Práctica 1

Comunicación de procesos usando sockets

Félix García Carballeira

Luis Miguel Sánchez García

Carlos Fómez Carrasco

Borja Bergua Guerra

Introducción

El objetivo de esta práctica es que el alumno llegue a conocer los principales conceptos relacionados con la comunicación de procesos usando sockets (*tcp*).

1. Descripción

El alumno deberá diseñar, codificar y probar, utilizando el lenguaje C y sobre sistema operativo UNIX/Linux un cliente y servidor que analiza y modifica ficheros de texto. A continuación se describe este programa.

1.1 Desarrollo del servicio

Se pretende la implementación de 2 programas:

- Un **servidor** que proporcione un servicio que analiza y modifica ficheros de texto.
- Un **cliente** que maneje este tipo de servidores.

1.2 Desarrollo del servidor

El servidor (`servidor_texto`) debe proporcionar un protocolo que permita realizar las siguientes operaciones:

- Contar el número de letras de un fichero de texto.
- Contar el número de palabras de un fichero de texto.
- Modificar un fichero de texto sustituyendo los espacios en blanco de un fichero por un salto de línea (‘\n’). De esta manera, los ficheros modificados contienen una palabra por línea eliminando todos los espacios en blando del nuevo fichero.

El servidor debe gestionar varias conexiones a la vez (*multithread*). El servidor utilizará sockets **TCP** orientados a conexión.

1.2.1 Uso

Se ejecutará de la siguiente manera:

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

O en caso de que se quiera realizar el modo depuración:

```
$ ./servidor_texto –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>
```

Ante una petición se mostrará la siguiente información:

- Si es contar letras de un fichero:

```
s> <IP cliente>: letras <nombre del fichero> <tamaño del fichero>
```

Al terminar la operación se mostrará la siguiente información:

```
s> <IP cliente>: letras <resultado>
```

- Si es contar palabras de un fichero:

```
s> <IP cliente>: palabras <nombre del fichero> <tamaño del fichero>
```

Al terminar la operación se mostrará la siguiente información:

```
s> <IP cliente>: palabras <resultado>
```

- Si es modificar el fichero origen:

```
s> <IP cliente>: cambiar  <nombre del fichero origen> <tamaño del fichero origen>
```

Al terminar la operación se mostrará la siguiente información:

```
s> <IP cliente>: cambiar  <nombre del fichero destino> <tamaño del fichero destino>
```

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

1.3 Desarrollo del cliente

El cliente es un intérprete de mandatos llamado `cliente_texto` funciona de la siguiente manera:

```
$ ./cliente_texto –s <servidor> -p <puerto>
```

Donde servidor puede ser el nombre o 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:

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

En caso de que la conexión funcione, deberá mostrarse un terminal:

```
c>
```

Para finalizar el cliente se debe escribir **quit**.

```
c> quit
```

1.3.1 Servicio contar letras

El cliente para poder acceder al servicio para contar el número de letras de un fichero, se usará el siguiente mandato:

```
c> letras <ruta del fichero>
```

El cliente debe mostrar el número de letras que contiene el fichero.

Ejemplo:

```
c> letras /tmp/foo.txt
10
```

NOTA: letras son las comprendidas entre A..Z y a..z. No se consideran el resto de caracteres alfabéticos (ñ, acentos, etc)

1.3.2 Servicio contar palabras

El cliente para poder acceder al servicio para contar el número de palabras de un fichero, se usará el siguiente mandato:

```
c> palabras <ruta del fichero>
```

El cliente debe mostrar el número de palabras que contiene el fichero.

Ejemplo:

```
c> palabras /tmp/foo.txt
5
```

1.3.3 Servicio modificar fichero de texto

El cliente para poder acceder al servicio para modificar un fichero de texto, se usará el siguiente mandato:

```
c> cambiar <ruta del fichero origen> <ruta del fichero destino>
```

El fichero destino almacenará el fichero modificado por el servidor. El fichero destino es similar al fichero origen, pero contiene una única palabra por cada línea.

El cliente debe mostrar el número de palabras que contiene el fichero.

Ejemplo:

```
c> cambiar /tmp/foo.txt /tmp/foo1.txt
```

Donde **`/tmp/foo.txt`** contiene:

Lorem ipsum id qui kasd necessitatibus, ne eum ornatus epicuri recteque. Pri atqui tempor inciderint ad, ad illum eruditi necessitatibus eam, sea debitis sadipscing no.

Y **`/tmp/foo1.txt`**

Lorem
ipsum
id
…

2. 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.

3. Documentación a entregar

Se debe 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**.

Además 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_texto.c

Implementación del cliente definido en la práctica de la práctica.

servidor_texto.c

Implementación del cliente definido en la práctica de la práctica.