LuaDebuger使用手册

LuaDebuger是一个嵌入式的Lua调试器，它可以很方便的嵌入到你的工程中，为你提供Lua调试的便利。我们只需要生成一个LuaDebuger对象，并进行初始化即可。

m\_pLuaDebuger = new LuaDebuger();

m\_pLuaDebuger->initialize( Lua::Instance().getState() );

initialize 方法需要传入一个lua\_State指针。

经过了上面的初始化后，当你的程序执行起来时也就具备了lua脚本的调试能力。之后你就可以打开LuaDebugConsole进行调试了。

LuaDebugConsole.exe是一个控制台程序，它通过\\.\pipe\lua\_debuger管道与你的程序进行连接，并发送调试命令以及获取返回的信息。在控制台里你可以输入以下命令：

1. dir  
   列出目录，列出当前目录下的文件和文件夹。
2. cd (..|direction)  
    更改当前目录，需要注意的是该方法可能会导致程序的运行路径被更改。从而使一些程序发生文件访问违规。  
    当没有参数的时候该命令返回当前目录的全路径。
3. open (index|filename)  
    没有参数的时候该命令列出已经打开的所有文件。当前文件的行首使用\*号标记  
    参数为数字时，打开无参数时罗列文件的索引所指定的文件。  
    参数为文件名时，打开指定路径的指定文件。
4. list ( begin (end) )  
    列出当前文件的20行。有断点的行第一个字符用 @ 标记。  
    begin ：起始行  
    end ：结束行
5. bp [line]  
    设置断点，line指定断点所在行索引。
6. clr [line|"all"]  
    清除断点,line指定清除的断点的行索引，all参数说明清除所有断点，不区分大小写。
7. stack (stack\_index )  
    无参数时，显示当前断点的堆栈。  
    有参数stack\_index的时候显示该堆栈索引所指的堆栈反推出的所有本地变量值。
8. step  
    单步跟踪。
9. stepout  
    跳出当前函数。
10. stepin  
     进入当前行内的未执行函数。