INFORME

S.O. - SISTEMAS OPERATIVOS

Trabajo Práctico 4

Programación de Sockets

- Grupo 8: Matías Heimann (57503), Johnathan Katan (56653), Lóránt Mikolás (57347) y Gabriel Silvatici(53122).

- Vencimiento de la entrega: 24/06/2018 (23:59).
- Profesor: Horacio Merovich.
- Adjuntos: Ariel Godio, Rodrigo Rearden.

Introducción

El objetivo de este trabajo consistió en implementar un sistema cliente-servidor concurrentes. Los diferentes pedidos de clientes fueron atendidos mediante la creación de threads. Por último se buscó hacer uso de sockets para la comunicación entre ambos. En particular se usaron los *unix domain sockets*.

Implementación

En la presente sección se detallarán los algoritmos y estructuras de datos utilizadas para el correcto funcionamiento del trabajo. Además se tiene que tener en cuenta que la temática elegida fue la de reservas de asientos de avión.

Sockets para la comunicación Cliente/Servidor

En cuanto a la comunicación entre los clientes y el servidor se usaron los *unix domain sockets*. Esto fue indicado con AF_UNIX en la creación del socket. Los *unix domain sockets* utilizan el file system como su *address namespace*. Se optó por este tipo de comunicación debido a que la comunicación se realizaba en el mismo host y suele ser más liviana y rápida que INET.

También vale la pena aclarar que se utilizó un SOCK_STREAM que utiliza el protocolo TCP para la transmisión de datos porque se buscaba que no haya pérdida de información, sea “confiable y se encuentre ordenada.

Los pasos a tomar para crear el socket y darle un nombre fueron basados en el algoritmo que se presenta a modo de ejemplo al ejecutar *man bind* en la terminal.

Threading

Uno de los desafíos principales del trabajo era atender los pedidos de varios clientes de manera concurrente. Con este fin se decidió utilizar la creación de threads (por sobre la creación de procesos).

Es más “barato” el uso de threads que de múltiples procesos mediante fork. Cada vez que se realiza el accept(...) de una conexión se crea un thread (POSIX thread, o pthread) que atiende la solicitud. Para ello se tuvo que agregar -lpthread a la compilación.

Base de datos

Para realizar la base de datos se utilizó SQLite 3 y se realizó una librería de funciones en c para modificarla fácilmente. Las tablas guardadas en la base de datos son (las claves se encuentran subrayadas):

- flight(flightID)
- user(userID)

- seat(seatNumber, seatState, flightNumber)
- reservation(user, seat, flight)
  Se aclara que seatState es de tipo text mientras que los campos restantes son ints.

Estándares utilizados para el estilo del código

En cuanto al estilo de código se usaron las mismas herramientas del trabajo práctico 3. Por lo tanto citamos lo que explicamos en esa entrega:

"Para formatear el código y que todos los archivos *.c* y *.h* respetaran el mismo estándar se utilizó la herramienta open source *clang-format*. Se usó un archivo llamado *.clang-format* que especifica los estándares del código, este archivo se puede encontrar en el directorio principal del proyecto. Se puede observar que el estándar utilizado es *LLVM,* con algunas modificaciones en cuanto a indentación con *tabs* que se puede ver en el archivo *.clang-format*.

Por último, como *clang-format* permite solo formatear un archivo en particular, se procedió a escribir un script para poder ejecutar esta herramienta sobre cada archivo del proyecto. Este script fue escrito en *python*, y se puede encontrar en el archivo *formatter.py* en el directorio principal del proyecto."

Instrucciones para ejecución

Para poder ejecutar el sistema se debe seguir los siguientes pasos:

En la carpeta TP-Sockets, donde está ubicado el archivo Makefile, ejecutar el siguiente comando en la shell: "make fromzero".

Luego, para ejecutar el servidor para que pueda recibir pedidos, se debe ejecutar ./server en la misma carpeta.

Por último, basta con ejecutar ./client en la misma carpeta, desde cualquier terminal, para poder conectarse con el servidor y usar el sistema de reservas, cuyas instrucciones de uso se presentan al principio de la ejecución del programa.

Conclusión

En conclusión este trabajo nos ayudó a ejercitar el uso de sockets y la comunicación entre procesos (IPC). Se pudieron investigar los diferentes tipos de sockets, compararlos y comprender sus ventajas y desventajas. También nos permitió hacer uso de los POSIX threads, que previamente no habíamos utilizado en c.

Por último el uso de una librería como sqlite 3 para la creación de una base de datos nos facilitó realizar una aplicación que tenga sentido práctico e integrar conocimientos de otras materias como Base de Datos I.

Problemas

Ante cualquier dificultad, nos pueden contactar en el correo institucional del ITBA a cualquiera de los integrantes del grupo.

Bibliografía

A continuación se encuentra la bibliografía utilizada para realizar el trabajo práctico. Se debe tener en cuenta que también fueron usados elementos teóricos y prácticos enseñados en clase y proveídos por la cátedra como parte del material que se encuentra en el campus.

- Código en el comando man bind para un sistema cliente servidor con sockets.
- Referencia para la creación de threads: https://www.geeksforgeeks.org/multithreading-c-2/
- Si bien se utilizaron unix domain sockets en una primera instancia se investigaron ejemplos de tipo INET: https://www.thegeekstuff.com/2011/12/c-socket-programming/?utm_source=feedburner
- http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html
- https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm