Trabajo practico 1 - Programación distribuida y tiempo real

Franco Dalla Gasperina

agosto 2025

1 Ejercicio 1

1) Teniendo en cuenta la comunicación con sockets (puede usar tanto los ejemplos provistos como también otras fuentes de información, que se sugiere referenciar de manera explıcita):

- a.- Identifique similitudes y diferencias entre los sockets en C y en Java.
- b.- ¿Por qué puede decirse que los ejemplos no son representativos del modelo c/s? Nota: corroborar con la clase donde se explica el modelo C/S.
- c.- ¿Qué cambio/s deberıan hacerse para que "cliente" provisto funcione como "servidor" y el "servidor" provisto funcione como "cliente"? Nota: corroborar con la clase donde se explica el modelo C/S.

Respuesta-1

1.1 a

Las similitudes entre los sockets de C y Java son:

- En ambos debe definirse si el socket utilizado sera para streams o datagramas.

Las diferencias entre ambos son:

- Java es un lenguaje de alto nivel. Por lo que hay mucha más abstracción en la realización de conexiones con sockets en este lenguaje en comparación a C. Por esta razón en C se puede, por ejemplo, no realizar algunas.
- en Java debe definirse el tipo de buffer (entrada o salida) a diferencia de C en el que no es necesario.

1.2 b

Se discutió mucho al respecto sobre esto en las clases. La constante fue que no se cumplía el modelo c/s a causa de que el código que funciona como servidor es muy simple y no tiene en cuenta ciertas tareas que debería cumplir un servidor. Tales como:

- Concurrencia. El servidor de ejemplo no puede manejar varios pedidos al mismo tiempo.
- Escala. El servidor de ejemplo solo responde una vez y finaliza la ejecución.

1.3 c

En el lenguaje C. cliente como servidor: Se necesitaría agregar bind, para asociar el socket creado a un puerto de la máquina. Además, en el cliente hay que poner listen, para que reciba una cantidad de conexiones entrantes y luego accept para cuando estas lleguen.

Servidor como cliente: Esta es más sencilla. Ya que se deberían quitar algunas cosas como el bind a un puerto y la recepción como el listen y accept y agregar solo elementos para conocer al host y ya funcionaría como cliente.

2 Ejercicio 2

2) Desarrolle experimentos para quede claro que:

- a.- Aunque un proceso "servidor" programado en C haya obtenido un socket y hecho un bind() no va a haber ningún otro proceso que pueda hacer una conexión con él a menos que se haya hecho el listen() ¿Por qué este mismo experimento no podrıa hacerse programando en Java?
- b.- Un proceso "cliente" puede tener una conexión con el proceso "servidor" aunque el "servidor" no haya ejecutado la operación accept().

Respuesta-2

2.1 a

Esto es cierto. Ya que con la instrucción listen se determina la cantidad de conexiones esperadas en el socket. Si esta instrucción no se usara no podría llevarse a cabo la conexión. En el caso de Java, la clase utilizada para las conexiones abstrae la tarea de la sentencia accept de C. Por lo que no podemos omitir este paso.

2.2 b