

2022unity接入Lua第三方库rapidjson过程

RapidJson 简介

RapidJSON 是一个 C++ 的 JSON 解析器及生成器。它的灵感来自 [RapidXml](#)。

- RapidJSON 小而全。它同时支持 SAX 和 DOM 风格的 API。SAX 解析器只有约 500 行代码。
- RapidJSON 快。它的性能可与 `strlen()` 相比。可支持 SSE2/SSE4.2 加速。
- RapidJSON 独立。它不依赖于 BOOST 等外部库。它甚至不依赖于 STL。
- RapidJSON 对内存友好。在大部分 32/64 位机器上，每个 JSON 值只占 16 字节（除字符串外）。它预设使用一个快速的内存分配器，令分析器可以紧凑地分配内存。
- RapidJSON 对 Unicode 友好。它支持 UTF-8、UTF-16、UTF-32 (大端序/小端序)，并内部支持这些编码的检测、校验及转码。例如，RapidJSON 可以在分析一个 UTF-8 文件至 DOM 时，把当中的 JSON 字符串转码至 UTF-16。它也支持代理对 (surrogate pair) 及 `"\u0000"` (空字符)。

官方文档: [RapidJSON: 首页](#)

当前版本的Lua第三方库情况

目前2022分支版本中的lua版本使用的是最新xLua官方最新的master版本，即2.1.16版本

相关源码已上传到分支目录中：

x1_cn_branch_unity2022\Dev\tools\xlua-2.1.16

在此源码基础上，项目中已接入了两个第三方库分别是

luamemstream

lua-protobuf

库的相关注册入口：

x1_cn_branch_unity2022\dev\client\Assets\3rdParty\XLua\Src\LuaLibs.cs

d:\x1_cn_branch_unity2022\dev\client\Assets\Scripts\FixClientManager.cs:

```
258 _luaEnv.AddBuildin("memstream", XLua.LuaDLL.Lua.LoadMemStream);
259 _luaEnv.AddBuildin("pb", XLua.LuaDLL.Lua.LoadPb);
260 _luaEnv.AddBuildin("pb.io", XLua.LuaDLL.Lua.LoadPbIO);
261 _luaEnv.AddBuildin("pb.buffer", XLua.LuaDLL.Lua.LoadPbBuffer);
262 _luaEnv.AddBuildin("pb.slice", XLua.LuaDLL.Lua.LoadPbSlice);
263 _luaEnv.AddBuildin("pb.conv", XLua.LuaDLL.Lua.LoadPbConv);
```

d:\x1_cn_branch_unity2022\dev\client\Assets\Scripts\LuaComponent\XLuaManager.cs:

```
77      DebugL8.Log("XLuaManager初始化OK!");
78      m_instance = this;
79      m_luaEnv = new LuaEnv();
80      m_luaEnv.GcPause = 100;
81      m_luaEnv.AddBuildin("memstream", XLua.LuaDLL.Lua.LoadMemStream);
82      m_luaEnv.AddBuildin("pb", XLua.LuaDLL.Lua.LoadPb);
83      m_luaEnv.AddBuildin("pb.io", XLua.LuaDLL.Lua.LoadPbIO);
84      m_luaEnv.AddBuildin("pb.buffer", XLua.LuaDLL.Lua.LoadPbBuffer);
85      m_luaEnv.AddBuildin("pb.slice", XLua.LuaDLL.Lua.LoadPbSlice);
86      m_luaEnv.AddBuildin("pb.conv", XLua.LuaDLL.Lua.LoadPbConv);
```

目前项目Plugins目录中引用的xlua库,都是添加过上述三方库扩展,且编译后的动态链接库文件。

```

|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/Android/libs/arm64-v8a/libxlua.so
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/Android/libs/armeabi-v7a/libxlua.so
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/Android/libs/x86/libxlua.so
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/WSA/ARM/xlua.dll
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/WSA/x64/xlua.dll
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/WSA/x86/xlua.dll
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/iOS/libxlua.a
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/x86/xlua.dll
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/x86_64/libxlua.so
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/x86_64/xlua.dll
|  /branches/x1_cn_branch_unity2022/Dev/Client/Assets/Plugins/xlua.bundle/Contents/MacOS/xlua
```

增量添加第三库步骤

目前按照官方文档中 Assets/XLua/Doc/XLua增加删除第三方lua库.md 的方法已经无法走通,相关文件以及命令脚本都已过时。

新的添加方法可以借助GitHub上 xLua官方在Action中部署的自动CI脚本,自动执行全平台的编译工作,免去需要到不同平台执行编译脚本以及搭建环境的步骤。

1. 先下载需要的rapidJson (0.7.1) 插件库源码,以及最新的xlua(2.1.16)官方主干源码

<https://github.com/xpol/lua-rapidjson>

<https://github.com/Tencent/xLua>

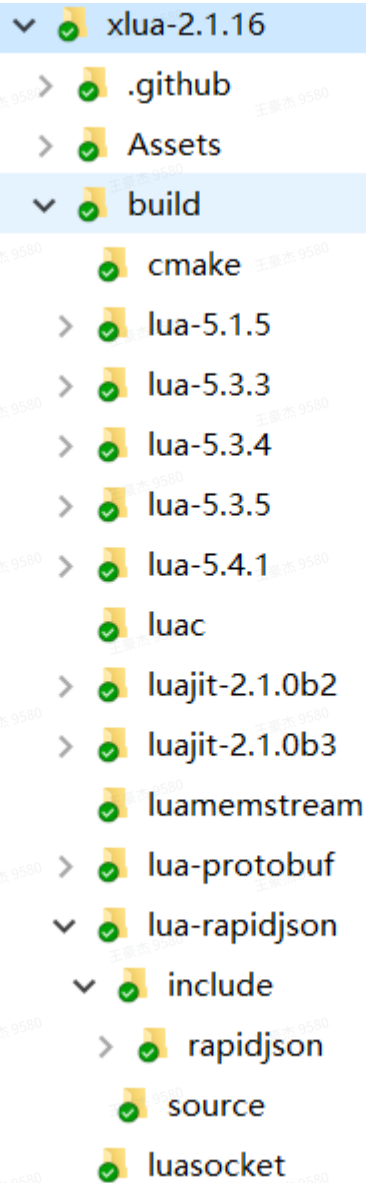
2. 在分支目录中保存最新的xlua源码,并且将之前已经接入的第三方库源码拷贝到源码的build目录下:

x1_cn_branch_unity2022\Dev\tools\xlua-2.1.16\build\luamemstream

x1_cn_branch_unity2022\Dev\tools\xlua-2.1.16\build\lua-protobuf

| 名称 | 修改日期 | 类型 | 大小 |
|--|-----------------|-----|----|
|  cmake | 2024/1/22 17:11 | 文件夹 | |
|  lua-5.1.5 | 2024/1/22 17:11 | 文件夹 | |
|  lua-5.3.3 | 2024/1/22 17:11 | 文件夹 | |
|  lua-5.3.4 | 2024/1/22 17:11 | 文件夹 | |
|  lua-5.3.5 | 2024/1/22 17:11 | 文件夹 | |
|  lua-5.4.1 | 2024/1/22 17:11 | 文件夹 | |
|  luac | 2024/1/22 17:11 | 文件夹 | |
|  luajit-2.1.0b2 | 2024/1/22 17:11 | 文件夹 | |
|  luajit-2.1.0b3 | 2024/1/29 16:57 | 文件夹 | |
|  luamemstream | 2024/1/22 17:11 | 文件夹 | |
|  lua-protobuf | 2024/1/29 15:30 | 文件夹 | |

3. 在xlua下的build文件夹下新建名为“lua-rapidjson”文件夹
4. 在“lua-rapidjson”下新建“include”和“source”文件夹
5. 将lua-rapidjson项目的“rapidjson\include”文件夹下的所有文件拷贝到xlua项目的“build\lua-rapidjson\include”文件夹下
6. 将lua-rapidjson项目的“src”文件夹下的所有文件拷贝到xlua项目的“build\lua-rapidjson\source”文件夹下



7. 打开“build\CMakeLists.txt”文件 此文件内已有原来的扩展命令行，需要添加新的扩展命令行：

```
1 #begin lua-rapidjson
2 set (RAPIDJSON_SRC
3     lua-rapidjson/source/Document.cpp
4     lua-rapidjson/source/rapidjson.cpp
5     lua-rapidjson/source/Schema.cpp
6     lua-rapidjson/source/values.cpp)
7 set_property(
8     SOURCE ${RAPIDJSON_SRC}
9     APPEND
10    PROPERTY
11    COMPILE_DEFINITIONS
12    LUA_LIB)
13 list(APPEND THIRDPART_INC lua-rapidjson/include)
14 set (THIRDPART_SRC ${THIRDPART_SRC} ${RAPIDJSON_SRC})
15 #end lua-rapidjson
```

8. 最关键的区别就在这里，原有教程里的方案是：执行xlua项目build目录下的对应的库文件生成脚本即可生成对应的链接库

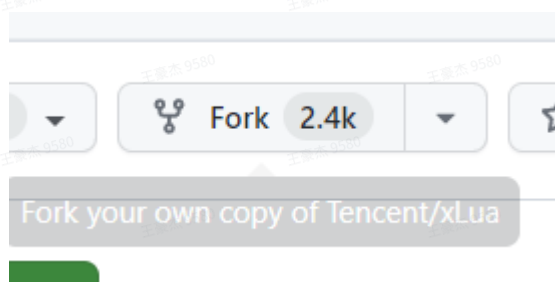
所有编译脚本都是按这个方式命名：make_平台_lua版本.后缀

比如windows 64位lua53版本是make_win64_lua53.bat，android的luajit版本是make_android_luajit.sh，要编译哪个版本就执行相应的脚本即可。

执行完编译脚本会自动拷贝到plugin_lua53或者plugin_luajit目录，前者是lua53版本放置路径，后者是luajit。

配套的android脚本是在linux下使用的，脚本开头的NDK路径要根据实际情况修改

9. 上一步清楚编译过程之后，在xLua的官方Github中,点击Fork按钮，创建一个属于自己的xlua仓库



Create a new fork

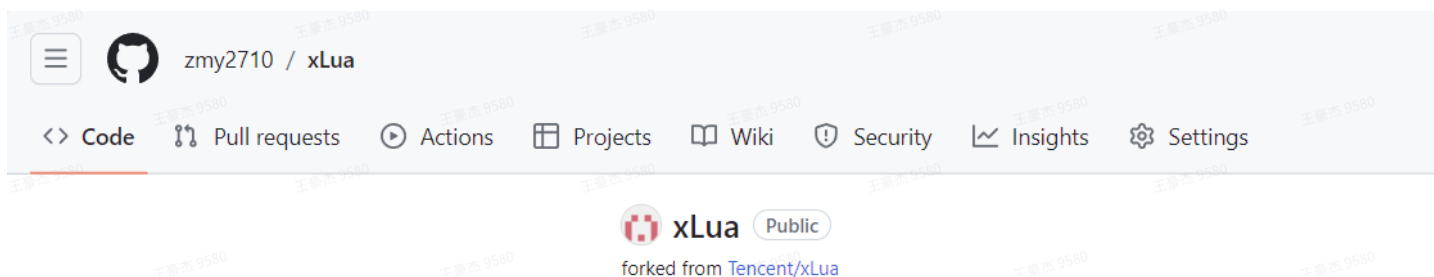
A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk ().*

No available destinations to fork this repository.

[zmy2710/xLua](#)

然后切换到自己的xlua仓库地址中：



10. 此时将上述第2步到第7步新增以及修改的文件和目录，上传到属于自己的xlua代码库的对应目录中，当提交完毕后，xlua代码库中的文件目录应该是和工程目录中的文件保持一致性：

zmy2710 Update CMakeLists.txt

This branch is 12 commits ahead of Tencent/xLua:master

| Name | Last commit message |
|--|---|
| .. | |
| cmake | 更新iOS cmake toolchain以支持macOS 10.15 (Tencent#657) |
| lua-5.1.5 | 51版本的一些平台兼容性改造 |
| lua-5.3.3 | WP下不能用getenv_popen_pclose_system, 统一在UWP下去掉 |
| lua-5.3.4 | ios 11兼容 |
| lua-5.3.5 | 加入字节码兼容的支持 |
| lua-5.4.1 | lua54的uwp兼容 |
| lua-protobuf | Add files via upload |
| lua-rapidjson | Add files via upload |
| luac | Windows下构建luac, 若使用vs 2015参数执行cmake失败, 则继续尝试使用vs 2017 (Tencent#488) |
| luajit-2.1.0b2 | ios 11兼容 |
| luajit-2.1.0b3 | 修复luajit2.1.0b3的编译报错 |
| luamemstream | Add files via upload |
| luasocket | fix Tencent#207 |
| plugin_lua53/Plugins/xlua.bundle/Contents | 多虚拟机, 并且在虚拟机中不断的新建及释放协程, 可能会出现协程指针重复 fix Tencent#257 |
| plugin_luajit/Plugins/xlua.bundle/Contents | 多虚拟机, 并且在虚拟机中不断的新建及释放协程, 可能会出现协程指针重复 fix Tencent#257 |
| vs2015 | window编译默认改为vs2017 |
| CMakeLists.txt | Update CMakeLists.txt |

11. 当仓库中的代码提交完毕后，GitHub的Actions页签下的CI脚步会自动执行编译过程，等待一会即可看到全平台的编译结果：

zmy2710 / xLua

<> Code

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

← build

Add files via upload #1

Summary

Jobs

Run details

android

android_luajit

linux

ios

osx

windows

windows_luajit-32

Usage

Workflow file

Triggered via push last week

zmy2710 pushed → 3d1dd09 master

Status

Success

Total duration

4m 7s

Artifacts

3

build.yml

on: push

android

1m 33s

android_luajit

2m 33s

linux

48s

ios

2m 31s

osx

1m 54s

windows

3m 54s

windows_luajit-32

1m 50s

在页面的最下方即使本次编译输出的文件结果：

| Artifacts | |
|-------------------------|---------|
| Produced during runtime | |
| Name | Size |
| plugin_lua53 | 10.4 MB |
| plugin_lua54 | 11.6 MB |
| plugin_luajit | 7.11 MB |

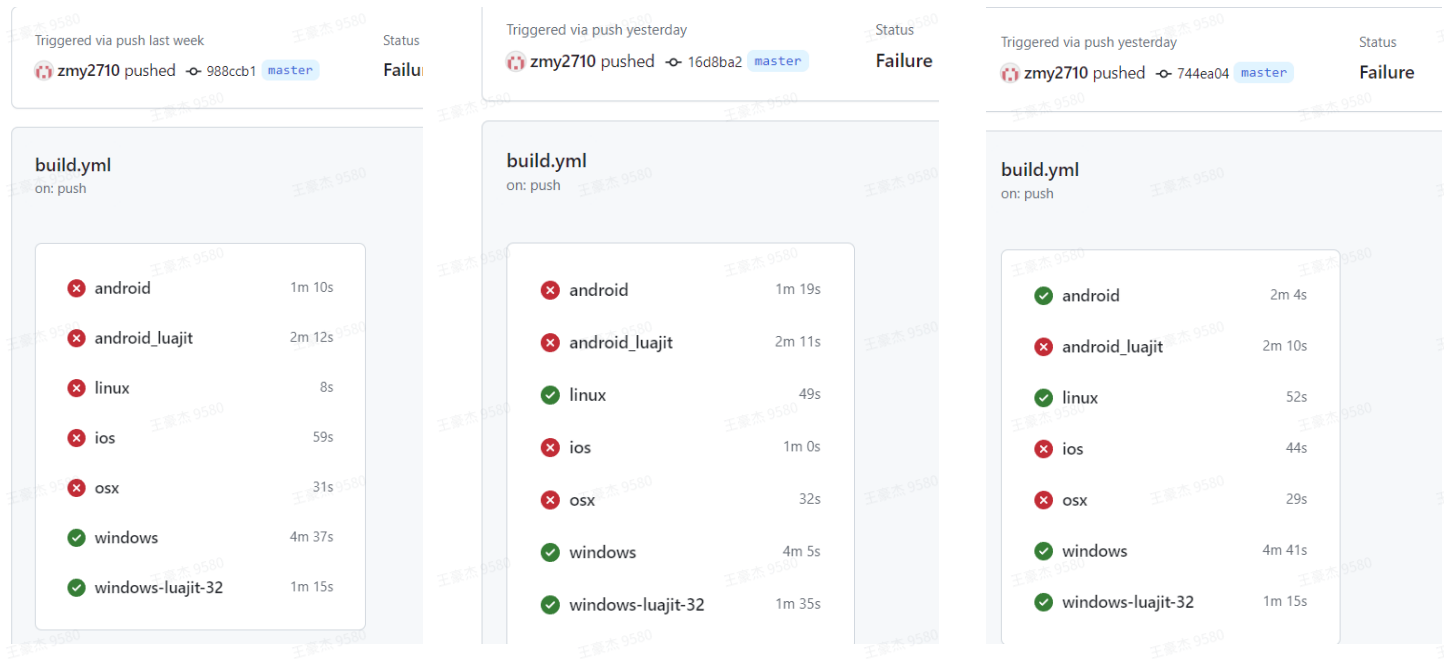
将上述文件下载本地解压缩后，将编译好的链接库覆盖掉项目中对应的老链接库文件即完成了整个编译替换步骤

12. 最后一步是在脚本中添加新的第三库的注册入口代码，这一步参考上面的 [2022unity接入Lua第三方库rapidjson过程](#) 步骤添加新的脚本接口即可

如果一切都如上述过程如此丝滑如此通顺就大吉大利，太玄幻剧了。下面补充一些遇到的问题

添加过程中遇到的问题

最大的门槛就是当提交完脚本以及三方源码后，遇到的各种平台的编译错误，比如这样的：



```
968 [ 81%] Building CXX object CMakeFiles/xlua.dir/luarapidjson/source/Document.cpp.o
969 In file included from /home/runner/work/xlua/xlua/build/luarapidjson/source/Document.cpp:14:
970 /home/runner/work/xlua/xlua/build/luarapidjson/source/values.hpp:232:10: error: calling a private constructor of class 'rapidjson::GenericValue<rapidjson::UTF8<char>, rapidjson::MemoryPoolAllocator<rapidjson::CrtAllocator>>'
971     return details::toValue(L, idx, 0, allocator);
972         ^
973 /home/runner/work/xlua/xlua/build/luarapidjson/include/rapidjson/document.h:690:5: note: declared private here
974     GenericValue(const GenericValue& rhs);
975     ^
976 /home/runner/work/xlua/xlua/build/luarapidjson/source/Document.cpp:115:12: error: calling a private constructor of class 'rapidjson::GenericValue<rapidjson::UTF8<char>, rapidjson::MemoryPoolAllocator<rapidjson::CrtAllocator>'
977     Value v = values::toValue(L, 3, doc->GetAllocator());
978         ^
979 /home/runner/work/xlua/xlua/build/luarapidjson/include/rapidjson/document.h:690:5: note: declared private here
980     GenericValue(const GenericValue& rhs);
981     ^
982 2 errors generated.
983 gmake[2]: *** [CMakeFiles/xlua.dir/build.make:314: CMakeFiles/xlua.dir/luarapidjson/source/Document.cpp.o] Error 1
984 gmake[1]: *** [CMakeFiles/Makefile2:83: CMakeFiles/xlua.dir/all] Error 2
985 gmake: *** [Makefile:91: all] Error 2
986 cp: cannot stat 'build_lj_x86/liblua.so': No such file or directory
987 Error: Process completed with exit code 1.
```

```
ios
failed last week in 44s

v Build 36s

598 /Users/runner/work/xlua/xlua/build/luamemstream/memstream.c:690:18: warning: implicit conversion loses integer precision: 'unsigned long' to 'int' [-Wshorten-64-to-32]
599     int strlength = strlen(strptr);
600
601
602 CompileC /Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/Objects-normal/arm64/memory_leak_checker.o /Users/runner/work/xlua/xlua/build/memory_leak_checker.c normal arm64 c
603 com.apple.compilers.llvm.clang.1_0.compiler (in target 'xlua' from project 'Xlua')
604 cd /Users/runner/work/xlua/xlua/build
/Applications/Xcode_14.2.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang -x c -target arm64-apple-ios9.0 -fmessage-length=0 -fdiagnostics-show-note-include-stack -fmacro-backtrace-limit=0 -Wno-trigraphs -fpascal-strings -O3 -Wno-missing-field-initializers -Wno-missing-prototypes -Wno-return-type -Wno-missing-braces -Wparentheses -Wswitch -Wno-unused-function -Wno-unused-label -Wno-unused-parameter -Wno-unused-variable -Wno-unused-value -Wno-empty-body -Wno-uninitialized -Wno-unknown-pragmas -Wno-shadow -Wno-four-char-constants -Wno-conversion -Wno-constant-conversion -Wno-int-conversion -Wno-bool-conversion -Wno-enum-conversion -Wno-float-conversion -Wno-non-literal-null-conversion -Wno-objc-literal-conversion -Wshorten-64-to-32 -Wpointer-sign -Wno-newline-eof -Wno-implicit-fallthrough -DMAKE_INTDIR=\"Release-iphonios\" -isysroot /Applications/Xcode_14.2.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS16.2.sdk -fstrict-aliasing -Wdeprecated-declarations -Wno-sign-conversion -Wno-infinite-recursion -Wno-comma -Wno-block-capture-autoreleasing -Wno-strict-prototypes -Wno-semicolon-before-method-body -I/Users/runner/work/xlua/xlua/build/build_ios_54/Release-iphonios/include -I/Users/runner/work/xlua/xlua/build -I/Users/runner/work/xlua/xlua/build/luarapidjson/include -I/Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/DerivedSources-normal/arm64 -I/Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/DerivedSources/arm64 -I/Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/DerivedSources -F/Users/runner/work/xlua/xlua/build/build_ios_54/Release-iphonios -fembed-bitcode -DDEBUG -DND -MT dependencies -MF /Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/Objects-normal/arm64/memory_leak_checker.d --serialize-diagnostics /Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/Objects-normal/arm64/memory_leak_checker.dia -c /Users/runner/work/xlua/xlua/build/memory_leak_checker.c -o /Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/Objects-normal/arm64/memory_leak_checker.o
605
606 ** BUILD FAILED **
607
608
609 The following build commands failed:
610 CompileC /Users/runner/work/xlua/xlua/build/build_ios_54/build/xlua.build/Release-iphonios/Objects-normal/arm64/values.o /Users/runner/work/xlua/xlua/build/luarapidjson/source/values.cpp normal arm64 c++
611 com.apple.compilers.llvm.clang.1_0.compiler (in target 'xlua' from project 'Xlua')
612 (1 failure)
613 cp: build_ios_54/Release-iphonios/liblua.a: No such file or directory
614 /Users/runner/work/xlua/xlua/build/build_ios_54/Xlua.xcodeproj: warning: The iOS deployment target 'IPHONEOS_DEPLOYMENT_TARGET' is set to 9.0, but the range of supported deployment target versions is 11.0 to 16.2.99. (in target 'ALL_BUILD' from project 'Xlua')
615 note: Run script build phase 'Generate CMakeFiles/ALL_BUILD' will be run during every build because the option to run the script phase "Based on dependency analysis" is unchecked. (in target 'ALL_BUILD' from project 'Xlua')
616 Error: Process completed with exit code 1.
```

其中有两个地方的修改最自闭，困扰了很久

一个是在luajit的编译错误中，由于路径未指定导致的编译错误（明明Set了，但是某一些平台就是识别不了，无语），需要在lua-rapidjson中添加如下命令行


```
include_directories(lua-rapidjson/include)
```

另外一个ios平台无论如何替换rapidjson版本以及luac工具版本都无法解决的错误（一度怀疑此三方库不兼容ios平台），需要添加以下两个命令行进行解决：

```
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
```

这是对C++11特性支持的命令行，否则ios os 和 android_luajit平台编译不通过







另外一个是在真机上运行时遇到的报错信息，提示rapidjson接入失败，错误关键字信息 format mismatch in precompiled chunk

这个依然是需要修改CMakeList编译脚本

```
LUAC_COMPATIBLE_FORMAT "compatible format" OFF )改为ON重新编译
```

完成上述修正后，即成功编译了包括三个第三方库支持的xlua动态链接库文件。

.1.16 > 编译结果 >

| 名称 | 修改日期 | 类型 | 大小 |
|--|-----------------|------------------|-----------|
|  lua53_xf_xlua_2.1.16.tgz | 2024/1/15 11:33 | WinRAR 压缩文件 | 4,484 KB |
|  lua54_xf_xlua_2.1.16.tgz | 2024/1/15 11:33 | WinRAR 压缩文件 | 4,988 KB |
|  luajit_xf_xlua_2.1.16.tgz | 2024/1/15 11:33 | WinRAR 压缩文件 | 3,240 KB |
|  plugin_lua53_2.zip | 2024/2/6 10:51 | WinRAR ZIP 压缩... | 10,781 KB |
|  plugin_lua54_2.zip | 2024/2/6 10:51 | WinRAR ZIP 压缩... | 11,290 KB |
|  plugin_luajit_2.zip | 2024/2/6 10:51 | WinRAR ZIP 压缩... | 8,027 KB |

经过和亮哥确认，目前项目中用到的都是Lua5.3版本的接口，所以这里我们解压缩lua53版的压缩包将里面的链接文件替换掉工程plugins目录下的文件即可。