Aplicación cliente-servidor con estado

Descripción

En esta práctica se pide el diseño e implementación de una aplicación cliente-servidor con estado para proporcionar un servicio calculadora. El protocolo se implementa sobre sockets TCP y en una arquitectura multi-hilo.

Diseño

El diseño consta de una aplicación servidor y una aplicación cliente. El servidor es multi-hilo y soporta el establecimiento de sesiones con varios clientes al mismo tiempo, para ello, delega la funcionalidad en hilos que son los que realmente responden a las peticiones de los distintos clientes.

Los usuarios hacen uso de la aplicación cliente por la cual se pueden conectar al servidor y enviar operaciones de dos operandos, que solamente soportan los operadores de suma, resta, división y multiplicación. De esta forma, los clientes podrán introducir operaciones del tipo 1+1, 2-1, 3*4 o 2/3.

Adicionalmente, el servidor dispone de un servicio de persistencia por el cual se almacena el último resultado obtenido por cada cliente. Para poder acceder a este resultado, el usuario puede hacer uso de la palabra clave ***ans*** como uno de los operandos en la petición (ans+1). Así, se sustituirá en la operación la palabra clave por el valor resultante de la última operación realizada. En caso de que todavía no se haya hecho ninguna, ans tomará el valor 0.

Protocolo

El protocolo consta de dos mensajes uno de petición y otro de respuesta. El mensaje de petición contiene el nombre de usuario y la información de la operación que deseamos realizar. El nombre de usuario identifica al cliente ante el servidor, de forma que se pueda usar la persistencia. A continuación, se incluye un carácter separador para facilitar el parseo del mensaje en el servidor. Finalmente, aparece la operación formada por los dos operandos y el operador. El mensaje finaliza con un salto de línea (*/n*).

NombreUsuario:Operando1OperadorOperando2

El mensaje de respuesta contiene únicamente el resultado de la ejecución de la operación seguido de un salto de línea (*/n*).

Ejemplos de uso

A continuación, se muestra un ejemplo de uso de la aplicación, donde dos clientes, Jorge y Pepe, realizan diversas peticiones al servidor. Se puede observar como el servidor acepta y responde a las dos sesiones de manera concurrente. De la misma forma, observamos el correcto funcionamiento del servicio de persistencia.

```
Introduzca su nombre de usuario:
Jorge
Introduzca la operacion o EXIT para finalizar la conexion:
1+2
Resultado: 2.0
Introduzca la operacion o EXIT para finalizar la conexion:
ans-2
Resultado: 0.0
Introduzca la operacion o EXIT para finalizar la conexion:
```

Ilustración 1: ejemplo de uso en el cliente Jorge

```
Introduzca su nombre de usuario:
Pepe
Introduzca la operacion o EXIT para finalizar la conexion:
1/2
Resultado: 0.5
```

Ilustración 2: ejemplo de uso en el cliente Pepe

```
Servidor ejecutándose. Esperando solicitudes...
Conexion aceptada: /127.0.0.1:25859
Cliente: Jorge
    Calculando 1*2
Conexion aceptada: /127.0.0.1:25863
Cliente: Pepe
    Calculando 1/2
Cliente: Jorge
    Calculando ans-2
```

Ilustración 3: ejemplo de uso en el servidor

Trazas

Mediante Wireshark, capturamos las tramas TCP que tienen lugar cuando usamos la aplicación. Gracias a esta herramienta somos capaces de encontrar los datos que enviamos dentro de los paquetes TCP.

Estas son las trazas que se intercambian en una petición:

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	TCP	108	25100 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=2053 Len=12
127.0.0.1	127.0.0.1	TCP	84	9999 → 25100 [ACK] Seq=1 Ack=13 Win=2053 Len=0
127.0.0.1	127.0.0.1	TCP	90	9999 → 25100 [PSH, ACK] Seq=1 Ack=13 Win=2053 Len=3
127.0.0.1	127.0.0.1	TCP	84	25100 → 9999 [ACK] Seq=13 Ack=4 Win=2053 Len=0
127.0.0.1	127.0.0.1	TCP	88	9999 → 25100 [PSH, ACK] Seq=4 Ack=13 Win=2053 Len=2
127.0.0.1	127.0.0.1	TCP	84	25100 → 9999 [ACK] Seq=13 Ack=6 Win=2053 Len=0

Ilustración 4: trazas intercambiadas

Si buscamos dentro de los mensajes podemos ver tanto la petición como la respuesta en la sección de datos de los paquetes 1 y 3, respectivamente:

```
0000  02 00 00 00 45 00 00 34  68 58 40 00 80 06 00 00   ....E..4 hX@.....
0010  7f 00 00 01 7f 00 00 01  62 0c 27 0f d9 2e b6 10   ........ b.'.....
0020  4b 14 46 57 50 18 08 05  d7 3d 00 00 4a 6f 72 67   K.FWP... .=..Jorg
0030  65 3a 61 6e 73 2b 32 0a                           e:ans+2.
```

Ilustración 5: datos del mensaje de petición

```
0000  02 00 00 00 45 00 00 2b  68 5a 40 00 80 06 00 00   ....E..+ hZ@.....
0010  7f 00 00 01 7f 00 00 01  27 0f 62 0c 4b 14 46 57   ........ '.b.K.FW
0020  d9 2e b6 1c 50 18 08 05  9a c1 00 00 35 2e 30      ....P... ....5.0
```

Ilustración 6: datos del mensaje de respuesta