**一、安装Protobuf编译器**

首先按照github上的顺序进行，

**brew install** automake

**brew install** libtool

**brew install** protobuf

如果直接按照github上的步骤，会出现很多error，没有走下去。

所以记下了自己实践成功步骤：

1、git clone <https://github.com/alexeyxo/protobuf-objc.git>

localhost:~ clf$ git clone https:*//github.com/alexeyxo/protobuf-objc.git*

Cloning into 'protobuf-objc'...

remote: Counting objects: 2896, done.

remote: Total 2896 (delta 0), reused 0 (delta 0), pack-reused 2895

Receiving objects: 100% (2896/2896), 33.48 MiB | 2.91 MiB/s, done.

Resolving deltas: 100% (1834/1834), done.

2、cd ~/protobuf-objc

localhost:~ clf$ **cd** ~/protobuf-objc

3、./autogen.sh

localhost:protobuf-objc clf$ ./autogen.sh

glibtoolize: putting auxiliary files **in** *'.'.*

glibtoolize: copying file *'./ltmain.sh'*

glibtoolize: putting macros **in** AC\_CONFIG\_MACRO\_DIRS, *'m4'.*

glibtoolize: copying file *'m4/libtool.m4'*

glibtoolize: copying file *'m4/ltoptions.m4'*

glibtoolize: copying file *'m4/ltsugar.m4'*

glibtoolize: copying file *'m4/ltversion.m4'*

glibtoolize: copying file *'m4/lt~obsolete.m4'*

configure.ac:13: installing *'./compile'*

configure.ac:9: installing *'./config.guess'*

configure.ac:9: installing *'./config.sub'*

configure.ac:10: installing *'./install-sh'*

configure.ac:10: installing *'./missing'*

src/compiler/Makefile.am:5: warning: source file *'google/protobuf/objectivec-descriptor.pb.cc' is in a subdirectory,*

src/compiler/Makefile.am:5: but **option** *'subdir-objects' is disabled*

automake: warning: possible forward-incompatibility.

automake: At least a source file **is** **in** a subdirectory, but the *'subdir-objects'*

automake: automake **option** hasn*'t been enabled. For now, the corresponding output*

automake: object file(s) will be placed **in** the top-level directory. However,

automake: this behaviour will change **in** future Automake versions: they will

automake: unconditionally cause object files **to** be placed **in** the same subdirectory

automake: **of** the corresponding sources.

automake: You are advised **to** start **using** *'subdir-objects' option throughout your*

automake: project, **to** avoid future incompatibilities.

src/compiler/Makefile.am: installing *'./depcomp'*

4、./configure

localhost:protobuf-objc clf$ ./configure

checking build system type... x86\_64-apple-darwin16.7.0

checking host system type... x86\_64-apple-darwin16.7.0

checking target system type... x86\_64-apple-darwin16.7.0

checking **for** a BSD-compatible install... /usr/local/bin/ginstall -c

checking whether build environment is sane... yes

checking **for** a thread-safe mkdir -p... /usr/local/bin/gmkdir -p

checking **for** gawk... no

checking **for** mawk... no

checking **for** nawk... no

checking **for** awk... awk

checking whether make sets $(MAKE)... yes

checking whether make supports nested variables... yes

checking **for** gcc... gcc

checking whether the C compiler works... yes

checking **for** C compiler default output file name... a.out

checking **for** suffix of executables...

checking whether we are cross compiling... no

checking **for** suffix of object files... o

checking whether we are using the GNU C compiler... yes

checking whether gcc accepts -g... yes

checking **for** gcc option **to** accept ISO C89... none needed

checking whether gcc understands -c **and** -o together... yes

checking whether make supports the include directive... yes (GNU style)

checking dependency style of gcc... gcc3

checking **for** g++... g++

checking whether we are using the GNU C++ compiler... yes

checking whether g++ accepts -g... yes

checking dependency style of g++... gcc3

checking C++ compiler flags...... use default: -g -O2 -DNDEBUG

checking how **to** print strings... printf

checking **for** a sed that does **not** truncate output... /usr/bin/sed

checking **for** grep that handles long lines **and** -e... /usr/bin/grep

checking **for** egrep... /usr/bin/grep -E

checking **for** fgrep... /usr/bin/grep -F

checking **for** ld used by gcc... /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld

checking **if** the linker (/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld) is GNU ld... no

checking **for** BSD- **or** MS-compatible name lister (nm)... /usr/bin/nm -B

checking the name lister (/usr/bin/nm -B) interface... BSD nm

checking whether ln -s works... yes

checking the maximum length of command line arguments... 196608

checking how **to** convert x86\_64-apple-darwin16.7.0 file names **to** x86\_64-apple-darwin16.7.0 format... func\_convert\_file\_noop

checking how **to** convert x86\_64-apple-darwin16.7.0 file names **to** toolchain format... func\_convert\_file\_noop

checking **for** /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld option **to** reload object files... -r

checking **for** objdump... objdump

checking how **to** recognize dependent libraries... pass\_all

checking **for** dlltool... no

checking how **to** associate runtime **and** link libraries... printf %s\n

checking **for** ar... ar

checking **for** archiver @FILE support... no

checking **for** strip... strip

checking **for** ranlib... ranlib

checking command **to** parse /usr/bin/nm -B output **from** gcc object... ok

checking **for** sysroot... no

checking **for** a working dd... /bin/dd

checking how **to** truncate binary pipes... /bin/dd bs=4096 count=1

checking **for** mt... no

checking **if** : is a manifest tool... no

checking **for** dsymutil... dsymutil

checking **for** nmedit... nmedit

checking **for** lipo... lipo

checking **for** otool... otool

checking **for** otool64... no

checking **for** -single\_module linker flag... yes

checking **for** -exported\_symbols\_list linker flag... yes

checking **for** -force\_load linker flag... yes

checking how **to** run the C preprocessor... gcc -E

checking **for** ANSI C header files... yes

checking **for** sys/types.h... yes

checking **for** sys/stat.h... yes

checking **for** stdlib.h... yes

checking **for** string.h... yes

checking **for** memory.h... yes

checking **for** strings.h... yes

checking **for** inttypes.h... yes

checking **for** stdint.h... yes

checking **for** unistd.h... yes

checking **for** dlfcn.h... yes

checking **for** objdir... .libs

checking **if** gcc supports -fno-rtti -fno-exceptions... yes

checking **for** gcc option **to** produce PIC... -fno-common -DPIC

checking **if** gcc PIC flag -fno-common -DPIC works... yes

checking **if** gcc static flag -static works... no

checking **if** gcc supports -c -o file.o... yes

checking **if** gcc supports -c -o file.o... (cached) yes

checking whether the gcc linker (/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld) supports shared libraries... yes

checking dynamic linker characteristics... darwin16.7.0 dyld

checking how **to** hardcode library paths into programs... immediate

checking whether stripping libraries is possible... yes

checking **if** libtool supports shared libraries... yes

checking whether **to** build shared libraries... yes

checking whether **to** build static libraries... yes

checking how **to** run the C++ preprocessor... g++ -E

checking **for** ld used by g++... /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld

checking **if** the linker (/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld) is GNU ld... no

checking whether the g++ linker (/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld) supports shared libraries... yes

checking **for** g++ option **to** produce PIC... -fno-common -DPIC

checking **if** g++ PIC flag -fno-common -DPIC works... yes

checking **if** g++ static flag -static works... no

checking **if** g++ supports -c -o file.o... yes

checking **if** g++ supports -c -o file.o... (cached) yes

checking whether the g++ linker (/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/ld) supports shared libraries... yes

checking dynamic linker characteristics... darwin16.7.0 dyld

checking how **to** hardcode library paths into programs... immediate

checking **for** ANSI C header files... (cached) yes

checking fcntl.h usability... yes

checking fcntl.h presence... yes

checking **for** fcntl.h... yes

checking **for** inttypes.h... (cached) yes

checking limits.h usability... yes

checking limits.h presence... yes

checking **for** limits.h... yes

checking **for** stdlib.h... (cached) yes

checking **for** unistd.h... (cached) yes

checking **for** working memcmp... yes

checking **for** working strtod... yes

checking **for** ftruncate... yes

checking **for** memset... yes

checking **for** mkdir... yes

checking **for** strchr... yes

checking **for** strerror... yes

checking **for** strtol... yes

checking google/protobuf/stubs/common.h usability... yes

checking google/protobuf/stubs/common.h presence... yes

checking **for** google/protobuf/stubs/common.h... yes

checking that generated files are newer than configure... done

configure: creating ./config.status

config.status: creating Makefile

config.status: creating src/compiler/Makefile

config.status: creating config.h

config.status: executing depfiles commands

config.status: executing libtool commands

5、make

localhost:protobuf-objc clf$ make

/Applications/Xcode.app/Contents/Developer/usr/bin/make all-recursive

Making all in src/compiler

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT main.o -MD -MP -MF .deps/main.Tpo -c -o main.o main.cc

mv -f .deps/main.Tpo .deps/main.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_enum\_field.o -MD -MP -MF .deps/objc\_enum\_field.Tpo -c -o objc\_enum\_field.o objc\_enum\_field.cc

mv -f .deps/objc\_enum\_field.Tpo .deps/objc\_enum\_field.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_file.o -MD -MP -MF .deps/objc\_file.Tpo -c -o objc\_file.o objc\_file.cc

mv -f .deps/objc\_file.Tpo .deps/objc\_file.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_message\_field.o -MD -MP -MF .deps/objc\_message\_field.Tpo -c -o objc\_message\_field.o objc\_message\_field.cc

mv -f .deps/objc\_message\_field.Tpo .deps/objc\_message\_field.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_enum.o -MD -MP -MF .deps/objc\_enum.Tpo -c -o objc\_enum.o objc\_enum.cc

mv -f .deps/objc\_enum.Tpo .deps/objc\_enum.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_generator.o -MD -MP -MF .deps/objc\_generator.Tpo -c -o objc\_generator.o objc\_generator.cc

mv -f .deps/objc\_generator.Tpo .deps/objc\_generator.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_primitive\_field.o -MD -MP -MF .deps/objc\_primitive\_field.Tpo -c -o objc\_primitive\_field.o objc\_primitive\_field.cc

mv -f .deps/objc\_primitive\_field.Tpo .deps/objc\_primitive\_field.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_extension.o -MD -MP -MF .deps/objc\_extension.Tpo -c -o objc\_extension.o objc\_extension.cc

mv -f .deps/objc\_extension.Tpo .deps/objc\_extension.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_helpers.o -MD -MP -MF .deps/objc\_helpers.Tpo -c -o objc\_helpers.o objc\_helpers.cc

mv -f .deps/objc\_helpers.Tpo .deps/objc\_helpers.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_field.o -MD -MP -MF .deps/objc\_field.Tpo -c -o objc\_field.o objc\_field.cc

mv -f .deps/objc\_field.Tpo .deps/objc\_field.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objc\_message.o -MD -MP -MF .deps/objc\_message.Tpo -c -o objc\_message.o objc\_message.cc

mv -f .deps/objc\_message.Tpo .deps/objc\_message.Po

g++ -DHAVE\_CONFIG\_H -I. -I../.. -g -O2 -DNDEBUG -MT objectivec-descriptor.pb.o -MD -MP -MF .deps/objectivec-descriptor.pb.Tpo -c -o objectivec-descriptor.pb.o `test -f 'google/protobuf/objectivec-descriptor.pb.cc' || **echo** './'`google/protobuf/objectivec-descriptor.pb.cc

mv -f .deps/objectivec-descriptor.pb.Tpo .deps/objectivec-descriptor.pb.Po

/bin/sh ../../libtool --tag=CXX --mode=link g++ -g -O2 -DNDEBUG -lprotobuf -lprotoc -o protoc-gen-objc main.o objc\_enum\_field.o objc\_file.o objc\_message\_field.o objc\_enum.o objc\_generator.o objc\_primitive\_field.o objc\_extension.o objc\_helpers.o objc\_field.o objc\_message.o objectivec-descriptor.pb.o

libtool: link: g++ -g -O2 -DNDEBUG -o protoc-gen-objc main.o objc\_enum\_field.o objc\_file.o objc\_message\_field.o objc\_enum.o objc\_generator.o objc\_primitive\_field.o objc\_extension.o objc\_helpers.o objc\_field.o objc\_message.o objectivec-descriptor.pb.o -Wl,-bind\_at\_load /usr/local/lib/libprotoc.dylib /usr/local/lib/libprotobuf.dylib -lz

make[2]: Nothing to be done for `all-am'.

6、make install

localhost:protobuf-objc clf$ make **install**

Making **install** **in** src/compiler

/usr/**local**/**bin**/gmkdir -p '/usr/local/bin'

/**bin**/sh ../../libtool *--mode=install /usr/local/bin/ginstall -c protoc-gen-objc '/usr/local/bin'*

libtool: **install**: /usr/**local**/**bin**/ginstall -c protoc-gen-objc /usr/**local**/**bin**/protoc-gen-objc

make[2]: **Nothing** **to** be done **for** `install-data-am'.

make[2]: Nothing to be done for `**install**-exec-am'.

make[2]: Nothing to be done for `install-data-am'.

localhost:protobuf-objc clf$

完成后，在/usr/local/bin/目录下的protoc-gen-objc就是我们需要的编译器。

**二、创建proto文件**

在桌面创建一个文件夹为ProtoBuf

实际工作中，proto文件都是后台提供给前端的，前端以此编译成.h和.m文件导入到工程中。

cd ~/Desktop/ProtoBuf

touch **person**.proto

vi **person**.proto

示例：一个message就是一个整体，里面有哪些必要的内容，哪些可选的内容。

**message** **LoginRequest**

{

**optional** bytes type =1;

**required** bytes username=2;

**required** bytes password =3;

}

**message** **LoginResponse**

{

**required** int32 code = 1;

**optional** bytes token = 2;

}

生成OC的.h和.m文件

localhost:~ clf$ cd /Users/clf/ProtoBuf

localhost:ProtoBuf clf$ protoc --plugin=/usr/local/bin/protoc-gen-objc person.proto --objc\_out="./"

localhost:ProtoBuf clf$

然后在当前目录下就生成了Person.pb.m和Person.pb.h文件。

**三、工程中使用Protobuf**

1、创建ProtoBuf工程

localhost:~ clf$ cd ~/Desktop/ProtoBuf

2、创建Podfile

localhost:~ clf$ touch Podfile

输入

**platform** :ios, ‘8.0’

use\_frameworks!

target ‘ProtoBuf’ **do**

pod 'ProtocolBuffers', '~> 1.9.11'

**end**

3、

pod **install**

4、将刚刚生成的.h和.m文件导入工程中，在工程中导入文件

#**import** <ProtocolBuffers/ProtocolBuffers.h>

#**import** "Person.pb.h"

就可以开始你的项目代码了。

+ (**void**)post

{

NSURL \*url = [NSURL URLWithString:@"http://www.baidu.com"];

NSMutableURLRequest \*request = [NSMutableURLRequest requestWithURL:url];

request.timeoutInterval = 0.5;

request.HTTPMethod = @"POST";

*// 设置请求体*

LoginRequestBuilder \*builder = [LoginRequest builder];

*// PdClickInfoBuilder \*builder = [PdClickInfo builder];*

NSString \*type = [NSString stringWithFormat:@"1"];

NSString \*nameStr = [NSString stringWithFormat:@"13714181323"];

NSString \*pwdStr = [NSString stringWithFormat:@"111111"];

builder.type = [type dataUsingEncoding:NSUTF8StringEncoding];

builder.username = [nameStr dataUsingEncoding:NSUTF8StringEncoding];

builder.password = [pwdStr dataUsingEncoding:NSUTF8StringEncoding];

LoginRequest \*req = [builder build];

NSLog(@"data = %@",[req data]);

request.HTTPBody = [req data];

NSURLSession \*session = [NSURLSession sessionWithConfiguration:[NSURLSessionConfiguration defaultSessionConfiguration]];

NSURLSessionDataTask \*task = [session dataTaskWithRequest:request completionHandler:^(NSData \* \_Nullable data, NSURLResponse \* \_Nullable response, NSError \* \_Nullable error) {

**if** (error){

NSLog(@"error = %@",error);

}

**else**{

*// NSLog(@"------ data = %@",data);*

NSLog(@"success");

*// 返回的数据*

NSString \*result = [[NSString alloc]initWithData:data encoding:NSUTF8StringEncoding];

NSLog(@"%@",result);

}

}];

[task resume];

}

感觉自己不是特别理解，又折腾了些[参看博客](http://www.cnblogs.com/tandaxia/p/6181534.html)

这里我直接拿博客中的.proto文件进行尝试

进行编译的时候出现了些问题

wifi:ProtoBuf clf$ vi myClass.proto

wifi:ProtoBuf clf$ protoc --plugin=/usr/local/bin/protoc-gen-objc myClass.proto --objc\_out="./"

myClass.proto:1:10: Unrecognized syntax identifier "proto3". This parser only recognizes "proto2".

wifi:ProtoBuf clf$ protoc --plugin=/usr/local/bin/protoc-gen-objc myClass.proto --objc\_out="./"

myClass.proto:4:9: Expected "required", "optional", or "repeated".

myClass.proto:5:9: Expected "required", "optional", or "repeated".

myClass.proto:13:9: Expected "required", "optional", or "repeated".

myClass.proto:16:17: Expected "required", "optional", or "repeated".

myClass.proto:17:17: Expected "required", "optional", or "repeated".

myClass.proto:26:9: Expected "required", "optional", or "repeated".

myClass.proto:27:9: Expected "required", "optional", or "repeated".

myClass.proto:28:9: Expected "required", "optional", or "repeated".

wifi:ProtoBuf clf$ vim class.proto

wifi:ProtoBuf clf$ protoc --plugin=/usr/local/bin/protoc-gen-objc class.proto --objc\_out="./"

修改之后

syntax = "proto2";

**message** **Person**

{

**optional** string name = 1;

**optional** int32 age = 2;

**enum** **DeviceType**{

IOS = 0;

Android = 1;

WP = 2;

}

**optional** DeviceType deviceType = 3; *//设备类型*

**message** **Result**{

**optional** string url = 1;

**optional** string title = 2;

}

**repeated** Result results = 4;

**repeated** Animal animals = 5;

}

**message** **Animal**{

**optional** float weight = 1;

**optional** double price = 2;

**optional** string namme = 3;

}

发现proto2和proto3生成的.pb.h和.pb.m文件差别好大，参考的博客中生成的代码方法完全不一样。不过proto2也就类似于上面写的post方法了。

这里是[proto2语法](https://www.cnblogs.com/resentment/p/6539021.html)

备注：

也可以安装别的版本的ProtocolBuffer

[Protobuf2.6集成到ios平台](https://www.jianshu.com/p/cbbb0bfd0bb6)

[ProtocolBuffer 历史版本](https://github.com/google/protobuf/releases)

[ProtocolBuffer官网](https://developers.google.com/protocol-buffers/docs/proto3)

参考博客：

<http://blog.csdn.net/samdy2008/article/details/52139047>

<https://www.2cto.com/kf/201503/382440.html>