Agenda Eletrônica com RPC

Anderson Hiroshi Hamamoto
Natan de Almeida Laverde

1 Introdução ao RPC

O *RPC(Remote Procedure Call)*[1] define um protocolo para execução de procedimentos remotos em computadore ligados em rede[2]. O RPC é uma técnica muito poderosa para criar aplicações no estilo cliente-servidor distribuídas[3]. O RPC consegue deixar transparente pro programador a interação entre a parte cliente e a parte servidor do sistema, assim, o servidor e o cliente podem estar ou não na mesma máquina, compartilhando o mesmo espaço físico, ou podem estar conectados por meio de um servidor.

Um exemplo de aplicação do RPC é o NFS(Network File System)[4]. Esse sistema provedencia um método transparente para para acesso remoto de dados compartilhados sobre a rede.

2 Tutorial de RPC

Nesse breve tutorial será feito uma aplicação que usa o RPC e retorna a soma e a subtração entre dois números.

Primeiro deve ser feito um arquivo com a extensão .x, o qual será usado para gerar os stubs. Stub é a parte responsável do sistema pra fazer com que a comunicaçnao das partes funcionam corretamente, garantir que o cliente e o servidor consigam se comunicar. Nesse caso faremos um arquivo calcula.x.

Código 1: calcula.x

```
struct operandos {
      int x;
      int y;
};

program PROG {
       version VERSAO {
```

```
                int ADD(operandos) = 1;
                int SUB(operandos) = 2;
        } = 100;
} = 55555555;
```

A struct é usada para passar parâmetros entre o cliente e o servidor. O PROG e a VERSAO são usadas como identificadores pelo RPC e o ADD e o SUB são as funções que o cliente pode requisitar do servidor com um identificador.

Agora é necessário usar o "rpcgen" para gerar os arquivos necessários. No terminal coloque o comando "rpcgen -a calcula.x", isso vai gerar os arquivos:

- calcula.h - cabeçalho da interface das funções e da struct que são usadas no programa
- calcula_svc.c - stub do servidor
- calcula_clnt.c - stub do cliente
- calcula_client.c - programador que define o que acontece no lado do cliente
- calcula_server.c - programador define o que o que o servidor faz com a requisição do cliente
- calcula.xdr - filtros de representações de dados externos(eXternal Data Representation)[5]

Os arquivos que precisamos nos preocupar são o calcula_client.c e o calcula_server.c, que são o que programador implementa para fazer uso do RPC. No arquivo calcula_client.c é colocado tudo o que o programador quer que ocorra no lado do cliente e calcula_server.c é colocado o que o servidor executa.

No nosso exemplo tanto o servidor como cliente não tem muito o que ser feito.

Código 2: client_client.c

```
int add(CLIENT *clnt, int x, int y) {
  operandos ops;
  int *result;

  ops.x = x;
  ops.y = y;
```

```
  result = add_100(&ops, clnt);
  if (result == NULL) {
    fprintf(stderr, "Problema na chamada RPC\n");
    exit(0);
  }
  return (*result);
}

int sub(CLIENT *clnt, int x, int y) {
  operandos ops;
  int *result;

  ops.x = x;
  ops.y = y;

  result = sub_100(&ops, clnt);
  if (result == NULL) {
    fprintf(stderr, "Problema na chamada RPC\n");
    exit(0);
  }
  return (*result);
}

int main(int argc, char **argv) {
  CLIENT *clnt;
  int x, y;

  if (argc != 4) {
    fprintf(stderr, "Usagem errada\n");
    exit(0);
  }
  clnt = clnt_create(argv[1], PROG, VERSAO, "udp");
  if (clnt == (CLIENT *) NULL) {
    clnt_pcreateerror(argv[1]);
    exit(1);
  }
  x = atoi(argv[2]);
  y = atoi(argv[3]);
  printf("%d + %d = %d\n",x,y, add(clnt,x,y));
  printf("%d - %d = %d\n",x,y, sub(clnt,x,y));

  return 0;
```

```
}
```

Código 3: calcula_server.c

```
int * add_100_svc(operandos *argp, struct svc_req *rqstp) {
  static int result;

  printf("Requisicao de adicao para %d e %d\n", argp->x,
      argp->y);
  result = argp->x + argp->y;

  return(&result);
}

int * sub_100_svc(operandos *argp, struct svc_req *rqstp) {
  static int result;

  printf("Requisicao de subtracao para %d e %d\n", argp->x,
      argp->y);
  result = argp->x - argp->y;

  return(&result);
}
```

Para compilar o programa feito, o cliente e o servidor devem ser compilados separadamente:

- servidor: "gcc -o servidor calcula.h calcula_svc.c calcula_server.c calcula_xdr.c"
- cliente: "gcc -o cliente calcula.h calcula_clnt.c calcula_client.c calcula_xdr.c"

Depois basta executar o servidor, e a máquina que irá rodar o servidor deve estar com o portmapper ativado, no Linux e no Mac Os tem programa chamado "rpcbind" que também pode ser usado. Para executar o cliente tem que passar o *host*, que no caso é o servidor pra poder executar o programa. Para executar o cliente também é necessário passar como argumento para o programa os números que o usuário quer que faça a soma e a subtração.

3 Projeto da Agenda Eletrônica

A agenda eletrônica envolve o uso de um banco de dados para armazenar entradas de uma agenda. Essa agenda armazena nome, endereço e telefone.

O banco de dados que escolhemos usar para esse projeto é o MySQL Community Server[6]. Tem uma base de dados com o nome de agenda, dentro desse banco tem uma tabela com o nome de "agendarpc". Essa tabela contém os campos nome, endereço e telefone, em que o nome é a chave primária da tabela.

Código 4: Comando no MySQL usado para criar a tabela

```
CREATE TABLE agendarpc(
        name VARCHAR(100) PRIMARY KEY,
        addr VARCHAR(100),
        phone VARCHAR(100)
);
```

Código 5: Struct para organizar os dados

```
struct entry {
        char *name;
        char *addr;
        char *phone;
};
```

O nosso programa tem 5 funções básicas: inicializar, inserir, alterar, remover e consultar. A função inicializar deleta todas as entradas da tabela do banco de dados e inserir todas as entradas no arquivo "entrada.txt".

Código 6: Formato do arquivo de entrada para a função inicializar

```
nome 1

endereco 1

telefone 1

nome 2

endereco 2

telefone 2
```

A função inserir tem como objetivo inserir um conjunto de dados (nome, endereço e telefone). Para implementar essa função, a query é montada em uma string que então é passada para o banco de dados. A função de alterar, ela modifica os dados que estão na tabela do banco de dados. Remover retira

uma tupla da tabela do banco de dados. A função de consultar retorna uma das tuplas usando o nome para buscar.

Abaixo tem as funções de inserir, alterar e remover, antes de aplicá-los no RPC.

```
typedef struct entry entry;

struct entry {
  char *name;
  char *addr;
  char *phone;
};

void insert(MYSQL *conn, entry *data) {
  int n;
  char query[100];
  n = sprintf(query, "INSERT INTO teste VALUES('%s', '%s',
      '%s')", data->name, data->addr, data->phone);
  if (!mysql_query(conn, query))
              printf("%s\n", mysql_error(conn));
}

int alter(MYSQL *conn, entry *old, entry *new) {
  int n;
  char query[100];
  n = sprintf(query, "UPDATE teste SET name='%s',
      address='%s', phone='%s' WHERE name='%s'", new->name,
      new->addr, new->phone, old->name);
  printf("%s\n", query);
  if (!mysql_query(conn, query))
              printf("%s\n", mysql_error(conn));
}

int delete(MYSQL *conn, entry *data) {
  int n;
  char query[100];
  n = sprintf(query, "DELETE FROM teste WHERE name='%s'",
      data->name);
  if (!mysql_query(conn, query))
              printf("%s\n", mysql_error(conn));
}
```

Referências

[1] Sun Microsystems. RPC: Remote Procedure Call Protocol specification. RFC 1050 (Historic), April 1988. Obsoleted by RFC 1057.

[2] Luís Fernando Fortes Garcia. Rpc - remote prodedure call. http://penta2.ufrgs.br/rc952/trab1/rpc.html, 2013.

[3] Dave Marshall. Remote procedure calls(rpc). http://www.cs.cf.ac.uk/Dave/C/node33.html.

[4] B. Nowicki. NFS: Network File System Protocol specification. RFC 1094 (Informational), March 1989.

[5] Sun Microsystems. XDR: External Data Representation standard. RFC 1014, June 1987.

[6] Mysql community server. https://dev.mysql.com/downloads/mysql/, 2013.