Universidad Carlos III

Sistemas Distribuídos

Curso 2023-24

Práctica 2

Sockets

Ingeniería Informática, Tercer curso

Adrián Fernández Galán (NIA: 100472182, e-mail: 100472182@alumnos.uc3m.es)
César López Mantecón (NIA: 100472092, e-mail: 100472092@alumnos.uc3m.es)

Prof . Félix García Caballeira
Grupo: 81

Índice

1 Introducción

El desarrollo de este proyecto consiste en implementar una aplicación cliente-servidor, donde los diferentes clientes podrán guardar información en tuplas a través del servidor, de forma transparente. La comunicación entre el cliente y el servidor se dará a través de una interaz de Sockets. Los mensajes transmitidos deben ser independientes del lenguaje de programación.

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

2 Diseño

La aplicación constará de dos partes diferenciadas: los clientes y el servidor.

2.1 Cliente y Biblioteca dinámica

Siguiendo la misma aproximación que en el ejercicio anterior, el cliente no ha necesitado de cambios ya que la única parte afectada por los requisitos de este ejercicio es la comunicación. La comunicación entre el cliente y el servidor está completamente encapsulada en la biblioteca dinámica *libclaves.so*. Por esto, el único archivo modificado será *src/claves.c*, que sirve como código fuente de la bibliteca dinámica.

Para compilar el archivo *claves.c* se ha empleado el siguiente comando en un archivo *Makefile*.

```
gcc -c -fPIC -shared -o lib.so claves.c
```

2.2 Servidor

En el servidor se encuentran las funcionalidades que se encargan de la comunicación con el cliente, en el fichero *src/servidor.c*, y la implementación de los servicios para las tuplas, en el fichero *src/imp_clave.c*. De nuevo, el único archivo que necesitó de modificaciones es *servidor.c* para adaptar el código a la comunicación mediante Sockets.

2.2.1 Estructura del Servidor

El servidor debe realizar dos acciones principales: aceptar la comunicación con un cliente y lanzar un hilo para gestionar dicha petición. Para la primera acción, el servidor debe estar alojado en un puerto que recibirá como argumento de programa. En dicho puerto esperará hasta recibir una petición de conexión con algún cliente. Para la segunda acción, usamos la interfaz ofrecida por *pthread.h* para lanzar un hilo indpendiente que tratará la petición y comunicará al cliente la respuesta.

2.2.2 Implementación en el servidor

La implementación de los servicios requeridos está contenida en el fichero *src/imp_claves.c*. Dado que los servicios coinciden exáctamente con los del ejercicio anterior, este fichero no ha sufrido ningún cambio. Se compilará junto con el servidor.

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

2.2.3 Concurrencia del Servidor

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

El servidor contará con un hilo principal (más adelante referido como *main*) encargado de escuchar y recibir peticiones y crear nuevos hilos *detached* para la gestión de dichas peticiones. Cada uno de los hilos creados por *main* ejecutará la función `tratar_peticion()`. En esta función, copiará el socket del cliente (protegiendo la copia por un *mutex* para resolver la condición de carrera) y se llamará al servicio adecuado según el código de operación de la petición. Después, se enviará la respuesta al cliente y se cerrará la conexión.

2.3 Comunicación

La comunicación, tal y como especificamos anteriormente, se realiza a través de Sockets. Este mecanismo permite comunicar dos procesos cualesquiera en la misma red, indpendientemente de la arquitectura de la máquina que los ejecuta o el lenguaje de programación en el que esté escrita la aplicaicón. Es por esto que es importante que la información transmitida por las partes cliente y servidor de nuestro sistema sea también independiente de estos menesteres.

2.3.1 Uso de cadenas de texto

Para hacer los mensajes indpendientes del lenguaje de programación o arquitectura de la máquina, transmitiremos los números enteros como cadenas de texto. Además, los mensajes serán transmitidos byte a byte para evitar la problemática presentada por la diferencia entre arquitecturas *little-endian* y *big-endian*.

2.3.2 Uso de doubles

Para transmitir los vectores de números en coma flotante de doble precisión, podemos transmitir la información tal y cómo está almacenada en memoria. Esto es porque su representación y uso queda recodigo en el estándar **IEEE 754** y, por tanto, son independientes de la arquitectura o lenguaje de programación.

2.3.3 Funciones auxiliares para la comunicación

Para facilitar la implementación de la comunicación 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, así como gestión de errores y envío y recepción de mensajes adaptados a las necesidades del sistema.

[[poner aquí cada función y lo que hace]]

3 Descripción de pruebas

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

[...]

4 Conclusiones

Este ejercicio ha servido como introducción al desarrollo de aplicaciones distribuídas mediante el uso de Sockets. Destacamos la importancia de diseñar un sistema modular, dado a que gracias a esta aproximación en el ejercicio 1, hemos necesitado muy pocas modificaciones a la hora de realizar esta segunda práctica.

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

Además, el desarrollo de esta práctica no solo nos ha permitido consolidar los conocimientos adquiridos en la asignatura de Sistemas Distribuídos; sino que también nos 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 informáticos según avanzamos en nuestro aprendizaje.