**[cocos2dx 3.x 集成protobuf](https://www.cnblogs.com/chevin/p/6001872.html)**

vs2013+cocos2dx 3.13.1

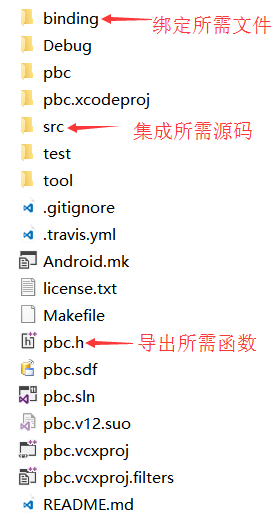
这篇博文是集成Lua版本的protobuf，集成C++版本的过程也可参考。

主要参考博文地址《[cocos2dx 3.x C++搭建protobuf环境](http://blog.csdn.net/ganpengjin1/article/details/50964961)》，这里的集成方式和参考博文有点不一样（这里是直接集成到引擎中，参考博文是当成一个库放到项目中），请仔细甄别。

1.下载pbc

①下载地址 pbc：https://github.com/cloudwu/pbc

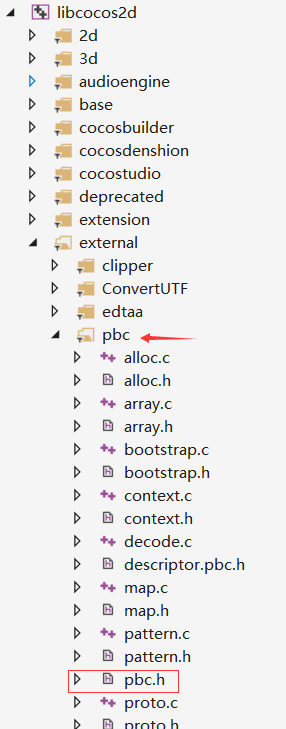
②解压后的文件列表，我们所需要的就是如下图的三个文件(夹)里的文件。（其中binding文件夹中包含了lua和lua53两个文件夹，因为cocos2dx集成的lua版本是5.1，所以我们使用lua文件夹，而不用lua53文件夹）



③其实就分两部分，一部分是C++部分(pbc.h和src文件夹)，一部分就是C++绑定到lua部分(binding文件夹中lua文件夹)

2.将C++部分的代码集成到libcocos2d项目中

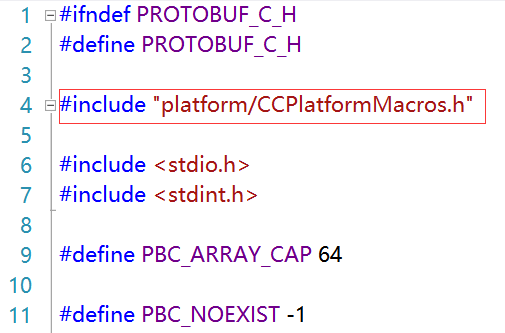
①在libcocos2d项目external文件夹中增加新的pbc文件夹，将pbc.h文件和src文件夹中的文件拷贝到里面去，并添加到项目中



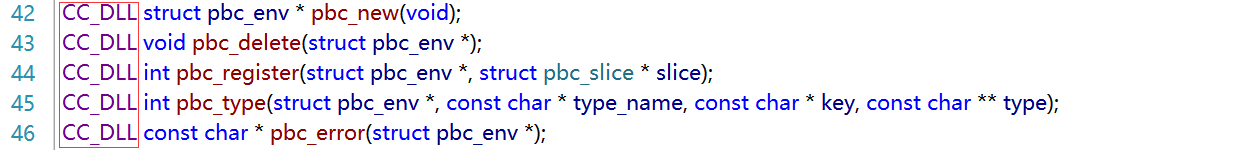
②修改pbc.h文件

因为libcocos2d项目是一个dll项目，所以需要导出必要的函数让libluacocos2d项目进行调用，导出的函数前需要增加CC\_DLL前缀。

首先include相应头文件，因为CC\_DLL的定义在CCPlatformMacros.h中



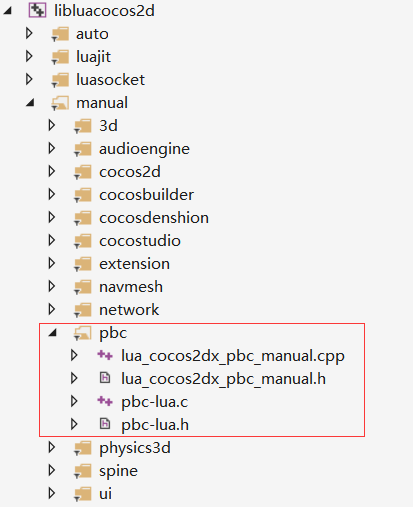
然后给所有（所有！）函数前增加CC\_DLL前缀，如下图只罗列了其中几个



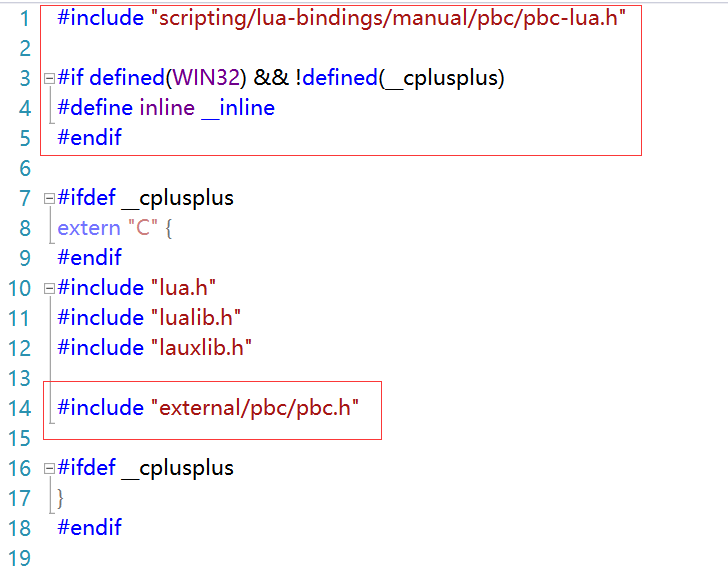
③编译libcocos2d项目

3.将C++绑定到lua部分集成到libluacocos2d项目中

①在libluacocos2d项目中manual文件夹中增加新的pbc文件夹，并将pbc-lua.c文件拷贝（从bunding/lua中拷贝）进去，并新增（新增！）三个用于绑定的文件pbc-lua.h/lua\_cocos2dx\_pbc\_manual.h/lua\_cocos2dx\_pbc\_manual.cpp。如下图



②修改pbc-lua.c文件（只修改前面部分），入下图



③pbc-lua.h内容

#pragma once

#ifdef \_\_cplusplus

extern "C" {

#endif

#include "lua.h"

#include "lualib.h"

#include "lauxlib.h"

#ifdef \_\_cplusplus

}

#endif

#ifdef \_\_cplusplus

extern "C" {

#endif

int luaopen\_protobuf\_c(lua\_State \*L);

#ifdef \_\_cplusplus

}

#endif

④lua\_cocos2dx\_pbc\_manual.h内容

#pragma once

#ifdef \_\_cplusplus

extern "C" {

#endif

#include "tolua++.h"

#ifdef \_\_cplusplus

}

#endif

TOLUA\_API int register\_pbc\_module(lua\_State\* L);

⑤lua\_cocos2dx\_pbc\_manual.cpp内容

#include "scripting/lua-bindings/manual/pbc/lua\_cocos2dx\_pbc\_manual.h"

#include "platform/CCPlatformConfig.h"

#include "base/ccConfig.h"

#include "scripting/lua-bindings/manual/tolua\_fix.h"

#include "scripting/lua-bindings/manual/LuaBasicConversions.h"

#include "scripting/lua-bindings/manual/CCLuaEngine.h"

#include "scripting/lua-bindings/manual/pbc/pbc-lua.h"

#include "cocos/platform/CCFileUtils.h"

int read\_protobuf\_file(lua\_State \*L){

const char \*buff = luaL\_checkstring(L, -1);

Data data = cocos2d::FileUtils::getInstance()->getDataFromFile(buff);

lua\_pushlstring(L, (const char\*)data.getBytes(), data.getSize());

return 1;

}

TOLUA\_API int register\_pbc\_module(lua\_State\* L)

{

lua\_getglobal(L, "\_G");

if (lua\_istable(L, -1))//stack:...,\_G,

{

lua\_register(L, "read\_protobuf\_file\_c", read\_protobuf\_file);

luaopen\_protobuf\_c(L);

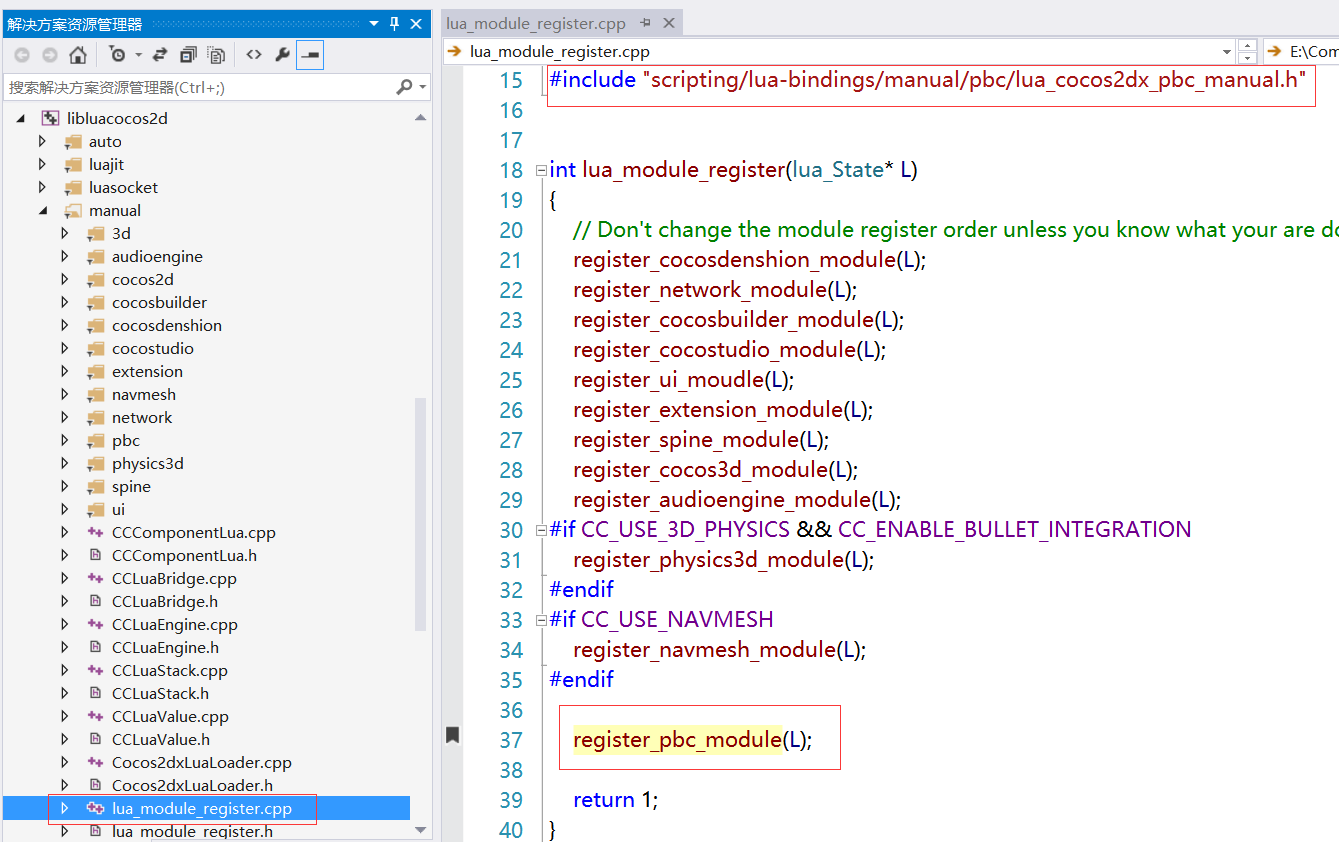
}

lua\_pop(L, 1);

return 1;

}

⑥在cocos2d-x\cocos\scripting\lua-bindings\manual\lua\_module\_register.cpp中进行注册

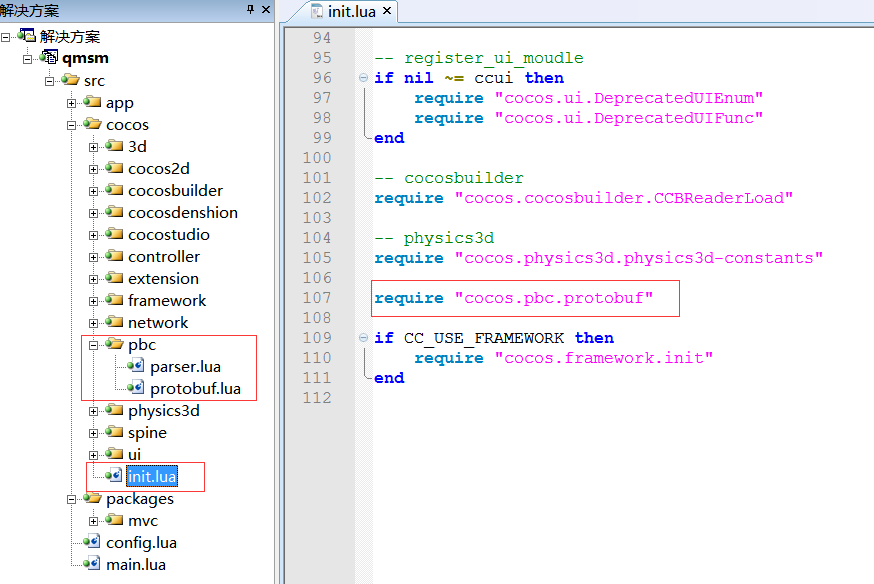


⑦编译libluacocos2d项目

以上，lua的在项目中的集成就完成了。下面来进行lua项目的验证。

4.在lua项目中的使用

①在bunding/lua中拷贝两个文件parser.lua/protobuf.lua到lua项目src中，然后在init.lua中进行require。



②定义MsgProtocol.proto文件

message Person {

required string name = 1;

required int32 id = 2;

optional string email = 3;

enum PhoneType {

MOBILE = 0;

HOME = 1;

WORK = 2;

}

message PhoneNumber {

required string number = 1;

optional PhoneType type = 2 [default = HOME];

}

repeated PhoneNumber phone = 4;

}

message AddressBook {

repeated Person person = 1;

}

③使用protoc.exe（这个工具需要到protobuf官方网站下载或者编译得到 官网:<https://github.com/google/protobuf/releases>）生成MsgProtocol.pb文件

protoc.exe --descriptor\_set\_out=./MsgProtocol.pb ./MsgProtocol.proto

④将生成的MsgProtocol.pb放到资源res文件夹中！

⑤相关验证使用的Lua代码

local pbFilePath = cc.FileUtils:getInstance():fullPathForFilename("MsgProtocol.pb")

release\_print("PB file path: "..pbFilePath)

local buffer = read\_protobuf\_file\_c(pbFilePath)

protobuf.register(buffer) --注:protobuf 是因为在protobuf.lua里面使用module(protobuf)来修改全局名字

local stringbuffer = protobuf.encode("Person",

{

name = "Alice",

id = 12345,

phone = {

{

number = "87654321"

},

}

})

local slen = string.len(stringbuffer)

release\_print("slen = "..slen)

local temp = ""

for i=1, slen do

temp = temp .. string.format("0xX, ", string.byte(stringbuffer, i))

end

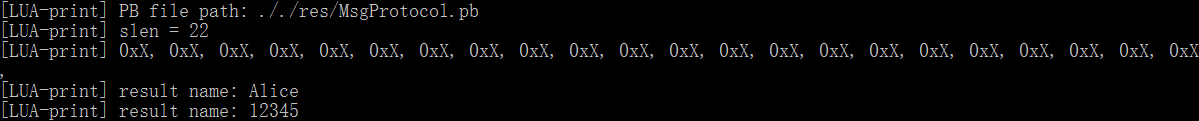
release\_print(temp)

local result = protobuf.decode("Person", stringbuffer)

release\_print("result name: "..result.name)

release\_print("result name: "..result.id)

⑥输出正确的结果

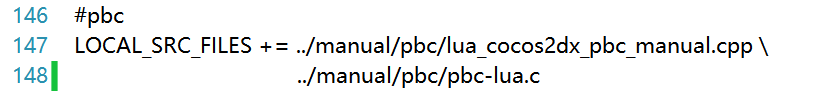


5.在windows上是能正确使用了，下面内容为修改mk文件，让protobuf在android上也能正常使用。

①修改libcocos2d项目对应的cocos2d-x\cocos\Android.mk文件，增加对应文件引用。



②修改libluacocos2d项目对应的cocos2d-x\cocos\scripting\lua-bindings\proj.android\Android.mk文件，增加对应文件引用。



③android上正常编译完成。

<https://www.cnblogs.com/chevin/p/6001872.html>