2017

Memoria Chat IRC

GRUPO 2311 PAREJA 01
OSCAR GARCÍA DE LARA PARREÑO
SANTIAGO GÓMEZ AGUIRRE

Contenido

Introducción 2
Librería 2
Makefile 2
Servidor 2
  Objetivo 2
  Diseño 2
  Ejecución 3
  Comentarios 3
Cliente 3
  Objetivo 3
  Diseño 3
  Ejecución 4
  Comentarios 4
SSL 4
  Objetivo 4
  Diseño 4
  Ejecucion 4
  Comentarios 4

Introducción

En la presente memoria vamos a hablar de las decisiones de diseño de las tres practicas desarrolladas, ya que hemos decidido que era más correcto que crear tres memorias separadas. De esta forma por lo tanto el nombre de la memoria no incluye la numeración de la práctica que es necesaria para pasar la prueba de empaquetado, si se quiere probar se debe modificar.

En cuanto a la estructura del documento primero hablaremos de la librería propia que hemos usado para el desarrollo, después pasaremos a explicar cada práctica individualmente por lo que el contenido de la memoria se encuentra separado en **Servidor**, **Cliente** y **SSL**.

Librería

Como parte del diseño global de la practica hemos desarrollado una librería que es el nexo común de todas las practicas, se puede crear al compilar con el makefile, se compone de distintos archivos con una funcionalidad distinta:

- **G-2311-01-P1_semaforos.c**: Conjunto de funciones que nos permiten crear, manejas y destruir semáforos que fue desarrollada por nosotros mismos el año pasado en las prácticas de SOPER.
- **G-2311-01-P1_socket.c**: En este archivo está relacionado con la conexión, tenemos un par de funciones que nos permiten crear un socket y daemonizar un programa.
- **G-2311-01-P1_utils.c**: En este archivo hemos añadido las funciones genéricas que no encajan en los otros apartados, tenemos una función para crear un array de funciones y otra para liberar matrices de char.
- **G-2311-01-P3_ssl.c**: Toda la funcionalidad para trabajar con el protocolo SSL.

Makefile

El makefile contiene las siguientes reglas:

- all: genera la librería, certificados y ejecutalbles necesarios.
- clean: limpia el proyecto.
- documentacion: Crea los archivos man.
- comprimir: genera un archivo tar.gz con toda la carpeta dentro
- certificados: crea los tres certificados

Servidor

Objetivo

El objetivo de la práctica era realizar un servidor IRC que cumpliera el estándar definido en los RFCs y que pudiera trabajar con varios clientes.

Diseño

Para diseñar nuestro server hemos decidido lanzar un hilo para procesar cada mensaje recibido, para esto usamos select que nos permite saber si hay un cambio en un socket, conjuntamente con un array de sockets de usuarios y de semáforos, estos últimos evitan que, si llegan varios mensajes del mismo usuario, no se procesen secuencialmente, si son de distintos usuarios daría igual el orden.

También hemos diseñado un array de funciones que nos permite separar cada tipo de mensaje en una función distinta, lo que hace todo más depurable y cómodo de trabajar.

Ejecución

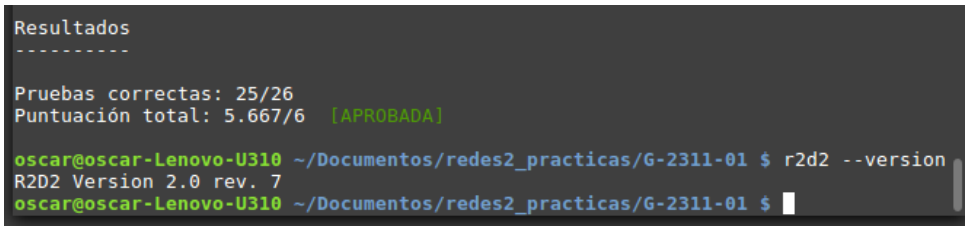
./server p 6667

O el puerto que se desee.

Comentarios

Hemos decidido comentar la función Quit del array de funciones porque falla y nos tiraba el resto de pruebas. Al parecer hay algún tipo de incompatibilidad entre select y la prueba quit porque todo el mundo que tiene esta combinación que pregunto le da problemas.

Al principio se nos congelaba el r2d2, después en la siguiente versión la pasábamos, pero esa versión volvía inestable otras y en la última que sacaron para corregir ese problema se volvió a romper Quit con un error muy curioso ya que nos informaba que el servidor parecía caído, pero seguía corriendo las demás sin cortarse que es lo que pasa cuando se cae, asi que tomamos la decisión de entregar con esa comentada mejor. Y como se ve en la siguiente captura es lo que pasamos actualmente:



Cliente

Objetivo

El objetivo de la práctica era realizar un cliente para el servidor IRC que cumpliera el estándar definido en los RFCs y que pudiera trabajar con varios clientes a la vez.

Diseño

La estructura y diseño de nuestro cliente se basa en crear un hilo un hilo adicional que le permita leer lo que recibe del servidor. Lo hemos dividido en tres partes:

- **G-2311-01-P2_client.c**: El esqueleto de este archivo, el cuál es la estructura base del cliente y su interfaz, nos fue proporcionado al comienzo de la práctica por lo que el diseño del cliente ha sido más guiado que en el caso del servidor. Aquí podemos encontrar las funciones principales de la estructura del cliente y la de envío de archivos y audio.
- **G-2311-01-P2_client_funciones.c**: En esta parte hemos desarrollado las funciones de los comandos que nuestro cliente recibe del servidor y que modifican la interfaz dependiendo de la funcionalidad que haya que ejercer en cada caso.
- **G-2311-01-P2_client_funciones_user.c**: En este archivo se encuentran las funciones del cliente que recogen la entrada de comandos en el sistema por parte del usuario para enviarlas al servidor.

Como en el servidor, hemos diseñado un array de funciones en los dos últimos archivos mencionados para separar cada tipo de comando y mensaje en funciones distintas.

En cuanto al sistema de transmisión de archivos lo hemos diseñado de manera que cada vez que el cliente desee hacer un envío cree un nuevo hilo en el que se crea un socket esperando la conexión del cliente al que se quiere enviar, el cual, tendrá que aceptar recibir el archivo

primero para posteriormente lanzar otro hilo que cree otro socket y realice dicha conexión que permita hacer efectiva la transmisión.

Ejecución

./client

Comentarios

El envío de audio no lo hemos realizado en esta entrega.

SSL

Objetivo

El propósito de la práctica era desarrollar una serie de funciones para poder usar conexiones con la capa SSL, además desarrollar un cliente y servidor echo de prueba y añadir esta capa a nuestro cliente y servidor IRC.

Diseño

Hemos añadido a nuestra librería las funciones de SSL necesarias como se indican en el manual de la práctica, el common name del certificado hemos tenido que poner P1 en vez de P3 que sería lo habitual para pasar la prueba y hemos desarrollado el cliente y servidor echo en unos ficheros aparte que solamente hacen esa función para poder generar los ejecutables como piden.

Ejecucion

./cliente_echo

./servidor_echo

Comentarios

En los echo hemos tenido que poner las rutas absolutas /Home/oscar/… hacia los certificados que se deben modificar, el motivo de esto es que si poníamos ../certs/ funcionaba al ejecutarlo manualmente, pero no pasábamos la prueba porque al parecer la prueba mueve los ejecutables de sitio y la única forma es ponerle la ruta absoluta.

No hemos implementado los cambios en el servidor y cliente IRC por falta de tiempo y el caos que supone tener que estar cambiando cada send y recv por si es ssl o no, además que nos podemos cargar algo que nos podría afectar en las anteriores entregas.