NDK功能使用文档

一、背景

- 解决客户特殊定制需求
- 解决用户隐私问题
- 与luatos无缝对接
- 跨平台使用无压力

二、原理

动态加载静态库

三、使用场景

用户有自己独立技术的库，不方便对外开放

四、相关知识

a. C语言基础
b. lua开发基础

五、API 说明

dl.open

导入静态库

- **语法**

```
handle=dl.open(lib)
```

导入静态库

- **参数**

	A	B	C	D
1	**参数**	**类型**	**释义**	**取值**
2	lib	string	lib库所在的路径。如：/lua/user.lib	一般随lua脚本一起下载后库都在/lua/目录下

- **返回值**

	A	B	C	D
1	**返回值**	**类型**	**释义**	**取值**
2	handle	number	库的句柄	

dl.sym

动态调用静态库函数

- **语法**

```
return = dl.sym(handle,fun,ret_type,arg1,arg2,arg3....)
```

- **参数**

	A	B	C	D
1	**传入值**	**类型**	**释义**	**取值**
2	handle	number	打开库返回的句柄	
3	fun	string	需要调用的函数名，字符串表示	
4	ret_type	number	返回值，目前有两种分别为：dl.RETURN_NUMBER(可以返回一个bool，int, 句柄等) dl.RETURN_STRING(返回一个字符串)	
5	argx	number，string等等	函数的参数，可以是number类型和string类型	

- **返回值**

动态调用静态库函数

	A	B	C	D
1	**返回值**	**类型**	**释义**	**取值**
2	return	number/string	根据设置的返回值参数而设置的	

dl.close

- **语法**

```
dl.close(handle)
```

- **参数**

	A	B	C	D
1	**传入值**	**类型**	**释义**	**取值**
2	handle	number	打开库返回的句柄	

- **返回值**

无

六、使用步骤

步骤1 创建工程

- 下载NDK环境。
- 用户将自己的源文件和头文件分别放在NDK工程的user目录下的include目录和src目录。

步骤2 编译C库

- 在NDK\platform\xxx\core 目录下存放的是底层的头文件，一般会全部放在core_cpi.h里面。
- 用户只能用core_api.h提供的外部接口，包括标准库的接口。
- 用户如果想调用底层的接口，只需要添加#include "core_api.h" 即可，不需要自行copy头文件。
- NDK根目录下，运行 start.bat RDA8910，即可进入自动编译流程。
- 编译完成后，NDK会自动生成out目录，里面存放的就是编译好的lib库。

问题：

- 用户的函数如果需要异步通知lua虚拟机怎么办？

  这个应用场景多数是异步操作，执行接口后，需要异步通知lua端，这里引入了消息机制，具体实现如下接口所示:

Objective-C

```
/*
发消息通知lua端，下面编写脚本部分会介绍lua端如何接收消息
函数名：OPENAT_msg_to_lua
参数：msg_id：用户自定义的消息id，用于区分多种消息
    result：用户自定义的消息结果，默认FALSE
    num：用户自定义，用于传输数值数据
    data：用户自定义，用于传输文本数据，用户如果malloc后需要调用该接口后free，底层会做copy
    dataLen：文本数据长度
*/

bool OPENAT_msg_to_lua(UINT8 msg_id,BOOL result,INT32 num,CHAR* data,UINT32 dataLen);
```

步骤3 编写脚本

- 载入库

  调用用户自定义的库接口之前，都需要先载入库，使用`dl.open`接口

- 如何调用库接口

  使用dl.sym接口调用库接口：

  `local ret_number = dl.sym(handle, "fun1", dl.RETURN_NUMBER, 100)`

  注意：如果没有参数传入，第四个参数可以不填，但是不管有没有参数返回，参数的返回类型必须指定。

- 订阅消息

  C与lua之间使用消息机制进行通信，上面说了C库中如何发消息给lua，这里说下lua如何异步接收C库传的消息：

Lua

```
local function dl_msg_pro(msg)
    print(msg.msg,msg.num,msg.data,msg.result)
end

rtos.on(rtos.MSG_DL_INFO, dl_msg_pro)
```

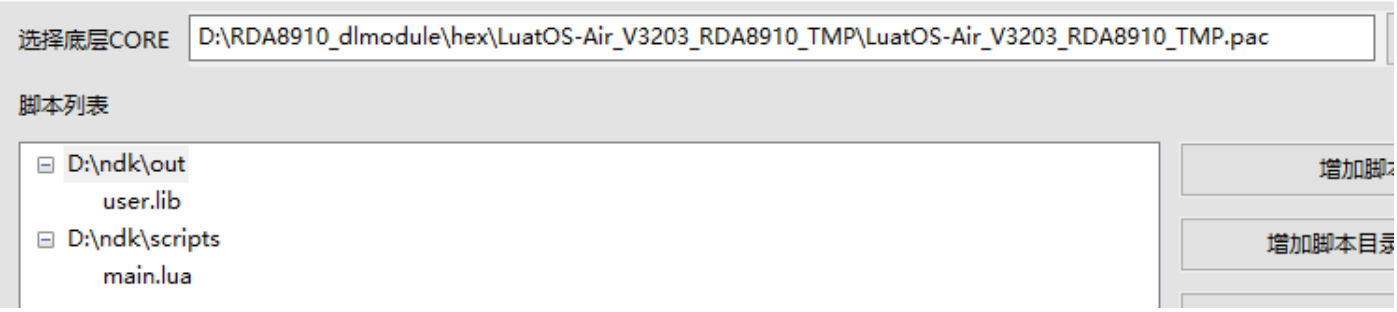
- 卸载库

  使用dl.close卸载库。

步骤4 下载

- 使用luatools选择支持NDK的固件库。

- 选择用户的脚本和NDK\out目录下的user.lib，如下图所示：



- 点击下载即可。

示例

c库编写：

C

```c
/*调用底层接口只需要调用该头文件即可，其它不需再调用*/
#include "core_api.h"

/*测试函数1，由lua直接调用*/
int fun1(int number)
{
    //调用了底层的 OPENAT_lua_print 接口
    OPENAT_lua_print("fun1 exe number=%d", number);
    return number;
}

/*测试函数2，由lua直接调用*/
char* fun2(char *string)
{
    OPENAT_lua_print("fun2 exe string=%s", string);
    fun1(10000);
    return string;
}
```

lua代码编写：

Lua

```lua
PROJECT = "DL_TEST"
VERSION = "1.0.0"

--加载日志功能模块，并且设置日志输出等级
--如果关闭调用log模块接口输出的日志，等级设置为log.LOG_SILENT即可
require "log"
LOG_LEVEL = log.LOGLEVEL_TRACE
require "sys"

rtos.sleep(3000)
--导入user.lib库
local handle = dl.open("/lua/user.lib")
--判断库是否导入成功
if handle then
    --调用fun1，并指定返回值为RETURN_NUMBER类型，传入的参数是100
    local ret_number = dl.sym(handle, "fun1", dl.RETURN_NUMBER, 100)
    local ret_string = dl.sym(handle, "fun2", dl.RETURN_STRING, "hello")

    print("ret_number", ret_number);
    print("ret_string", ret_string);
    --卸载库
    dl.close(handle)
end

--启动系统框架
sys.init(0, 0)
sys.run()
```

运行结果：

Apache

```apache
fun1 exe number=100
fun2 exe string=hello
fun1 exe number=10000
ret_number 100
ret_string hello
```