**Instituto Tecnológico de Costa Rica**



**Administración en Tecnología de la Información**

Lenguajes de Programación

Prof. Andrei Fuentes

**Tarea programada # 1**

**Clonación WhatsApp**

José Daniel Chinchilla Cerdas – 201211069

Josue Masis Álvarez – 201240146

Verónica Vargas Mora – 201269405

10 de setiembre del 2013

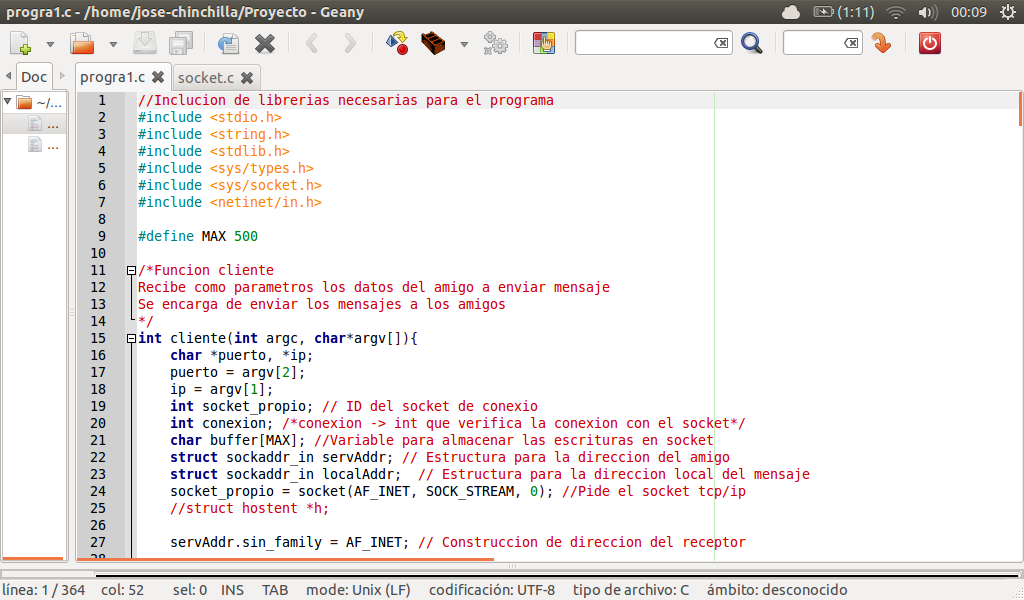
**Descripción del problema**

Para la tarea programada #1 se debió realizar un programa en el lenguaje C, en si el problema de este programa consiste en buscar la forma de realizar un sistema de mensajería instantánea entre dos computadoras, para realizarlo se utiliza como base la lógica del sistema de mensajería instantánea “WhatsApp”.

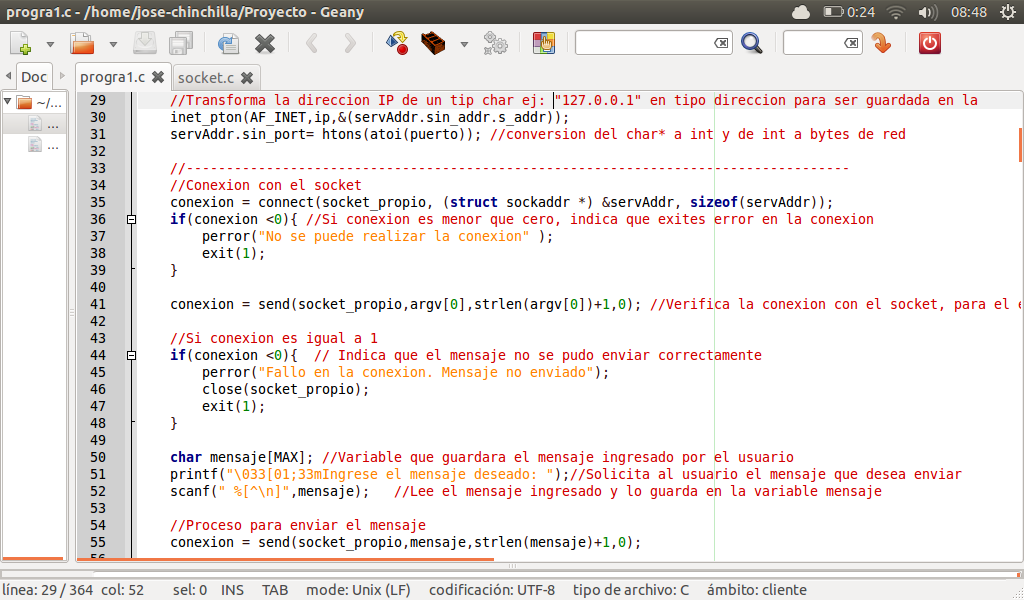
Algunos puntos importantes de esta tarea programada es que se debió aprender el manejo de sockets, el cual permite la interconexión entre las dos computadoras, otro aspecto es el manejo de hilos la cual permite el manejo de dos procesos en el procesador a la misma vez, lo cual facilita el ingreso y salida de mensajes.

**Diseño del programa**

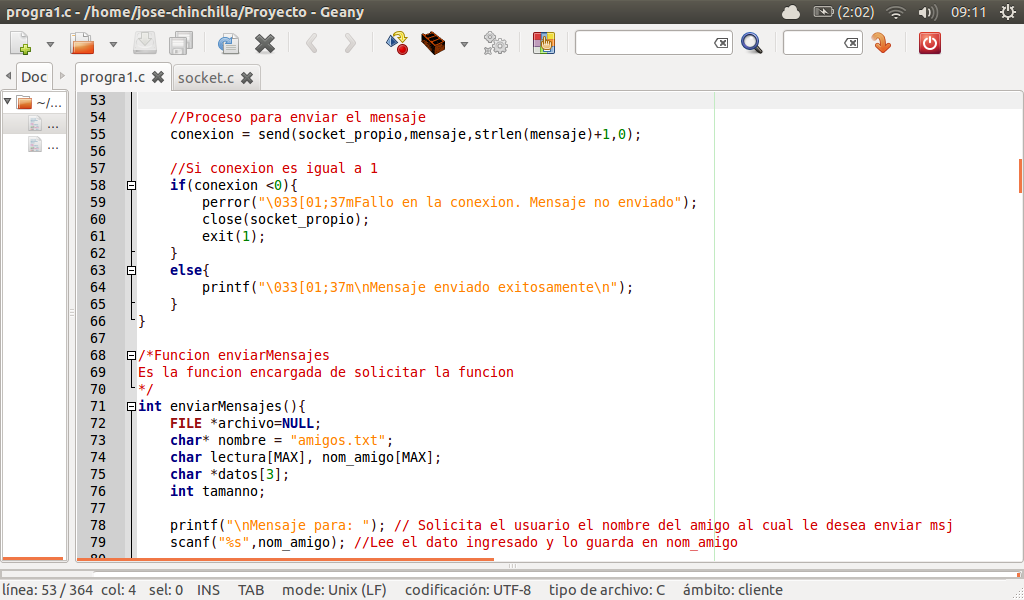
En esta parte se va a explicar el código utilizado para la realización del sistema de mensajería instantánea.



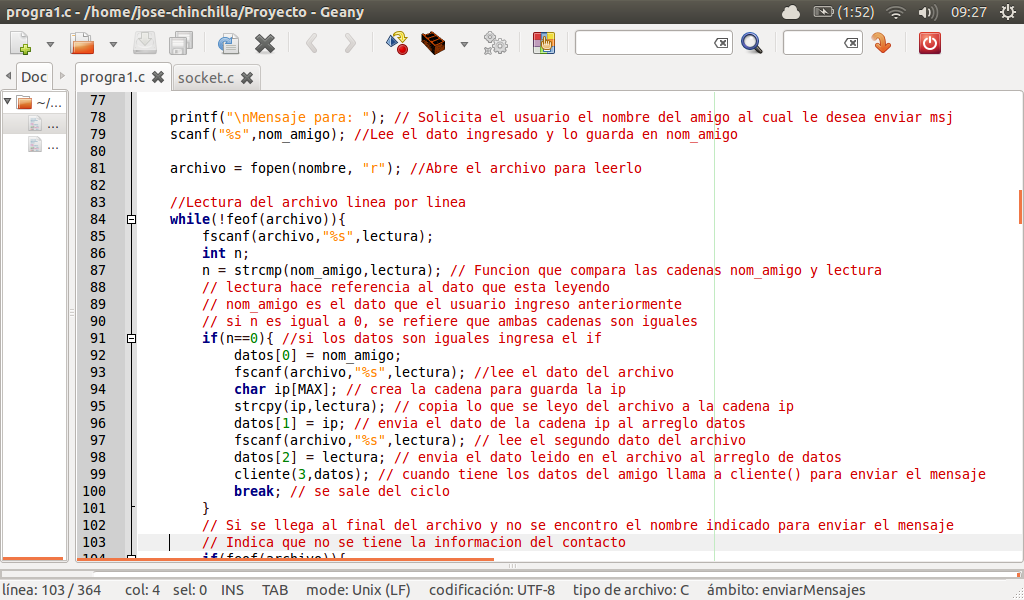
Como se muestra en la imagen anterior al principio se incluyen todas las librerías necesarias para el correcto funcionamiento de las funciones determinadas, como al incluir “sus/socket.h” esto permite el manejo y creación de sockets en el código. Una vez determinada todas las librerías se crea la función cliente, la cual es de tipo int y recibe un argumento de tipo int y un arreglo de char\*, posteriormente se definen e inicializan una serie de valores que se van a utilizar en el resto de la función. Esta función es la que permite la conexión del cliente con los sockets.



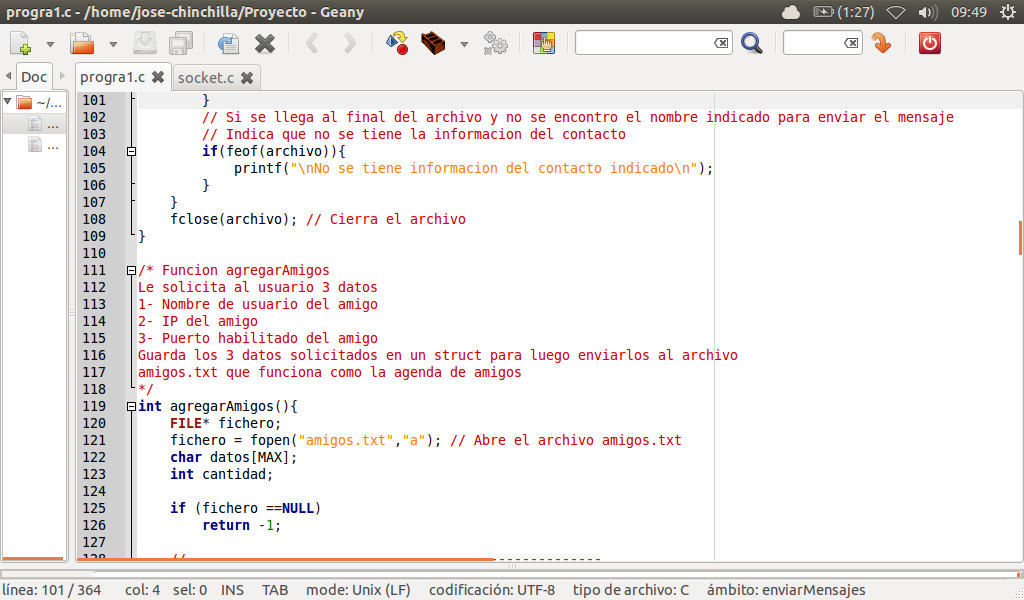
Es esta imagen se ve donde se trasforma la ip a char, esto para guardarla como parámetro para posteriormente ser utilizado en los sockets. Después comienza la conexión con el socket (todo esto a un se esta realizando en la función cliente), en esta parte se valida si la conexión fue exitosa, o si el mensaje fue enviado, para esto se utiliza la función send la cual es propia de sockets.



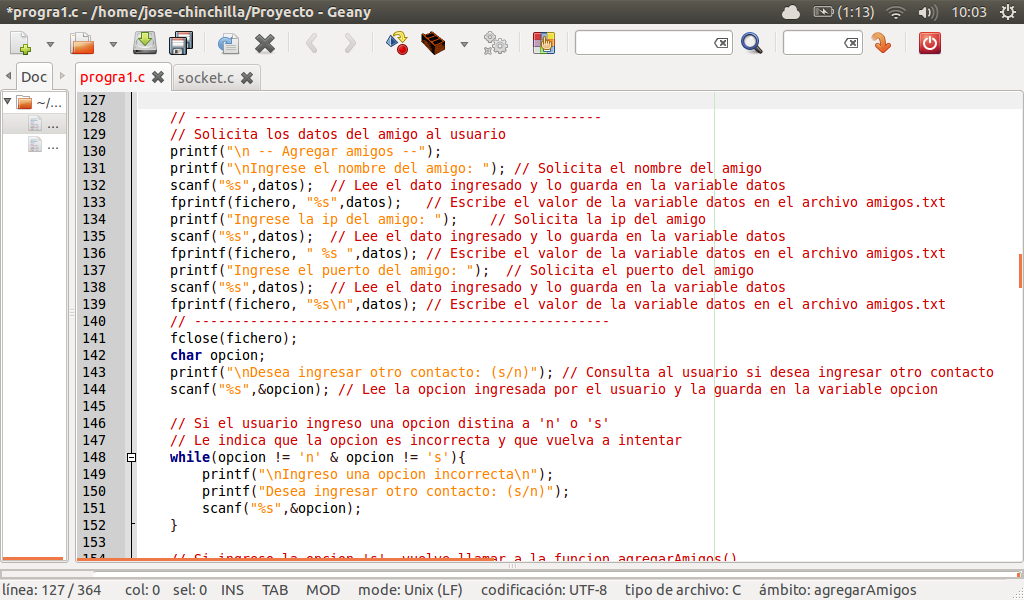
En esta parte se ve donde se termina de validar el envío de mensajes y ahí termina la función cliente(). Después se comienza la función enviarMensajes esta función permite el envío de mensajes leyendo desde un archivo los contactos de los amigos del cliente. En esta imagen se ve por el momento donde se definen las variables a utilizar en esta función.



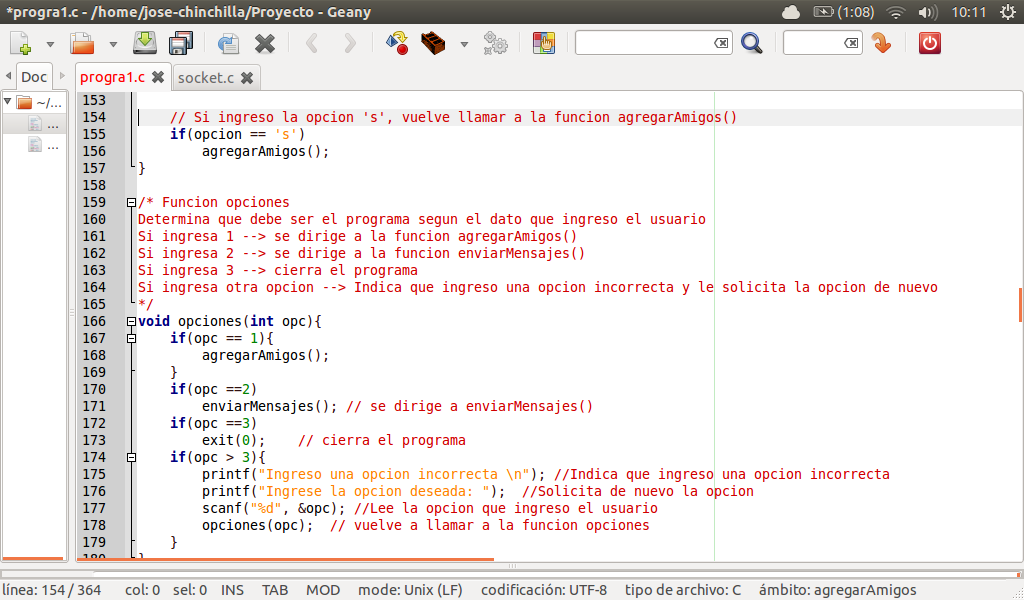
Aquí se ve donde se le pide al usuario que ingrese el nombre del amigo que quiere contactar, después se abre el archivo de amigos, que es el que contiene los contactos, y con un while empieza a recorrer el archivo en busca del nombre anteriormente ingresado, si lo encuentra ingresa al if, ahí lee los datos, una vez con los datos llama la función cliente que ya fue especificada y se sale del while.



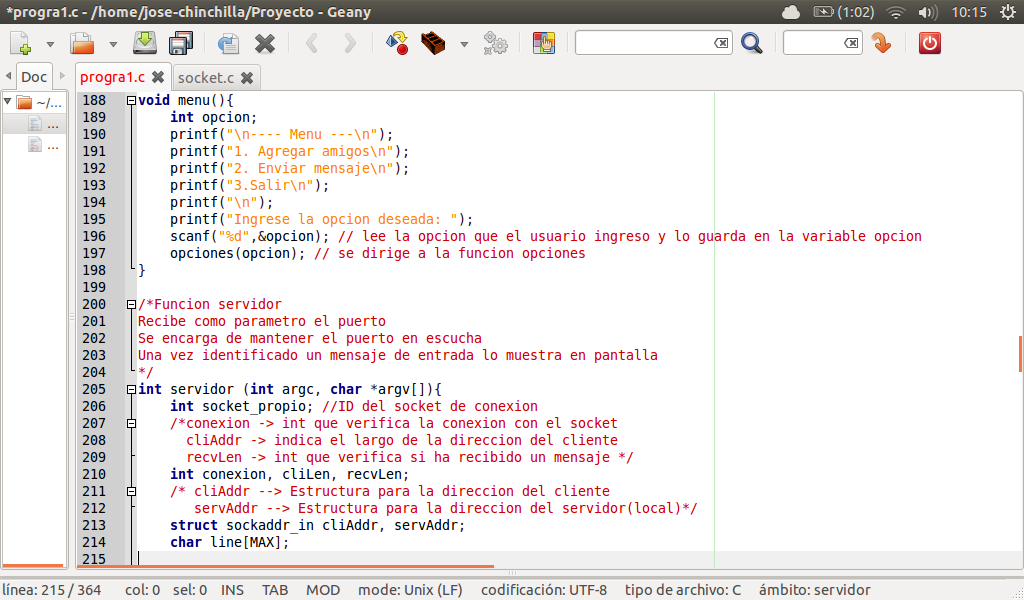
Aquí se indica que si se llego al final de la función enviarMensajes se imprime que no se encontró el cliente deseado, se cierra el archivo y ahí termina la función enviarMensajes. Después se encuentra la función agregarAmigos, aquí se ve donde se abre el archivo amigos.txt que es donde se va a ingresar los datos del nuevo amigo.



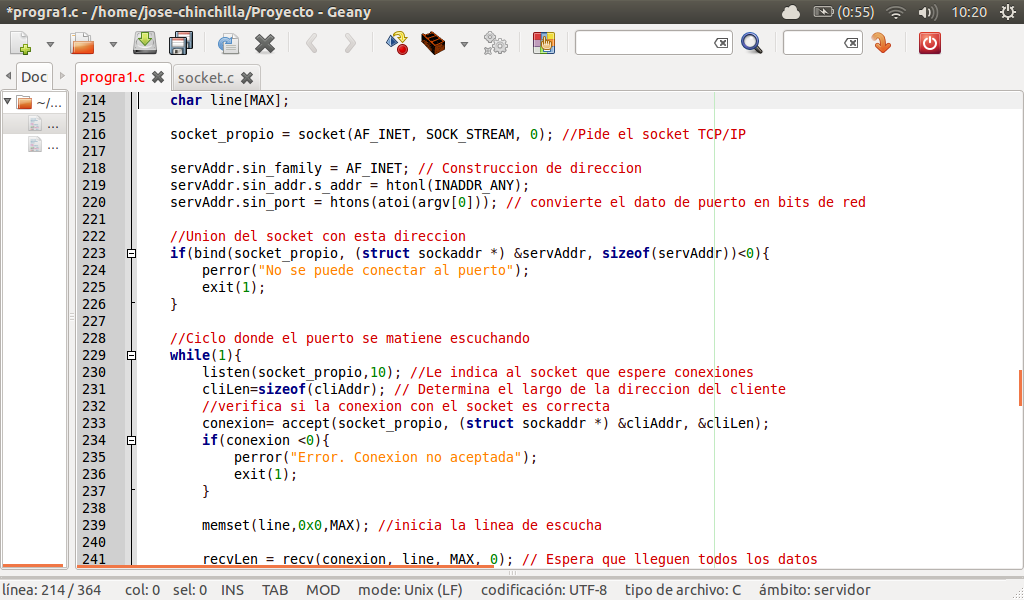
Esto es la continuación de la función agregarAmigos, en esta parte lo que se hace es solicitar al cliente los datos necesarios del amigo, estos serian nombre, Ip y el puerto que utiliza su amigo. Después viene la opción de si quiere volver a ingresar otro amigo, si se responde que si vuelve a comenzar la función agregarAmigos.



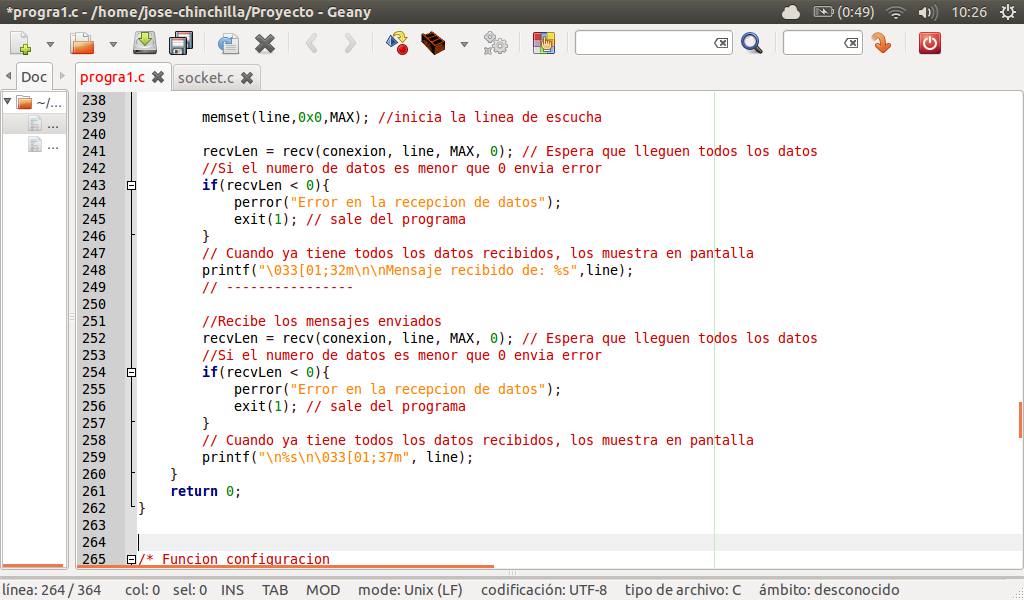
Aquí se ve donde comienza la función opciones, esta recibe un int, el cual es el que se va utilizar para llamar la opción deseada por el usuario, ya sea agregar amigo, enviar mensaje o cerrar el programa.

[](http://c.learncodethehardway.org/book/ex2.html)

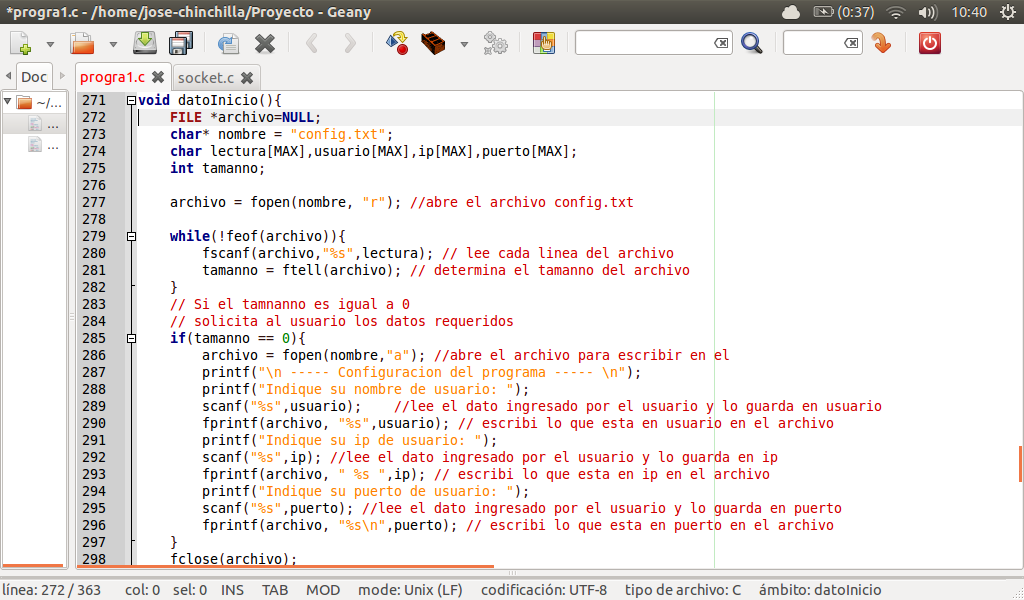
Aquí se muestra la función menú, esta lo que hace es imprimir en pantalla las funciones que se pueden hacer y después solicita una opción, la cual al ser ingresada se llama a la función opciones con este dato para realizar la función deseada. Posteriormente se comienza la función servidor, esta función lo que hace es mantener el puesto de escucha abierto y cuando llega un mensaje lo muestra en pantalla.



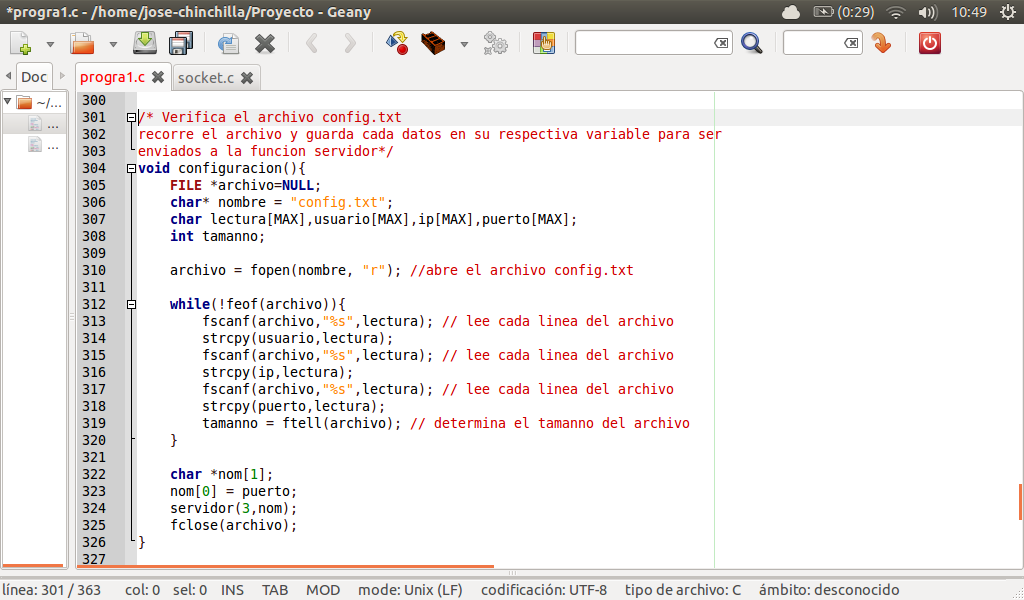
Aquí lo que se ve es donde se define el socket propio del usuario, agregándole al dirección del servidor la dirección ip y el puerto, después estas dos se unen con la función bind(). Luego se crea un ciclo que es el que permite tener el puerto escuchando por la llegada de mensajes.



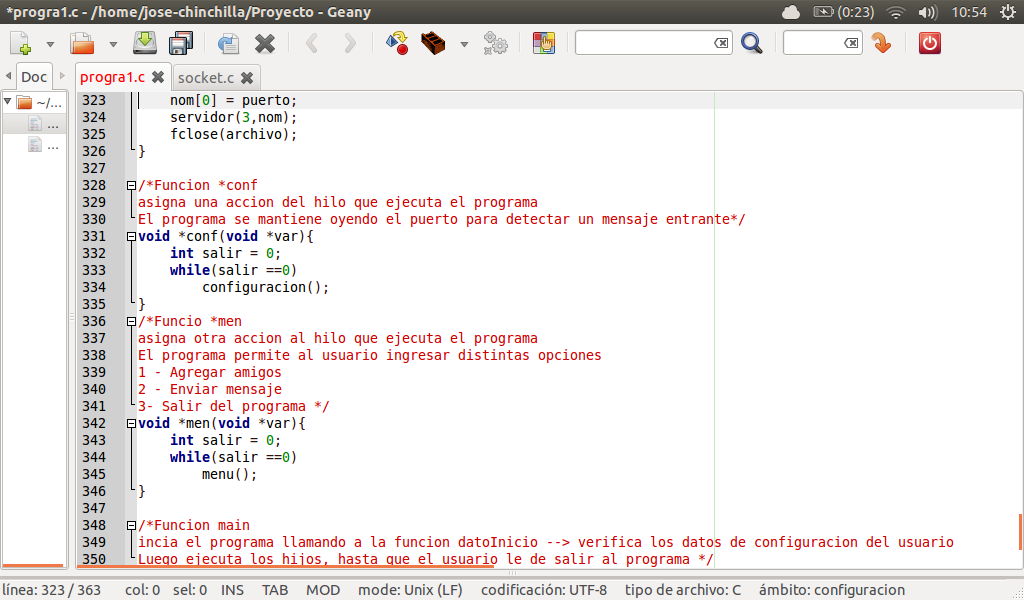
Aquí se muestra como se inicia la linea de escucha, con recv() espera la llegada de mensajes entrantes, si vienen todos los datos imprime el mensaje nuevo, sino tira error en la recepción de datos.



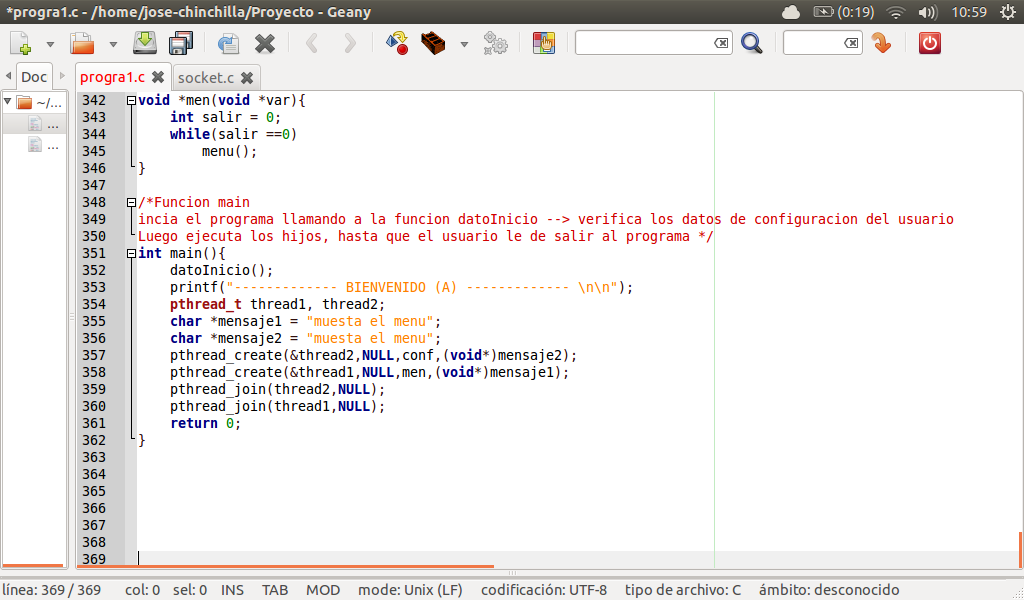
Esta función datoInicio() lo que hace es verificar si el archivo config.txt , con un while revisa si el archivo esta vacio de ser así le solicita al usuario, su nombre de usuario, la dirección Ip y el puerto a utilizar, y estos datos se ingresan en el archivo.



Aquí se muestra la función configuración, esta lo que hace es con un while volver a leer el archivo config.txt y asigna los datos a variables establecidas, para después pasarlas como parámetros a la funcionservidor().

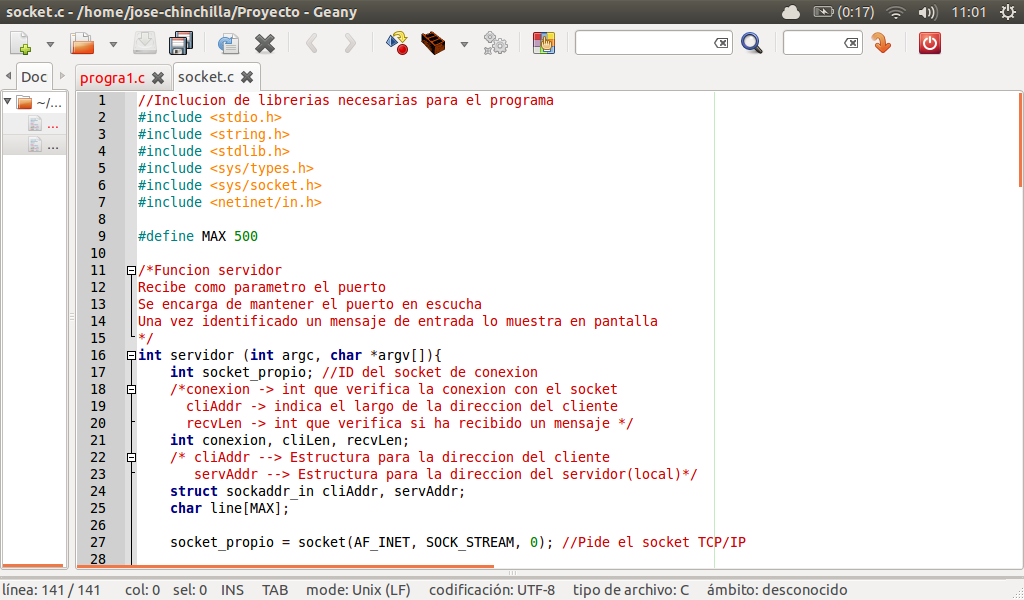


Es esta imagen se muestran dos funciones \*conf() esta asigna una acción al hilo de ejecutar el programa, lo hace llamando a la función configuración. La otra función que se muestra es \*men esta asigna otra acción lo que hace que se realice diferentes opciones, esto se hace llamando la función menú().

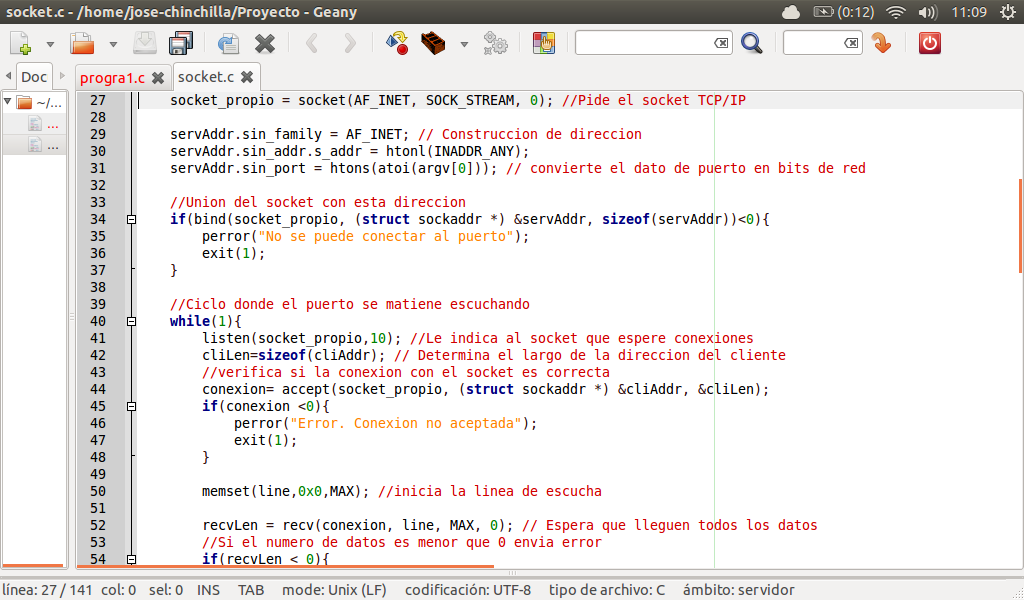


En esta imagen se muestra el main lo que permite el inicio del programas.

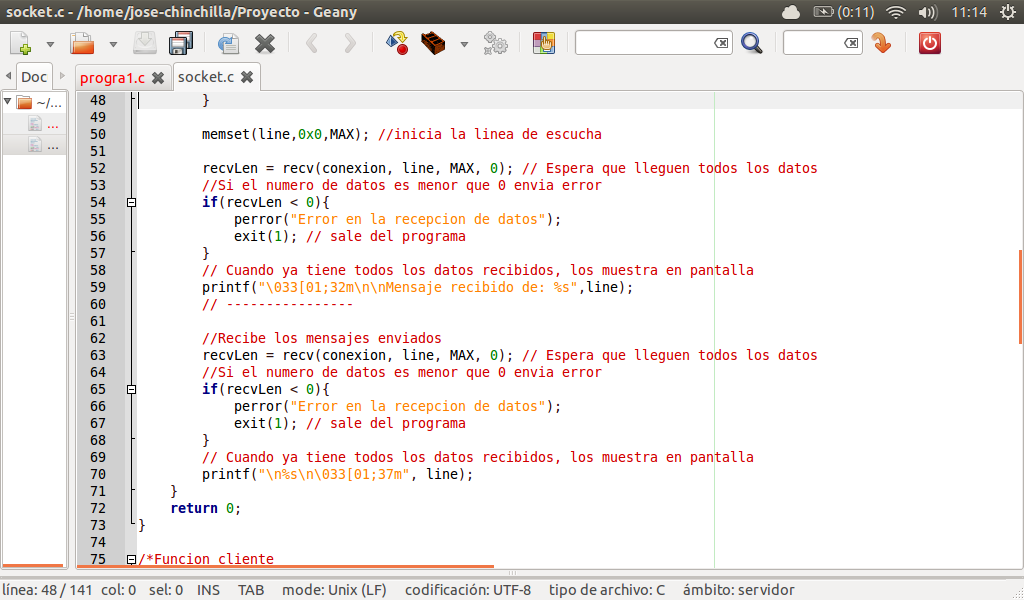
Acontinuacion se mostrara el código. del archivo socket.c



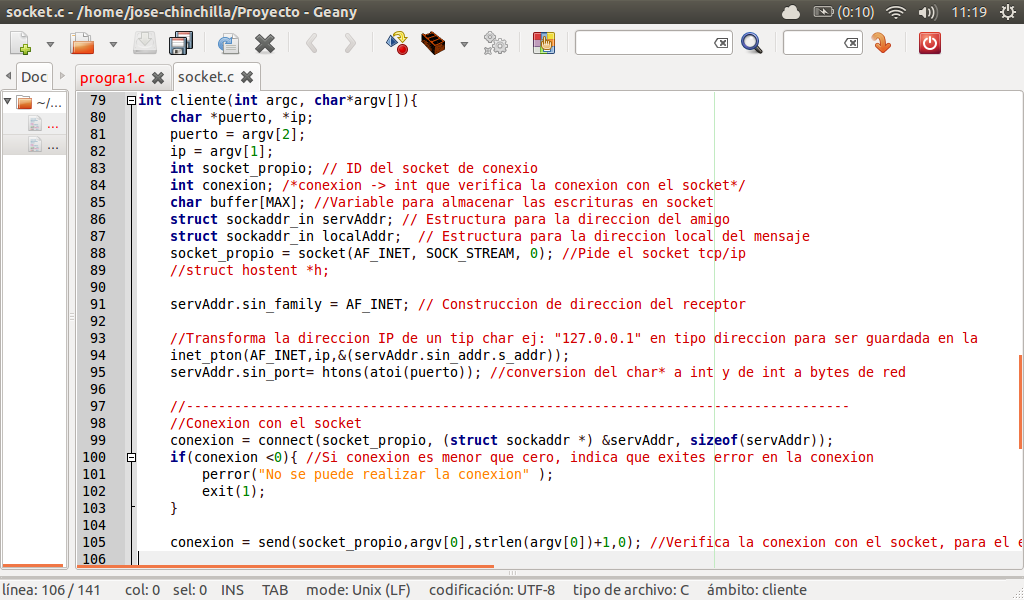
Al igual que en el archivo progra1.c se incluyen las librerías necesarias para le programa. Después se inicia la función servidor(), esta función recibe el puerto y esta función es la encargada de mantener el puerto en escucha .



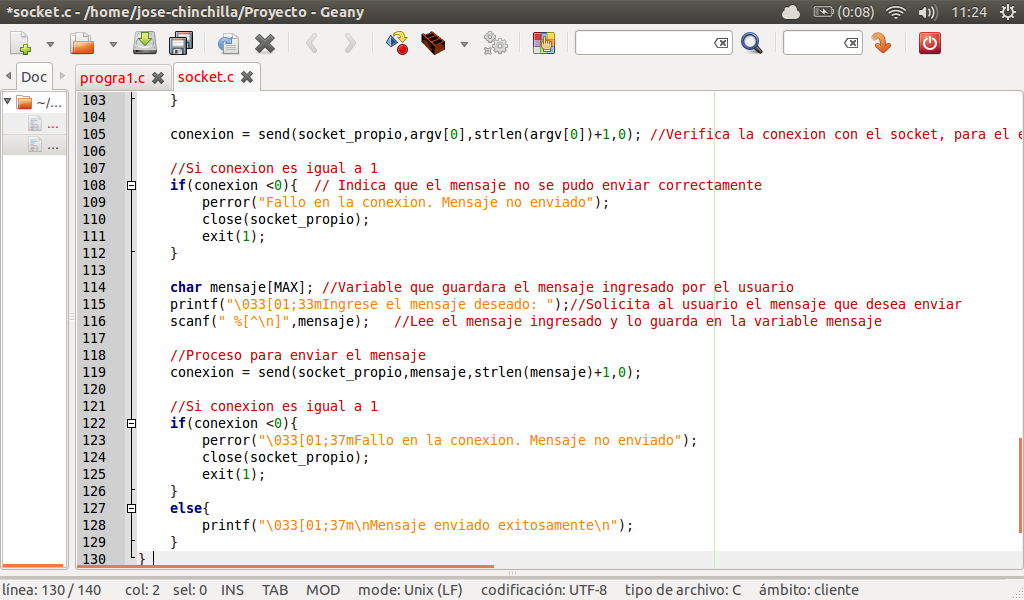
Aquí se muestra donde se construye la dirección ip, combinando la ip y el puerto en la función bind() y con el ciclo se mantiene el puerto de escucha en escucha si la conexión con el socket es correcta.

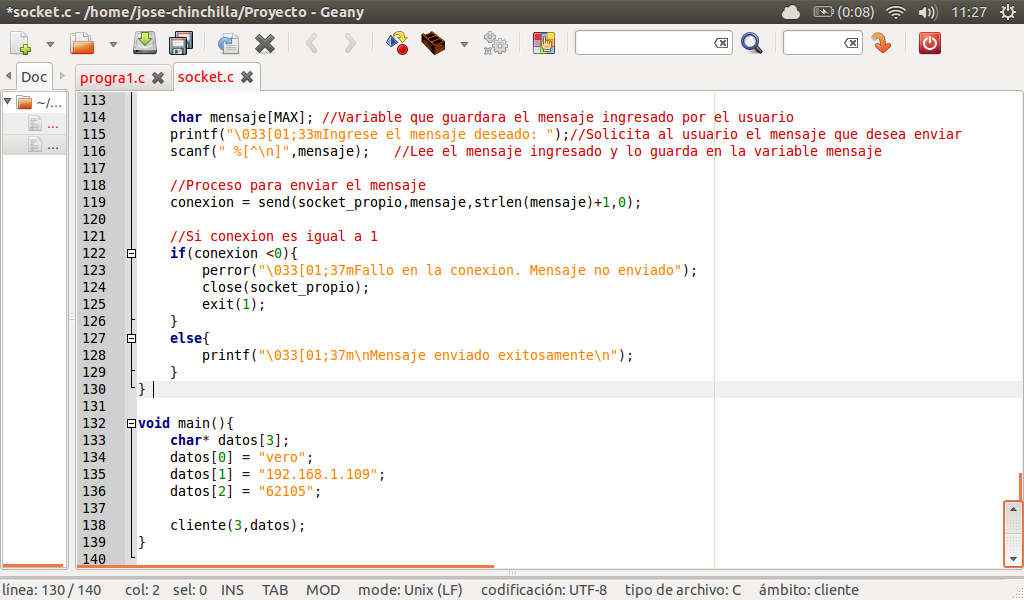


Aquí se inicia el puerto de escucha, lo que hace es ver si hay un mensaje de entrada, si ciene con todos los datos se imprime en pantalla sino tira error.



Aquí se muestra la función cliente la cual permite el envío de mensajes, se comienza con la construcción de la dirección del cliente receptor y la conexión con el socket



Aquí se muestra la validación para enviar mensajes si se puede conectar con el socket lo envía con la función send(), sino muestra mensaje de error. Por ultimo se muestra el main de los sockets.

**Librerías utilizadas**

**stdio.h:** Librería que nos permite definir tres tipos de variables, variables macros y diversas funciones para la realización de entradas y salidas datos de nuestro programa. Con esta librería se pueden declarar variables como:

* size\_t: Variable que nos brinda el mismo resultado que la palabra reservada sizeof
* printf: Función que nos permite realizar impresiones en consola
* scanf: Función que permite leer entradas de texto al programa principalmente de la entrada estándar de C (teclado)

**string.h:** Librería que permite definir macros, funciones para realizar la manipulación de un conjunto de caracteres. Con esta librería se pueden utilizar funciones como:

* strcmp: Función que permite realizar la comparación entre dos cadenas de caracteres
* strlen: Función que da como resultado el tamaño de una cadena de caracteres

**stdlib.h:** Librería que permite definir funciones generales, como las siguientes:

* atof: Función que convierte un arreglo de caracteres (char\*) a un punto flotante (float)
* atoi: Función que convierte un arreglo de caracteres (char\*) a un entero (int)
* abs: Función que realiza el valor absoluto a un valor numérico
* malloc: [Función que devuelve un puntero a un espacio de memoria que se encuentra vacío](http://www.tutorialspoint.com/c_standard_library/)

**sus/types.h:** Librería que permite definir símbolos y estructuras para utilizar sockets. Se pueden definir estructuras como:

* ip\_addr\_t: Función para la dirección IP
* u\_long: Tipo de dato long sin signo
* in\_addr\_t: Función para direcciones de internet

**sus/socket.h:** Librería que contiene la definición y estructura de un socket.

* Struct sockaddr\_in

**netinet/in.h:** Librería que contiene las definiciones de toda la familia de protocolos de Internet.

**Lecciones aprendidas**

* **Josue**

Con la realización de esta tarea programada aprendí como se le puede dar un gran uso a las conexiones de internet, como mediante una simple dirección IP que todas las computadoras y redes poseen se puede dar el traslado de caracteres y de hasta archivos de una computadora a otra mediante una conexión TCD-IP.

Por otro lado se hizo una gran investigación sobre el manejo de archivos txt en el lenguaje de programación C, de la cual se aprendió que en lenguaje cuenta con una librería en la cual están previamente definidas las funciones que nos permiten y nos ayudan a abrir, leer y modificar archivos txt, como por ejemplo las funciones que utilizamos en nuestro programa como: fopen, fscanf, ftell, entre otra que no se utilizaron pero que fueron aprendidas para ser utilizadas en otras ocasiones.

Concluyendo con una tarea programada que nos llevo tiempo en realizarla por la dificultad de los sockets y mas en un lenguaje de programación levemente desconocido, pero dando como resultado un gran aprendizaje tanto en los dos segmentos en que se dividió la tarea (manejo de archivos y sockets), como en el aprendizaje del lenguaje de programación C.

* **José Daniel**

Al realizar esta tarea programada puede aprender mas a fondo sobre el manejo del lenguaje C y sobre el programa git, siendo manejado ya fuera desde consola o desde la pagina web de github. Sin embargo esta tarea programada se baso mas que todo en el aprender el manejo de sockets y de la función fork() entonces estas fueron de las cosas que mas aprendí.

Sin embargo, esta fue una progra que fomento el trabajo en equipo, y esta es una de las cosas que forman parte de las lecciones aprendidas para esta tarea programada.

* **Verónica**

Durante el desarrollo e implementación de esta tarea programada, la cual consiste en la clonación de la aplicación WhatsApp, aprendí muchas funcionalidades del lenguaje de programación C y además de la utilización de los repositorios Git.

El manejo de archivos txt, durante el desarrollo de esta tarea aprendí como se puede leer y escribir en un archivo de texto utilizando las librerías con que cuenta el lenguaje C, y sus funcionalidades especificas para cada acción deseada como lo son:

* fopen: nos permite abrir el archivo, ya sea para leer o escribir.
* fprintf: permite escribir en el archivo.
* fscanf: permite la lectura del archivo
* fclose. Permite el cierre del archivo.

La utilización de hilos, con la implementación de hilos en el programa aprendí como se puede realizar dos acciones en un mismo momento durante le ejecución del programa. En este caso, el programa mientras se navega en las distintas opciones que tiene el programa (agregar amigos y envió de mensajes), también el programa se encuentra en modo de escucha para detectar en el momento que ingresa un mensaje.

Con la implementación de socket, aprendí como se realizar las conexiones por medio de internet para la comunicación entre distintas computadoras, en nuestro caso siempre que estas se encuentren en una misma red, utilizando la dirección IP versión 4 y un puerto habilitado en cada computadora. Lo primero permite identificar a cual ordenar se desea enviar el mensaje y el segundo permite al ordenador la recepción de los mensaje.

**Manual de usuario**

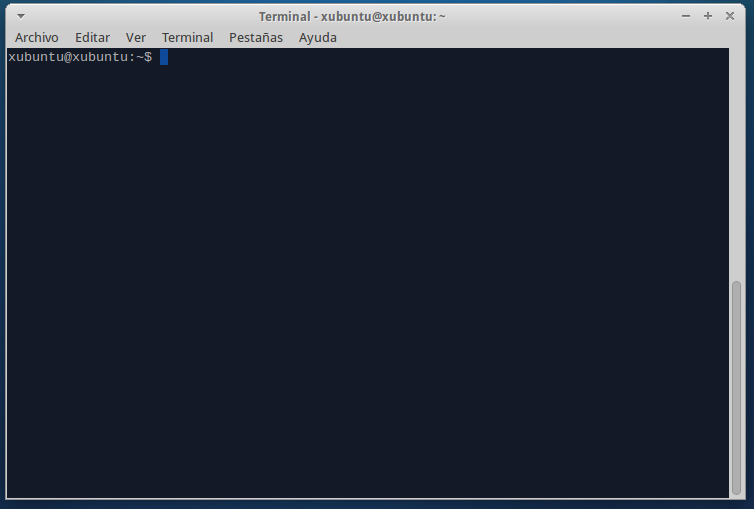
ClonWhatsApp es un programa implementado y desarrollado por José Daniel Chinchilla Cerdas, Josué Masis Álvarez y Verónica Vargas Mora; realizado con el fin de comunicarse entre distintas personas que se encuentren conectadas bajo una misma red.

La comunicación entre las distintas terminales que tengan ejecutando el programa, será siempre efectiva cuando todas las terminales que desean enviar mensajes el una a la otra se encuentren bajo la misma red, y todas tengas acceso a internet.

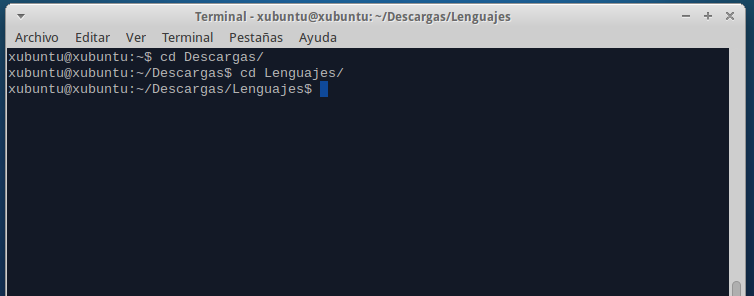
Este manual es un instrumento de ayuda para la operación del sistema.

**Compilación del sistema.**

* Abrir la consola del sistema

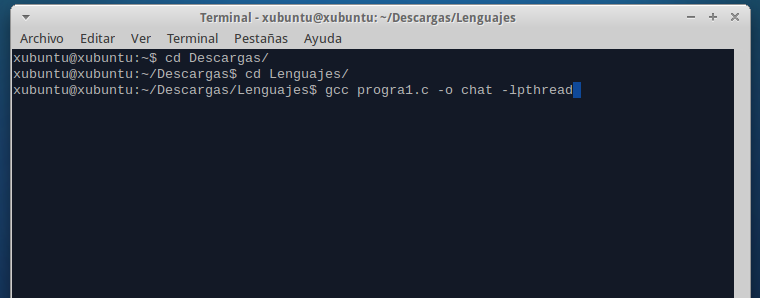


* Ubicarse en la carpeta que se encuentra el programa



* Escribir la siguiente línea de comando

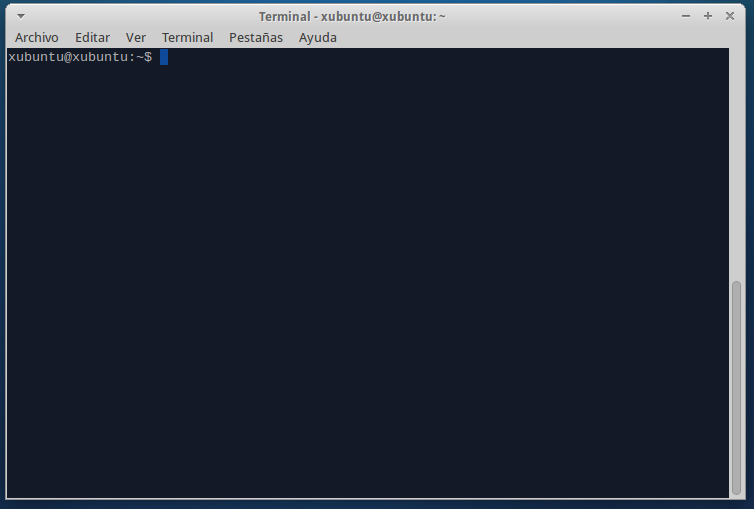
***gcc “nombre\_del\_programa” –o “nombre\_ejecutable” –lpthread***y luego darle Enter



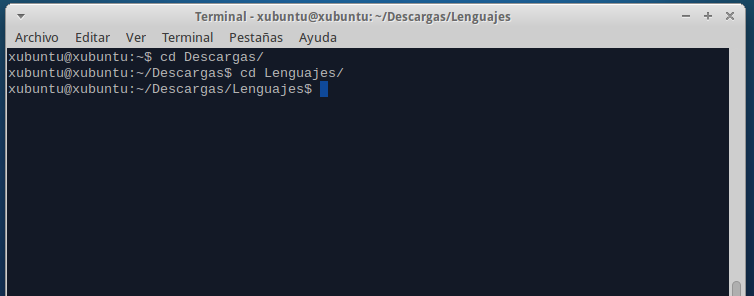
**Finalizado estos tres pasos, el programa se encuentra compilado**

**Ejecución del sistema.**

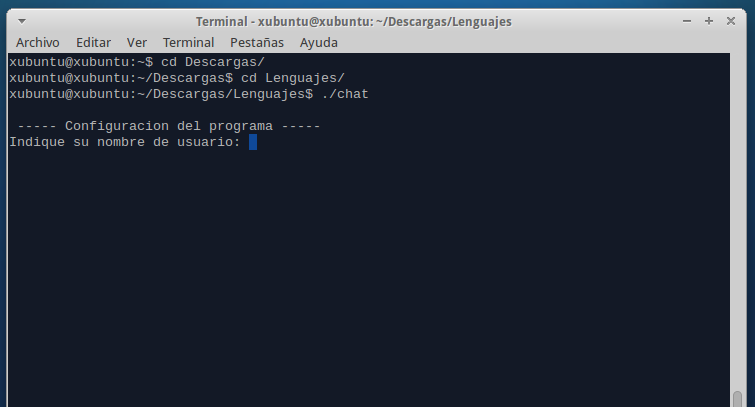
* Abrir la consola del sistema



* Ubicarse en la carpeta que se encuentra el programa



* Escribir la siguiente línea de comando .***/”nombre\_ejecutable*** y luego darle Enter



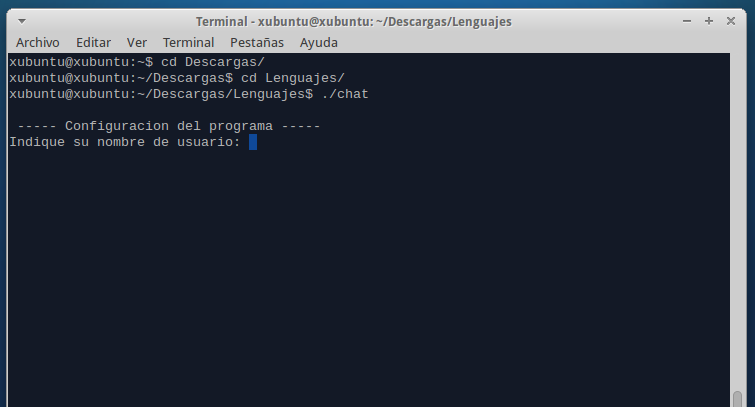
**Finalizado estos tres pasos el programa empieza a ejecutarse.**

**Nota**: Además debe tener en la misma ubicación del archivo ejecutable los dos archivos txt: amigos.txt y config.txt. Cuya función se explicará mas adelante.

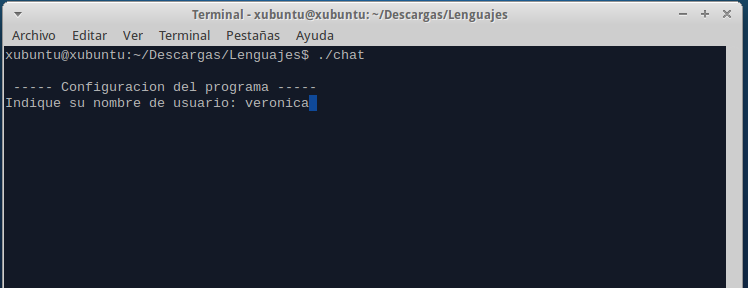
**Instrucciones de uso**

* ***Configuración de usuario***

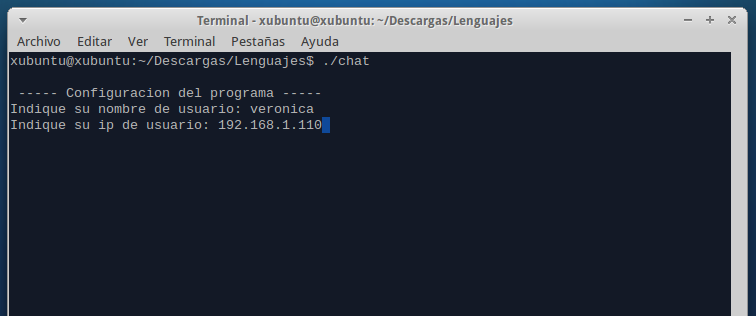
Si es la primera vez que ejecuta el programa, este le solicitara la información de configuración de usuario



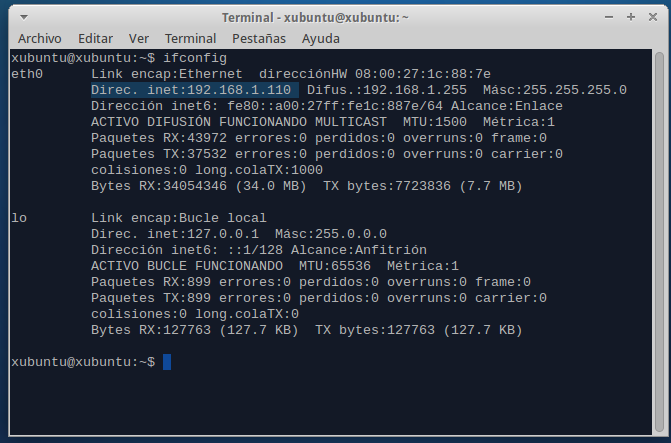
**Solicita el nombre de usuario:** El usuario debe ingresar el su nombre de usuario, este puede ser cualquier nombre, que el usuario desee.



**Solicita la ip del usuario:** El usuario debe ingresar la ip de su computadora, si ingresa otro número de ip, el programa no funcionara con normalidad.



**Nota:** Esta información la puede averiguar al ingresar el comando ***ifconfig***

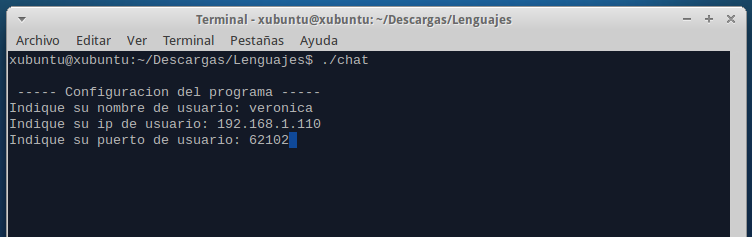


***La ip que se utiliza es la que esta en la tercera linea la que dice Direc. Inet:***

***En este caso la dirección es 192.168.1.110***

**Solicita el puerto del usuario:** El usuario debe ingresar el numero de puerto que desea utilizar.

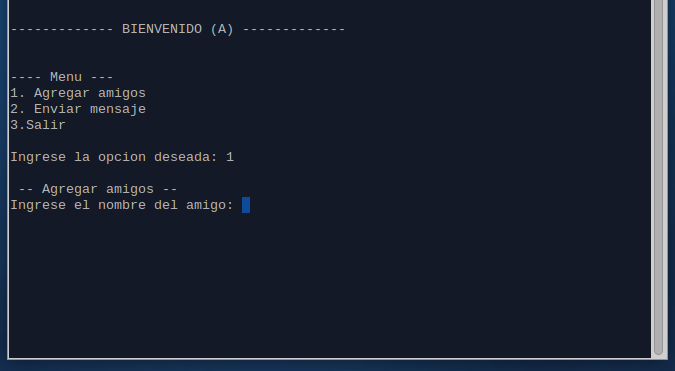
Debe tomar en cuenta que sea un puerto habilitado en la red.



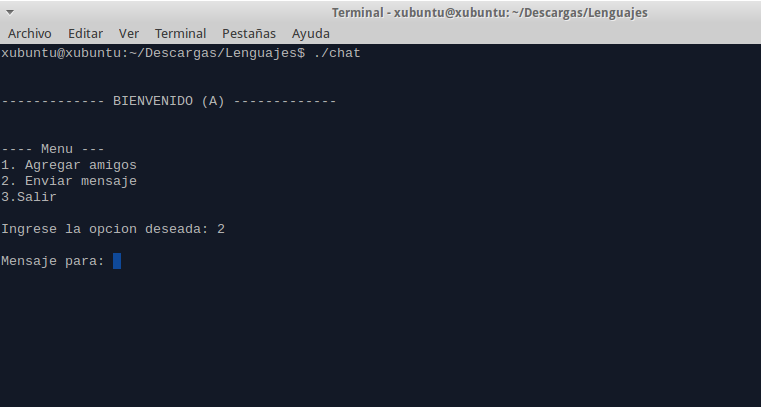
* ***Ingreso al sistema***

Luego de configurar la información de usuario, o cada vez que se vuelva abrir el programa se desglosará el siguiente menú. El menú le presenta al usuario tres opciones por realizar, el usuario debe ingresar el número de la opción que desea realizar,

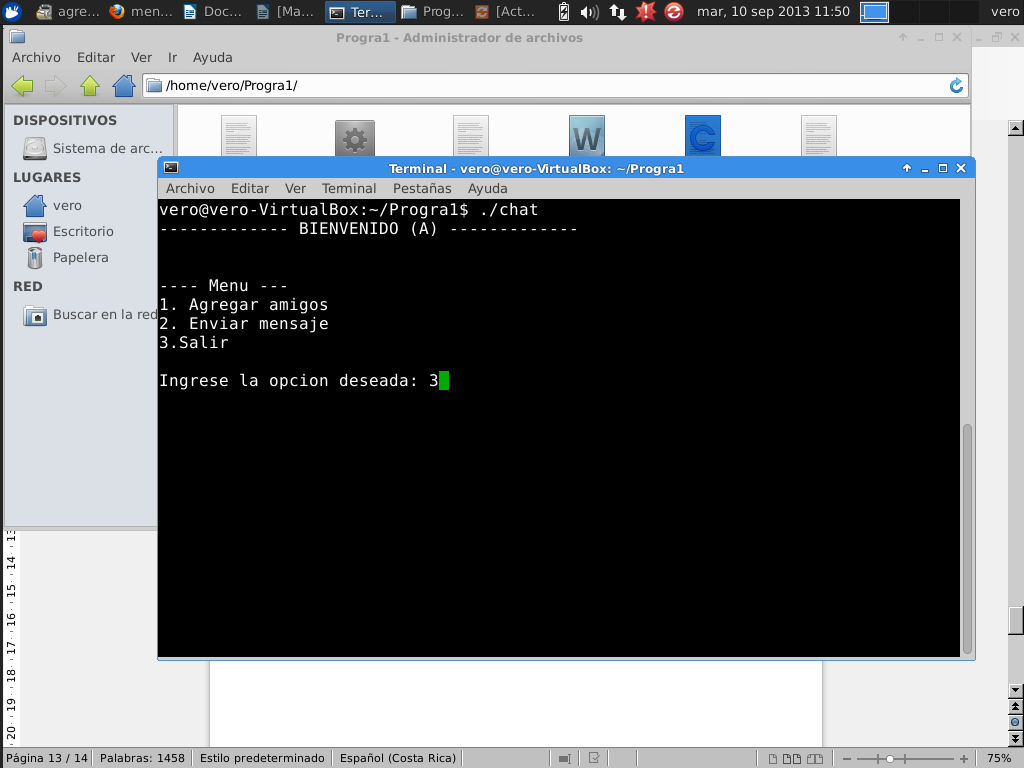
1. Agregar amigos *ver punto* ***Agregar amigos***



1. Enviar mensajes *ver punto* ***Enviar mensajes***

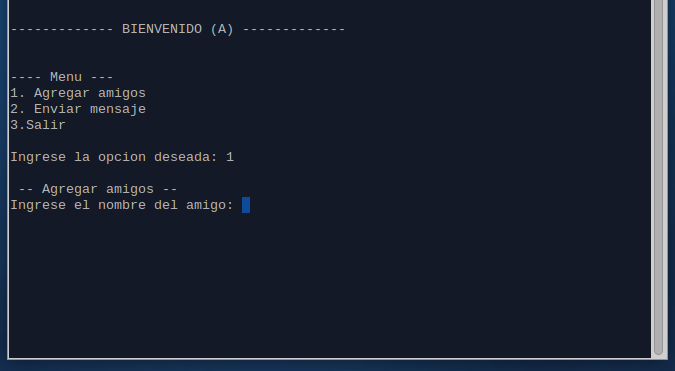


1. Salir del sistema *ver punto* ***Salir***

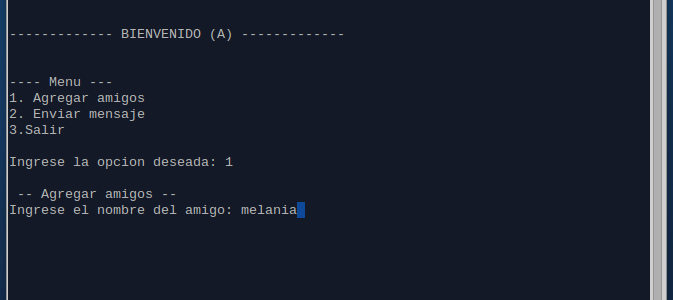


* ***Agregar amigos***

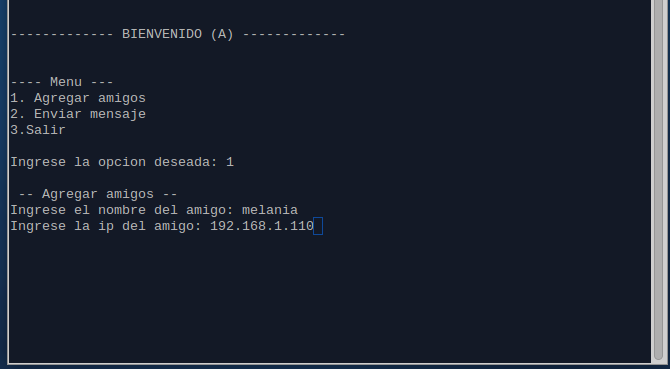
Ingreso a la función de agregar amigos, cada amigo que agrega en el programa queda guardado en el archivo amigos.txt. El archivo amigos.txt funciona como la agenda de contacto del programa, es una lista de amigos que contienen nombre de usuario, ip de usuario y puerto de usuario, información importante para el envió de mensajes.



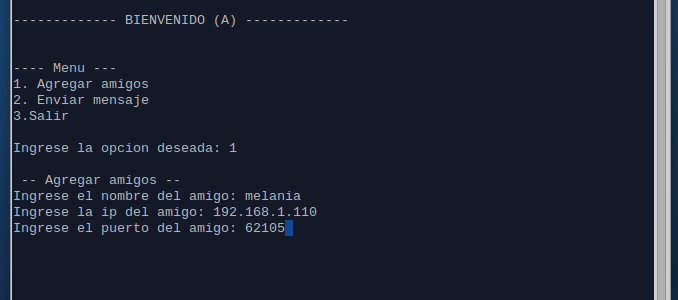
**Solicita el nombre del amigo:** El usuario debe ingresar el nombre del amigo



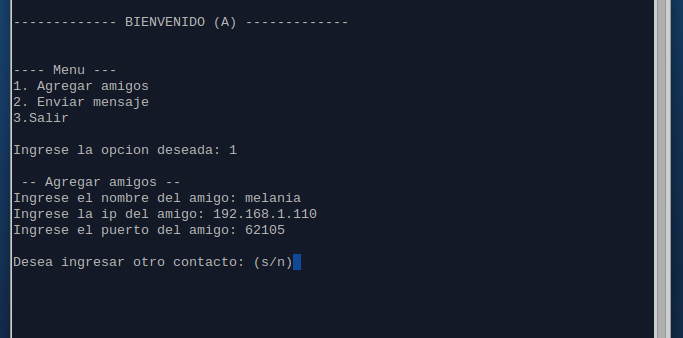
**Solicita la ip del amigo:** El usuario debe ingresar la ip del amigo.



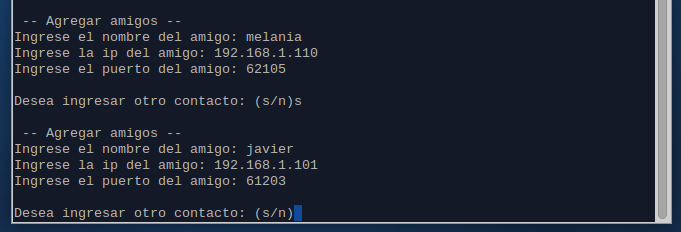
**Solicita el puerto del amigo:** El usuario debe ingresar el número de puerto del amigo.



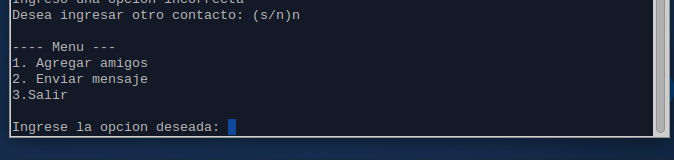
Una vez ingresado los 3 datos anteriores, el sistema le pregunta si quiere agregar otro amigo.



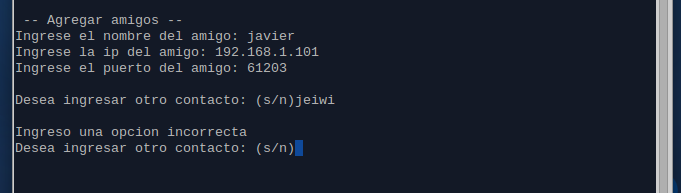
Si la respuesta es –s, el programa se devuelve a la funcionalidad agregarAmigos();



Si la respuesta es –n, el programa muestra el menú principal

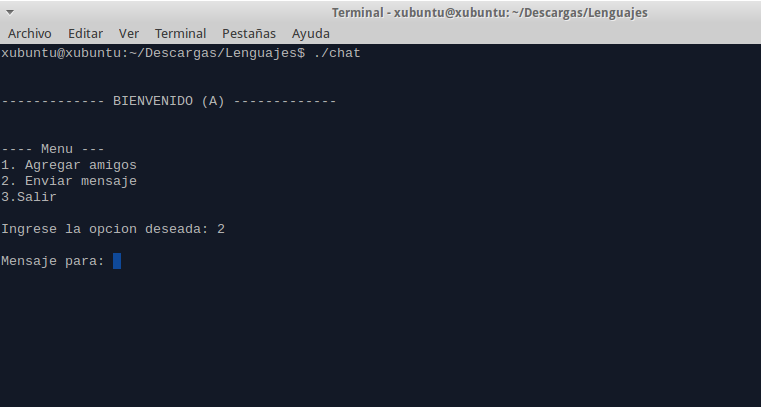


Si la respuesta es distinta de –s ó –n le indica al usuario que se produjo un error y le vuelve a solicitar al usuario la opción deseada.

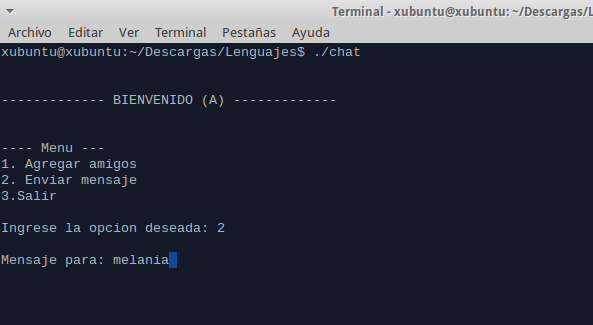


* ***Enviar mensajes***

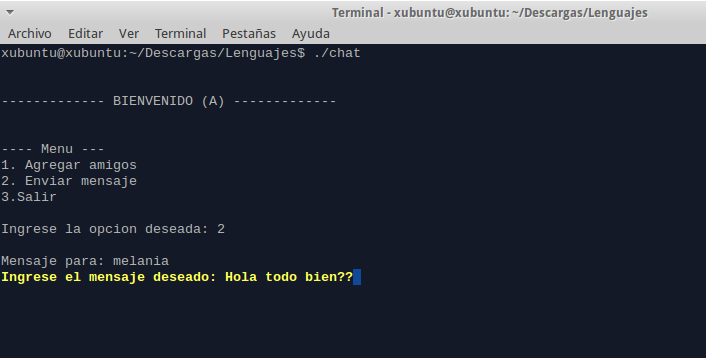
Ingreso a la función enviar mensaje, permite enviar un mensaje a un amigo que se encuentre en la lista de contacto (archivo amigos.txt)



**Solicita el mensaje para:** en esta opción el usuario debe ingresar el nombre del amigo al cual le desea enviar un amigo.



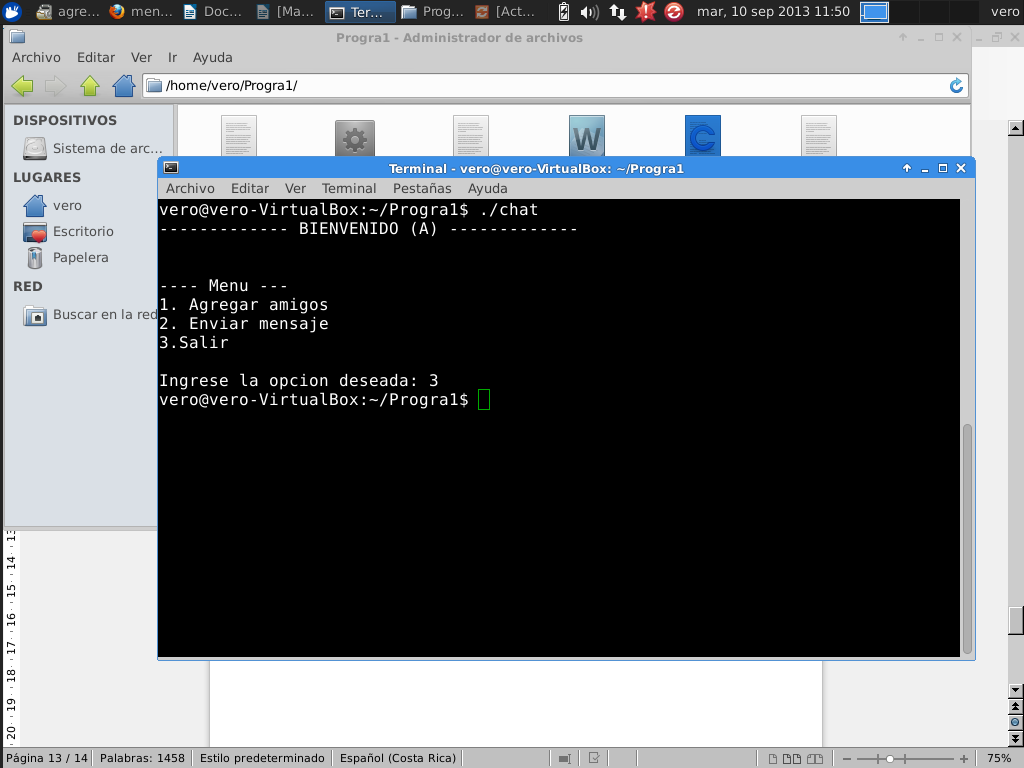
**Solicita el mensaje:** el usuario puede ingresar el mensaje que desea enviar a su amigo



Si el mensaje se logro enviar correctamente muestra un mensaje de afirmación. Sino envía un mensaje de error, según el error que se presentó.

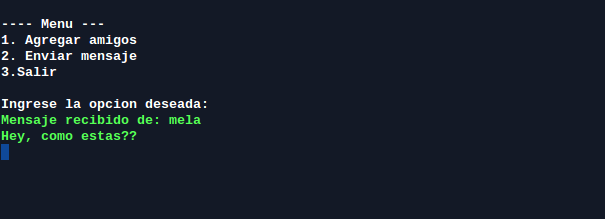
* ***Salir***

Función que cierra la ejecución del programa.



* ***Recibir mensajes***

El programa siempre esta ejecutando esta opción por debajo del sistema, para poder recibir en cualquier momento el mensaje de algún amigo.



**Bibliografía**

IBM (2010). Z/OS V1R11 information center (April 2010) Recuperado de <http://publib.boulder.ibm.com/infocenter/zos/v1r11/index.jsp?topic=/com.ibm.zos.r11.bpxbd00/typesh.htm>

The C Standard Library (s. f.). Recuperado de http://www.tutorialspoint.com/c\_standard\_library/