Redes y Comunicaciones

Trabajo Práctico

2do Cuatrimestre de 2019

|  |  |  |
| --- | --- | --- |
| Documento | Nombre | Evaluación Individual |
|  |  |  |
| 36068006 | José Victor Ibañez |  |
|  |  |  |
|  |  |  |
| Evaluación Trabajo |  | |

Manejo de comandos:

La aplicación cuenta con tres comandos:

* (M): envía un mensaje al otro usuario. Ej: (M) ¿Hola como estas?
* (T): transfiere un archivo. Ej: (T) C:\nombrearchivo.extencion (sin comillas).
* (F): cierra la comunicación entre los usuarios. Ej: (F).

Durante el envío de un mensaje lo primero que se hace es determinar que comando ingreso el usuario. Para esto una vez que se recibe la cadena que se ingresó por pantalla, se cortan los primeros 4 caracteres con la función substr() y se los guardan en otro string recientemente creado. A través de la función strcpy() se transforma el string en una cadena para poder usar el comando strcmp() y poder comparar esos primeros 4 caracteres con las funciones pre-designadas. Para poder determinar que comando se ingresó se usa una secuencia de if anidados que determinan que comando ingreso el usuario. De acuerdo al comando se realiza la acción correspondiente, pero si se ingresó una función inexistente si le pide al usuario que lo vuelva a intentar.

Para recibir un mensaje se utiliza el mismo método para determinar de qué comando se trata y así poder realizar la acción correcta, ya sea solo imprimir un mensaje, cerrar la aplicación o en su defecto guardar un archivo.

Envío y recepción de mensajes:

Para poder enviar un mensaje primero se crea un string en donde se guardará la cadena que ingrese el usuario. Luego se llamará a la función pedirCadena() la cual se encarga de mostrarle un mensaje al usuario y recibir la cadena que ingrese que luego devolverá para determinar qué tipo de comando ingresó. Si es un comando de mensaje entonces se lo enviará utilizando el comando send() que provee la librería winsock.

Para recibir un mensaje es similar al envío. Primero se comprueba a que comando pertenece y si es un mensaje lo recibe y lo muestra por pantalla.

Transferencia de archivos:

Para la transferencia de archivos, primero se le avisa al usuario que inicio un envío de algún archivo, y se abre el archivo para ver si existe, en caso de que no existe se le avisa al usuario el error. Si no hay error al abrir el archivo primero se crea una variable entera en la cual se guardara el tamaño total del archivo y se la enviará al usuario receptor para que lo pueda recibir correctamente y sin errores. Luego dentro de un bucle while se comenzará a leer y se enviará de a solo 100 bites hasta que se pasen todos los datos. Se le avisará al usuario que el archivo se envió de forma correcta y quedará listo para poder recibir un mensaje.

A la hora de recibir un archivo, se le avisa al usuario que esta por recibir uno, luego con el nombre y la ruta que se pasó desde el otro usuario se crea el archivo y se controla que no haya error, si lo hay se emitirá un mensaje. Si el archivo se pude crear de forma correcta el usuario pasará a estar en “recibiendo” hasta que le llegue el tamaño total. Una vez tenga el tamaño se armará un bucle while que se cortará una vez se llegue al tamaño total del archivo. Dentro del bucle se toman los datos que van llegando en el buffer y los va escribiendo en el archivo con la función fwrite(). Si el elemento no se pudo transferir completamente se emitirá un mensaje, caso contrario se le indicará al usuario que lo recibió sin errores.

Manejo de Socket:

Para poder realizar las llamadas de sockets, desde el servidor, junto con un cliente, nuestro programa siendo primitivo, el código del servidor se puede compilar y ejecutar en cualquier maquina Windows conectada a una red local, también se puede compilar y ejecutar el código del cliente en cualquier otra máquina que posee dicho Sistema operativo.

Primero explicó el código del servidor. Requerirá un PUERTO como 8080, este número es elegido al azar, todo número desde 1024 al 65535 funcionara siempre y cuando otro proceso no lo esté utilizando. Esto permite al socket inicializarse y verifica que no tenga errores. continua con verificar si la IP cargada por código fuente es correcta llamando la función “bind”. si fue correcto, continua y escucha la petición y lista el socket para anunciar que el servidor esta dispuesto a aceptar llamadas entrantes e indicar al sistema que almacene la llamada de un cliente, una vez realizado correctamente todo, el servidor esta iniciada y espera una conexión del cliente en dicho puerto.

Desde el cliente se debe iniciar y cargar la IP y un número de Puerto, este se obtiene desde el código fuente del servidor, y configuraciones cargada de dicha aplicación. A continuación, inicializa un socket para establecer la conexión TCP con el servidor mediante la función “connect”. Si el servidor está activo y ejecutándose en la máquina especificada y enlazada al Puerto, y si esta vacía su cola de “listen” se realiza la conexión. El cliente puede comenzar enviando comando por consola al servidor. Esto lo lleva a un bucle esperando peticiones por teclado hasta que la variable “quit” este en true y termina la conexión cerrando estos sockets.

Elección del Sistema operativo:

Se realizo pruebas tanto en Windows como en Linux, logrando mejores expectativas en Windows con la compatibilidades y acceso de nuestros equipos, se eligió el S.O debido por su interface cómoda y compatibilidad con el lenguaje de programación en C++.

Organización del bucle del Programa:

La aplicación se organiza de la siguiente manera, el siguiente diagrama de flujo demuestra de como se maneja las funciones en nuestra aplicación del servidor:

La aplicación comienzo esperando un comando de cliente, este lo recibe y lo lee , si es >>(M) va dirigido a la función “receiveMessage()” , si es >>(T) va a la función de recibir datos “recvData()”, y si es >>(F) finaliza y cierra el bucle caso contrario continua, los siguientes pasos son, pedir por consola un comando y realiza si es >>(M) enviar mensaje por la función “send()”, si es >>(T) a la función de enviar datos “sendData()”, y si es (F) finaliza la aplicación y cierra el socket.

Con respecto a Cliente se maneja de igual manera solamente que esta invertido, solicita primero un comando y luego espera respuesta del servidor

