简单来讲， ProtoBuf 是结构数据序列化方法，可简单类比于 XML，其具有以下特点：

语言无关、平台无关。即 ProtoBuf 支持 Java、C++、Python 等多种语言，支持多个平台

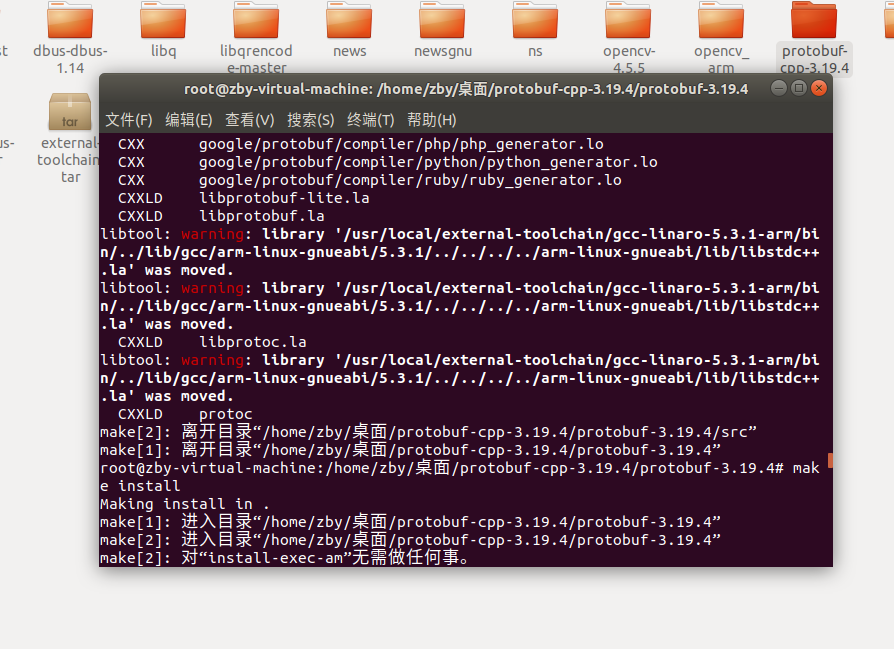
高效。即比 XML 更小（3 ~ 10倍）、更快（20 ~ 100倍）、更为简单。相比于Json，Protobuf有更高的转化效率，时间效率和空间效率都是JSON的3-5倍。

扩展性、兼容性好。你可以更新数据结构，而不影响和破坏原有的旧程序

序列化：将结构数据或对象转换成能够被存储和传输（例如网络传输）的格式，同时应当要保证这个序列化结果在之后（可能在另一个计算环境中）能够被重建回原来的结构数据或对象。

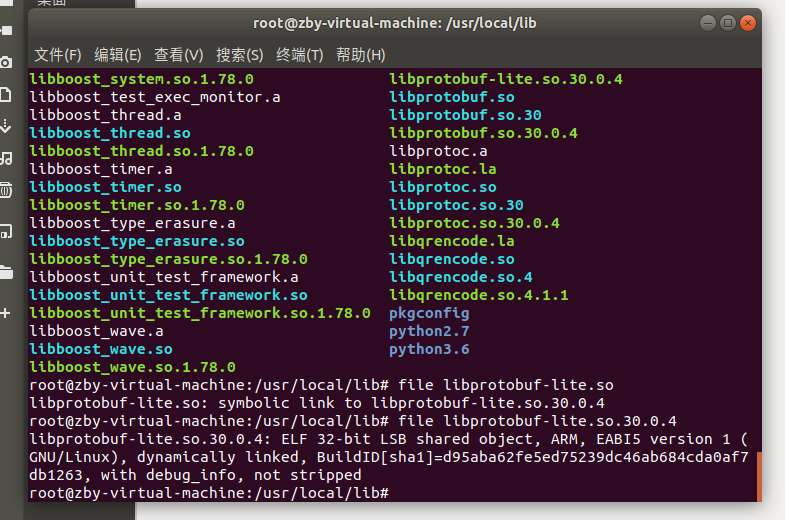
API封装。

Import Message 的用处主要在于提供了方便的代码管理机制，类似 C 语言中的头文件。您可以将一些公用的 Message 定义在一个 package 中，然后在别的 .proto 文件中引入该 package，进而使用其中的消息定义。



安装完成后，使用file命令验证

file xxxx.so



protobuf-lite 是 专门为嵌入式系统优化的runtime，牺牲了一些特性，换来了一些相对小的库以及不错的性能.

要使用protobuf-lite ，需要在所定义的proto文件中需要声明：

option optimize\_for = LITE\_RUNTIME;

使用arm-linux-c++编译运行

使用Gcc（默认）重新编译一遍protoc

生成Bin中的protoc（用来生成proto编译出的.cc和.h文件）

/home/protobuf/ubuntu/bin/protoc -I=./ --cpp\_out=./ ./eg.proto

<https://blog.csdn.net/orange_littlegirl/article/details/98478537>

编译proto：/home/protobuf/ubuntu/bin/protoc -I=./ --cpp\_out=./ ./addressbook.proto

编译.cc文件：/usr/local/external-toolchain/gcc-linaro-5.3.1-arm/bin/arm-linux-gnueabi-c++ -std=c++11 -g -o main main.cc addressbook.pb.cc -I. -I/usr/local/include/ -L/usr/local/lib/ -lprotobuf-lite -pthread

usr/local/external-toolchain/gcc-linaro-5.3.1-arm/bin/arm-linux-gnueabi-c++ -std=c++11 -g -o testjson test\_json.o test.pb.o json2pb.o -I. -I/usr/local/include/ -L/usr/local/lib/ -lprotobuf -pthread -lcurl -ljansson

strip命令：/usr/local/external-toolchain/gcc-linaro-5.3.1-arm/bin/arm-linux-gnueabi-strip libprotobuf-lite.so.30

消息内容赋值：<https://herbort.blog.csdn.net/article/details/115474254>

