ugr | Universidad de Granada

Desarrollo de Sistemas Distribuidos

Práctica 2:
SUN RPC

José Manuel Navarro Cuartero

Índice

Introducción

La independencia del transporte de RPC aísla a las aplicaciones de los elementos físicos y lógicos de los mecanismos de comunicaciones de datos y permite a la aplicación utilizar varios transportes. RPC permite a las aplicaciones de red utilizar llamadas a procedimientos que ocultan los detalles de los mecanismos de red subyacentes.
En esta práctica, partiendo de un archivo base llamado “calculadora.x”, en el que definimos las operaciones que va a realizar el servidor, generamos una serie de archivos con “rpcgen”. Posteriormente, modificamos dos de éstos, “calculadora_client.c” y “calculadora_server.c”, y generamos los dos ejecutables que se utilizarán.
Describiremos el proceso paso a paso.

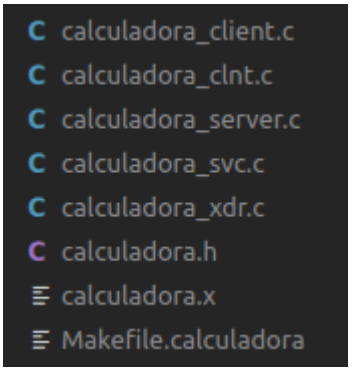
Calculadora.x

```
program CALCULADORA {
    version BASICA {
        double SUMA (double,double) = 1;
        double RESTA (double,double) = 2;
        double MULTIPLICACION (double,double) = 3;
        double DIVISION (double,double) = 4;
        int MODULO (int,int) = 5;
        double POTENCIA (double,int) = 6;
        double RAIZ (int, double) = 7;
        double LOG_NAT (double) = 8;
        double LOG10 (double) = 9;
        double SENO (double) = 10;
        double COSENO (double) = 11;
        double TANGENTE (double) = 12;
    }=2;
} = 0x20000002;
```

Como puede verse, hemos definido 12 operaciones distintas:

- Operaciones básicas: suma, resta, multiplicación y división.
- Operaciones con enteros: módulo, potencia y raíz.
- Operaciones logarítmicas: logaritmo natural y logaritmo en base 10.
- Operaciones trigonométricas: seno, coseno y tangente.

A partir de este archivo, con la orden “rpcgen -NCa” se generarán el resto de archivos necesarios.



Programación de las funciones

Hay 3 archivos que modificaremos, de los que se nos han generado:

Makefile

Para comodidad nuestra a la hora de trabajar, modificaremos el nombre del Makefile de “Makefile.calculadora” a simplemente “Makefile”.
Además, cambiaremos las órdenes para que tenga en cuenta la biblioteca de rpc y la de cmath (que usaremos más adelante).

```
CFLAGS += -g -I /usr/include/tirpc/ # Añadimos el path a la biblioteca RPC
LDLIBS += -lnsl -ltirpc -lm # Tambien añadimos la biblioteca de cmath
RPCGENFLAGS =
```

Server

El archivo del servidor, “calculadora_server.c” tendrá definidas las funciones que incluimos en el “calculadora.x” pero estarán vacías.

```
double * suma_2_svc(double arg1, double arg2,  struct svc_req *rqstp){
    static double  result;

    /*
     * insert server code here
     */

    return &result;
}
```

Será nuestro trabajo aquí incluir donde se nos indica el código para que cada función de el resultado que se busca.

```
double * suma_2_svc(double arg1, double arg2,  struct svc_req *rqstp){
    static double  result;
    result = arg1 + arg2;
    return &result;
}
```

Client

En el archivo del cliente, “calculadora_client.c” es donde se encontrará la mayor parte de nuestro trabajo. Distinguiremos varias partes del código.

main()

```
if (argc <= 3 || argc >= 6) {
    instrucciones(argv[0]);
    exit (1);
}
host = argv[1]; // localhost en nuestro sistema
```

En el main, primero se filtra si la llamada al cliente tiene el número de argumentos necesario (en nuestro caso, o 4 o 5). Si no, se sacan por pantalla las instrucciones de uso, y termina la ejecución. Si la llamada es correcta, se guarda cuál es el servidor.
Posteriormente, en función de si la operación es con 1 o 2 números, se identifica qué operación, se extraen los miembros y se llama a la función principal.

```
switch (argc){
    case 4: operacion = identifica(argv[2]);
    x = strtod(argv[3],NULL);
    printf("Operacion --> %s %f\n",argv[2],x);
    break;
    case 5: operacion = identifica(argv[3]);
    x = strtod(argv[2],NULL);
    y = strtod(argv[4],NULL);
    printf("Operacion -> %f %s %f\n",x,argv[3],
    break;
}
calculadora_2 (host,operacion,x,y);
```

instrucciones()

Sacamos por pantalla unas instrucciones básicas con las operaciones disponibles.

```
void instrucciones(char* prog){
    printf("Uso: %s localhost operacion\n",prog);
    printf("f = flotantes, i = enteros, g = grados\n");
    printf("Operaciones basicas: f +-x/ f\n");
    printf("Modulo: i mod i\n");
    printf("Potencia y raiz: f pow i , i root f\n");
    printf("Logaritmos: ln f , log10 f\n");
    printf("Geometricas: sen f , cos f , tan f\n");
}
```

identifica()

En esta función auxiliar transformaremos la operación que el usuario introduce por consola, que es una cadena de caracteres, a un entero que tendremos identificado como la operación que se desea hacer, de 1 a 12.

```
}else if(strcmp(operacion,"root") == 0){
    return 6;
}else if(strcmp(operacion,"ln") == 0){
    return 7;
}else if(strcmp(operacion,"log10") == 0){
    return 8;
```

Función principal

En esta función, “calculadora_2”, primero establecemos conexión con el servidor.

```
clnt = clnt_create (host, CALCULADORA, BASICA, "udp");
if (clnt == NULL) {
    clnt_pcreateerror (host);
    exit (1);
}
```

Posteriormente entramos en un switch que en función del entero que identifica a la operación, hace una llamada u otra, y devuelve el resultado por pantalla. Además, se hacen las comprobaciones necesarias para no hacer cálculos imposibles (como dividir por cero, la tangente de 90º, o el logaritmo de un número negativo).

```
case 7: if(x<=0){
    printf("Math error: Logaritmo negativo\n");
} else {
    result_double = log_nat_2(x, clnt);
    printf("logatirmo natural de %f = %f\n",x,*result_double);
} break;
```

Como se puede ver, si la operación es válida, se llama a la función apropiada del servidor, y se saca por pantalla el resultado.

Ejecutables

Haciendo make generaremos todos los archivos necesarios, en particular “calculadora_server” y “calculadora_client”, los ejecutables.

```
pepe@PepePC:~/Desktop/UGR-DSD/Practica 2/Entregar$ make
cc -g -I /usr/include/tirpc/    -c -o calculadora_clnt.o calculadora_clnt.c
cc -g -I /usr/include/tirpc/    -c -o calculadora_client.o calculadora_client.c
cc -g -I /usr/include/tirpc/    -c -o calculadora_xdr.o calculadora_xdr.c
cc -g -I /usr/include/tirpc/     -o calculadora_client  calculadora_clnt.o calcula
dora_client.o calculadora_xdr.o -lnsl -ltirpc -lm
cc -g -I /usr/include/tirpc/    -c -o calculadora_svc.o calculadora_svc.c
cc -g -I /usr/include/tirpc/    -c -o calculadora_server.o calculadora_server.c
cc -g -I /usr/include/tirpc/     -o calculadora_server  calculadora_svc.o calculad
ora_server.o calculadora_xdr.o -lnsl -ltirpc -lm
```

Uso

Por último, para utilizarlo en nuestro propio PC, primero ejecutaremos el servidor con “./calculadora_server”, que se mantendrá a la espera de llamadas. En otra terminal, haremos la llamada al cliente con la operación.

```
pepe@PepePC:~/Desktop/UGR-DSD/Practica 2/Entregar$ ./calculadora_client
localhost 8.3 pow 3
Operacion -> 8.300000 pow 3.000000
Resultado -> 8.300000 elevado a 3 = 571.787000
```

Si descargamos el paquete subido a Prado, antes de poder usarlo, simplemente habrá que hacer “make”. Ésto generará todos los archivos necesarios para el uso.