Práctica 2
Creación de un cliente IRC

Eloy Anguiano

Oscar Delgado

Carlos García

Antonio Calleja

Fecha de entrega optativa: 13 de Marzo a las 9:00.
Fecha de entrega obligatoria: lunes 8 de mayo a las 9:00.

Cualquier archivo fuente de los proporcionados que sea modificado por el usuario, deberá cambiar de nombre y usar la nomenclatura propia del usuario.

1 Diseño de un cliente IRC

Uno de los objetivos de esta asignatura es la de la capacidad de interpretar e implementar un protocolo según los RFCs correspondientes y por tanto no se va a describir el protocolo (aunque los profesores de prácticas ayudarán a cada alumno en las dudas que tengan acerca de los protocolos). Sin embargo, para facilitar la implementación se aporta información a continuación.

Esta práctica consiste en el diseño de un cliente IRC. El protocolo IRC (*Internet Relay Chat*) es un protocolo de nivel de aplicación ideado para el intercambio de mensajes de texto. Los detalles de este protocolo pueden verse en los RFCs ("Request For Comments") indicados en la página web: `http://metis.ii.uam.es/redes2`

En estos RFCs se describe cómo se deben realizar las comunicaciones entre los distintos agentes que intervienen en el intercambio de mensajes. Sin embargo se puede encontrar información adicional de varias formas. Una de ellas consiste en utilizar un cliente de GNU/linux llamado `xchat` y que está disponible en los repositorios oficiales de casi todas las distribuciones. Se aconseja utilizar este cliente antes de leer los RFCs para tener una idea somera del funcionamiento del protocolo IRC. Así mismo, una de las opciones de este programa permite guardar las comunicaciones realizadas y este fichero es completamente legible por cualquier editor de texto además de verlo en directo en una ventana especial del propio programa.

Existen también multitud de fuentes de información que puede buscar el alumno. Es sin embargo importante diferenciar entre las comunicaciones IRC y los comandos de usuario. Los comandos de usuario son los que introduce el usuario en los clientes IRC y que deben ser traducidos al protocolo IRC aunque muchas veces haya una correspondencia directa entre el comando de usuario y el mensaje IRC correspondiente.

1.1 Funcionalidad aportada

Para simplificar la programación también se ha creado una librería para parsear los mensajes recibidos y parsear los comandos de usuario (la misma que se utilizó durante la implementación del servidor, documentada en `http://metis.ii.uam.es/redes2`). El motivo de no explicar en detalle algunas funciones es porque su uso tiene que ver con la interpretación de los RFCs correpondientes y esto se corresponde con las habilidades que se deben adquirir en estas prácticas. Las funciones que se explican es porque no se corresponden directamente con los RFCs sino que son funciones auxiliares. Así mismo se proporcionan también funciones que construyen los mensajes a enviar. Por el mismo motivo sólo se aporta en dicho documento un listado de dichas funciones con la explicación sólo de las funciones auxiliares.

2 Interfaz gráfico

Para poder implementar adecuadamente un cliente se proporciona un interfaz gráfico en C disponible en el repositorio de la asignatura. Del interfaz se adjunta un fuente y una librería. Parte de los comandos IRC se pueden realizar a través del interfaz y otros deberán implementarse como comandos en el texto introducido. Por ese motivo será necesario parsear adecuadamente este texto para interpretar los comandos. Se recomienda el uso tradicional de comandos descrito en `http://personales.mundivia.es/papi/comirc.html`, que son los implementados en el parser de comandos de usuario. Así mismo se pueden usar otros comandos de usuario pero que deberán ser parseados por funciones propias y documentados.

Para compilar la práctica es necesario disponer del paquete de desarrollo de gnome (GTK)[1]. Así mismo el interfaz gráfico es modificable si se desea.

2.1 Funcionalidad aportada

Se pueden encontrar todas las funciones proporcionadas en la web de la librería de redes2: `http://metis.ii.uam.es/redes2`. Éstas pueden ser enlazadas con los distintos programas que se realicen con los parámetros adecuados en la línea de compilación, descritos en dicha página. Además de las que ya se han utilizado en la práctica I, se dispone de funciones de callbacks e interacción con el interface (`IRCInterface_*`), así como de manejo de sonido (`IRCSound_*`).

Es importante tener en cuenta que si se utilizan múltiples hilos o procesos, el acceso al interfaz gráfico deberá ser ordenado usando los semáforos del propio interfaz gráfico. En caso de estar en el hilo principal del interfaz como son las funciones "callback" descritas posteriormente, no es necesario usar semáforos de los hilos puesto que ya están en el hilo principal. Se proporcionan funciones para interactuar con el interfaz desde hilos (son las terminadas en `Thread`).

Banderas de ejecución

Por último, el cliente deberá soportar una serie de banderas de ejecución, cuya gestión ya se entrega simplificada en la librería. Esencialmente estas banderas sirven para modificar el comportamiento del cliente desde su ejecución en la línea de comandos:

- `-ssl`: indica al cliente que éste debe utilizar SSL en su conexión con el servidor. Si se utiliza esta bandera, debe incluirse también `-port`.
- `-ssldata ["datos"]`: indica al cliente que éste debe utilizar SSL en su conexión y, simultáneamente, enviar la cadena *datos* al servidor. En este caso, ni siquiera debe lanzarse el interfaz gráfico: simplemente se crea el socket SSL, se envían los datos y se acaba la ejecución del cliente.
- `-data ["datos"]`: indica al cliente que debe enviar la cadena *datos* al servidor. En este caso, ni siquiera debe lanzarse el interfaz gráfico: simplemente se crea el socket, se envían los datos y se acaba la ejecución del cliente
- `-port`: indica el puerto de conexión.

[1] `libgtk-3-dev`, que se instala automáticamente con el paquete debian `eps-redes2`

2.2 Implementación de funcionalidad

Sólo es necesario un fuente en C para la realización de estas prácticas. Es el fuente en el que están definidas y vacías las funciones llamadas por el interfaz de usuario (callbacks) y el main del interfaz de usuario. Con la instalación de los paquetes (`http://metis.ii.uam.es/redes2`) este fuente se haya disponible en en

`/usr/local/src/redes2/xchat2.c`

Deberán implementarse las funciones presentadas en `xchat2.c`. Si no se implementan es necesario dejar la implementación con la que se aportan originalmente. Si se desea implementar una mayor funcionalidad puede añadirse en este u otros ficheros. Es necesario cambiar el nombre de este fuente al propio del grupo y por tanto es divisible en múltiples archivos fuente.

En el texto devuelto introducido por el usuario en el interfaz deberá procesarse para determinar si se ha introducido un comando de usuario y se ha realizado determinar cuál. El parseo de los comandos de usuario se pueden realizar con la familia `IRCUserParse`. También se dispone de un conjunto de funciones útiles para interaccionar con el interfaz.

Así mismo será necesario un hilo que esté constantemente recibiendo y procesando los mensajes del servidor y, al menos, una rutina de atención a una señal (ALARM) para atender al protocolo PING-PONG.

3 Envío de ficheros

Al protocolo de IRC utilizado por el cliente de la práctica 2 se van a añadir los comandos `/fsend`, `/faccept` y `/fcancel`. Desde el punto de vista de los usuarios el primer comando sólo lleva dos parámetros: un nick de usuario y una ruta a un fichero en el disco. Los demás pueden llevar algún identificador de la transferencia con la que operarán.

Estos comandos se utilizarán para que un cliente de IRC pueda realizar el envío de un fichero a otro cliente IRC mediante un canal de comunicación seguro. Desde el punto de vista de la comunicación el primer comando enviará internamente un PRIVMSG en el que se indicará que es una petición de envío de fichero, y deberá incluir el nick del usuario, el nombre del fichero y su tamaño en bytes. El segundo comando enviara otro PRIVMSG con información sobre el nick del usuario, su IP y el puerto donde espera recibir el fichero. En el caso de `/fcancel` se cerrará la conexión y se enviará un PRIVMSG indicando que se ha cerrado la conexión. En ningún caso deberán presentarse estos mensajes al usuario.

Los puertos para las comunicaciones deberán ser elegidos por el sistema operativo y no por el programa.

4 Comunicación de audio

Se propone una modificación del protocolo IRC que consiste en la comunicación privada usando sonido. Para ello será necesario utilizar el protocolo RTP: RFC 3550[2]. Para que el alumno no tenga que realizar las funciones de adquisición y reproducción de sonido se aporta un objeto con esta funcionalidad. Como

[2] `http://tools.ietf.org/html/rfc3550`

la comunicación se realizará vía UDP será necesario encapsular nuevas funciones encapsuladas para la comunicación vía UDP.

Para elegir un puerto aleatorio que esté libre se deberá utilizar en el `bind` como número de puerto el 0 y después llamar a la función `getsockname` para obtener la información necesaria acerca de nuestra IP y el puerto que ha sido abierto.

Se dispone de una librería de audio que ya graba y reproduce en formatos que pueden ser enviados usando el protocolo RTP. En el interfaz gráfico se puede abrir una ventana para realizar esta comunicación y debe realizarse el diseño completo de esta funcionalidad.

En el fuente `sound.c` y en su correspondiente `sound.h` se crean y prototipan respectivamente las siguientes funciones:

int open_record(char *identificacion): Inicia la captura. Devuelve un error estándar de PulseAudio o 0 si no ha habido error. El parámetro de identificación es una cadena de caracteres cualquiera. Se suele utilizar el nombre del programa que usa esta función.

int open_play(char *identificacion): Inicia la reproducción. Devuelve un error estándar de PulseAudio o 0 si no ha habido error. El parámetro de identificación es una cadena de caracteres cualquiera. Se suele utilizar el nombre del programa que usa esta función.

void close_record(void): Cierra la captura.

void close_play(void): Cierra la reproducción.

int record_sound(char * buf, int size): Almacena una parte de la grabación en el buffer. Se aconseja el uso de buffers con tamaños relacionados con los paquetes RTP. Devuelve un error estándar de PulseAudio o 0 si no ha habido error. Los parámetros son el buffer en el que se almacenan los datos y el tamaño en bytes a leer. El buffer debe tener espacio reservado suficiente.

int play_sound(char * buf, int size): Reproduce la parte de la grabación almacenada en el buffer. Se aconseja el uso de buffers con tamaños relacionados con los paquetes RTP. Devuelve un error estándar de PulseAudio o 0 si no ha habido error. Los parámetros son el buffer en el que se almacenan los datos y el tamaño en bytes a leer.

int sample_format(enum pa_sample_format format, int nChanels): Tipo de muestreo de la señal. Devuelve el valor del "payload" de RTP correspondiente. El parámetro de entrada es un valor de los presentados en la tabla inferior. Si no se usa, el valor por defecto es PA_SAMPLE_S16BE. Así mismo, el segundo valor es el número de canales que se quiere muestrear. Este parámetro puede tomar el valor 1 en todos los casos y también 2 en el caso de PA_SAMPLE_S16BE. Las funciones internas de PulseAudio permiten otro tipo de muestreos pero ninguno compatible con un payload RTP. Devuelve -1 en caso de formato incorrecto.

Valor	Significado
PA_SAMPLE_ALAW	8 Bit a-Law 8 kHz
PA_SAMPLE_ULAW	8 Bit mu-Law 8 kHz
PA_SAMPLE_S16BE	16 Bit PCM con signo, big endian, 44.1 kHz

Si se desea modificar la funcionalidad de sonido es necesario disponer del paquete de PulseAudio[3] y leer esto. En realidad estas son funciones síncronas. Si se desean funciones asíncronas es necesario utilizar otras funciones de PulseAudio. Es más, a través de un valor para `pkg-config` distinto se puede integrar con

[3] libpulse-dev

el *main loop* de GTK. Aunque no es necesaria esta implementación, se tendrá en cuenta positivamente si se realiza.

Como en el caso de los ficheros en la primera práctica los puertos deben ser elegidos por el sistema. En este caso el envío de sonido debe hacerse en modo seguro también.

5 Consideraciones de diseño

Es importante realizar el diseño de tal forma que sea fácil añadir funcionalidad. Esto es debido a que en la siguiente práctica se va a añadir a la comunicación la posibilidad de utilizar comunicaciones seguras con SSL.

6 Cronograma orientativo

- El la primeras dos semanas deberán quedar implementadas la estructura de hilos y señales así como las funciones de conexión, desconexión, envío y recepción TCP, así como UDP (necesario para la transmisión RTP de audio).
- Tras otras dos semanas debería estar implementado por completo el protocolo de comunicaciones del cliente.
- Las últimas semanas se utilizarán para implementación de la comunicación de sonido y de ficheros.