 Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ing. en Tecnologías de la información

Computo Distribuido

Resumen

Implementación de chat y envío de archivos cliente-servidor

Profesor:

Fernando Rodríguez Bernal

Alumnos:

Kevin Armas Hernández

Vicente Zavaleta Sanchez

Yosselin Pablo Ruiz

Matriculas:

201933809

202062102

201926783

A 19 de mayo de 2023

# Introducción

El objetivo principal de este proyecto es proporcionar una solución eficiente y confiable para la comunicación y el intercambio de archivos, utilizando los conceptos y las técnicas propias de la programación en red. Para ello, se ha desarrollado una aplicación de chat basada en sockets TCP/IP, que permite la comunicación en tiempo real entre un cliente y un servidor.

A lo largo de este reporte, se detallarán los aspectos técnicos de la implementación, incluyendo la estructura del código, los protocolos de comunicación utilizados, así como las técnicas de manejo de archivos y la gestión de errores. También se presentarán los resultados obtenidos en términos de eficiencia, confiabilidad y escalabilidad del sistema.

En resumen, este informe describe la implementación de un sistema cliente-servidor en C, capaz de activar un chat bidireccional y facilitar la transferencia de archivos entre ambas partes. Este sistema tiene aplicaciones prácticas en diversos contextos, como la colaboración en línea, el intercambio de datos y la comunicación en tiempo real. A lo largo del informe, se explorarán los detalles de su implementación y se evaluarán sus funcionalidades y desempeño.

## Desarrollo

La implementación del sistema cliente-servidor consta de dos partes principales: el cliente y el servidor. El cliente se encarga de establecer la conexión con el servidor, enviar mensajes de chat y solicitar la transferencia de archivos. Por otro lado, el servidor acepta conexiones entrantes de los clientes, administra los mensajes recibidos y facilita la transferencia de archivos.

El chat implementado permite una conversación fluida y segura entre el cliente y el servidor, asegurando que los mensajes se entreguen de manera confiable y en el orden correcto. Además, se han implementado comandos específicos para solicitar la transferencia de archivos entre ambas partes, lo que brinda la posibilidad de enviar archivos desde el cliente al servidor y viceversa.

## Implementación

En la siguiente imagen podemos observar como inicializamos el cliente en este caso montado en una maquina virtual con Alma Linux como sistema operativo y el compilador GCC.

A screenshot of a computer

Description automatically generated

Posteriormente vemos lo mismo pero el inicio del lado del servidor.

A screenshot of a computer

Description automatically generated

Es importante en nuestro caso inicializar primero el servidor ya que el cliente hace la solicitud de comunicación al servidor y por lo tanto si no este iniciado marcara error. En la siguiente imagen podemos observar que una vez inicializado el servidor nos muestra que ya está escuchando al puerto en el cual se llevara a cabo la comunicación.

A screenshot of a computer

Description automatically generated

En la siguiente imagen podemos ver que se muestra un menú una vez ejecutado el código y que se establece la comunicación con el servidor. En el menú vemos las opciones de:

1.- Enviar un archivo. Esto habilitara que el cliente pueda enviar un archivo al servidor.

2.- Solicitar un archivo al servidor. Esto hará que le solicitemos un archivo al servidor y el servidor lo envíe al cliente.

3.- Chat con el servidor. Esto hará que podamos empezar un chat con el servidor, el servidor siempre iniciándolo; esto hasta que terminemos la conexión con el servidor.

A screenshot of a computer

Description automatically generated

Aquí podemos observar lo que nos muestra si elegimos la primera opción “Enviar un archivo” la cual nos pide el nombre del archivo a enviar junto con su extensión y de ser encontrado se lo enviara al servidor.

A screenshot of a computer

Description automatically generated

La siguiente imagen muestra la confirmación del envió del lado del servidor.

A screenshot of a computer

Description automatically generated

En la siguiente imagen podemos ver lo que sucede con la opción 2 “Solicitar archivo al servidor”. Donde pedirá el nombre del archivo que deseas recibir con su extensión. Una vez recibido nos dará la confirmación de recepción.

A screenshot of a computer

Description automatically generated

En la siguiente imagen vemos el texto de confirmación del envío del archivo del lado del servidor.

A screenshot of a computer

Description automatically generated

En la siguiente imagen podemos observar lo que sucede si seleccionas la opción 3 “Chat con el servidor” en donde una vez seleccionada tiene que primero escribir el servidor y posteriormente el chat y así sucesivamente hasta que terminemos la comunicación.

A screenshot of a computer

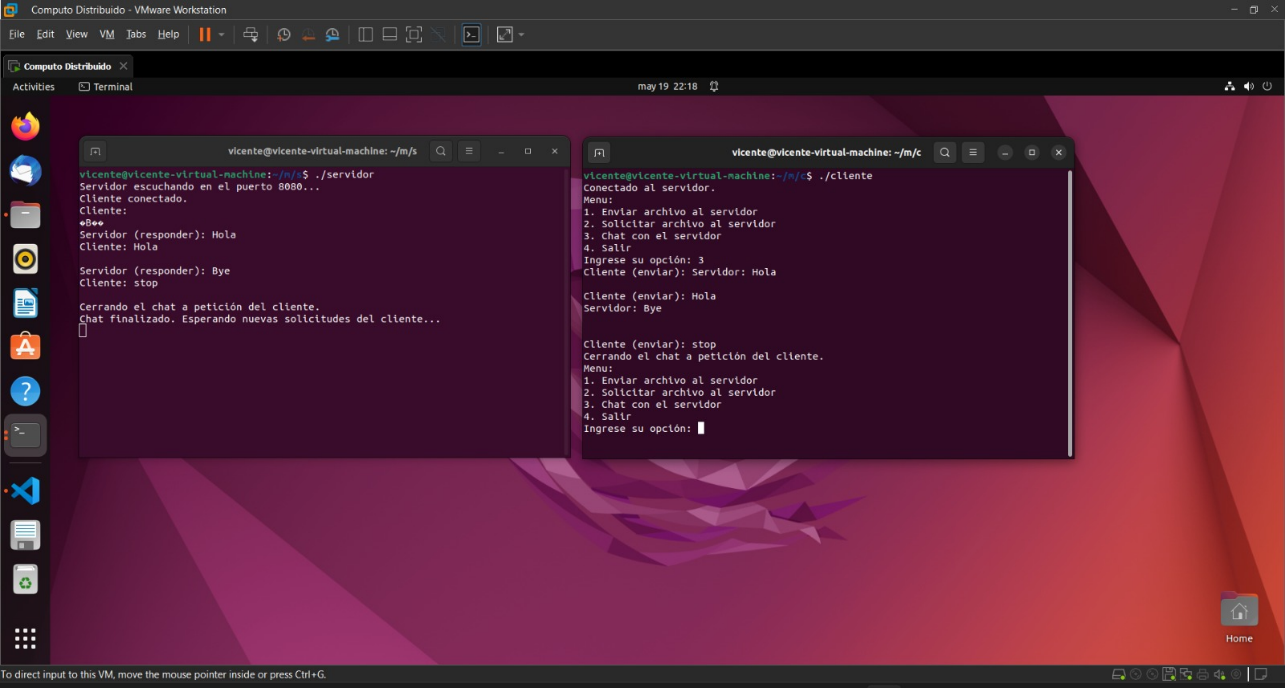
Description automatically generated

En la siguiente imagen podemos observar como se lleva a cabo el chat del lado del servidor.

A screenshot of a computer

Description automatically generated

Aquí podemos observar como dentro del mismo chat se integran las funciones de mandar y solicitar un archivo para lo que a nivel lógica se tiene que dentro de la función de chat mandamos a llamar a las funciones de recibir archivo o enviar archivo a lo cual al ser otros hilos interrumpe temporalmente o inhabilita el chat ya que el canal es ocupado por el envío del archivo.



## Código del servidor

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <string.h>*

*#include <unistd.h>*

*#include <arpa/inet.h>*

*#define MAX\_BUFFER\_SIZE 1024*

*void receiveFile(int clientSocket) {*

*// Recepción del nombre de archivo del cliente*

*char filename[MAX\_BUFFER\_SIZE];*

*read(clientSocket, filename, sizeof(filename));*

*// Recepción del contenido del archivo del cliente*

*FILE \*file = fopen(filename, "wb");*

*if (file == NULL) {*

*perror("Error al crear el archivo");*

*return;*

*}*

*char buffer[MAX\_BUFFER\_SIZE];*

*int bytesRead;*

*while ((bytesRead = read(clientSocket, buffer, sizeof(buffer))) > 0) {*

*fwrite(buffer, 1, bytesRead, file);*

*}*

*printf("Se ha recibido el archivo '%s' del cliente.\n", filename);*

*fclose(file);*

*}*

*void sendFile(int clientSocket) {*

*// Recepción del nombre de archivo solicitado por el cliente*

*char filename[MAX\_BUFFER\_SIZE];*

*read(clientSocket, filename, sizeof(filename));*

*FILE \*file = fopen(filename, "rb");*

*if (file == NULL) {*

*perror("Error al abrir el archivo");*

*return;*

*}*

*// Envío del contenido del archivo al cliente*

*char buffer[MAX\_BUFFER\_SIZE];*

*int bytesRead;*

*while ((bytesRead = fread(buffer, 1, sizeof(buffer), file)) > 0) {*

*write(clientSocket, buffer, bytesRead);*

*}*

*printf("Se ha enviado el archivo '%s' al cliente.\n", filename);*

*fclose(file);*

*}*

*void chat(int clientSocket) {*

*char message[MAX\_BUFFER\_SIZE];*

*while (1) {*

*// Recepción del mensaje del cliente*

*int bytesRead = read(clientSocket, message, sizeof(message));*

*if (bytesRead <= 0) {*

*break;*

*}*

*// Imprimir el mensaje del cliente*

*printf("Cliente: %s\n", message);*

*// Envío de la respuesta al cliente*

*printf("Servidor (responder): ");*

*fgets(message, sizeof(message), stdin);*

*write(clientSocket, message, strlen(message));*

*}*

*}*

*int main() {*

*// Configuración del servidor*

*struct sockaddr\_in serverAddress, clientAddress;*

*serverAddress.sin\_family = AF\_INET;*

*serverAddress.sin\_port = htons(8080);*

*serverAddress.sin\_addr.s\_addr = INADDR\_ANY;*

*// Creación del socket*

*int serverSocket = socket(AF\_INET, SOCK\_STREAM, 0);*

*if (serverSocket == -1) {*

*perror("Error al crear el socket");*

*exit(EXIT\_FAILURE);*

*}*

*// Vinculación del socket al puerto*

*if (bind(serverSocket, (struct sockaddr \*)&serverAddress, sizeof(serverAddress)) == -1) {*

*perror("Error al vincular el socket al puerto");*

*exit(EXIT\_FAILURE);*

*}*

*// Espera de conexiones entrantes*

*if (listen(serverSocket, 5) == -1) {*

*perror("Error al escuchar");*

*exit(EXIT\_FAILURE);*

*}*

*printf("Servidor escuchando en el puerto 8080...\n");*

*while (1) {*

*// Aceptación de la conexión del cliente*

*socklen\_t clientAddressLength = sizeof(clientAddress);*

*int clientSocket = accept(serverSocket, (struct sockaddr \*)&clientAddress, &clientAddressLength);*

*if (clientSocket == -1) {*

*perror("Error al aceptar la conexión del cliente");*

*exit(EXIT\_FAILURE);*

*}*

*printf("Cliente conectado.\n");*

*int option;*

*read(clientSocket, &option, sizeof(option));*

*switch (option) {*

*case 1:*

*receiveFile(clientSocket);*

*break;*

*case 2:*

*sendFile(clientSocket);*

*break;*

*case 3:*

*chat(clientSocket);*

*break;*

*default:*

*printf("Opción inválida recibida del cliente.\n");*

*}*

*// Cierre de la conexión con el cliente*

*close(clientSocket);*

*printf("Cliente desconectado.\n");*

*}*

*// Cierre del socket del servidor*

*close(serverSocket);*

*return 0;*

*}*

## Codigo del cliente

#include <stdio.h>

*#include <stdlib.h>*

*#include <string.h>*

*#include <unistd.h>*

*#include <arpa/inet.h>*

*#define MAX\_BUFFER\_SIZE 1024*

*void chat(int serverSocket) {*

*char message[MAX\_BUFFER\_SIZE];*

*while (1) {*

*// Envío del mensaje al servidor*

*printf("Cliente (enviar): ");*

*fgets(message, sizeof(message), stdin);*

*write(serverSocket, message, strlen(message));*

*// Recepción de la respuesta del servidor*

*int bytesRead = read(serverSocket, message, sizeof(message));*

*if (bytesRead <= 0) {*

*break;*

*}*

*// Imprimir la respuesta del servidor*

*printf("Servidor: %s\n", message);*

*// Chequear si el usuario quiere salir*

*if (strcmp(message, "stop") == 0) {*

*break;*

*}*

*}*

*}*

*int main() {*

*// Configuración del servidor*

*struct sockaddr\_in serverAddress;*

*serverAddress.sin\_family = AF\_INET;*

*serverAddress.sin\_port = htons(8080);*

*serverAddress.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); // Cambia esta dirección IP por la del servidor*

*// Creación del socket*

*int clientSocket = socket(AF\_INET, SOCK\_STREAM, 0);*

*if (clientSocket == -1) {*

*perror("Error al crear el socket");*

*exit(EXIT\_FAILURE);*

*}*

*// Conexión al servidor*

*if (connect(clientSocket, (struct sockaddr \*)&serverAddress, sizeof(serverAddress)) == -1) {*

*perror("Error al conectar con el servidor");*

*exit(EXIT\_FAILURE);*

*}*

*printf("Conectado al servidor.\n");*

*int option;*

*printf("Menu:\n");*

*printf("1. Enviar archivo al servidor\n");*

*printf("2. Solicitar archivo al servidor\n");*

*printf("3. Chat con el servidor\n");*

*printf("Ingrese su opción: ");*

*scanf("%d", &option);*

*// Envío de la opción al servidor*

*write(clientSocket, &option, sizeof(option));*

*switch (option) {*

*case 1: {*

*// Envío del nombre de archivo al servidor*

*char filename[MAX\_BUFFER\_SIZE];*

*printf("Ingrese el nombre del archivo a enviar: ");*

*scanf("%s", filename);*

*write(clientSocket, filename, sizeof(filename));*

*// Lectura y envío del contenido del archivo al servidor*

*FILE \*file = fopen(filename, "rb");*

*if (file == NULL) {*

*perror("Error al abrir el archivo");*

*exit(EXIT\_FAILURE);*

*}*

*char buffer[MAX\_BUFFER\_SIZE];*

*int bytesRead;*

*while ((bytesRead = fread(buffer, 1, sizeof(buffer), file)) > 0) {*

*write(clientSocket, buffer, bytesRead);*

*}*

*fclose(file);*

*printf("Archivo enviado al servidor.\n");*

*break;*

*}*

*case 2: {*

*// Envío del nombre de archivo solicitado al servidor*

*char filename[MAX\_BUFFER\_SIZE];*

*printf("Ingrese el nombre del archivo a solicitar: ");*

*scanf("%s", filename);*

*write(clientSocket, filename, sizeof(filename));*

*// Recepción y escritura del contenido del archivo recibido*

*FILE \*file = fopen(filename, "wb");*

*if (file == NULL) {*

*perror("Error al crear el archivo");*

*exit(EXIT\_FAILURE);*

*}*

*char buffer[MAX\_BUFFER\_SIZE];*

*int bytesRead;*

*while ((bytesRead = read(clientSocket, buffer, sizeof(buffer))) > 0) {*

*fwrite(buffer, 1, bytesRead, file);*

*}*

*fclose(file);*

*printf("Archivo recibido del servidor.\n");*

*break;*

*}*

*case 3:*

*do {*

*chat(clientSocket);*

*break;*

*}while(opción!='stop')*

*default:*

*printf("Opción inválida.\n");*

*}*

*// Cierre de la conexión con el servidor*

*close(clientSocket);*

*return 0;*

*}*

## Conclusión

En conclusión, la implementación de un sistema cliente-servidor para chat y transferencia de archivos en C ha demostrado ser una solución eficiente y confiable para facilitar la comunicación en tiempo real y el intercambio de archivos en el ámbito de la informática.

A lo largo del proyecto, se ha logrado desarrollar una aplicación basada en sockets TCP/IP que permite a los clientes establecer una conexión con el servidor, enviar mensajes de chat y solicitar la transferencia de archivos. El sistema ha demostrado ser capaz de mantener una conversación fluida y segura, asegurando que los mensajes se entreguen de manera confiable y en el orden correcto.

En resumen, la implementación de este sistema cliente-servidor en C ha cumplido con éxito los objetivos propuestos, brindando una solución completa y robusta para la comunicación en tiempo real y la transferencia de archivos. Este proyecto sienta las bases para futuros desarrollos en el ámbito de las aplicaciones de red, y ofrece una sólida base de conocimientos y experiencia en la implementación de sistemas de cómputo distribuido.