一、简介

LuatOS NDK定义

LuatOS NDK(Native Development Kit，以下简称为NDK)是一种本地化接口（芯片硬件提供的原始接口）开发工具集。可以简单的理解为 **使用C/C++开发的接口，可以在lua上直接调用。** 对于商用的LuatOS系统，Lua虚拟机部分是闭源的，用户无法直接集成C/C++代码。NDK正是针对这一场景提供的解决方案，开发简单，容易上手，无需过多关注本地化实现部分。

应用场景

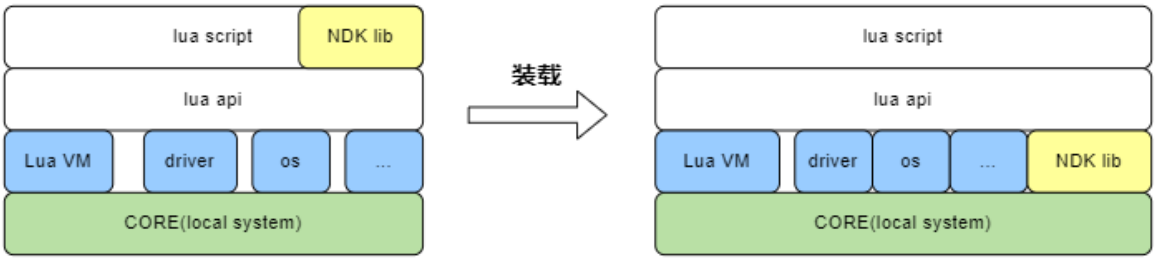
NDK主要针对LuatOS闭源版本（例如LuatOS-Air/LuatOS-HMI/LuatOS-iRTU等)的以下几种场景：

- lua语言运行效率达不到要求，需要用C/C++这类底层语言。
- 已有成熟的C/C++代码需要集成到LuatOS中。
- 希望关键代码能够受到保护，避免lua容易被反编译的风险。
- 向第三方提供闭源lib库。

当然开源版本的LuatOS理论上也可以使用，不过针对开源版本，更建议直接将库源码放到LuatOS源码中构建编译。

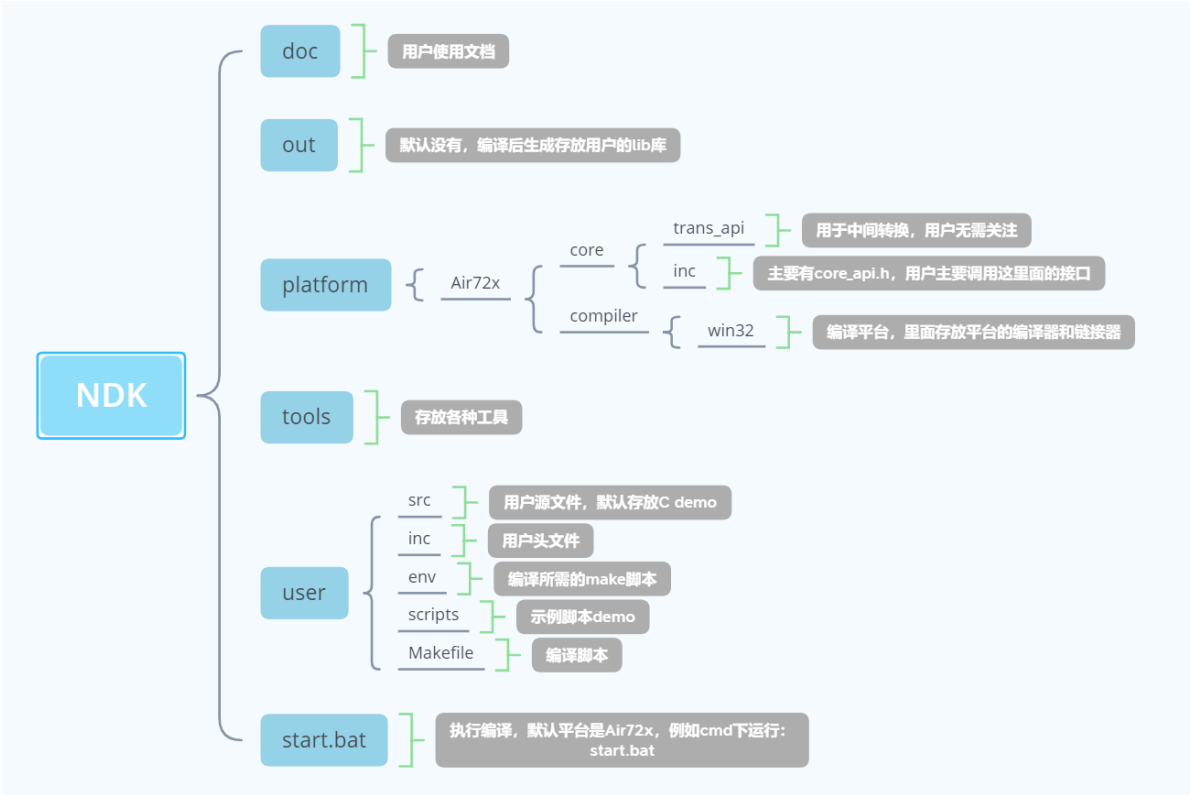
原理介绍

用户使用NDK将C代码编译成lib库后，和lua脚本文件一起打包下载到硬件中。LuatOS系统将lib库动态加载到内存中运行，并将lib库中提供的API接口注册到系统中。这样lua脚本就可以通过dl模块进行加载，并使用这类接口。



二、使用介绍

NDK目录结构



1. 添加源文件

将源文件copy到NDK的`user/src`目录下，头文件相应放到`user/include`目录下。

> 注意：目前的NDK环境默认支持Air72x平台，NDK自带该平台编译器。其他平台需要额外安装编译器，具体步骤待支持其他平台后再补充。

2. 修改源文件

- 在NDK\platform\xxx\core目录下存放的是底层的头文件，底层提供的接口会全部放在core_api.h里面。
- 用户如果想调用底层的接口，只需要在源码中添加#include "core_api.h" 即可
- 编写好的C库文件仅需要为lua提供相应函数即可。

3. 生成lib库文件

- 在NDK根目录下运行`start.bat`。
- 运行结束后会在NDK的根目录下生成`out`文件夹，里面有`user.lib`，就是合成好的静态库。

3.1 lua接口介绍

可以参考代码示例

dl.open

加载静态库并执行入口函数

- 语法

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

- 参数

传入值	释义
lib	lib库所在的路径。如：/lua/user.lib
main	lib库的入口函数

- 返回值

  handle：库的句柄

dl.close

卸载动态库

- 语法

  `dl.close(handle)`

- 参数

传入值	释义
handle	打开库返回的句柄

- 返回值

  nil

MSG_DL_INFO

lib库消息上报

- 语法

```
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)
```

3.2 lua和c常见的参数传递接口

可以参考代码示例

luaL_checklstring

获取字符串类型参数

- 语法

```
const char *luaL_checklstring (void *L, int narg, size_t *len)
```

- 参数

传入值	释义
L	状态机句柄
narg	参数索引
len	获取字符串参数的长度

- 返回值

  const char*类型的字符串

luaL_checkinteger

获取lua_Integer类型参数

- 语法

```
lua_Integer *luaL_checkinteger (void *L, int narg)
```

- 参数

传入值	释义
L	状态机句柄
narg	参数索引

- 返回值

  lua_Integer类型的数值

lua_pushstring

返回字符串类型参数

- 语法

```
void lua_pushstring (lua_State *L, const char *s)
```

- 参数

传入值	释义
L	状态机句柄
s	const char*类型字符串

- 返回值

  无

lua_pushinteger

返回lua_Integer类型参数

- 语法

```
void lua_pushinteger (lua_State *L, lua_Integer n)
```

- 参数

传入值	释义
L	状态机句柄
s	lua_Integer类型数值

- 返回值

  无

lua_pushinteger

返回lua_Integer类型参数

- 语法

```
void lua_pushinteger (lua_State *L, lua_Integer n)
```

- 参数

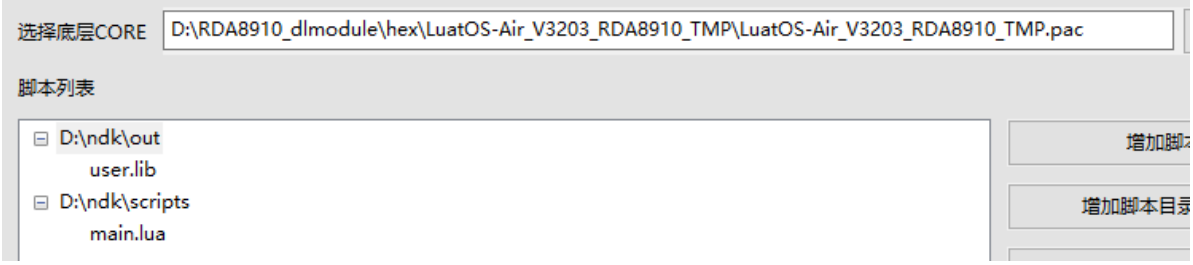
传入值	释义
L	状态机句柄
s	lua_Integer类型数值

- 返回值

  无

三、固件下载

- 底层固件选择支持NDK的固件库。
- 通过增加脚本文件按钮，选择lua脚本和NDK\out目录下的user.lib，如下图所示：



- 点击下载即可。

四、代码示例

C库demo

```c
/*编译成user.lib*/
#include "core_api.h"

int test(void *L)
{
    /*获取第一个参数*/
    int strSize;
    char *string  = luaL_checklstring(L,1, &strSize);
    /*获取第二个参数*/
    int num  = luaL_checkinteger(L, 2);
    OPENAT_lua_print("test exe %s,%d", string, num);
    /*返回第一个参数*/
    lua_pushinteger(L, 100);
    /*返回2个参数*/
    lua_pushstring(L, string);
    /*返回参数个数*/
    return 2;
}

/*C函数表*/
const luaL_Reg user_map[] =
{
    {"test", test},
    {NULL, NULL}
};

/*入口函数*/
int user_main(void *L)
{
    OPENAT_lua_print("user_main exe");

    /*C函数注册*/
    luaI_openlib(L, "user", user_map, 0);

    /*其他初始化*/
    return 0;
}
```

lua脚本demo

```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)
```

```
--通过dl.open接口加载user.lib文件，并执行user_main入口函数
--user_main入口函数会将user_map注册到虚拟机中，然后使用user.test（）进行接口调用
local handle = dl.open("/lua/user.lib","user_main")
if handle then
    print("user",user);
    local len,string = user.test("hello", 99)
    print("string",string)
    print("len",len)
    dl.close(handle)
end

--启动系统框架
sys.init(0, 0)
sys.run()
```

运行结果：

```
user_main exe
user table: 0x80a203a0
test exe hello,99
string hello
len 100
```