Communicator

Aplicación de mensajería realizada en el lenguaje de programación C

Tarea Programada I

Realizada por:

Jimmy Granados

Fabían Fernández

Esteban Leandro

Table of Contents

[Descripción del Problema. 2](#_Toc384083369)

[Librerías usadas. 2](#_Toc384083370)

[Decisiones del programa. 3](#_Toc384083371)

[Algoritmos Implementados. 3](#_Toc384083372)

[Análisis de resultados. 6](#_Toc384083373)

[Conclusión. 6](#_Toc384083374)

Documentación Communicator

# Descripción del Problema.

Se nos pidió crear un programa de mensajería que creara una conexión entre 2 computadoras por medio de sockets siguiendo el modelo TCP/IP.

Era necesario que un solo programa funcionara tanto como cliente como servidor, esto para poder enviar mensajes entre 2 computadoras sin la necesidad de crear más de una aplicación.

Además de esto, debía poder almacenar y consultar contactos de una manera persistente en un archivo de texto.

# Librerías usadas.

Para la realización de nuestro programa, utilizamos las siguientes librerías:

* **stdio.h**

Librería que incluye la declaración de las funciones más populares del lenguaje, además de eso, es la que permite la lectura y escritura en los estándar streams.

* **<sys/types.h>**

Una librería que permite el uso de distintos tipos dentro del lenguaje de programación.

* **<sys/socket.h>**

Es la librería que define la estructura socketaddr, y permite usar funciones como socklen\_t otras similares para el uso de sockets.

* **<netinet/in.h>**

Es la librería que define la estructura sockaddr\_in, en la cual se almacenan las dirección de internet utilizadas en la tarea programada.

* **<netdb.h>**

Librería que se utiliza a la hora de realizar operaciones en la red.

* **<stdlib.h>**

Librería que se puede utilizar para la conversión de tipos, asignación de memoria, control de procesos y operaciones matemáticas.

* **<String.h>**

Librería que incluye las funciones utilizadas para manejo de Strings y arreglos de caracteres en C.

# Decisiones del programa.

Para la realización de la tarea programada, tomamos distintas decisiones de diseño las cuales afectaron las formas en las que se comporta el programa, entre ellas están:

* Cargar los contactos a memoria desde que inicia el programa y no hacer la búsqueda dentro del TXT, para aumentar el desempeño del programa ya que se evita la apertura del mismo archivo varias veces.
* Separar el programa en distintos módulos, como cliente y servidor separados, para así aumentar la portabilidad del código escrito
* Crear un programa estilo menú que lo que hace es cargar el código escrito en diferentes módulos
* Uso de colores para mensajes de estado del programa como advertencia, error e información.
* Cargar la IP del usuario y el puerto a usar desde un archivo conf.txt

# Algoritmos Implementados.

Para la implementación del programa no creamos tantos algoritmos específicos, ya que usamos funciones de librerías del sistema que realizaban el trabajo, los algoritmos creados se detallan a continuación:

* A la hora de guardar/cargar/consultar contactos, se utiliza un arreglo de struct contacto, en el cual se guarda el nombre, dirección IP y puerto por medio del cual se realiza la comunicación. Este struct se llena una vez que inicia el programa y carga los datos desde el archivo contactos.txt, en el cual se almacenan los datos de todos los contactos y se utiliza como una “base de datos” de clientes.

Diagrama UML de la función utilizada para cargar los contactos a un arreglo.

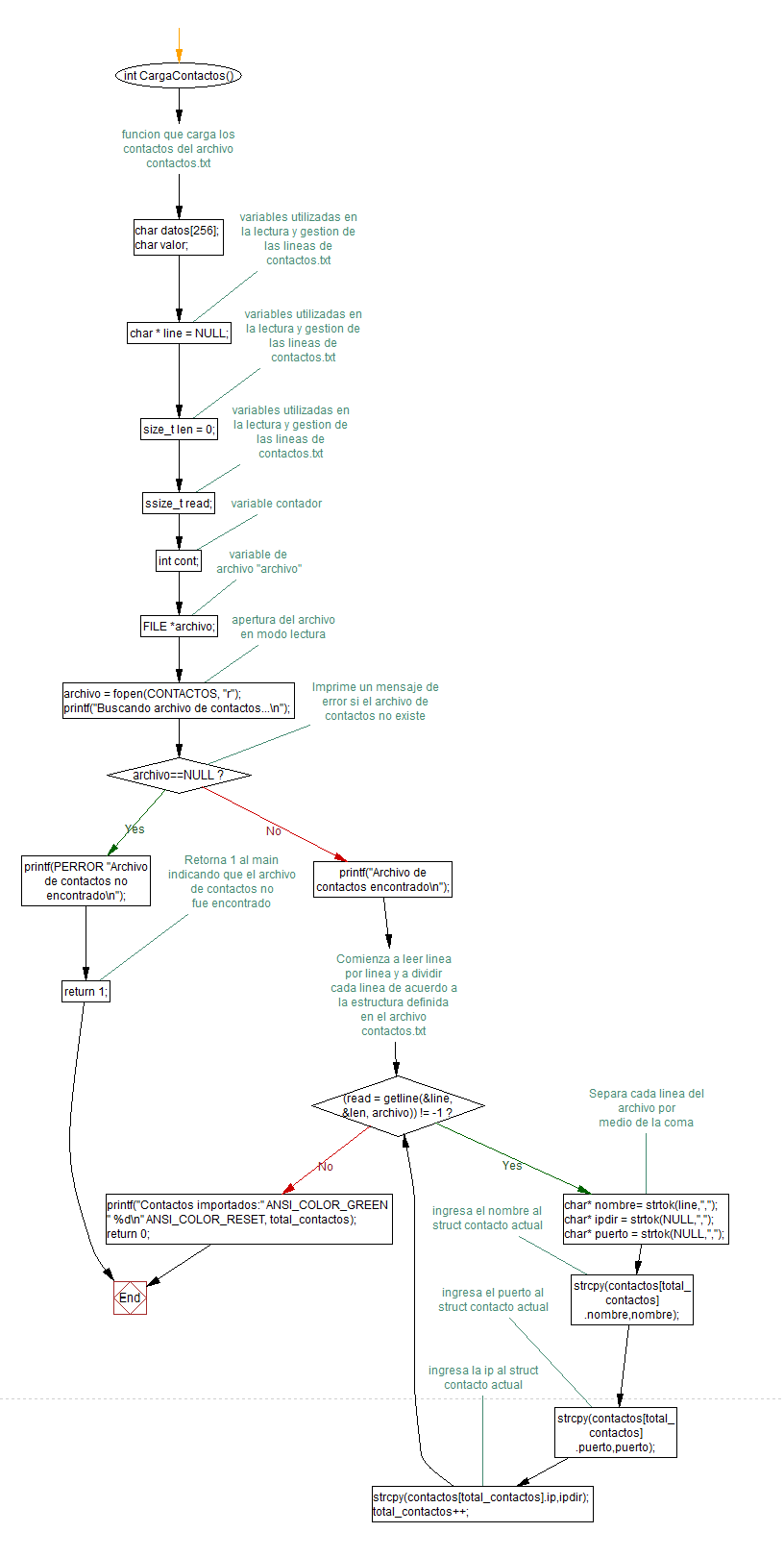
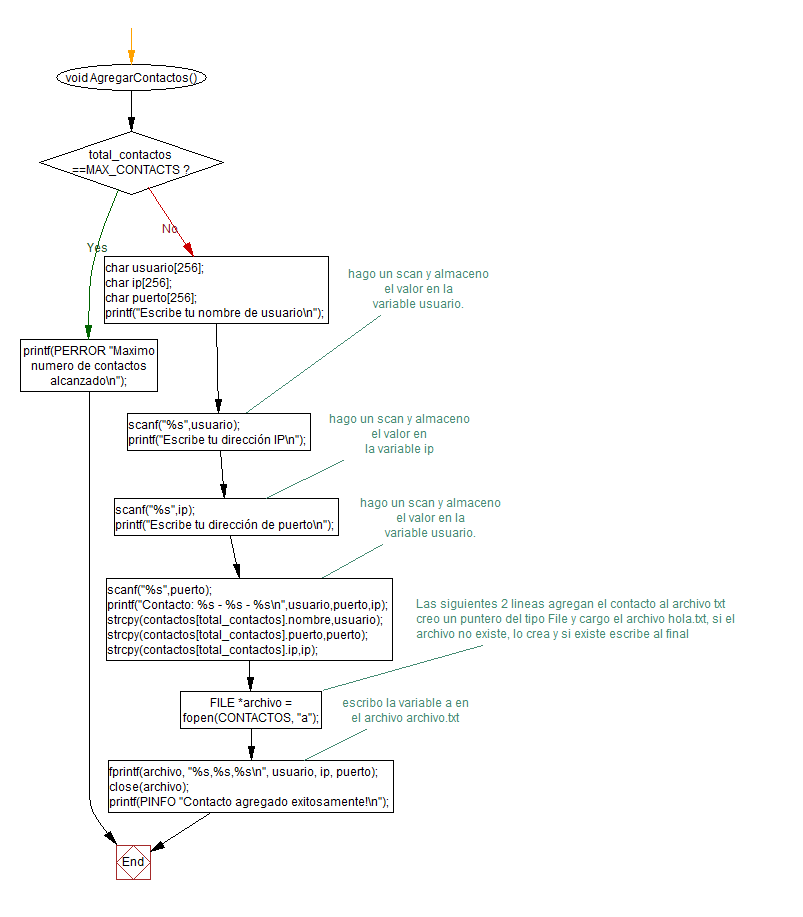
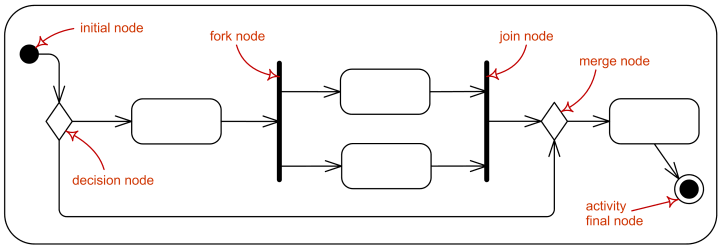


Diagrama UML de la función que agrega un nuevo contacto al arreglo y al archivo de texto.



Y un diagrama (no representativo de nuestro programa) UML de un fork



# Análisis de resultados.

Para la tarea programada, pudimos realizar todo lo solicitado en la especificación, dentro de los objetivos solicitados que cumplimos se cumplen:

* Comunicación entre equipos
* Cambio de colores en la consola
* Bifurcación del programa por medio de la función fork()
* Envío de mensajes por medio de sockets
* Almacenamiento y consulta de contactos
* Implementación de una archivo de tipo MAKEFILE para la compilación

# Conclusión.

Durante la realización de la tarea programada, llegamos a darnos cuenta que el lenguaje de programación C es un lenguaje muy poderoso, que permite la realización de diversos tipos de programas, y que está muy bien documentado en internet y libros debido a los más de 40 años de existencia del mismo.

También concluimos que es un lenguaje mucho más completo que lenguajes de más alto nivel como Python y Ruby, y que realizar tareas simples, a veces se puede tornar complicado si no se tiene conocimiento previo del lenguaje de programación.

Además, durante la realización de la tarea programada, aprendimos conceptos bastante importantes como lo son:

* El modelo TCP/IP
* Modelo OSI
* Comunicación por medio de sockets
* Uso de sistemas operativos alternativos a MS Windows
* Conceptos de UNIX

Por lo que consideramos que la tarea programada nos dejó mucho conocimiento técnico que podemos utilizar más adelante en nuestras carreras y en el ámbito profesional.