## 

Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Redes de Computadoras I CI-4835

Octubre 2013 - Enero 2014

**Proyecto 1**

Autores:

Marrero Erick 09-10981

Pereira Juan 09-11173

Sartenejas, Noviembre de 2013

**Informe Proyecto I**

El programa que se desea realizar es un chat de mensajes cortos que tenga la posibilidad de tener multiusuarios y multisalas dentro del mismo, y para ello es necesario establecer un servidor que permita las entradas o conexiones provenientes de otros computadores.

Para poder permitir la comunicación entre los clientes, se requiere crear un servidor y este servidor tendrá el nombre de schat, mientras que para los clientes que deseen establecer conexión con el mismo tendrán que invocar el programa cchat. De esta manera se implementa una comunicación a nivel de capa de aplicación entre servidor y cliente(s).

En el modulo del servidor se manejaron las múltiples conexiones de los usuarios mediante el uso de los hilos, esto es, cada vez que el programa reciba una conexión entrante se creará un hilo y se le asignará un socket a cada hilo. Se verifica además que el usuario a conectarse no exista ya en los usuarios conectados al servidor.

También se utilizó una estructura **struc** que contiene el nombre de la sala, un arreglo de usuarios y un arreglo de sus sockets, que están suscritos a esta sala. Primeramente a pesar que se conecten varios usuarios al servidor, si no se suscribe alguno a alguna sala, no contendrán ningún usuario conectado a las mismas, razón por la que si un usuario se conecta y, sin suscribirse a alguna sala, intenta mandar un mensaje, este no se enviará a nadie puesto que no está suscrito a ninguna sala.

Otro aspecto importante, es que cada vez que un cliente se conecta al servidor, se guarda esa conexión en un arreglo de usuarios (*lista\_conexiones*) que se conectan al servidor, esto sin aun estar suscritos a ninguna sala. Junto con este arreglo de clientes que se conectan se almacenan también sus sockets (*lista\_sockets*) para cuando se necesiten enviar mensajes a los que estén suscritos en salas comunes.

En el momento en que un cliente o usuario se conecta al servidor y además se suscribe a alguna sala, se busca en el arreglo de la estructura definida previamente, la posición en la que se encuentra la sala a la cual se desea suscribir el cliente y se agrega el cliente a la sala deseada.

Nótese además que en el modulo del servidor no se puede quedar sin sala asignada, por lo que si un usuario trata de borrar una sala, y esta es la ultima restante, se arroja un error indicando que solo queda una sola sala y no puede ser eliminada, por el contrario la sala si podrá eliminarse sin problema alguno.

En el manejo de los comandos enviados por el cliente hacia el servidor se tienen que cuidar las variables globales que se manejan con los hilos y para ello se utilizan los semáforos, y así evitar la no concurrencia entre los hilos.

En el modulo del cliente se maneja de forma similar, se emplean los hilos y se tiene como variable global el nombre del usuario que se está conectando con el servidor. Esta variable global será asignada siempre y cuando la secuencia de llamada que se realiza del programa cchat es correcta. Se verifican estos parámetros de llamada y si falta alguno de los parámetros que no es el archivo a procesar, pues manda un mensaje de error y no crea la conexión.

Si por el contrario la llamada fue exitosa, se procede a crear el socket y establecer la conexión con el servidor. Justo después se crea el hilo que se encargará de recibir la información que enviara el servidor a través del socket y estará a la espera de la información que corresponde como respuesta del servidor.

Si la conexión se hace efectiva, el programa verifica si existe un archivo que contendrá una serie de comandos a ejecutar, pertinentes a los comandos que son posible ejecutar dentro del contexto del CCHAT. Claro esta que, el archivo se verifica que exista, de lo contrario el programa se salta esta parte y entra directamente a la introducción de comandos por pantalla, indicando un fallo en la apertura del archivo. Si es válido, se debe procesar línea a línea, chequeando que los comandos suministrados en el archivo son correctos y además son posibles de efectuar entre el usuario y el servidor.

Ya a este punto se debe conocer como será la estructura de la trama que debe manejar el servidor y recibir el cliente y para ello las describiremos de la siguiente forma:

<Nombre\_cliente> <Comando\_a\_ejecutar> <Mensaje/Sala>

Cada campo está bastante explicito en lo que su función se refiere. Cuando se toma cada línea del archivo se procesa a conformar un string de forma que quede con ese formato, y se envía a través del socket para que el servidor reciba la indicación y así mismo como se envía la trama a través del socket, el servidor la recibe y la procesa de igual forma. Cuando el servidor termina de procesar el comando deseado envía una respuesta al cliente que es atajada por el hilo antes mencionado.

También es importante que el usuario se dé cuenta de cómo se van ejecutando sus comandos dentro del archivo que será procesado por lo que se coloca un *sleep(1)* luego de ejecutar cada comando leído del archivo para que el cliente se percate de que es lo que está ocurriendo en el servidor.

Es de notarse que si en el archivo viene el comando ***fue*** el usuario no se va a desconectar del servidor, sino que permanecerá en la introducción de comandos, y así permitirle al cliente ingresar algún comando o mensaje que desee luego de la lectura y procesamiento de su archivo.

Si no hay un archivo que deba ser procesado, el programa cae directamente en la estancia de introducción de comandos. Si dentro de esta estancia, se indica el comando ***fue****,* se desconectara el usuario del servidor.

Según las pruebas que se realizaron con la comunicación cliente/servidor, no se encontraron fallos entre la ejecución de los comandos. El único problema podría ser en el momento en el que un usuario ejecuta la opción "*CTRL + C*". En el servidor permanecerá aún conectado el usuario, aunque obviamente la información que se envié a través de su socket podrá generar un mensaje de error.