# Google\_Protobuf协议——protobuf代码中的通讯

[序列化和反序列化](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#序列化和反序列化)

* [计算序列化后的长度](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#计算序列化后的长度)
* [代码实现](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#代码实现)
  + [定义发送与接收基类以及函数](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#定义发送与接收基类以及函数)
  + [序列化到数组](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#序列化到数组)
  + [数组反序列化](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#数组反序列化)
  + [序列化到字符串](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#序列化到字符串)
  + [字符串反序列化](https://blog.csdn.net/zxng_work/article/details/78943167?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control#字符串反序列化)

在类google::protobuf::Message中有protobuf通讯的函数，将信息序列化和反序列化，在由proto文件生成的.cc和.h文件中，消息体生成由命名空间匡制，继承google::protobuf::Message的类。  
在生成的test.pb.h中

class TestSend : public ::google::protobuf::Message {

public:

TestSend();

virtual ~TestSend();

TestSend(const TestSend& from);

...

}

# 序列化和反序列化

序列化函数组

*//序列化到编码流*

bool SerializeToCodedStream(io::CodedOutputStream\* output) const;

*//序列化到字符串*

bool SerializeToString(string\* output) const;

*//序列化为字符串*

string SerializeAsString() const;

*//序列化到数组*

bool SerializeToArray(void\* data, int size) const;

反序列化数组

*//编码流反序列化*

bool ParseFromCodedStream(io::CodedInputStream\* input);

*//字符串反序列化*

bool ParseFromString(const string& data);

*//数组反序列化*

bool ParseFromArray(const void\* data, int size);

//stream stream

bool ParseFromIstream(istream\* input);

# 计算序列化后的长度

int ByteSize() const {

return internal::ToIntSize(ByteSizeLong());

}

# 代码实现

C数组的序列化和反序列化API

//C数组的序列化和序列化API

bool ParseFromArray(const void\* data, int size);

bool SerializeToArray(void\* data, int size) const;

//使用

void set\_people()

{

wp.set\_name("sealyao");

wp.set\_id(123456);

wp.set\_email("sealyaog@gmail.com");

wp.SerializeToArray(parray,256);

}

void get\_people()

{

rap.ParseFromArray(parray,256);

cout << "Get People from Array:" << endl;

cout << "\t Name : " <<rap.name() << endl;

cout << "\t Id : " << rap.id() << endl;

cout << "\t email : " << rap.email() << endl;

}

C++ String的序列化和反序列化API

//C++string序列化和序列化API

bool SerializeToString(string\* output) const;

bool ParseFromString(const string& data);

//使用：

void set\_people()

{

wp.set\_name("sealyao");

wp.set\_id(123456);

wp.set\_email("sealyaog@gmail.com");

wp.SerializeToString(&pstring);

}

void get\_people()

{

rsp.ParseFromString(pstring);

cout << "Get People from String:" << endl;

cout << "\t Name : " <<rsp.name() << endl;

cout << "\t Id : " << rsp.id() << endl;

cout << "\t email : " << rsp.email() << endl;

}

文件描述符序列化和反序列化API

//文件描述符的序列化和序列化API

bool SerializeToFileDescriptor(int file\_descriptor) const;

bool ParseFromFileDescriptor(int file\_descriptor);

//使用:

void set\_people()

{

fd = open(path,O\_CREAT|O\_TRUNC|O\_RDWR,0644);

if(fd <= 0){

perror("open");

exit(0);

}

wp.set\_name("sealyaog");

wp.set\_id(123456);

wp.set\_email("sealyaog@gmail.com");

wp.SerializeToFileDescriptor(fd);

close(fd);

}

void get\_people()

{

fd = open(path,O\_RDONLY);

if(fd <= 0){

perror("open");

exit(0);

}

rp.ParseFromFileDescriptor(fd);

std::cout << "Get People from FD:" << endl;

std::cout << "\t Name : " <<rp.name() << endl;

std::cout << "\t Id : " << rp.id() << endl;

std::cout << "\t email : " << rp.email() << endl;

close(fd);

}

C++  stream 序列化和反序列化API

//C++ stream 序列化/反序列化API

bool SerializeToOstream(ostream\* output) const;

bool ParseFromIstream(istream\* input);

//使用：

void set\_people()

{

fstream fs(path,ios::out|ios::trunc|ios::binary);

wp.set\_name("sealyaog");

wp.set\_id(123456);

wp.set\_email("sealyaog@gmail.com");

wp.SerializeToOstream(&fs);

fs.close();

fs.clear();

}

void get\_people()

{

fstream fs(path,ios::in|ios::binary);

rp.ParseFromIstream(&fs);

std::cout << "\t Name : " <<rp.name() << endl;

std::cout << "\t Id : " << rp.id() << endl;

std::cout << "\t email : " << rp.email() << endl;

fs.close();

fs.clear();

}