Universidad Carlos III

Sistemas Distribudos

Curso 2023-24

Prctica 2

Sockets

Ingeniera Informtica, Tercer curso

Adrin Fernndez Galn (NIA: 100472182, e-mail: 100472182@alumnos.uc3m.es)
Csar Lpez Mantecn (NIA: 100472092, e-mail: 100472092@alumnos.uc3m.es)

Prof . Flix Garca Caballeira
Grupo: 81

ndice

1 Introduccin

El desarrollo de este proyecto consiste en implementar una aplicacin cliente-servidor, donde los diferentes clientes podrn guardar informacin en tuplas a travs del servidor, de forma transparente. La comunicacin entre el cliente y el servidor se dar a travs de una interaz de Sockets. Los mensajes transmitidos deben ser independientes del lenguaje de programacin.

Cabe destacar que el tratamiento de peticiones en el lado del servidor ser de forma concurrente, mediante el uso de procesos ligeros.

2 Diseo

La aplicacin constar de dos partes diferenciadas: los clientes y el servidor.

2.1 Cliente y Biblioteca dinmica

Siguiendo la misma aproximacin que en el ejercicio anterior, el cliente no ha necesitado de cambios ya que la nica parte afectada por los requisitos de este ejercicio es la comunicacin. La comunicacin entre el cliente y el servidor est completamente encapsulada en la biblioteca dinmica *libclaves.so*. Por esto, el nico archivo modificado ser *src/claves.c*, que sirve como cdigo fuente de la bibliteca dinmica.

2.2 Servidor

En el servidor se encuentran las funcionalidades que se encargan de la comunicacin con el cliente, en el fichero *src/servidor.c*, y la implementacin de los servicios para las tuplas, en el fichero *src/imp_clave.c*. De nuevo, el nico archivo que necesit de modificaciones es *servidor.c* para adaptar el cdigo a la comunicacin mediante Sockets.

2.2.1 Estructura del Servidor

El servidor debe realizar dos acciones principales: aceptar la comunicacin con un cliente y lanzar un hilo para gestionar dicha peticin. Para la primera accin, el servidor debe estar alojado en un puerto que recibir como argumento de programa. En dicho puerto esperar hasta recibir una peticin de conexin con algn cliente. Para la segunda accin, usamos la interfaz ofrecida por *pthread.h* para lanzar un hilo independientemente que tratar la peticin y comunicar al cliente la respuesta.

2.2.2 Implementacin en el servidor

La implementacin de los servicios requeridos est contenida en el fichero *src/imp_claves.c*. Dado que los servicios coinciden exctamente con los del ejercicio anterior, este fichero no ha sufrido ningn cambio. Se compilar junto con el servidor.

Siguiendo la misma idea que en la prctica anterior, guardaremos las tuplas en ficheros dado su caracter permanente entre sesiones y las facilidades que presentan para la gestin de la concurrencia.

2.2.3 Concurrencia del Servidor

Seguiremos un esquema de hilos bajo demanda. Es decir, crearemos un hilo por cada peticin recibida para su gestin.

El servidor contar con un hilo principal (ms adelante referido como *main*) encargado de escuchar y recibir peticiones y crear nuevos hilos *detached* para la gestin de dichas peticiones. Cada uno de los hilos creados por *main* ejecutar la funcin `tratar_peticion()`. En esta funcin, copiar el socket del cliente (protegiendo la copia por un *mutex* para resolver la condicin de carrera) y se llamar al servicio adecuado segn el cdigo de operacin de la peticin. Despus, se enviar la respuesta al cliente y se cerrar la conexin.

2.3 Comunicacin

La comunicacin, tal y como especificamos anteriormente, se realiza a travs de Sockets. Este mecanismo permite comunicar dos procesos cualesquiera en la misma red, independientemente de la arquitectura de la mquina que los ejecuta o el lenguaje de programacin en el que est escrita la aplicacn. Es por esto que es importante que la informacin transmitida por las partes cliente y servidor de nuestro sistema sea tambin independiente de estos menesteres.

2.3.1 Uso de cadenas de texto

Para hacer los mensajes independientes del lenguaje de programacin o arquitectura de la mquina, transmitiremos la informacin como cadenas de texto. Adems, los mensajes sern transmitidos byte a byte para evitar la problemtica presentada por la diferencia entre arquitecturas *little-endian* y *big-endian*.

2.3.2 Funciones auxiliares para la comunicacin

Para facilitar la implementacin de la comunicacin mediante Sockets se han implementado una serie de funciones auxiliares recogidas en el fichero *src/common.c*. Estas implementan acciones repetidas tanto en el lado cliente y servidor, as como gestin de errores y envo y recepcin de mensajes adaptados a las necesidades del sistema.

- `serverSocket()`: crea y devuelve un *socket* para el servidor en un nmero de puerto dado. Adems, se ejecuta la llamada `liste()` para permitir al servidor aceptar conexiones ms adelante.
- `serverAccept()`: esta funcin permite al servidor aceptar una conexin con un cliente.
- `clientSocket()`: crea y devuelve un socket para un cliente en el sistema. Adems, este *socket* estar conectado al servidor.
- `sendMessage()`: enva un mensaje contenido en un *buffer* a travs de un *socket*.
- `recvMessage()`: recibe un mensaje y lo almacena en n *buffer* dado.
- `writeline()`: se apoya de la funcin `sendMessage()` para enviar un mensaje hasta el final.
- `readLine()`: se apoya de `recvMessage()` para recibir un menesaje hasta el final de la cadena de texto.

3 Compilacin

En esta seccin se describe la forma de compilar el proyecto para generar dos ejecutables cliente y servidor. Todos los comandos que se exponen a continuacin quedan recogidos en un archivo *Makefile* encargado de generar ambos ejecutables.

3.1 Biblioteca Dinmica

Tal y cmo hemos descrito anteriormente, el cdigo fuente de la biblioteca dinmica est contenido en el archivo *claves.c*. Para compilarlo como una biblioteca dinmica se han empleado los siguientes comandos.

```
# c digo objeto
gcc -c -fPIC claves.c
# biblioteca din mica
gcc -shared -o libclaves.so claves.o
```

Con estos comandos generamos un archivo de cdigo para enlazar con otro ejecutable. Este cdigo ser independiente de la posicin.

3.2 Cliente

El cdigo fuente del cliente est contenido en el archivo *cliente.c*. Este se compila enlazndolo con la biblioteca dinmica a travs de los siguientes comandos:

```
# generacion del objeto
gcc -c cliente.c

# generar ejecutable con biblioteca din mica
gcc -L. -Wl,-rpath=. -lclaves -o cliente cliente.o
```

3.3 Servidor

4 Descripcin de pruebas

Para validar elcorrecto funcionamiento del sistema, hemos lanzado varios clientes en mquinas con sistemas operativos distintos utilizando la herramienta *docker*.

[...]

5 Conclusiones

Este ejercicio ha servido como introduccin al desarrollo de aplicaciones distribudas mediante el uso de Sockets. Destacamos la importancia de disear un sistema modular, dado a que gracias a esta aproximacin en el ejercicio 1, hemos necesitado muy pocas modificaciones a la hora de realizar esta segunda prctica.

El uso de Sockets, junto con una correcta codificacin de la informacin por parte del programador, es una herramienta muy potente para el desarrollo de sistemas distribudos. Esto es porque ya no estamos limitados a una nica arquitectura o sistema operativo, sino que nos permiten ejecutar y comunicar procesos en distintos ordenadores con especificaciones variadas.

Adems, el desarrollo de esta prctica no solo nos ha permitido consolidar los conocimientos adquiridos en la asignatura de Sistemas Distribudos; sino que tambin nos ha incentivado a rescatar conceptos presentados en la asignatura de Redes de Ordenadores. Es importante, a la par que satisfactorio, ver como los conocimientos adquiridos en asignaturas previas nos sirven de base para nuestro desarrollo como informticos segn avanzamos en nuestro aprendizaje.