Programa de chat 1 a 1

Objetivos

El objetivo de esta práctica es familiarizarse con el uso de sockets de forma *simultánea* para recibir y enviar utilizando varios procesos.

Cliente/servidor de chat

Realice un programa para enviar texto a través de una conexión TCP. El programa podrá actuar como cliente o como servidor según se indique, y permitirá a dos usuarios comunicarse entre sí, sólo con el uso del programa `chat`.

La especificación será la siguiente:

FORMATO

```
chat -l <puerto>
chat -c <destino> <puerto>
```

DESCRIPCIÓN

Si el comando se lanza con la opción **-l**, el programa actuará como servidor TCP aceptando una conexión por el puerto indicado (en este caso no hace falta especificar el host). Si el comando se lanza con la opción **-c**, el programa actuará como cliente TCP realizando una conexión al host y puerto indicados.

<puerto> puerto de destino o de escucha.

<destino> nombre o dirección IP del destino de la conexión.

En cualquier caso, una vez establecida la conexión, el programa permitirá al usuario escribir en la entrada estándar y cada vez que escriba una línea la enviará al otro extremo. **Al mismo tiempo**, cuando lleguen datos del otro extremo, el programa los presentará al usuario por la salida estándar. Cuando cualquiera de los dos extremos cierre la conexión, el otro extremo deberá darse cuenta y cerrarse apropiadamente.

EJEMPLO

El uso será algo así:(marcamos con **$** los comandos del shell y con **>** lo que escribe el usuario del programa chat)

En **10.1.1.50**	En **10.1.1.16**
$ chat –l 9999	
	$ chat –c 10.1.1.50 9999
Conexión aceptada desde 10.1.1.16	
	Conexión establecida
> Hola	
	Hola
	> Me oyes?
Me Oyes?	
^D	
$	El otro extremo ha cerrado la conexión
	$

Utilice la función select() para leer simultáneamente lo que escribe el usuario y lo que llega por la conexión. En el primer caso, deberá de hacer uso del descriptor de la entrada estándar STDIN_FILENO, para ello consulte el manual (`man stdin`).

Pruebe su programa conectándolo con clientes y servidores usando `nc`, y lanzando su programa `chat` en varios ordenadores. Pruebe, también, con los programas creados por sus compañeros para comprobar la interoperatividad.

Cambiando el protocolo

Hasta ahora el programa `chat` envía directamente lo que escribe el usuario. Pero podemos separar y utilizar diferente interfaz con el usuario de lo que enviamos por la conexión.

Interfaz de usuario

Consideremos más opciones en el interfaz con el usuario. A partir de ahora, en la siguiente versión del programa, el usuario puede escribir comandos que empezarán con el carácter "\". Si el usuario escribe algo que no empieza por \ quiere decir que es una frase que debe enviarse al otro extremo de la conexión. Si el usuario escribe algo que empieza por \ lo que sigue es un comando que tendrá un significado que debe interpretar el programa `chat`.

La siguiente versión del programa `chat` que realice deben interpretar los siguientes comandos y debe ignorar los comandos que no entienda. Es decir, aunque lo que haya escrito el usuario sea un comando no reconocido, no hay que enviarlo por la conexión.

```
Comandos del usuario

\quit   dice al programa que debe cerrar la conexión, lo mismo que hacer ^D

\info   pide al programa que muestre la información del otro extremo, es decir
        con quién estoy conectado y en qué puerto.

\log <nombredefichero>
        el programa chat debe guardar la conversación. Para ello debe abrir un
        fichero con el nombre indicado, y a partir de ese momento guardar en
        dicho fichero, todo lo que escribe el usuario y todo lo que llega por la
        conexión.

\nolog  para de guardar la conversación en fichero.
```

Realice una nueva versión del programa `chat` que interprete los comandos descritos. Sólo es necesario entregar una versión del programa `chat` que incluya todas las funciones que haya programado. Se evaluará la práctica según cuáles de estos comandos haya conseguido programar para `chat`.

Entrega de la práctica

Para entregar la práctica debe crear un directorio `prac4` dentro de su directorio home (`/opt3/rc/rc<numerodegrupo>/prac4`)

En el directorio `prac4` debe dejar las fuentes necesarias para compilar el programa `chat` con todas las funciones que haya programado. Para la corrección se borrarán los programas ejecutables y se recompilarán de forma automática por lo que en el directorio debe existir un fichero `Makefile` que permita construir todos los programas, lanzando el comando `make` sin argumentos en el directorio `prac4`

Los programas deben cumplir *exactamente* las especificaciones y se probarán en la medida de lo posible mediante "scripts de corrección" que esperarán que los programas se comporten tal y como se pide. Asegúrese, antes de entregar los programas, que cumplen las especificaciones.

Al finalizar el curso se cerrarán las cuentas y se considerará entregada la práctica que haya en el directorio `prac4`.