Departamento de Ingeniería Telemática
Escuela Superior de Ingenieros
Universidad de Sevilla

PRÁCTICA 1.
LLAMADAS A PROCEDIMIENTO REMOTO

1. OBJETIVO

El objetivo es mostrar el funcionamiento de las Llamadas a Procedimiento Remoto (RPC, Remote Procedure Call) mediante la prueba de un ejemplo simple.

2. DESCRIPCIÓN

Las llamadas a procedimiento remoto permiten crear servidores de una forma fácil y cómoda. Para la especificación de un servicio en las RPCs se debe definir un conjunto de procedimientos dentro de un programa. Cada programa puede tener varias versiones. Esto permite cambiar la interfaz del servicio permitiendo que clientes que utilizan versiones anteriores sigan funcionando. Los programas, las versiones y procedimientos se identifican mediante números enteros. Un procedimiento queda totalmente identificado mediante la terna:

<programa, versión, procedimiento>

Los números de versión se utilizan en las RPCs para permitir que existan servidores similares (mismo programa) que exportan interfaces distintas (distintas versiones).

El primer paso que hay que dar para la creación de un servidor es especificar su interfaz, donde se indican los procedimientos que el servidor exporta. Esta especificación se escribe en un fichero cuya extensión es `.x`.

Por ejemplo, para un servidor con dos procedimientos remotos, sumar y restar, la interfaz es la siguiente:

```
program CALCULAR {
    version UNO {
        int sumar (int a, int b) = 1;
        int restar(int a, int b) = 2;
    }=1;
}=999999999;
```

calcular.x

En calcular.x se ha definido la versión `UNO` (con identificador `1`) del programa `CALCULAR` (con identificador `999999999`) que contiene el procedimiento `sumar` (con identificador `1`) y el procedimiento `restar` (con identificador `2`). No se puede utilizar como número de procedimiento el valor cero.

⇒ 1. Arranque el ordenador con Ubuntu.

⇒ 2. Abra un terminal pulsando el icono correspondiente en la pantalla.

⇒ 3. Cree un directorio cuyo nombre sea su login en la plataforma de enseñanza virtual (este directorio es el que se entregará en la tarea correspondiente una vez finalizada la práctica). Cree dentro de este directorio, otro directorio para trabajar con las RPC's de nombre calcularRPC.

⇒ 4. Vaya a este directorio y realice los siguientes pasos.

⇒ 5. Cree mediante algún editor (por ejemplo gedit) el fichero calcular.x con el contenido dado previamente. NO haga un copiar y pegar desde el pdf para los ficheros de esta práctica, edítelos a mano completamente.

Una vez definida la interfaz del servidor, se utiliza el programa `rpcgen` para generar automáticamente el soporte (stub) del cliente y del servidor. Debido a que se ha utilizado más de un argumento en los procedimientos hay que indicárselo explícitamente a este programa con la opción `-N` (`-C` para generar código ANSI C):

```
rpcgen -N -C calcular.x
```

con esta orden se generarán automáticamente los siguientes ficheros:

`calcular.h`: fichero de cabecera a incluir en el cliente y en el servidor.
`calcular_svc.c`: soporte del servidor.
`calcular_clnt.c`: soporte del cliente.
`calcular_xdr.c`: funciones para la transformación de tipos.

⇒ 6. En el terminal, en el directorio calcularRPC, ejecute rpcgen para la generación de los soportes mediante la orden:

$ rpcgen –N –C calcular.x

⇒ 7. Muestre los ficheros que hay en el directorio calcularRPC mediante la orden:

$ ls

⇒ 8. Muestre el contenido de los ficheros que se han creado mediante:

$ more calcular.h
$ more calcular_svc.c
$ more calcular_clnt.c
$ more calcular_xdr.c

NOTA: Todo este código ha sido generado automáticamente por rpcgen y no hay que entenderlo, pero sí reconocer en él los métodos que hemos puesto en la interfaz.

A continuación el programador debe codificar el programa cliente y el programa servidor.

Los servidores deben implementar cada una de las funciones especificadas en el archivo de definición de interfaces. Estas funciones, que deben ser implementadas por el programador, tienen la siguiente forma:

```
tipo_resultado *procedimiento_V_svc(tipo_argumento1 arg1, struct svc_req
*sr);
```

Al nombre del procedimiento se le añade el número de versión seguido de `_svc` (porque pertenece al servidor). Aunque sólo se ha indicado el argumento arg1, debe haber un argumento para cada uno de los argumentos especificados en el fichero de especificación del servidor. El segundo argumento permite acceder, aunque no se utiliza normalmente, a diferentes características como el número de versión o el número de procedimiento.

En el ejemplo, el programa servidor debe incluir la implementación de las funciones `sumar` y `restar` de la siguiente forma:

```
#include "calcular.h"
int * sumar_1_svc(int a, int b, struct svc_req *rqstp)
{
    static int r;
    r = a + b;
    return (&r);
}

int * restar_1_svc(int a, int b, struct svc_req *rqstp)
{
    static int r;
    r = a - b;
    return (&r);
}
```

calcular_servidor.c

Las variables que almacenan el resultado deben ser de tipo static (`static int r`), debido a que las funciones anteriores devuelven un puntero a entero.

Los prototipos de las funciones están especificados en el fichero calcular.h que ha generado rpcgen.

⇒ 9. Cree el fichero calcular_servidor.c con el contenido indicado anteriormente usando la siguiente orden:

$ gedit calcular_servidor.c &

El cliente, para poder ejecutar un procedimiento remoto debe en primer lugar establecer una conexión con el servidor, mediante el siguiente servicio:

```
CLIENT *clnt_create(char *host,u_long prognum,u_long versnum,char *protocol);
```

El primer argumento especifica el nombre de la máquina donde ejecuta el servidor. Los dos siguientes argumentos especifican el número de programa y número de versión del procedimiento remoto a ejecutar. El último argumento especifica el protocolo de transporte empleado (UDP o TCP). Devuelve un manejador de cliente (`CLIENT *`) que permite al cliente realizar llamadas al servidor. Si falla devuelve `NULL` y se puede usar `clnt_pcreateerror()` para imprimir la razón del error.

El prototipo que debe emplear el cliente para las llamadas a procedimientos remotos es:

```
tipo_resultado *procedimiento_V(tipo_argumento arg, CLIENT *cl);
```

donde el sufijo `V` representa el número de versión. El argumento cl es el resultado obtenido en la llamada a clnt_create.

El programa cliente para el ejemplo del servidor para el programa `CALCULAR` es el siguiente:

```
#include "calcular.h"
int main (int argc, char **argv)
{
    char *host;
    CLIENT *sv;
    int *res;
    if (argc!=2)
        printf("Uso: %s  <host>\n", argv[0]);
    else
      {
        host = argv[1];
        sv = clnt_create(host, CALCULAR, UNO, "tcp");
        if (sv != NULL)
          {
            res = sumar_1(5, 2, sv);
            if (res != NULL)
            {
              printf("5 + 2 = %d\n", *res);
            }
            else
            {
               clnt_perror(sv, "error en RPC");
            }
            clnt_destroy(sv);
          }
        else
          {
            clnt_pcreateerror(host);
          }
      }
   return(0);
}
```

calcular_cliente.c

El programa cliente recibe el nombre de la máquina donde ejecuta el servidor en la línea de argumentos y la almacena en la variable host (hay que darle como primer argumento localhost o 127.0.0.1 si el servidor está en local). Se ha utilizado la función clnt_destroy que permite romper la asociación entre el cliente y el servidor.

⇒ 10. Cree el fichero calcular_cliente.c con el contenido indicado anteriormente usando la siguiente orden:

$ gedit calcular_cliente.c &

⇒ 11. Compile el programa servidor (se producen algunos warnings de parámetros sin uso)

$ gcc –W –Wall calcular_xdr.c calcular_svc.c calcular_servidor.c –o servidor

⇒ 12. Compile el programa cliente mediante la orden:

$ gcc –W –Wall calcular_xdr.c calcular_clnt.c calcular_cliente.c –o cliente

Para ejecutar el servidor y luego el cliente, en nuestra máquina debe haber un proceso especial que se conoce como enlazador y que mapea el nombre del servidor a un puerto concreto para que el cliente lo pueda localizar.

⇒ 13. Ejecute el servidor usando la siguiente orden:
$ servidor

⇒ 14. Abra otro terminal y cambie al directorio calcularRPC

⇒ 15. Ejecute rpcinfo para comprobar que el programa enlazador considera nuestro servidor
$ rpcinfo –p

⇒ 16. Ejecute el cliente mediante la siguiente orden:
$ cliente localhost

⇒ 17. Para terminar con el servidor se da en el terminal donde se ejecuta el servidor control-c

⇒ 18. Una vez terminado el servidor ejecute de nuevo el cliente para ver el mensaje mostrado.

⇒ 19. Vuelve a ejecutar el servidor y en otra máquina y con el mismo entorno prueba a ejecutar el cliente. La dirección que hay que indicar al cliente para que sepa dónde encontrar al servidor se puede obtener en la máquina del servidor con la orden:
$ ifconfig

⇒ 20. Copie el directorio calcularRPC en otro directorio al mismo nivel que éste en la estructura de directorios de nombre calcularRPCarg. En este otro directorio, modifica el programa cliente para que sume dos números dados por la línea de órdenes. Para convertir argv[2] que es una cadena de caracteres a un entero hay que utilizar la función atoi(argv[2]) y también para argv[3].

⇒ 21. Copie el directorio calcularRPCarg en otro directorio al mismo nivel que éste en la estructura de directorios de nombre calcularRPCoper. Modifica el cliente para que sume o reste dos números dados por la línea de órdenes depende de si se da s o r como último argumento (argv[4]). Para ello compara argv[4][0] con el carácter ‘s’ o ‘r’.

⇒ 22. Copie el directorio calcularRPCoper en otro directorio al mismo nivel que éste en la estructura de directorios de nombre calcularRPCcomp. Modifica la descripción del servidor para incluir el procedimiento multiplicar en la interfaz de la versión UNO que sirva para multiplicar dos números (ver NOTA). Modifica el servidor para incluir este nuevo procedimiento. Modifica el cliente para que llame también a este procedimiento en caso de indicárselo con el carácter ‘m’ en la línea de órdenes.

NOTA: En caso de usar la opción –a en rpcgen se generan ficheros adicionales, que nos indican los prototipos que deben tener los procedimientos en el servidor, y cómo se llaman desde el

cliente. Puede utilizar esta opción y ver lo que se genera de forma adicional a lo ya indicado anteriormente. Tenga cuidado ya que si vuelve a ejecutar rpcgen, se vuelven a generar estos ficheros y por lo tanto, no debe modificarlos, ya que las modificaciones se perderían.

⇒ 23. En la tarea habilitada para la práctica, entregue un fichero comprimido (.zip) con nombre su login que sea el directorio de nombre su login comprimido.

⇒ 24. Eche un vistazo a la RFC 4506. En el directorio correspondiente a esta práctica se ha incluido un resumen en español de esta RFC. En ella se pueden ver los tipos de los datos que se pueden utilizar en una especificación de un servidor. También se puede ver como se codifican estos tipos para la comunicación entre cliente y servidor.