Documento de pruebas del proyecto

Programación de Sistemas Concurrentes y Distribuidos

2º curso, Grado de Ingeniería en Informática, 2018-2019

Número de equipo: *1*

Integrantes:

- *Germán Garcés Latre, 757024*
- *Sergio García Esteban, 755844*
- *Diego Caballé Casanova, 738712*
- *Patricia Briones Yus, 735576*

Índice de contenidos

1. Introducción

Hemos realizado una serie de pruebas en nuestro programa para verificar el correcto funcionamiento del servicio Linda, las cuales son descritas en detalle en las secciones siguientes.

La metodología utilizada fue dividir el trabajo en casos concretos más pequeños (“*divide y vencerás”, Julio César):*

- Crear una estructura de datos genérica de tipo lista.
- Un servidor al que se pudiesen conectar varios clientes, que añadiesen y eliminasen información en las listas según el mensaje recibido.
- Hacer los servidores multicliente y que realicen las operaciones en exclusión mutua.
- Crear otro servidor (LindaServer) al que se conectasen clientes y pudieran usar las operaciones ofrecidas por el servicio Linda a través del monitor LindaDriver, sin llegar a enviar la información a los servidores.
- Crear el TAD tuplas con las operaciones que venían en el mainLindaDriver.cpp.
- Llevar el control de procesos cliente, tuplas en pizarra, etc de LindaServer con el monitor ControlLS.
- Hacer actuar a LindaServer como servidor y cliente a la vez.
- Crear un proceso administrador para cerrar de manera controlada el servicio Linda. Mejorar los cierres de comunicaciones entre procesos.
- Ir añadiendo funciones que el código requería para ir completando los requisitos del servicio y proyecto.

2. Funcionalidad clave del sistema y alcance de los correspondientes casos de prueba

El sistema requiere poder conectar uno o más clientes al servicio Linda y que pueda añadir, eliminar o leer tuplas de la memoria las veces que quiera, así como desconectarse de este.

Las pruebas hechas para asegurarnos de que la implementación de nuestro código se ajusta a los requisitos del enunciado y del servicio Linda han sido:

1. Las tuplas de diferentes tamaños que son introducidas se guardan en la memoria de los servidores que les corresponden.
2. No se puede introducir un patrón (tupla con variables) en memoria.
3. Ejecutar leer con un patrón, y que en cada una de las variables devuelva el valor de la tupla encontrada en memoria.

4. Ejecutar eliminar con un patrón, que cada una de las variables devuelva el valor de la tupla encontrada en memoria y se elimine la tupla de memoria.

5. Si el patrón ejecutado con leer o eliminar no existe, el cliente se queda esperando.

6. Si un patrón tiene el formato [?A,?A,?X], las variables de letra A tendrán que ser iguales. Ejemplo: la tupla [hola, hola, pepe] sería reconocida.

7. Cierre del servicio de manera controlada con un proceso administrador habiendo clientes conectados, pudiendo estar algunos bloqueados.

8. Un cliente se desconecta del servicio.

9. Si un cliente1 se queda bloqueado esperando una tupla, cuando otro cliente2 introduce esa tupla, cliente1 sale de su bloqueo y recibe la tupla.

3. Definición de los casos de prueba

1.

Seguimiento tuplas hasta servidor X	**Responsable**	**Patricia**
	Fecha	**14/01/2019**
Descripción: Las tuplas de diferentes tamaños que son introducidas se guardan en la memoria de los servidores que les corresponden. Probaremos con una tupla de longitud 6.		
Pasos (metodología): El cliente conectado al servicio realiza un postnote de una tupla de longitud 6.		
Resultado esperado: Tupla enviada al servidor 3 Esa información la escribirá LindaServer por pantalla.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

2.

Inserción de patrones	Responsable	Patricia
	Fecha	14/01/2019
Descripción: Los patrones (tupla con variables) no se pueden almacenar en memoria. Por tanto, devolverá un mensaje de error.		
Pasos (metodología): El cliente conectado al servicio realiza un postnote de un patrón (ej. [?A, hola]).		
Resultado esperado: El servidor correspondiente informará por pantalla de que no se ha añadido la tupla, y LindaServer escribirá por pantalla la razón por la que no se ha introducido.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

3.

Lectura de patrones	Responsable	Sergio
	Fecha	14/01/2019
Descripción: Realizar una operación readN con comodín.		
Pasos (metodología): El cliente conectado al servicio realiza un readN de un patrón (ej. [?A, hola]).		
Resultado esperado: El servidor correspondiente devolverá la tupla que coincida a la enviada sustituyendo el comodín por el valor en la tupla almacenada.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

4.

Eliminación de patrones	Responsable	Sergio
	Fecha	14/01/2019
Descripción: Realizar una operación RN con comodín.		
Pasos (metodología): El cliente conectado al servicio realiza un RN de un patrón (ej. [?A, hola]).		
Resultado esperado: El servidor correspondiente devolverá la tupla que coincida a la enviada sustituyendo el comodín por el valor en la tupla almacenada y eliminará esta tupla de memoria.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

5.

Lectura/eliminado de patrones con bloqueo	Responsable	Sergio
	Fecha	14/01/2019
Descripción: Realizar una operación readN con la memoria vacía.		
Pasos (metodología): El cliente conectado al servicio realiza un readN de una tupla.		
Resultado esperado: El cliente se quedará bloqueado.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

6.

Lectura de patrones múltiples	Responsable	Sergio
	Fecha	14/01/2019
Descripción: Realizar una operación ReadN con comodines.		
Pasos (metodología): El cliente conectado al servicio realiza un ReadN de una tupla con comodines repetidos (ej. [?A,?A,?X]).		
Resultado esperado: El servidor correspondiente devolverá la tupla que coincida a la enviada sustituyendo los comodines repetidos por los valores repetidos de la tupla almacenada en memoria (ej. [hola,hola,pepe]).		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

7.

Finalización controlada con proceso administrador	Responsable	Sergio
	Fecha	14/01/2019
Descripción: Lanzar un cliente privado que manda un mensaje especial para que se cierren todos los servidores.		
Pasos (metodología): Lanzar cliente.		
Resultado esperado: LindaServer se cerrará y los tres servidores de memoria también.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

8.

Un cliente se desconecta	Responsable	Sergio
	Fecha	14/01/2019
Descripción: Un cliente se desconecta.		
Pasos (metodología): Desconectar al cliente mediante la función de LindaDriver		
Resultado esperado: Se desconecta el cliente y Linda sigue funcionando y recibiendo nuevos clientes.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		

9.

Desbloqueo de un cliente bloqueado	Responsable	Sergio
	Fecha	14/01/2019
Descripción: Un segundo cliente inserta una tupla que busca un cliente bloqueado.		
Pasos (metodología): Hacer un PN de la tupla buscada por el cliente bloqueado.		
Resultado esperado: El PN se realiza correctamente y el cliente bloqueado se desbloquea y obtiene la información de la tupla que buscaba.		
Resultado obtenido: Resultado obtenido = Resultado esperado.		