UNLu - Departamento de Tecnología

Trabajo Practico N°1
Conceptos básicos para la construcción de Sistemas Distribuidos

1) Escriba un servidor que, usando sockets, reciba un mensaje de texto y lo repita a su cliente. Programe también el cliente para verificar y probar el comportamiento del servidor. Realice la implementación en TCP y comente los resultados.

Pasos para correr el ejercicio:
1. Ejecutar: Servidor (tiene su propio, sin pasar argumentos)
2. Ejecutar: Cliente (tiene su propio, sin pasar argumentos)

2) Modifique el programa anterior para que pueda atender varios clientes a la vez.

Pasos para correr el ejercicio:
1. Ejecutar: Servidor (tiene su propio, sin pasar argumentos)
2. Ejecutar: Cliente (tiene su propio, sin pasar argumentos)

3) Escriba un servidor de mensajes en colas, que permita a los clientes dejar un mensaje (identificando de alguna forma a quién se lo dejan), y bajar los mensajes que le están dirigidos. La comunicación entre cliente y servidor debe ser mediante sockets, y el servidor debe poder atender varios clientes a la vez.

Pasos para correr el ejercicio:
1. Ejecutar: Servidor (tiene su propio, sin pasar argumentos)
2. Ejecutar: Cliente (tiene su propio, sin pasar argumentos)

4) Modifique el programa anterior para que el mensaje de la cola sea borrado por el servidor una vez que el cliente confirma, mediante un mensaje de tipo ack, que efectivamente recibió el mensaje que estaba en la cola.

Pasos para correr el ejercicio:
1. Ejecutar: Servidor (tiene su propio, sin pasar argumentos)
2. Ejecutar: Cliente (tiene su propio, sin pasar argumentos)

5) Escribir un servicio que devuelva información de clima del lugar donde reside el servidor. Esta información podrá generarse de forma aleatoria. Deberá ser realizado con RMI. Para ello considere la interface remota, la clase (lado servidor) que implementa esa interface, el servidor, y un cliente que permita probar este funcionamiento.

Pasos para correr el ejercicio:
1. Ejecutar: ServidorRMI (tiene su propio, sin pasar argumentos)
2. Ejecutar: Cliente (tiene su propio main, sin pasar argumentos)

6) Escribir un servidor utilizando RMI, que ofrezca la posibilidad de sumar y restar vectores de enteros. Introduzca un error en su código que modifique los vectores recibidos por parámetro. Qué impacto se genera? Qué conclusión saca sobre la forma de pasaje de parámetros en RMI? Mostrar claramente los valores de los vectores del lado cliente, antes y después de la ejecución de la suma o resta.

Pasos para correr el ejercicio:
1. Ejecutar: ServidorRMI (tiene su propio, sin pasar argumentos)
2. Ejecutar: Cliente (tiene su propio main, sin pasar argumentos)

¿Qué impacto se genera? Qué conclusión saca sobre la forma de pasaje de parámetros en RMI?
En el servidor se modifico al vector v1 poniendo todos sus elementos en 0, pero como el vector fue pasado por valor y no referencia no se modifica del lado del cliente. Si se quisiese modificar, habría que devolver el vector modificado y asignarlo al vector original del lado del cliente.

7) Implemente un servidor RMI que resuelva tareas genéricas. Para ello tener en cuenta la interface Tarea, que tendrá un método ejecutar(). El objetivo es que desde el cliente se puedan escribir objetos (que implementen la interface Tarea) que hagan un cálculo concreto (calcular un número aleatorio, un primo, el valor de Pi con cierta precisión, etc.), y que esa tarea se pase al servidor RMI, quien hará el cálculo y devolverá el valor al cliente.

Pasos para correr el ejercicio:
1. Ejecutar: ServidorRMI (tiene su propio, sin pasar argumentos)
2. Ejecutar: Cliente (tiene su propio main, sin pasar argumentos)