

DOCUEMENTACIÓN EXTERNA

Contenido

Descripción de problema	2
Diseño de programa	3
Estructura de Cliente y Servidor del programa	4
Librerías utilizadas:	5

Descripción de problema

Se debe realizar un programa que permita la mensajería instantánea similar al chat de Facebook, WhatsApp, etc. El programa debe realizar la conexión entre dos computadoras (clientes), las cuales deben estar conectadas a un servidor central. La conexión se define como cliente-servidor-cliente. El proyecto debe ser realizado con el lenguaje de programación; C.



El programa realiza las siguientes funciones:

- **Registrar el usuario:** El usuario ingresa su nombre de usuario, el cual se almacena en un archivo (user.txt) ubicado en el servidor. En el momento del registro se obtiene la dirección de la IP del usuario de forma automática, sin necesidad de especificarlo, con el fin de evitar una mayor complejidad de su uso. Con respecto al puerto, siempre se usará un valor predeterminado que estará especificado en un archivo de configuración (**config_client.txt**). En caso de ser necesario el usuario podría modificar el número de puerto accediendo al archivo mencionado previamente.
- **Envío de mensajes:** El programa permite al usuario enviar mensajes a alguno de sus contactos. Para realizarlo debe especificar el nombre de usuario con que desea chatear y el mensaje. El programa envía el mensaje a dicho usuario, a través del servidor, entendiéndose mensaje como texto. Al momento de enviar el mensaje pasa por un proceso de encriptación en el cliente emisor, pasando encriptado en el servidor y luego se procede a su respectiva desencriptación en el cliente receptor. En caso de los archivos el usuario especifica el archivo por medio de la ruta completa del mismo. Los mensajes, tanto de texto como archivos, no serán enviados de un usuario directamente a otro usuario, los archivos también serán

enviados por cliente servidor cliente.

- **Recibir mensajes:** Para que los mensajes lleguen, el usuario debe estar conectado al servidor. Antes de llegar el mensaje pasa primero por el servidor el cual realiza un proceso de bifurcación, creando una copia del socket que proviene del usuario que envió el mensaje, dicho proceso actúa como un proceso hijo del proceso originario. Para realizarlo se utilizó hilos con la librería “lpthread”, que permite la distribución de la información.

Diseño de programa

La elaboración del programa es en el lenguaje de programación C. Para que su compilación sea efectiva se deben poseer los siguientes archivos:

1. **Server.c**, Contiene toda la información del servidor y utiliza la librería –lpthread para hacer la bifurcación de envío y recibido de mensajes.
2. **Client.c** Contiene toda la información de los clientes y también usa esa librería para poder enviar la información al servidor y recibirla de este
3. **archivos_servidor.c**, Contiene toda la información pero del servidor que recibe archivos
4. **archivos_cliente.c**, Contiene toda la información pero del cliente que envía y recibe archivos
5. **config_client.txt**, Posee la información de red del servidor como lo es la IP el puerto de red al que desea conectarse. En caso que el puerto se bloquee o esté en uso puede ser cambiado accediendo al archivo.
6. **Config_server.txt**, Contiene el puerto de red al que se deberá acceder con el fin de establecer la respectiva conexión con los clientes.
7. **User.txt**, Contiene los nombres de usuarios registrados en el programa.

Estructura de Cliente y Servidor del programa

Colores: Los colores utilizados en este programa son, rojo, verde, amarillo, azul, morado, celeste y se definen con un #define “nombredelcolor” “x1.[...]”

Cliente: Se crean e inicializan variables para almacenar los nombres de usuarios y el contenido del mensaje que será enviado. Algunas variables creadas son char mensaje [250] (almacena el mensaje que será enviado y encriptado al llegar al servidor). Luego abre el config_client.txt y obtiene la dirección del IP y el puerto al que deberá conectarse para establecer la conexión con el servidor. Luego se procede a crear la conexión con el servidor el cual ya debe estar disponible debido a que es el primero en iniciarse. El proceso se resume a continuación mediante los siguientes hechos.

- Se crea el socket y recibe el dominio, protocolo y el tipo del socket.
- Se establece la estructura de la dirección e indica al socket donde debe conectarse, después convierte el número de puerto para la orden de red.
- Se solicita al usuario el nombre de usuario por medio del comando fgets(“se pone información para el usuario”), cuando este cumplió con las respectivas normas envía el nombre de usuario al server a través de un socket manipulado por la función “send”.
- Realiza la lectura del archivo que tiene los usuarios y utilizó la función fopen(), y muestra los usuarios registrados en el sistema. Contemplando los usuarios en línea y los que no están conectados al servidor.
- Solicita al usuario digitar el mensaje que desea enviar pero en la variable donde se va a guardar la información primero se anulan todos sus caracteres para que el arreglo quede libre por medio de la función “bzero” (pone en nulo a la cantidad de caracteres indicada de un arreglo) y luego se procede a almacenarlo.
- Se realiza la encriptación del mensaje por medio del método de cifrado Cesar. Por aparte se tiene otra variable que contiene el nombre del usuario que envía el mensaje. Finalmente se concatenan los arreglos de caracteres para que se envíe el mensaje encriptado con el nombre de usuario lo envía y al que lo desea enviar, debido que es información relevante para el servidor para realizar el proceso de redistribución del mensaje.
- La función recibir() tiene como funcionalidad recibir el socket enviado por el servidor. Se realiza un proceso de lectura del mensaje para la respectiva des encriptación y que el mensaje sea claro.

Servidor. El proceso de su ejecución se resume a continuación:

- En primera instancia se comienza realizando la lectura del txt de usuarios (User.txt) con el fin de obtener los usuarios conectados en el programa.
- Luego lee el archivo config_server.txt pero para poder obtener el puerto de configuración de red, con el fin de dejar la puerta de red abierta en espera a la conexión de los clientes y así al compilar el programa (Cliente.c), pueda establecer la conexión con el servidor a través del puerto proporcionado.
- Recibe el socket a través de la función server(), que le recibe del usuario emisor para poder comunicarse con el usuario receptor y viceversa. Esta función vuelve a verificar la existencia de los usuarios. Se optó por el uso de etiquetas (go to:) para poder manejar las órdenes que se le desean enviar al computador.
- Al recibir la información debe realizar una lectura de la información para ordenarla de acuerdo a la lógica empleada para el respectivo proceso de des encriptación del mensaje. Un ejemplo del formato que se busca definir es: Juan:dglsng8|Pedro (se debe saber que el mensaje desde que fue enviado por el usuario emisor, siempre ha estado encriptado y será claro hasta que sea recibido por el otro usuario.

Librerías utilizadas:

Las librerías son archivos de la biblioteca estándar en C que contienen un conjunto de funciones, tipos relacionados y macros, y estos 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:

- **#include <stdio.h>=** Contiene las definiciones de macros, constantes, declaraciones de funciones y definición de tipos, usados por operaciones estándar de entrada y salida.
- **#include <stdlib.h>=** Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.
- **#include <unistd.h>=** Concede acceso a POSIX del api del Sistema operativo
- **#include <string.h>=** Es un archivo de la Biblioteca estándar del lenguaje de programación C que contiene la definición de macros, constantes, funciones y tipos de utilidad para trabajar con cadenas de caracteres y algunas operaciones de manipulación de memoria
- **#include <errno.h>=** Definen las macros que presentan un informe de error a través de códigos de error.
- **#include <sys/types.h>=** Definen las macros que brindan informes de los sockets

- **#include <sys/socket.h>**= Librería que permite la comunicación entre dos programas o procesos. Contiene las funciones de sockets, y sus correspondientes constantes.
- **#include <netinet/in.h>**=Permite usar la variable errno.
- **#include <arpa/inet.h>**= Pone a disposición del tipo in_port_t y el tipo in_addr_t tal como se define en la descripción de <netinet / in.h> .
- **#include <pthread.h>**= Es la librería que permite usar hilos para la creación del socket y que bifurque.
- **#define MAX 250**= color.
- **#define Rojo "\x1b[31m"**= color.
- **#define Verde "\x1b[32m"**= color.
- **#define Amarillo "\x1b[33m"**= color.
- **#define Azul "\x1b[34m"**= color.
- **#define Morado "\x1b[35m"**= color.
- **#define Celeste "\x1b[36m"**= color.
- **#define ResetColor "\x1b[0m"**= color.

Análisis de los resultados

Para la implementación de este chat, se logró realizar la conexión cliente servidor cliente el cual envía mensajes pero este también envía archivos, la diferencia es que esto se realiza por aparte debido a diferentes problemas presentados durante el desarrollo del programa.

El sistema de mensajería sirve muy bien sin embargo presenta varias imperfecciones que si no se está consiente de estas el programa presentara errores, por ejemplo el mayor inconveniente que se nos presento fue el archivos de los usuarios, ya que este es utilizado por el cliente y el servidor lo que obliga a estos 2 archivos estar juntos en una carpeta para la correcta ejecución del programa, además de esto los usuarios solo se almacenan mientras el servidor este corriendo ya que cada vez que se abre se reinicia la lista de contactos. Otro problema es que debido a esto el primer cliente que se conecte no le aparecerá contactos y deberá esperar a que alguien más se conecte para así poder iniciar una conversación.

Otro de los principales problemas fue que no se pudo unir la parte de archivos al chat por lo que es un programa totalmente aparte.

Manual de usuario

Para poder ejecutar este programa es necesario que el sistema operativo del equipo en el que se encuentre sea Linux, ya que en otro no servirá. Una vez que se esté trabajando en Linux se procede a abrir la terminal, para esto puede utilizar el comando *CTRL+ALT+T*.

Después de esto se escribe en la ventana negra el siguiente comando.

```
$ sudo apt-get update  
$ sudo apt-get upgrade  
$ sudo apt-get install build-essential  
$ gcc -v  
$ make -v
```



```
jur@ubuntu: ~  
jur@ubuntu:~$ sudo apt-get update  
jur@ubuntu:~$ sudo apt-get upgrade  
jur@ubuntu:~$ sudo apt-get install build-essential  
jur@ubuntu:~$ gcc -v  
jur@ubuntu:~$ g++ -v  
jur@ubuntu:~$
```

Luego de esto se le da a la tecla enter, y ya está instalado el lenguaje c y c++ en su computadora.

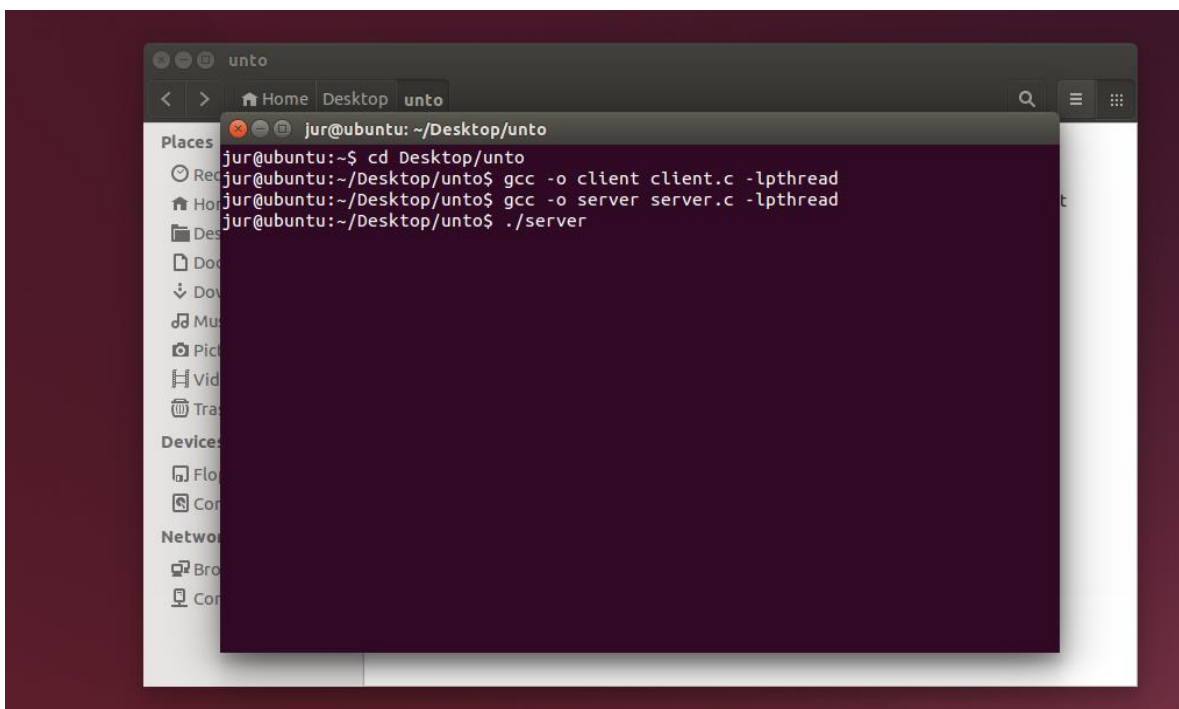
Después volvemos a escribir con el teclado CTRL+ALT+T y se nos volverá a abrir la ventana anterior mente mostrada y escribiremos `cd` “Nombre de la dirección de la carpeta contenedora” para acceder a esta, por ejemplo `cd Desktop`, `cd Tarea_Programada` o `cd Desktop/Tarea_programada`. Una vez que se accede a la carpeta desde la consola puede utilizar el comando `ls`² para verificar los archivos que se encuentren en esta.

```
jur@ubuntu: ~/Desktop/unto  
jur@ubuntu:~$ cd Desktop/unto  
jur@ubuntu:~/Desktop/unto$ ls  
client      config_client.txt  config_server.txt  server.c  
client.c    config_client.txt~  server             user.txt  
jur@ubuntu:~/Desktop/unto$
```

En caso de que los archivos no se encuentren compilados se debe usar el siguiente comando para el servidor y el cliente respectivamente: `gcc server.c -o server -lpthread`, `gcc client.c -o client -lpthread`.

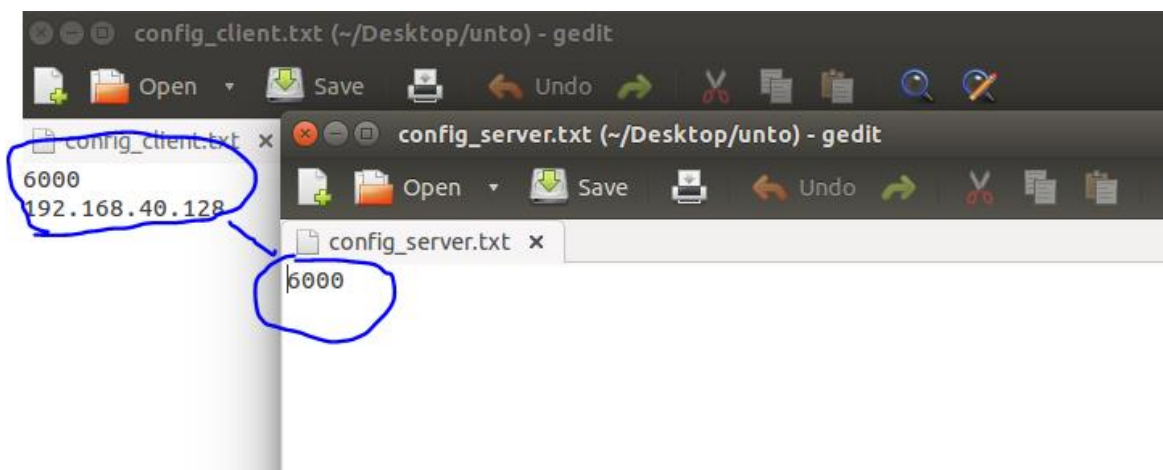
1 cd: Se accede a la carpeta donde se encuentren los programas

2 ls: Se puede ver los elementos que hay en la carpeta que nos encontramos

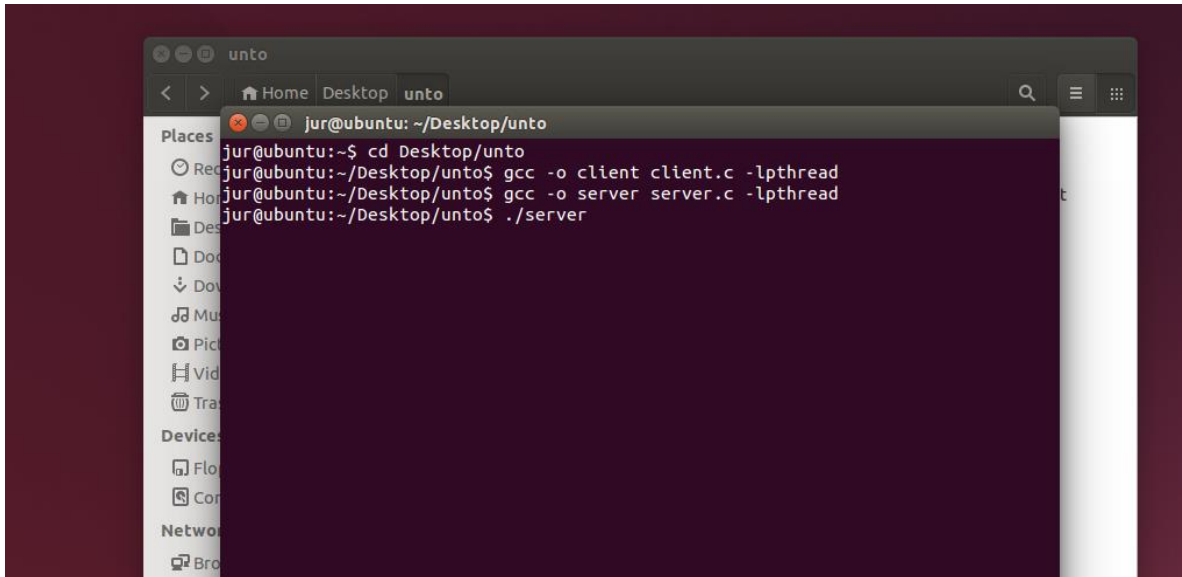


```
jur@ubuntu: ~/Desktop/unto
jur@ubuntu:~$ cd Desktop/unto
jur@ubuntu:~/Desktop/unto$ gcc -o client client.c -lpthread
jur@ubuntu:~/Desktop/unto$ gcc -o server server.c -lpthread
jur@ubuntu:~/Desktop/unto$ ./server
```

Una vez hecho esto ya se podrá ejecutar los programas pero se recomienda antes revisar los archivos *config_server.txt* y *config_client.txt* para asegurarse que la dirección de la IP sea la correcta y que los puertos sean los mismos.

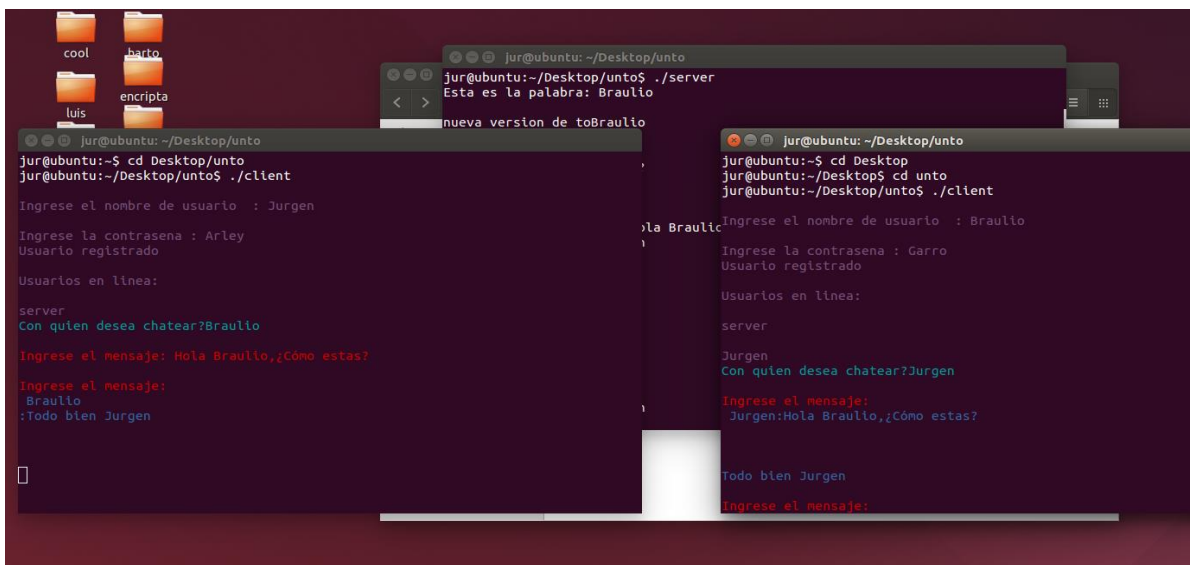


Lo primero es ejecutar el servidor para esto se introduce en la consola *./server*.

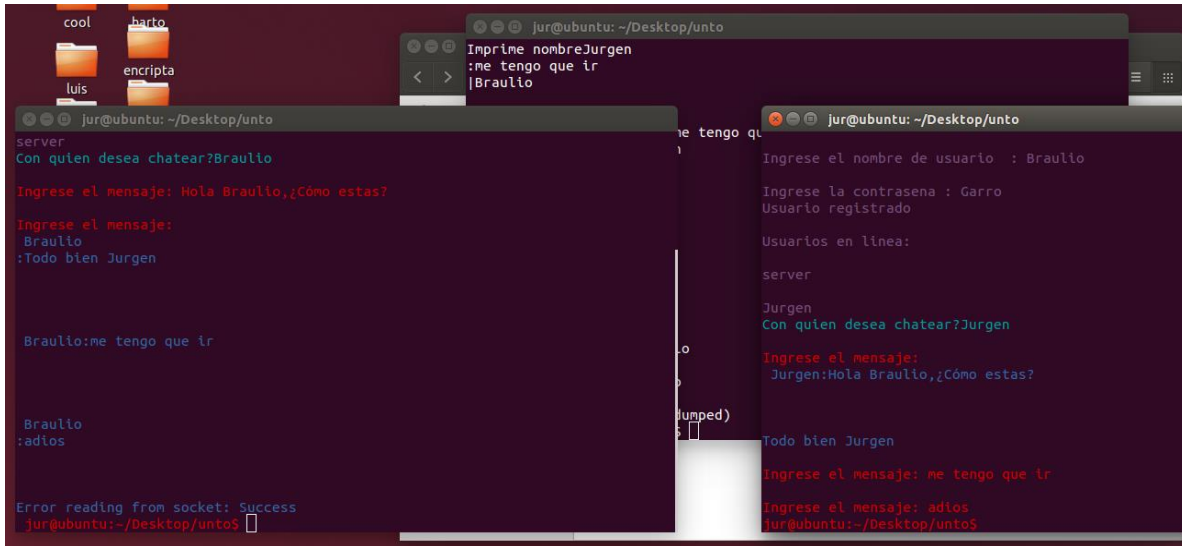


Luego si se va a correr en el mismo equipo se deben abrir dos consolas más y dirigirse a la carpeta donde se encuentre el programa siguiendo los pasos mencionados anteriormente, una vez que se esté ahí se inserta en la consola `./client`, esto en ambas.

Luego de esto el programa le solicitará un nombre de usuario y una contraseña, en caso de que nunca haya usado el chat el mismo programa lo agregará a la lista de contactos, si es un usuario registrado verificará su contraseña. Posteriormente le preguntará con quien desea chatear, luego de elegir debe esperar a que el otro cliente lo seleccione a usted antes de enviar un mensaje. Cuando ambos han seleccionado con quien hablar se puede proceder a hablar, los mensajes que un usuario escribe se verán de **color rojo**, y los mensajes recibidos de **color azul**.



Para finalizar la conversación se debe enviar el mensaje “adiós”. Esto es para que no se dañen los puertos y se guarde la información correctamente, esto es como cerrar sesión.



```
jur@ubuntu: ~/Desktop/unto
server
Con quien desea chatear?Braulio
Ingrese el mensaje: Hola Braulio,¿Cómo estas?
Ingrese el mensaje:
Braulio
:Todo bien Jurgen

Braulio:me tengo que ir

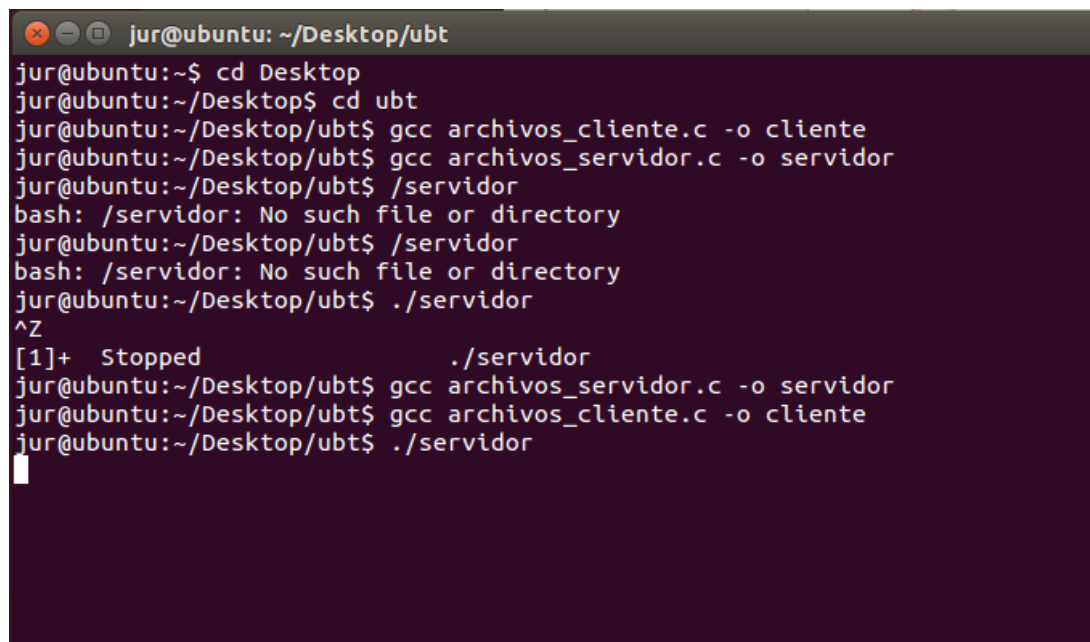
Braulio
:adios

Error reading from socket: Success
jur@ubuntu:~/Desktop/unto$

jur@ubuntu: ~/Desktop/unto
Imprime nombreJurgen
:me tengo que ir
|Braulio

jur@ubuntu: ~/Desktop/unto
Ingrese el nombre de usuario : Braulio
Ingrese la contraseña : Garro
Usuario registrado
Usuarios en línea:
server
Jurgen
Con quien desea chatear?Jurgen
Ingrese el mensaje:
Jurgen:Hola Braulio,¿Cómo estas?
Todo bien Jurgen
Ingrese el mensaje: me tengo que ir
Ingrese el mensaje: adios
jur@ubuntu:~/Desktop/unto$
```

Por problemas de desarrollo el de archivos se maneja aparte del chat, para esto se deben realizar los primero pasos para acceder a la carpeta donde se encuentren los programas archivos_clientes y archivos_servidor, el método de compilación es más sencillo ya que se omite la sentencia `-lpthread`, ejemplo: `gcc archivos_clientes.c -o archivos_clientes`, `gcc archivos_servidor.c -o archivos_servidor`. Antes de enviar un archivo es importante que tenga el archivo a enviar en la misma carpeta del cliente. Cuando se ejecuta el programa este le solicita el nombre del archivo a enviar y este llegara a donde se encuentre el servidor.



```
jur@ubuntu: ~/Desktop/ubt
jur@ubuntu:~$ cd Desktop
jur@ubuntu:~/Desktop$ cd ubt
jur@ubuntu:~/Desktop/ubt$ gcc archivos_cliente.c -o cliente
jur@ubuntu:~/Desktop/ubt$ gcc archivos_servidor.c -o servidor
jur@ubuntu:~/Desktop/ubt$ ./servidor
bash: ./servidor: No such file or directory
jur@ubuntu:~/Desktop/ubt$ ./servidor
bash: ./servidor: No such file or directory
jur@ubuntu:~/Desktop/ubt$ ./servidor
^Z
[1]+  Stopped                  ./servidor
jur@ubuntu:~/Desktop/ubt$ gcc archivos_servidor.c -o servidor
jur@ubuntu:~/Desktop/ubt$ gcc archivos_cliente.c -o cliente
jur@ubuntu:~/Desktop/ubt$ ./servidor
```

Conclusión

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.

La parte de envíos de archivos se tuvo como referencia un proyecto encontrado en internet que enviaba archivos peer to peer y para hacer el envío de mensajes nos basamos en un proyecto que utilizaba pthread de librería para realizar la tarea que sustituía al fork, el cual nos facilitó a realizar esa parte.

Uso de sockets: Se buscó información del uso de sockets, entendiendo con esta, el funcionamiento de las IP's y puertos para la conexión entre dos usuarios. Además, se aprendió del uso de los hilos para bifurcar la señal del servidor.

El manejo de archivos en C: El grupo de trabajo aprendió información de cómo se gestionan los archivos en c, lo que es leer y escribir.

Encriptación y des encriptación: Se aprendió a encriptar y des encriptar archivos y la información antes y después de enviarla al servidor para que esta información no sea robada.