SISTEMAS DISTRIBUIDOS

INGENIERÍA INFORMÁTICA

UNIVERSIDAD CARLOS III DE MADRID

Práctica 2:
Llamadas a procedimientos remotos (RPC)

Mª Gregoria Casares Andrés
Alejandro Calderón

Marzo 2012

Índice de contenidos

1. Objetivo de la práctica.....5

2. Consideraciones previas.....5

3. Descripción de la funcionalidad.....5

3.1 Desarrollo del servicio.....5

3.2 Desarrollo del servidor.....6

3.2.1 Uso.....6

3.2.2 Servicio de ping.....6

3.2.3 Servicio de intercambio de letras.....6

3.2.4 Servicio de cálculo de la función resumen.....7

3.2.5 Servicio de comprobación del resumen de un texto.....8

3.2.6 Servicio de obtención de estadísticas.....8

3.3 Desarrollo del cliente.....9

3.3.1 Uso.....9

3.3.2 Servicio ping.....10

3.3.3 Servicio de intercambio de letras.....10

3.3.4 Servicio de cálculo de la función resumen.....11

3.3.5 Servicio de comprobación del resumen de un texto.....11

3.3.6 Servicio de obtención de estadísticas.....12

3.4 Bibliografía.....13

4. Normas y recomendaciones generales.....14

5. Documentación a entregar.....15

5.1 autores.txt.....15

5.2 memoria.pdf.....15

5.3 text.x.....16

5.4 client.c.....16

5.5 server.c.....16

5.6 Fichero a entregar.....17

6. Entrega.....18

7. Tareas a realizar.....19

1. Objetivo de la práctica

El objetivo de esta práctica es que el alumno llegue a conocer los principales conceptos relacionados con la comunicación de procesos usando llamadas a procedimientos remotos (RPC).

2. Consideraciones previas

Es responsabilidad del alumno leer y comprender este enunciado de esta práctica. Las sesiones de laboratorio estarán dedicadas a resolver todas aquellas dudas puntuales que hayan podido surgir durante el trabajo del alumno, con el fin de poder solucionar los problemas que puedan surgir antes de que acabe el plazo de entrega de la práctica.

3. Descripción de la funcionalidad

El alumno deberá diseñar, codificar y probar, utilizando el lenguaje C y sobre sistema operativo UNIX/Linux un cliente y un servidor que analice y modifique textos. El servidor y el cliente en C **tendrán la misma funcionalidad** que los desarrollados en la práctica anterior, con las modificaciones que sean **necesarias** para que se comuniquen usando RPC en lugar de sockets. A continuación se describe este programa.

3.1 Desarrollo del servicio

Se pretende la implementación de dos programas:

- Un **servidor en C** que proporcione un servicio que analice y modifique textos. **Tendrá la misma funcionalidad** que el desarrollado en la práctica anterior, con las modificaciones que sean **necesarias** para que se comunique con el cliente en C usando RPC en lugar de sockets.
- Un **cliente en C** que se comunique con este servidor. **Tendrá la misma funcionalidad** que el desarrollado en la práctica anterior, con las modificaciones que sean **necesarias** para que se comunique con el servidor en C usando RPC en lugar de sockets.

3.2 Desarrollo del servidor

El **servidor en C** (`server`), que **tendrá la misma funcionalidad** que el de la práctica anterior, con las modificaciones que sean **necesarias** para que se comunique con el cliente en C usando RPC, debe proporcionar un protocolo que permita realizar las siguientes operaciones:

- Devolver al cliente una respuesta a un *ping* emitido por éste.
- Intercambiar letras (comprendidas entre a-zA-Z) de minúsculas a mayúsculas y viceversa de un texto. Cualquier otro carácter se debe dejar intacto.
- Calcular una función resumen (o *hash*) sobre un texto.
- Comprobar el resumen (o *hash*) de un texto.
- Contabilizar el número de veces que se le ha solicitado cada una de las operaciones anteriores.

El servidor utilizará llamadas a procedimientos remotos (**RPC**).

3.2.1 Uso

Se ejecutará de la siguiente manera:

```
$ ./server
```

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

3.2.2 Servicio de *ping*

Ante una petición de *ping* se mostrará el siguiente mensaje:

```
s> <IP cliente>:<puerto cliente> ping
```

Y se devolverá al cliente un ACK (1 byte) tan pronto como sea posible.

3.2.3 Servicio de intercambio de letras

Ante una petición de intercambio de minúsculas y mayúsculas se mostrará el siguiente mensaje:

```
s> <IP cliente>:<puerto cliente> init swap <longitud del texto>
```

El servidor recibirá un texto enviado por el cliente y devolverá al cliente el texto con las letras intercambiadas: todas las minúsculas (a...z) por mayúsculas (A...Z) y viceversa; y un `unsigned int` (4 bytes), que indique cuántas letras se han intercambiado.

AVISO: Para enviar la cantidad de letras intercambiadas es OBLIGATORIO que se envíe en formato numérico, es decir, un `unsigned int` (4 bytes). Queda EXPRESAMENTE PROHIBIDO pasar dicho número a cadena de caracteres (tal y como hace *sprintf* y familia) y enviar los dígitos del número como si se tratara de un *char* *.

Hay que tener en cuenta que el texto enviado por el cliente al servidor puede ser de cualquier tamaño.

Al terminar la operación se mostrará la siguiente información:

```
s> <IP cliente>:<puerto cliente> swap = <numero de letras intercambiadas>
```

3.2.4 Servicio de cálculo de la función resumen

Ante una petición de cálculo de la función resumen se mostrará el siguiente mensaje:

```
s> <IP cliente>:<puerto cliente> init hash <longitud del texto>
```

En este momento, el servidor recibirá un texto enviado por el cliente y debe calcular la función resumen usando el siguiente algoritmo:

```
hash = 0
For caracter in texto do:
  hash = (hash + caracter) mod 1.000.000.000
End do
```

AVISO: Para enviar el resumen del texto es OBLIGATORIO que se envíe en formato numérico, es decir, un `unsigned int` (4 bytes). Queda EXPRESAMENTE PROHIBIDO pasar dicho número a cadena de caracteres (tal y como hace *sprintf* y familia) y enviar los dígitos del número como si se tratara de un *char* *.
Hay que tener en cuenta que el texto enviado por el cliente al servidor puede ser de cualquier tamaño.

Al terminar la operación se mostrará la siguiente información:

```
s> <IP cliente>:<puerto cliente> hash = <resultado>
```

3.2.5 Servicio de comprobación del resumen de un texto

Ante una petición de comprobación del resumen de un texto se mostrará el siguiente mensaje:

```
s> <IP cliente>:<puerto cliente> init check <longitud del texto> <resumen>
```

En este momento, el cliente enviará al servidor un texto y un resumen y el servidor debe calcular la función resumen usando el algoritmo del apartado anterior y comprobar si coincide con el resumen que le ha enviado el cliente. Finalizada la comprobación, el servidor enviará al cliente dos posibles respuestas: `OK` (representado con un entero de un byte cuyo valor es 0) o `FAIL` (representado con un entero de un byte cuyo valor es 1).

AVISO: Para enviar el resumen del texto es OBLIGATORIO que se envíe en formato numérico, es decir, un `unsigned int` (4 bytes). Queda EXPRESAMENTE PROHIBIDO pasar dicho número a cadena de caracteres (tal y como hace *sprintf* y familia) y enviar los dígitos del número como si se tratara de un *char* *.

Hay que tener en cuenta que el texto enviado por el cliente al servidor puede ser de cualquier tamaño.

Al terminar la operación se mostrará la siguiente información:

```
s> <IP cliente>:<puerto cliente> check = [OK|FAIL]
```

3.2.6 Servicio de obtención de estadísticas

Ante una petición de obtención de estadísticas se mostrará el siguiente mensaje:

```
s> <IP cliente>:<puerto cliente> init stat
```

Se devolverá al cliente cinco valores de tipo `unsigned int` (4 bytes cada uno) correspondientes al número de veces que se le ha pedido cada una de las operaciones que es capaz de realizar el servidor (sin contar esta).

AVISO: Para enviar las estadísticas es OBLIGATORIO que se envíen en formato numérico, es decir, cinco `unsigned int` (4 bytes cada uno). Queda EXPRESAMENTE PROHIBIDO pasar

dichos números a cadenas de caracteres (tal y como hace *sprintf* y familia) y enviar los dígitos de los números como si se tratara de un *char* *.

Al terminar la operación se mostrará la siguiente información:

```
s> <IP cliente>:<puerto cliente> stat = <num_pings> <num_swaps> <num_hashs> <num_checks> <num_stats>
```

3.3 Desarrollo del cliente

El **cliente en C** es un intérprete de mandatos (`client`), que **tendrá la misma funcionalidad** que el de la práctica anterior, con las modificaciones que sean **necesarias** para que se comunique con el servidor en C usando RPC.

3.3.1 Uso

Se ejecutarán de la siguiente manera:

```
$ ./client –s <servidor>
```

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 con el servidor, se deberá mostrar el siguiente mensaje:

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

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

```
c>
```

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

```
c> quit
```

3.3.2 Servicio ping

El cliente, para poder acceder al servicio de *ping*, usará el siguiente mandato:

```
c> ping
```

El cliente debe enviar un mensaje tan pequeño como sea posible al servidor y esperar la respuesta de éste. Una vez recibida la respuesta, debe calcular la diferencia de tiempo desde que lo envió, dividirla por 2 y escribir este tiempo por pantalla. Algo así como esto:

```
init = gettimeofday()
send(servidor, mensaje)
receive(servidor, mensaje)
end  = gettimeofday()

time = (end – init)/2   /* OJO con los truncamientos */
printf(time)
```

Ejemplo:

```
c> ping
0.000341 s
```

3.3.3 Servicio de intercambio de letras

El cliente, para poder acceder al servicio de intercambio de letras de un texto, usará el siguiente mandato:

```
c> swap <ruta a fichero origen> <ruta a fichero destino>
```

El cliente debe leer el contenido del fichero origen, enviárselo al servidor y esperar la respuesta de éste. Una vez recibida la respuesta, debe escribir el texto modificado y recibido del servidor al fichero de destino y mostrar la cantidad de letras intercambiadas (cantidad que habrá recibido del servidor, junto con el texto modificado).

Ejemplo:

```
c> swap /tmp/foo.txt /tmp/foo2.txt
53
```

AVISO: El cliente debe enviar al servidor el contenido del fichero, **NO LA RUTA DEL FICHERO**, puesto que cliente y servidor podrán estar y estarán en máquinas distintas y no se puede asumir que compartan un mismo sistema de ficheros.

Hay que tener en cuenta que el texto enviado por el cliente al servidor puede ser de cualquier tamaño, es decir el fichero puede tener cualquier tamaño.

3.3.4 Servicio de cálculo de la función resumen

El cliente, para poder acceder al servicio de cálculo de la función resumen de un texto, usará el siguiente mandato:

```
c> hash <ruta a un fichero>
```

El cliente debe leer el contenido del fichero, enviárselo al servidor y esperar la respuesta de éste. Una vez recibida la respuesta, debe mostrar por pantalla el valor de la función resumen generado por el servidor.

Ejemplo:

```
c> hash /tmp/foo.txt
12345678
```

AVISO: El cliente debe enviar al servidor el contenido del fichero, **NO LA RUTA DEL FICHERO**, puesto que cliente y servidor podrán estar y estarán en máquinas distintas y no se puede asumir que compartan un mismo sistema de ficheros.

Hay que tener en cuenta que el texto enviado por el cliente al servidor puede ser de cualquier tamaño, es decir el fichero puede tener cualquier tamaño.

3.3.5 Servicio de comprobación del resumen de un texto

El cliente, para poder acceder al servicio de comprobación de la función resumen de un texto, usará el siguiente mandato:

```
c> check <ruta a un fichero> <resumen>
```

El cliente debe leer el contenido del fichero, enviárselo al servidor junto con el resumen introducido por teclado y esperar la respuesta de éste. Una vez recibida la respuesta, debe mostrar por pantalla el valor de la comprobación de la función resumen generado por el servidor (`OK` o `FAIL`).

Ejemplo:

```
c> check /tmp/foo.txt 12345678
OK

c> check /tmp/foo.txt 12345679
FAIL
```

AVISO: El cliente debe enviar al servidor el contenido del fichero, **NO LA RUTA DEL FICHERO**, puesto que cliente y servidor podrán estar y estarán en máquinas distintas y no se puede asumir que compartan un mismo sistema de ficheros.

Hay que tener en cuenta que el texto enviado por el cliente al servidor puede ser de cualquier tamaño, es decir el fichero puede tener cualquier tamaño.

3.3.6 Servicio de obtención de estadísticas

El cliente, para poder acceder al servicio de obtención de estadísticas, usará el siguiente mandato:

```
c> stat
```

El cliente debe enviar un mensaje al servidor y esperar la respuesta de éste. Una vez recibida la respuesta, debe mostrar por pantalla los cinco valores.

Ejemplo:

```
c> stat
ping 11
swap 22
hash 33
check 44
stat 55
```

3.4 Bibliografía

A tutorial on ONC RPC by Dr Dave Marshall of Cardiff University:
http://www.cs.cf.ac.uk/Dave/C/node33.html

A developer's introduction to RPC and XDR, from SGI IRIX documentation:
http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=0650&db=bks&srch=&fname=/SGI_Developer/IRIX_NetPG/sgi_html/ch04.html

4. Normas y recomendaciones generales

Es responsabilidad del alumno leer y comprender este enunciado de práctica, las sesiones de laboratorio estarán dedicadas a resolver todas aquellas dudas puntuales que hayan podido surgir durante el trabajo del alumno, con el fin de poder solucionar los problemas que puedan surgir antes de que acabe el plazo de la entrega.

Se valorará el **uso correcto del lenguaje** en la memoria.

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 intérprete 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.

ADVERTENCIA: Se va a dedicar especial interés en comprobar la existencia de **copias** entre prácticas de distintos grupos, tanto en convocatorias del presente curso como de cursos anteriores.

5. Documentación a entregar

Se debe entregar un archivo comprimido en formato zip con el nombre **ssdd_p2_A_B.zip** donde **A** y **B** son los NIAs de los integrantes del grupo.

Por ejemplo: ssdd_p2l_100012345_100067890.zip

El archivo debe contener:

- **autores.txt**
- **memoria.pdf**
- **text.x**
- **client.c**
- **server.c**

5.1 autores.txt

El fichero de autores contendrá los datos personales de los autores de la práctica separados por tabulador utilizando el siguiente formato por línea:

```
Apellidos, Nombre   NIA
Apellidos, Nombre   NIA
```

5.2 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**.

No descuide la calidad de la memoria de su práctica. Aprobar la memoria es imprescindible para aprobar la práctica, tanto como el correcto funcionamiento de la misma. **Si al evaluarse la memoria de su práctica, se considera que no alcanza el mínimo admisible, su práctica estará suspensa**.

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.

5.3 text.x

Implementación del servicio RPC definido en este enunciado de práctica.

5.4 client.c

Implementación del **cliente en C** definido en este enunciado de práctica. Misma funcionalidad que el de las prácticas anteriores pero usando RPC.

5.5 server.c

Implementación del **servidor en C** definido en este enunciado de práctica. Misma funcionalidad que el de las prácticas anteriores pero usando RPC.

5.6 Fichero a entregar

Para crear el fichero a entregar se deben seguir los siguientes pasos:

1. Creamos el directorio para preparar los materiales a entregar y comprobamos que nos encontramos en el directorio de la entrega:

```
$ cd
$ mkdir ssdd_p2_AAAAAAAAA_BBBBBBBBB
$ cd ssdd_p2_AAAAAAAAA_BBBBBBBBB
$ pwd
<path>/ssdd_p2_AAAAAAAAA_BBBBBBBBB
```

2. Después procedemos a copiar los ficheros **autores.txt**, **memoria.pdf** y los ficheros con las fuentes (**text.x**, **client.c** y **server.c**) al directorio de la entrega. Una vez se haya realizado la copia, comprobamos el contenido:

```
$ ls
autores.txt memoria.pdf text.x client.c server.c
```

3. Y procedemos a generar el fichero zip a ser entregado:

```
$ cd ..
$ ls
... ssdd_p2_AAAAAAAAA_BBBBBBBBB...
$ zip -r ssdd_p2_AAAAA_BBBBB.zip ssdd_pN_T_AAAAA_BBBBB/
updating: ssdd_p2_AAAAA_BBBBB/ (stored X%)
  adding: ssdd_p2_AAAAA_BBBBB/autores.txt (stored X%)
  adding: ssdd_p2_AAAAA_BBBBB/memoria.pdf (stored X%)
  adding: ssdd_p2_AAAAA_BBBBB/text.x  (stored X%)
  adding: ssdd_p2_AAAAA_BBBBB/client.c (stored X%)
  adding: ssdd_p2_AAAAA_BBBBB/server.c (stored X%)
```

6. Entrega

Para entregar la práctica deberá utilizarse el entregador de Aula Global 2 en el apartado entrega de prácticas. **No se permite la entrega** por correo electrónico sin autorización previa. En caso de que se entregue la práctica varias veces sólo será válida la última entrega.

IMPORTANTE: La práctica **sólo** deberá ser entregada por **un único integrante** del grupo de prácticas. No se debe entregar la misma práctica de forma repetida por todos los integrantes del grupo.

La entrega incluye los siguientes apartados:

- **Fichero IDL** especificando el servicio**.**
- **Cliente desarrollado en C** usando RPC que se comunique con el servidor de procesamiento de textos.
- **Servidor desarrollado en C** usando RPC que implemente el servicio de procesamiento de textos y se comunique con los clientes que desean usar este servicio.
- **Memoria de la práctica**
- **Fichero de autores**

El plazo de entrega de la entrega final se especificará en el entregador.

7. Tareas a realizar

	Tareas	Hecho	Probado
Práctica 2	**Llamadas a procedimientos remotos (RPC)**	-	-
	Implementar el servicio *ping*		
	Comunicar el cliente con el servidor usando el servicio *ping*		
	Implementar el servicio *intercambiar letras*		
	Comunicar el cliente con el servidor usando el servicio *intercambiar letras*		
	Implementar el servicio *calcular función resumen*		
	Comunicar el cliente con el servidor usando el servicio *calcular función resumen*		
	Implementar el servicio *comprobar función resumen*		
	Comunicar el cliente con el servidor usando el servicio *comprobar función resumen*		
	Implementar el servicio *obtener estadísticas*		
	Comunicar el cliente con el servidor usando el servicio *obtener estadísticas*		