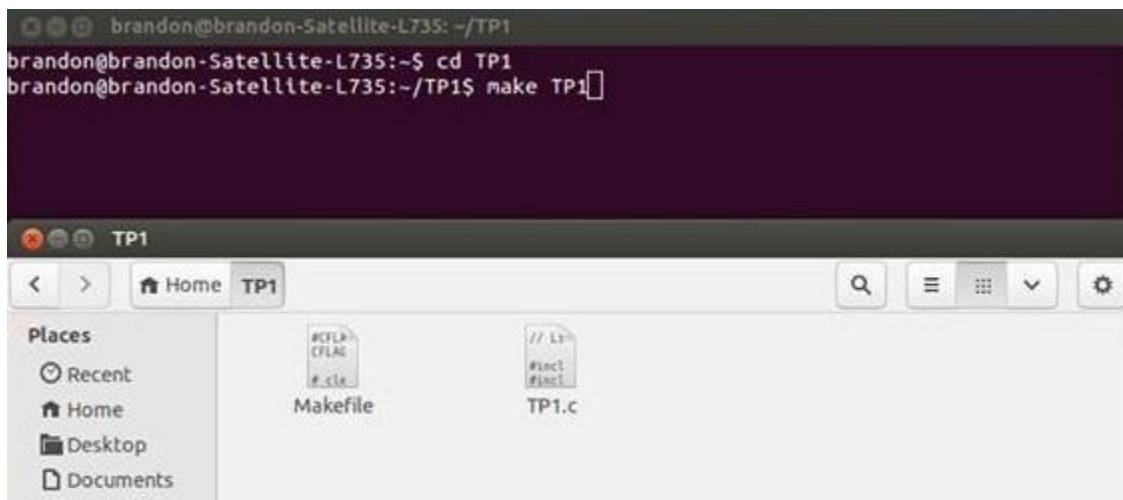


# Manual de Usuario

Para iniciar el programa, se comienza por abrir la terminal del sistema operativo. Lo primero que se debe hacer, es localizar el archivo donde está ubicado el programa, para ello utilizamos el comando “ls” que nos dará un menú de los archivos que se encuentran dentro de la carpeta actual así:

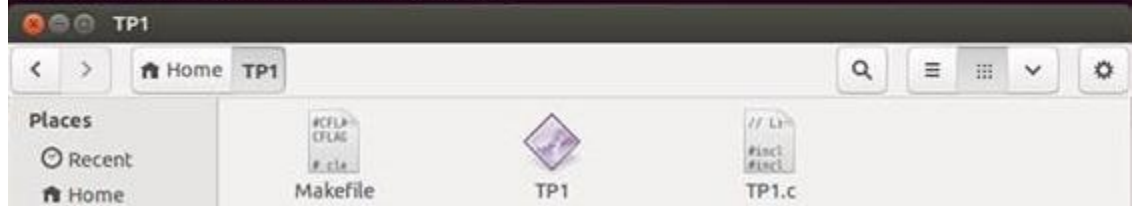
```
brandon@brandon-Satellite-L735:~$ ls
Desktop  examples.desktop  Pictures  Templates  Videos
Documents holamundo.c~      Project  TP1
Downloads Music          Public   Ubuntu One
brandon@brandon-Satellite-L735:~$
```

Con el comando “cd”, seguido del nombre del archivo donde se encuentra ubicado el programa, podemos volver a utilizar el comando “ls” para obtener el nuevo menú de la carpeta visitada. En este caso, el programa llamado TP1.c se encuentra en la carpeta “TP1”.



Como vemos en la imagen anterior, existe un archivo llamado “Makefile”, y este será el encargado de compilar el programa. Por lo tanto, para compilar, escribimos “make” y esto creará un archivo ejecutable con el mismo nombre (TP1).

```
brandon@brandon-Satellite-L735:~$ cd TP1
brandon@brandon-Satellite-L735:~/TP1$ make TP1
cc -Wall -g TP1.c -o TP1
TP1.c: In function 'cliente':
TP1.c:211:41: warning: suggest parentheses around assignment used as truth value [-Wparentheses]
TP1.c: At top level:
TP1.c:253:6: warning: return type of 'main' is not 'int' [-Wmain]
brandon@brandon-Satellite-L735:~/TP1$
```



En el caso de que se quiera eliminar el archivo ejecutable, se debe de enviar “make clean” así

```
brandon@brandon-Satellite-L735:~/TP1$ make clean
rm -f TP1
brandon@brandon-Satellite-L735:~/TP1$
```

Ahora sí, para ejecutar el programa, se debe llamar al ejecutable con “./TP1”. Esto iniciará el programa.

```
brandon@brandon-Satellite-L735:~$ cd TP1
brandon@brandon-Satellite-L735:~/TP1$ make TP1
cc -Wall -g TP1.c -o TP1
TP1.c: In function 'cliente':
TP1.c:211:41: warning: suggest parentheses around assignment used as truth value [-Wparentheses]
TP1.c: At top level:
TP1.c:253:6: warning: return type of 'main' is not 'int' [-Wmain]
brandon@brandon-Satellite-L735:~/TP1$ ./TP1
```

Al iniciar el programa, el menú de acciones se mostrará, dando como opciones crear un contacto nuevo, ver la agenda de contactos, escoger contacto para comunicarse y empezar una conversación. Cada opción con un número a la izquierda el cual lo identifica y servirá para escoger la opción deseada. El menú se mira así:

```

brandon@brandon-Satellite-L735:~/TP1$ ./TP1
0 Contactos Existentes en la Agenda
Mensajería Instantanea

Elija una opcion

1.- Nuevo Contacto
2.- Ver agenda de contactos
3.- Escoger los datos de un contacto para comunicarse
4.- Agregar o actualizar puerto de escucha local
5.- Continuar al envio y recepcion de mensajes

```

1- Nuevo Contacto: Para indicar que se desea crear un nuevo contacto, se debe indicar con el número 1. En esta parte, se debe indicar cual es el nombre del usuario, luego el puerto de conexión con ese usuario, y finalmente la dirección de IP del mismo. Después de eso, el usuario quedará guardado y el menú aparecerá de nuevo.

```

1
Añadiendo nuevo contacto

Escriba el nombre del contacto nuevo:
Andres
Escriba el puerto del nuevo contacto:
8002
Escriba la direccion ip del nuevo contacto:
192.168.181.148
Contacto guardado exitosamente!

Mensajería Instantanea

Elija una opcion

1.- Nuevo Contacto
2.- Ver agenda de contactos
3.- Escoger los datos de un contacto para comunicarse
4.- Agregar o actualizar puerto de escucha local
5.- Continuar al envio y recepcion de mensajes

```

2- Ver agenda de contactos: Después de haber ingresado contactos, puede visualizarlos con esta opción. Al indicar la segunda opción, se desplegará una lista con los nombres de los usuarios que han sido agregados al programa así:

```

2
Lista de contactos

Andres
Brandon
Adrian
David

Mensajería Instantánea

Elija una opción

1.- Nuevo Contacto
2.- Ver agenda de contactos
3.- Escoger los datos de un contacto para comunicarse
4.- Agregar o actualizar puerto de escucha local
0.- Continuar al envío y recepción de mensajes

```

3- Escoger los datos de un contacto para comunicarse: Esta opción recibe el nombre de alguno de los usuario al cual se desea obtener información. Al ingresar el nombre, retornará los mismos datos que fueron indicados durante la creación de un usuario nuevo.

```

3
Escriba el nombre del contacto:
Andres
Tenemos los siguientes datos en memoria para comunicarse con:
Nombre: Andres
Puerto: 8002
Direccion IP: 192.168.181.148
Si desea empezar a comunicarse con esta persona digite 0

Mensajería Instantánea

Elija una opción

1.- Nuevo Contacto
2.- Ver agenda de contactos
3.- Escoger los datos de un contacto para comunicarse
4.- Agregar o actualizar puerto de escucha local
0.- Continuar al envío y recepción de mensajes

```

4- Agregar o actualizar puerto de escucha local: Esta opción permite actualizar el puerto de escucha local. Se indica cual es el puerto actual, en este caso es 0, se cambiara por el puerto 8000 y posteriormente, el menú se mostrará de nuevo.

```

4
Puerto de escucha local actual: 0
Escriba el puerto de escucha local que desea guardar:
8000
El puerto local se ha guardado correctamente

Mensajería Instantánea

Elija una opción

1.- Nuevo Contacto
2.- Ver agenda de contactos
3.- Escoger los datos de un contacto para comunicarse
4.- Agregar o actualizar puerto de escucha local
0.- Continuar al envío y recepción de mensajes

```

5- Continuar al envío y recepción de mensajes: Finalmente, para comenzar el envío y recepción de mensajes, se indica con un 0. Este indicará cuáles son los puertos y la IP establecida. Después de eso, se indicará que hay conexión y se habilita el intercambio de mensajes:

```

0
soy el proceso padre
Este es el puerto servidor que mandaron desde el main: 8000
soy el proceso hijo
Este es el puerto cliente que mandaron desde el main: 8002
Este es el ip cliente que mandaron desde el main: 192.168.181.148
HAY CONEXION

Escriba su mensaje:

```

Al escribir un mensaje, el mensaje será enviado diferenciándose con color distinto al recibido. Al enviarse, se vuelve a habilitar para continuar el envío de mensajes. Así es como se verá el intercambio de mensajes:

Envío de mensajes:

```

HAY CONEXION

Escriba su mensaje:

Mensaje Enviado: Hola! Como esta?
Escriba su mensaje:

```

Recepción de mensajes:

```

Mensaje : Hola! Como esta?

```

Al finalizar el intercambio de mensajes, y cerrar el programa, se indica con CTRL+Z y éste terminará el proceso. Después de esto, la terminal queda disponible para seguir utilizándola o volver a llamar el programa.

```

^Z
[2]+  Stopped                  ./cliente 192.168.15.105 20000
brandon@brandon-Satellite-L735:~/TP1$

```



# Conclusión Personal

Para el desarrollo de esta primera tarea programada, se necesitaba conocer acerca del lenguaje de programación C y del uso de algunas librerías fundamentales, entre ellas las librerías para la definición y manejo de sockets. Con esto logramos comprender la importancia del uso de la función fork y de las ventajas que aportan los sockets cuando se quiere desarrollar un programa que pueda utilizarse en más de una computadora y requieran conectarse. Y el manejo de hilos para ejecutar varias funciones a la misma vez.

Además resultó sumamente provechoso conocer la herramienta Github que facilita el proceso a la hora de programar grupalmente. Y brinda opciones para realizar el “branching” de diferentes versiones del proyecto. Gracias a este control de versiones es que facilita la programación en un grupo porque al almacenar todas las versiones del código que se han incluido en el repositorio, resulta sencillo poder observar y comprender los cambios que realiza algún compañero y de ser necesario se pueden encontrar las versiones anteriores a un cambio.

Tuvimos extensas búsquedas de información porque tuvimos que aprender de cosas que nunca habíamos visto entonces la información de tutoriales y foros en internet nos ayudó mucho para comprender el manejo de sockets y el manejo de fork. También el manejo de archivos en C.

Gracias al desarrollo de esta tarea logramos ampliar y mejorar nuestros conocimientos del lenguaje C; adquirimos conocimientos acerca de los Sockets, manejo de archivos, conexiones a través de la red en otros. También se pudo analizar y comprender las formas de resolver el problema que propone cada compañero, y con esto poder dar con la mejor combinación de ideas y obtener el mejor resultado.

# Bibliografía Consultada

- Sockets en C de Unix/Linux (2007, 04 de Febrero). Recuperado el 10 de Septiembre del 2013, de [http://www.chuidiang.com/clinix/sockets/sockets\\_simp.php](http://www.chuidiang.com/clinix/sockets/sockets_simp.php).
- Cabanes, N. (2010). *Curso de C, por Nacho Cabanes-Tema 6-Ficheros* Recuperado el 10 de Septiembre del 2013, de <http://www.nachocabanes.com/c/curso/cc06.php>.
- MANUAL DE SOCKETS EN C (2012). Recuperado el 10 de Septiembre del 2013, de [http://atc2.aut.uah.es/~jmruiz/Descarga\\_LE/PRACTICA2-MANUAL\\_SOCKETS.pdf](http://atc2.aut.uah.es/~jmruiz/Descarga_LE/PRACTICA2-MANUAL_SOCKETS.pdf).
- Exercise 2: Make Is Your Python Now (s. f.). Recuperado el 10 de Septiembre del 2013, de <http://c.learnthecodethehardway.org/book/ex2.html>