Práctica 2

Heterogeneidad en la comunicación usando sockets

Félix García Carballeira

Luis Miguel Sánchez García

Carlos Fómez Carrasco

Borja Bergua Guerra

El objetivo de esta práctica es que el alumno llegue a conocer aspectos un poco más avanzados relacionados con la comunicación de procesos usando sockets (*TCP*) para permitir comunicar procesos desarrollados en diferentes lenguajes.

Descripción

El alumno deberá diseñar, codificar y probar, usando el sistema operativo UNIX/Linux, un cliente desarrollado en C, un cliente desarrollado en Java y un servidor implementado en C que usan un protocolo definido. A continuación se describen estos programas.

1.1 Desarrollo del servicio

Se pretende la implementación de dos programas:

- Un **servidor** que proporcione un servicio que realiza operaciones aritméticas sobre una matriz de enteros.
- Un **cliente** que maneje este tipo de servidores desarrollado en C.
- Un **cliente** que maneje este tipo de servidores desarrollado en Java.

1.2 Protocolo

El protocolo de comunicaciones entre el cliente y el servidor será el siguiente:

1. Una vez realizada la operación de establecimiento de la conexión, el cliente enviará un entero (4 bytes) al servidor, que representará el número de elementos de valor entero que se desean procesar.
2. El servidor devolverá el valor entero 1234567 (4 bytes) para indicar que se ha recibido correctamente el entero anterior. En caso de recibir cualquier otro valor, eso significará que se debe reintentar el paso 1.
3. Una vez realizada la operación de establecimiento de la conexión, el cliente enviará los N enteros (4 bytes cada uno) del array al servidor.
4. El servidor devolverá el valor entero 1357924 (4 bytes) para indicar que se ha recibido correctamente el entero anterior. En caso de recibir cualquier otro valor, eso significará que se debe reintentar el paso 3.

5. El cliente enviará el mandato a realizar en formato texto (“MEDIA”, “MAXIMO”, “MINIMO”, o “INVERTIR”)
6. El servidor en caso de recibir correctamente los datos, enviará 2 caracteres (“OK”) más los dos caracteres de fin de línea (’\r’+’\n’). En caso contrario, el cliente deberá reenviar el mandato de nuevo (paso 5).
7. El servidor espera un tiempo aleatorio entre 1 y 5 segundos (**man 3 rand**).
8. El servidor realizará la operación y enviará al cliente el resultado de la operación (un entero en el caso de las 3 primeras operaciones y un array de igual tamaño que el enviado por el cliente), y cerrará la conexión con el cliente.

La figura 1 muestra un esquema del protocolo establecido entre el cliente y el servidor.

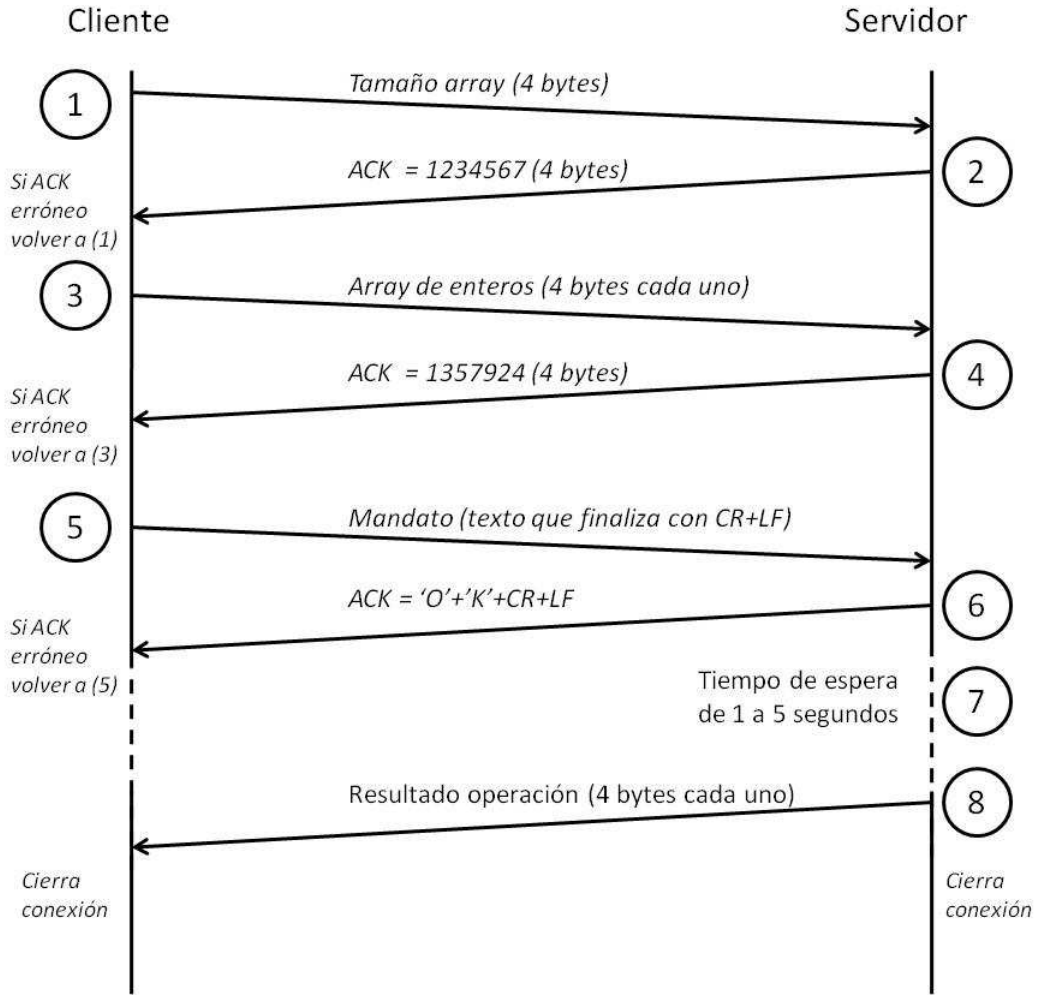

Ilustración 1

1.3 Desarrollo del servidor

Se ejecutará de la siguiente manera:

```
$ ./servidor_enteros –p <puerto>
```

O en caso de que se quiera ejecutar en modo depuración:

```
$ ./servidor_enteros –p <puerto> -d
```

Los mensajes que aparecerán en modo depuración serán los siguientes:

- Al iniciar saldrá el siguiente mensaje:

```
s> init server <IP local>:<port>
```

- Antes de recibir una petición se debe mostrar el siguiente mensaje:

```
s> waiting
```

- Si recibe una conexión se mostrará el siguiente mensaje:

```
s> accept <IP cliente>:<port>
```

Ante una petición se mostrará la siguiente información:

- Tamaño del array:

```
s> <IP cliente>:<port> Recv Array size <tamaño>
```

- Respuesta del servidor:

```
s> <IP cliente>:<port> Send <codigo>
```

- Elementos del array:

```
s> <IP cliente>:<port> Recv Array <elemento 0> … <elemento n-1>
```

- Respuesta del servidor:

```
s> <IP cliente>:<port> Send <codigo>
```

- Mandato recibido:

```
s> <IP cliente>:<port> Recv cmd <mandato>
```

- Respuesta del servidor al mandato:

```
s> <IP cliente>:<port> Send <codigo>
```

- Resultado devuelto al cliente:

```
s> <IP cliente>:<port> Send Resultado <datos enviados>
```

- Cuando se cierra la conexión:

```
s> <IP cliente>:<port> close
```

El programa terminará al recibir una señal SIGINT (*Ctrl+c*)

1.4 Desarrollo del cliente

El cliente es un programa llamado `cliente_enteros` que funciona de la siguiente manera:

```
$ ./cliente_enteros –s <servidor> -p <puerto> -m <mandato> -f <fichero> -d
```

Donde **<servidor>** puede ser tanto el nombre como la IP del servidor. De esta manera se debe conectar el cliente con el servidor. En caso de que no se pueda conectar, se deberá mostrar el siguiente mensaje:

```
Error en la conexión con el servidor <servidor>:<puerto>
```

El mandato puede ser (siempre en mayúsculas):

- MEDIA: para obtener la media aritmética (entera) del array de enteros.
- MAXIMO: para obtener el valor máximo del array de enteros.
- MINIMO: para obtener el valor máximo del array de enteros.
- INVERTIR: para obtener la matriz en el orden contrario al enviado.

El fichero contiene los números enteros que se van a enviar al servidor para su tratamiento.

Si se pasa la opción –d, el cliente debe mostrar la siguiente información:

- Tamaño del array:

```
c> <IP servidor>:<port> Send size <tamaño>
```

- Respuesta del servidor:

```
c> <IP servidor>:<port> Recv <codigo>
```

- Elementos del array:

```
c> <IP servidor>:<port> Send Array <elemento 0>  … <elemento n-1>
```

- Respuesta del servidor:

```
c> <IP servidor>:<port> Recv <codigo>
```

- Mandato enviado:

```
c> <IP cliente>:<port> Send cmd <mandato>
```

- Respuesta del servidor al mandato:

```
c> <IP servidor>:<port> Recv <codigo>
```

- Resultado devuelto del servidor:

```
c> <IP servidor>:<port> Recv Resultado <datos recibidos>
```

- Cuando se cierra la conexión:

```
c> <IP servidor>:<port> close
```

En todos los casos el o los valores obtenidos se deben mostrar por pantalla usando un salto de línea para separa los valores. Ej:

```
$ cat foo.txt
1
2
3
4
5
6
7
```

```
$ ./cliente_enteros –s localhost -p 9000 -m INVERTIDO -f foo.txt
7
6
5
4
3
2
1
```

```
$ ./cliente_enteros –s localhost -p 9000 -m MAXIMO -f foo.txt
7
```

Recomendaciones generales

Es importante analizar el **código de apoyo** proporcionado con la práctica ya que será el punto de partida para la realización de la misma. Se va a proporcionar el esqueleto de los programa cliente y servidor. Hay que utilizar estos dos programas como punto de partida. Todo el tratamiento de los argumentos y del inteprete de mandatos **está ya implementado**.

El alumno tiene libertad a la hora de diseñar el sistema siempre que proporcione la funcionalidad pedida.

Otro aspecto que conviene resaltar es que, debido al esquema de compilación usado en la práctica, puede ocurrir que un error de programación (como, por ejemplo, usar *print* en vez de *printf*) aparezca simplemente como un *warning* en la fase de compilación y enlazado. El error como tal no aparecerá hasta que se ejecute el sistema. En resumen, vigile los *warnings* que se producen durante la compilación.

Documentación a entregar

Se deben entregar los siguientes archivos

memoria.pdf

En ella se deben comentar los aspectos del desarrollo de su práctica que considere más relevantes. Asimismo, puede exponer los comentarios personales que considere oportunos. **Se deberá entregar un documento en formato pdf**.

Se deberá cumplir los siguientes requisitos:

- Presentar una estructura lógica en sus contenidos (índice de contenidos).
- Estar convenientemente formateada, para facilitar su lectura.
- Describir con claridad, y en profundidad los puntos recogidos en este cuaderno de prácticas, así como los que decidan incluir como complemento.
- Incluir las pruebas realizadas.
- Incluir los comentarios personales que considere oportunos.
- NO incluir el código fuente.

La memoria debe incluir como mínimo los siguientes puntos:

1. Índice de contenidos.
2. Diseño del programa, usando diagramas de flujo para explicar el funcionamiento del mismo.
3. Pruebas realizadas para probar la aplicación.
4. Conclusiones del alumno.
5. Descripción de las tareas realizadas y tiempo dedicado a cada tarea.

El documento no debe sobrepasar las 20 hojas.

cliente_enteros.c

Implementación del cliente desarrollado en C definido en este enunciado de la práctica.

Cliente_enteros.java

Implementación del cliente desarrollado en Java definido en este enunciado de la práctica.

servidor_ enteros.c

Implementación del servidor definido en este enunciado de la práctica.