**RPC实践记录**

1. 定义与特性
   1. 定义

RPC中文叫远程函数调用，它是一种通信方式，只是看起来像普通的函数调用。它包括基本的三个要素：

①服务端注册相应的（服务）函数。

②调用方通过调用的方式将一些信息和参数打包到消息，然后发送消息给 被调用方。

③被调用方收到消息后，提取信息和参数，调用相应函数。

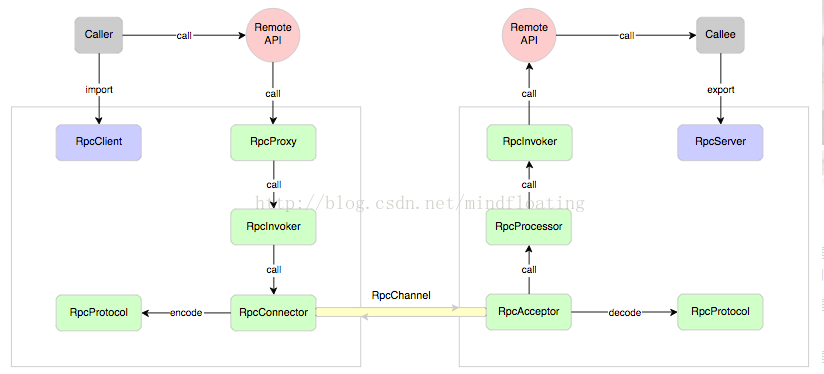
* 1. 特性

1.2.1.RPC数据的传输。使用原始tcp还是http，负责连接，属于传输层。

1.2.2.RPC消息。数据的传输，编解码，序列化与反序列化，xml,json,proto等。属于协议层。

1.2.3.RPC服务注册。服务端具体支持哪些调用，收到来自客户端的RPC请求后，怎样去调用真正的需要执行的方法，这些内容也是一个完整的RPC框架必须考虑的

1.2.4.RPC消息处理。这里其实与RPC本身关系不大，一般就是考虑支持异步/同步调用。

****

1. Protobuf简介和使用

2.1 Protobuf是什么？

Google Protocol Buffer(简称Protobuf)是一种轻便高效的结构化数据存储格式，平台无关，语言无关，可扩展，可用于通讯协议和数据存储等领域。

Protobuf可用于做序列化和反序列化。在将.proto文件编译成.h和.cpp文件以后，会提供很多操纵数据的函数，方便给消息类型的成员变量赋值和取值。

2.2 为什么要用？

平台无关，语言无关，可扩展

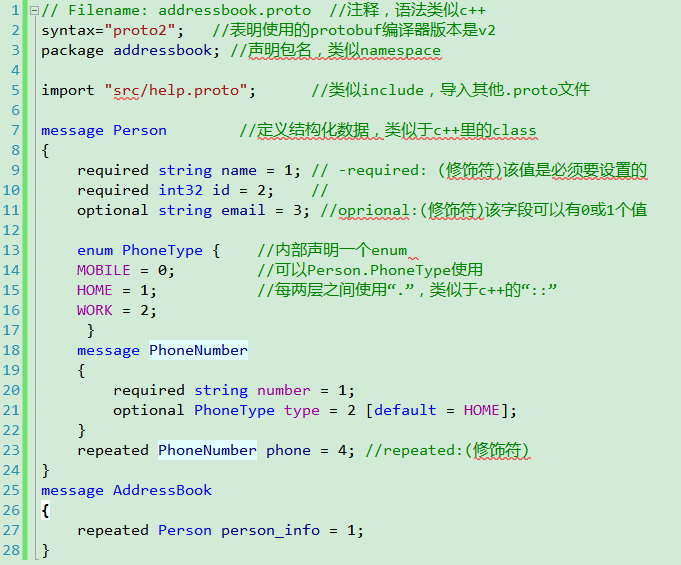
提供了友好的动态库，使用简单

解析速度快，比对应的xml快约20-100倍。

序列化数据非常简洁，紧凑，与xml相比，其序列化之后的数据量约为1/3 到1/10.

2.3 写proto文件

创建一个后缀为.proto的文件，并进行编辑，定义自己需要的结构类型，这样生成.h和.cpp以后，程序里便可以直接使用该类型，同时生成的.h提供了序列化和反序列化的函数，方便使用。



2.4 生成c++文件

Protoc 是proto文件的编译器，目前可以将proto文件编译成c++，Java，Python三种代码文件，编译格式：

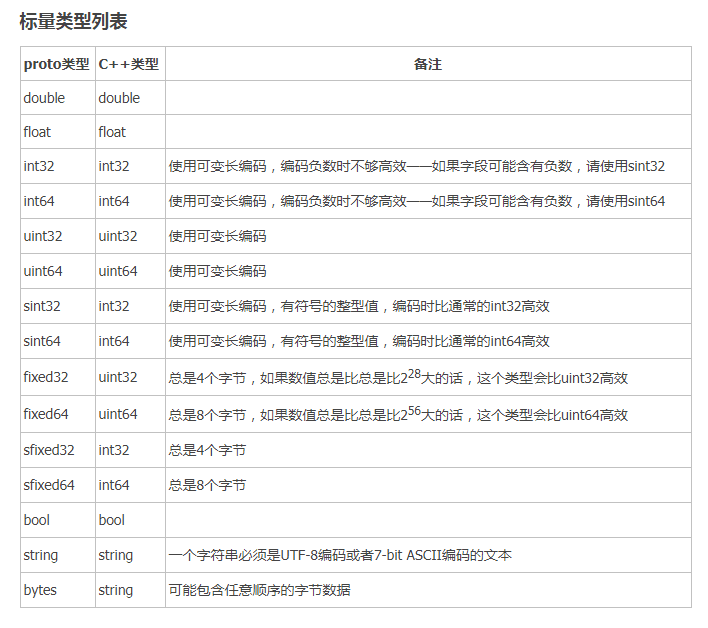
Protoc -I=$SRC\_DIR --cpp\_out=$DST\_DIR /path/to/file.proto

-I : 包含的.proto文件。

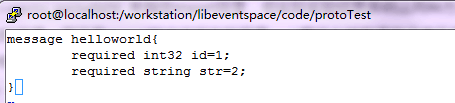
--cpp\_out: 指定生成.cpp和.h文件的路径。

/path/to/file.proto: 该编译文件完整路径。

2.5 常用数据类型



2.6 应用举例

2.6.1 编辑helloworld.proto文件。  


2.6.2 编译

X915T$VNO7{QY88OTMU`7CX4TI0DHVPB5J(4$KJR{3ABLX

2.6.3 编译可执行文件

5%}L~EE`5LM6ZG%7S${J}(9

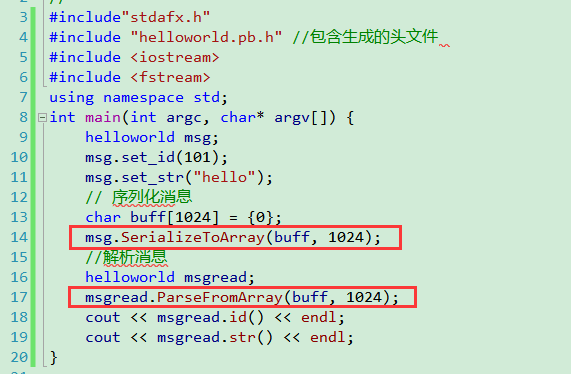
编译时，helloworld.pb.cc 和这两个库一定要带上。

2.6.3序列化和反序列化应用举例

第10，11 给消息类型变量msg赋值。

第14行，将消息类型序列化成字符数组。

第17行，将数组反序列化成消息变量，获取成员元素。



2.7 对比xml json。

在对象序列化的时候，xml，json也能做到将一个类存成字符数组的形式，将每一个元素存成一个节点，相比protobuf，可能效果差一点。

Json串的序列化，抓包数据

9]$MH]DT@N%XX}7$@3D5KU2

Xml序列化，抓包数据

E(`L@}@J~~IB9QOS)2DQ5]M

Protobuf序列化，抓包数据

N{8S1~AUB[HW90RFRCBT)F9

1. Libevent+protobuf

这便是很简单的远程调用，只不过没有区分客户端/客户代理，服务端/服务代理。

使用gsoap编译wsdl文档，形成接口文件，对接ipc设备也可以算是典型的远程调用。

1. 使用实例

4.1 Rpcgen 为linux 内置程序。在使用的时候分为3个步骤。

①rpc 规约文件

②客户端接口过程

③服务端接口过程

4.2 分布式程序的生成原理

RPC的实现包括一个工具，它自动地生成实现分布式程序所需要的大多数代码。（我理解的就是生成框架，留下接口供客户端调用）。这个工具叫rpcgen，它读取一个规约文件作为输入，生成C的源文件作为输出。（类型与proto文件，只不过定义的是接口）。规范为念包含常量，全局数据类型，全局数据以及远程过程（包括过程参数和结果类型）的声明。Rpcgen产生的代码包含了实现客户端和服务端程序所需的大部分源代码。具体地说，rpcgen包括参数整理，发送rpc报文，包传入调用分派到正确的过程，发送应答，在参数和结构的外部表示和本地数据之间进行转换。Rpcgen的输入与应用程序和程序员编写的少数文件相结合后，便产生了完整的客户和服务端程序。

Repgen 读取输入文件，该文件包括远程过程的说明。它产生四个输出文件，每个文件都包含有源代码，如果输入的文件（规约文件）具有名字q.x ,则输出文件如下所示：

q.h: 常量和类型的声明

q\_xdr.h : XDR过程调用

q\_clnt.c : 客户端的通信接口

q\_svc.c : 服务端的通信接口

同时也会生成q\_client.c : 对客户端接口的简单调用，可修改。

q\_server.c: 服务端对应接口具体做什么，必须修改。

4.3 .x文件

一个远程过程是要有三个要素来唯一确定：程序号，版本号，过程号。

程序号是用来区别一组相关的并且具有唯一过程号的远程过程。一个程序可以有一个或者几个不同的版本，而每个版本的过程都包含一系列能被远程过程调用的过程，通过版本的引进，使得不同版本下的RPC能同时提供服务。每个版本都包含许多可供远程调用的过程，每个过程则有其唯一标识的过程号。

4.3.1 挂载 （遇到一个问题error=112，加版本号解决了）

BE]875SMXY_[39%1BD$V@1T

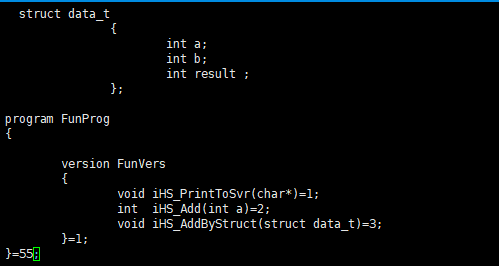
4.3.2 创建.x文件

.x文件里定义了一个结构体，还有一个程序集合。

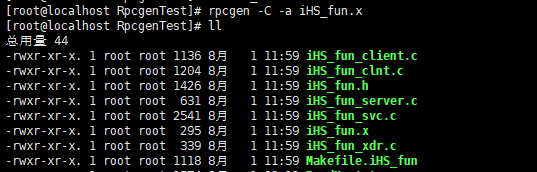
集合里有一个版本，版本里有3个函数：

iHS\_PrintToSvr: 在服务端输出一段文字。

iHS\_Add: 返回\*2后的结果。iHS\_AddByStruct 返回计算以后的结构体。



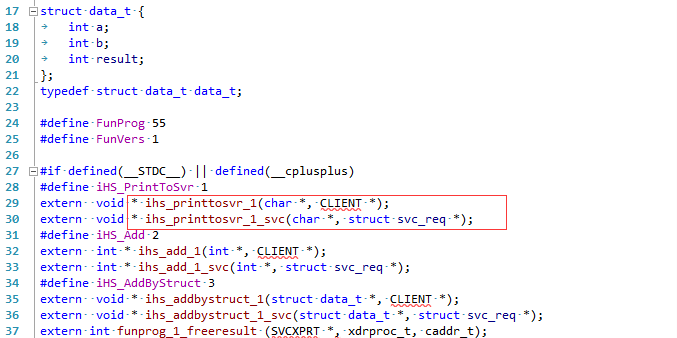
4.3.3 编译.x文件



可以看到生成了7个文件。

对这几个文件进行分析：

iHS\_fun.h :生成的c头文件，包括结构体和函数接口，函数接口都是成对的。一个供客户端调用，一个供服务端实现。



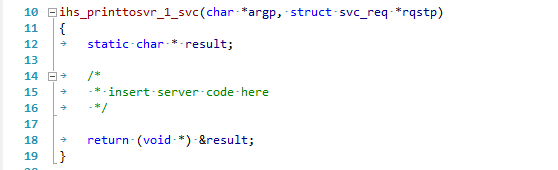
iHS\_fun\_clnt.c: 是向rpc服务调用对应服务端远程接口的过程，不需要修改。

iHS\_fun\_client.c: 是使用示例，可字节修改，调用想调用的接口。

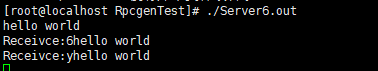
iHS\_fun\_svc.c 向rpc服务注册的过程，不需要修改。

iHS\_fun\_xdr.c :公用的结构体转换文件。

iHS\_fun\_server.c: 具体接口的实现，这里只生成了框架，具体需要自己实现。



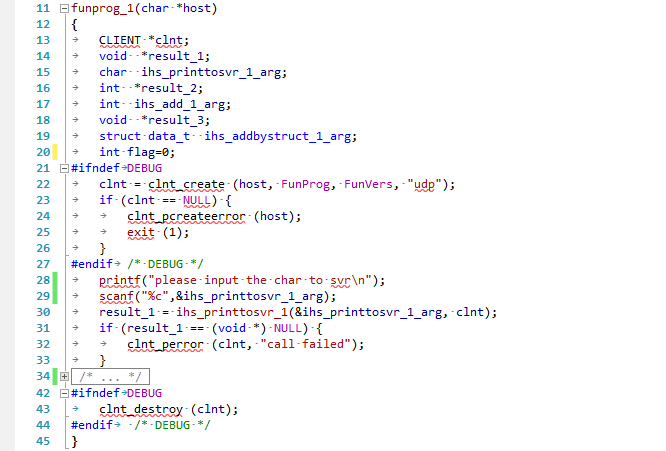
4.4 编译例程，运行。

JH]AK$MRM6VBAXLFV4Q{LO2

T[5@TZ91NG$2%9)2{(HUD9Y

~V_W5V`8]P~Q$5R`{GV{KNU

4.5 关于客户端调用的过程：



第22行，创建客户端对象。

第30行，调用ihs\_printtosvr\_1()函数。

第43行，销毁对象。