REDES DE COMUNICACIONES II

PRÁCTICA 1
SERVIDOR IRC

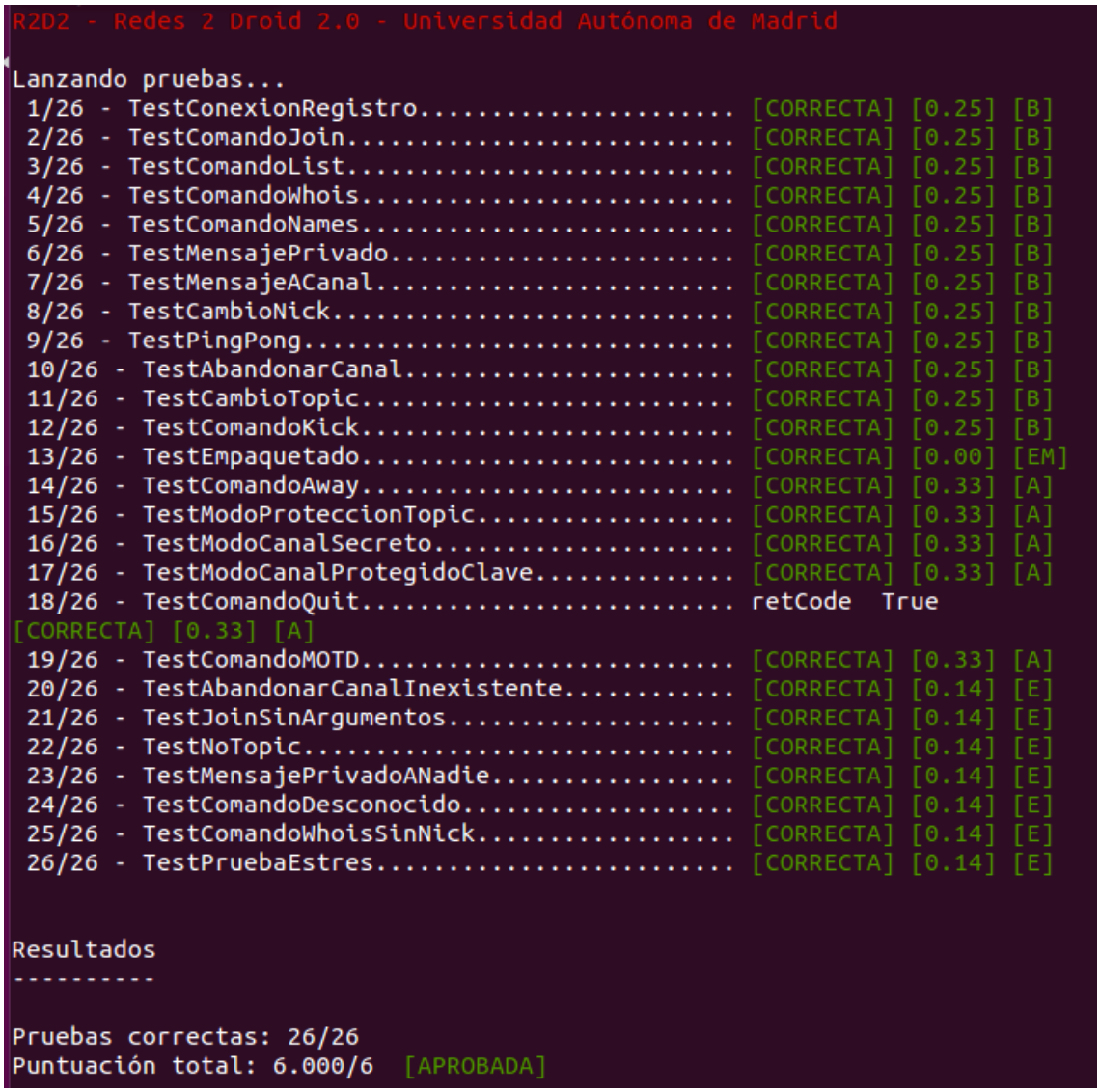
Jorge Parrilla Llamas
(jorge.parrilla@estudiante.uam.es)
Javier de Marco Tomás
(javier.marco@estudiante.uam.es)

1. INTRODUCCIÓN

La práctica consiste en la implementación de un servidor IRC programado bajo el lenguaje C. Para el desarrollo del servidor se ha seguido la implementación del RFC 2812 y 2813 según se solicitaba como requisito técnico del enunciado de la práctica.

Además, se ha seguido la implementación proporcionada por la librería eps-redes2.

Una vez desarrollada la práctica hemos realizado una serie de pruebas basándonos en los tests proporcionados por el comando de la librería eps-redes2 r2d2.



Además de probar el servidor con el r2d2 hemos accedido al servidor utilizando la tecnología que ofrece XChat para comprobar que el funcionamiento con dicha herramienta también es correcto.

2. DISEÑO

La práctica ha sido desarrollada bajo un diseño fácil e intuitivo, por un lado se utiliza *G-2313-06-P1_server.c* que contiene la base de las funciones de conexión del servidor: inicialización del socket, aceptación de conexiones, puesta en marcha de los hilos de los clientes, manejadores de los hilos principales, array de punteros a función para los comandos del servidor, etc.

Se utilizan también varios ficheros en los que se encuentran el resto de funciones implementadas para la realización de la práctica:

- G-2313-06-P1_thread_pool.c: Contiene las funciones que inican un pool de procesos que se van distribuyendo de forma equitativa a los usuarios según van llegando al servidor, en caso de quedarse sin hilos se aumenta de forma automática el número de procesos en el pool para seguir atendiendo las conexiones entrantes al servidor.
- G-2313-06-P1_function_handlers.c: Contiene todas las funciones handler de los comandos del servidor IRC, es decir, las funciones que controlan la ejecución de los distintos comandos: NICK, USER, QUIT, NAMES, LIST… A dichas funciones se accede desde el main (*G-2313-06-P1_server.c*) mediante el array de punteros que apuntan a dichas funciones. Dicha implementación la hemos realizado con un array de punteros a función porque nos parecía lo más limpio y óptimo posible, la otra opción era un switch con los cases para cada comando pero nos parecía algo totalmente desorganizado.
- G-2313-06-P1_common_functions.c: En este fichero se encuentran todas las funciones auxiliares que utilizan los handlers de los comandos, por ejemplo: actualizar el nick de un usuario, buscar un usuario por nick o socket, obtener el mensaje de away de un usuario, etc.

(Más información en la web de documentación detallada)
G-2313-06-P1/doc/html/index.html

3. FUNCIONALIDAD IRC

Como se puede ver en la página web de documentación detallada (Doxygen), se han implementado las siguientes funciones relativas a los comandos del servidor IRC:

- ***server_command_nick:*** Esta función implementa el handler del comando NICK. En ella se realizan varias comprobaciones previas: que el Nick no sea nulo, no esté mal escrito y que no esté en uso. Si alguna de esas tres condiciones se cumple, se devuelve un mensaje de error al usuario. En caso contrario, el Nick de guarda para ser usado en el resto de funciones.
- ***server_command_user:*** Esta función implementa el handler del comando USER. Lo primero que hace es comprobar que se haya recibido previamente el comando de NICK y se haya parseado el Nick de forma correcta, en caso contrario no se procederá al registro del usuario.
- ***server_command_join:*** Esta función implementa el handler del comando USER. Comprueba que el parseo del comando JOIN sea correcto, en caso contrario devuelve un mensaje de error generado por la función del TAD IRCMsg_ErrNeedMoreParams que se envía mediante el descriptor del socket del cliente recibido por el parámetro int desc. Comprueba también que el modo del canal no tenga contraseña, en caso afirmativo comprueba que la contraseña introducida sea correcta.

  Si no existe ningún problema, une al usuario al canal.
- ***server_command_quit:*** Esta función implementa el handler del comando USER. Parsea el comando para saber si se ha introducido correctamente, en caso afirmativo, elimina al usuario de la lista de usuarios y cierra la conexión con el descriptor del socket del usuario.
- ***server_command_ping***: Esta función implementa el handler del comando PING. Parsea el comando para comprobar que ha sido recibido correctamente, en caso afirmativo devuelve un PONG al usuario con la misma cadena que ha recibido mediante PING.
- ***server_command_list***: Esta función implementa el handler del comando LIST. Devuelve al usuario una lista de los canales activos en el servidor que sean públicos (comprueba que no se muestren canales con modo privado), así mismo también envía el número de usuarios en el canal y el topic del canal.
- ***server_command_privmsg***: Esta función implementa el handler del comando PRIVMSG. Envía un mensaje a todos los usuarios activos en un canal (realiza un for sobre la lista de usuario para entregarles el mensaje a todos) menos al usuario que envía dicho mensaje. Además, en caso de que el target sea un usuario en concreto le envía el mensaje de forma privada.
- ***server_command_part***: Esta función implementa el handler del comando PART. Elimina a un usuario de un canal en caso de que el comando se reciba correctamente y el canal exista en la lista de canales del TAD.
- ***server_command_names***: Esta función implementa el handler del comando NAMES. Devuelve al usuario la lista de usuarios activos en el canal, además,

comprueba los usuarios que son operadores del canal y les introduce un @ delante del nombre para que el usuario que solicita la lista de usuarios pueda saber fácilmente cuáles son los operadores del canal.

- *server_command_kick*: Esta función implementa el handler del comando KICK. Expulsa a un usuario de un canal, para ello previamente se comprueba que el usuario que envía el comando tenga el modo necesario para poder ejecutar dicho comando (es decir, sea operador del canal). En caso afirmativo el usuario es expulsado del canal.
- *server_command_mode*: Esta función implementa el handler del comando MODE. Se puede ejecutar de dos maneras, si no se especifica un modo a la hora de enviar el comando la función handler devolverá el modo actual del canal solicitado. En caso de que se especifique un canal, el handler actualizará el modo del canal al nuevo modo introducido por el usuario.
- *server_command_away*: Esta función implementa el handler del comando AWAY. Actualiza el modo away de un usuario, tiene la opción de añadir un mensaje o de desactivar el modo away.
- *server_command_whois:* Esta función implementa el handler del comando WHOIS. Devuelve la información sobre un usuario especificado en el comando.
- *server_command_topic*: Esta función implementa el handler del comando TOPIC. Actualiza el topic de un canal especificado por un usuario con permisos para ello o devuelde dicho topic en caso de que no se especifique ningún topic en concreto.
- *server_command_motd:* Esta función implementa el handler del comando MOTD. Devuelve el mensaje diario del servidor (MOTD) al usuario.

4. CONCLUSIONES TÉCNICAS

Realmente hemos aprendido mucho sobre el funcionamiento de los sockets en C. Además, una de las cosas que más clara nos ha quedado es saber manejar una librería externa y llegar a cubrir los requisitos de un RFC.

También hemos aprendido a desenvolvernos bien con los punteros, algo que hasta ahora habíamos tenido un tanto de miedo. Por eso nos hemos atrevido a desarrollar la práctica sobre la base del array de punteros a función, algo que nos ha servido para modular el servidor de forma totalmente elástica y no nos hemos visto limitados por un switch como el resto de implementaciones.

5. CONCLUSIONES PERSONALES

Ha sido una de las prácticas con mayor trabajo de lo que llevamos de carrera, en parte por la longitud de código y sobre todo por que en un principio estábamos muy perdidos ya que no sabíamos ni qué teníamos que desarrollar ni cómo. Poco a poco nos hemos ido centrando, tanto gracias a la ayuda de la profesora de prácticas como al empeño que hemos puesto por nuestra cuenta al leernos la documentación y el RFC.