文档编号：

luatable使用说明

luatable Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **文档状态** | [√]初稿  [ ] 评审通过  [ ] 修改  [ ] 发布  [ ] 作废 | **密级** |  |
| **当前版本** | V0.1 |

中国外汇交易中心

全国银行间同业拆借中心

变更履历

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **版本号** | **变更时间** | **修改人** | **变更简要说明** | **变更**  **类别** |
| 1 | 0.1 |  |  | 初始化创建文档 | N |

**Luatable使用说明**

1. **Luatable配置**
   1. **安装Luatable库到环境中**

下载Luatable库：

|  |
| --- |
| https://bitbucket.org/MartinFelis/luatables/src/default/ |

编译Luatable库：

|  |
| --- |
| mkdir build &&cd build  cmake .. |

安装Luatable库：

|  |
| --- |
| make && make linux install (若环境不是linux则根据实际情况修改) |

注意事项：

1. CMakeLists.txt中默认设置find\_package函数寻找Lua51库，若环境Lua版本不是5.1，需要修改find\_package函数为对应版本。

|  |
| --- |
| FIND\_PACKAGE (Lua51 REQUIRED) |

2.CMakeLists.txt中默认加入unittest++进行单元测试，若系统中未安装unittest++，可将相应test部分删除直接生成链接库。

|  |
| --- |
| SUBDIRS (  tests/  ) |

* 1. **使用luatable库**

1.包含头文件

|  |
| --- |
| #include luatables.h |

2.链接到LuaTable库

|  |
| --- |
| FIND\_PACKAGE (Lua51 REQUIRED)  LINK\_DIRECTORIES(<your path to libluatables.so>)  INCLUDE\_DIRECTORIES (  ${LUA\_INCLUDE\_DIR}  src/  )  include\_directories(inc)  add\_executable(<your project> <your source file>)  TARGET\_LINK\_LIBRARIES (  <your project>  luatables  ${LUA\_LIBRARIES}  ) |

1. **luatables使用**
   1. **luatables主要函数**

1. 读取table函数：

（1）从文件中读取：

|  |
| --- |
| LuaTable fromFile (const char \*\_filename); |

（2）从表达式中读取：

|  |
| --- |
| LuaTable fromLuaExpression (const char\* lua\_expr); |

（3）从语句中读取

|  |
| --- |
| LuaTable fromLuaState (lua\_State \*L); |

2. 读取table中不同类型内容函数：

每个函数中都需要输入默认值（default\_value），即若未读取到相应内容就存入默认值。

（1）bool类型

|  |
| --- |
| template<> bool LuaTableNode::getDefault<bool>(const bool &default\_value); |

（2）double类型

|  |
| --- |
| template<> double LuaTableNode::getDefault<double>(const double &default\_value); |

（3）float类型

|  |
| --- |
| template<> float LuaTableNode::getDefault<float>(const float &default\_value); |

（4）string类型

|  |
| --- |
| template<> std::string LuaTableNode::getDefault<std::string>(const std::string &default\_value); |

* 1. **使用示例**

1. 从lua文件中读取信息

文件内容：

|  |
| --- |
| -- myfile.lua  person = {  name = "James",  height = 1.87,  address = {  country = "United Kingdom"  }  }  return person |

读取内容操作：

|  |
| --- |
| LuaTable ltable = LuaTable::fromFile("myfile.lua")  std::string name = ltable["name"].get<std::string> ("Unknown");  double height = ltable["id"].get<double> (-1.);  std::string country = ltable["address"]["country"].get<std::string> ("Unknown"); |

2. 为已知结构体编写对应操作函数：

结构体示例：

|  |
| --- |
| struct CustomType {  string name;  double age;  bool drunken;  double position[3];  }; |

从LuaTable中读取：

|  |
| --- |
| template<> CustomType LuaTableNode::getDefault<CustomType>(const CustomType &default\_value) {  CustomType result = default\_value;  if (exists()) {  LuaTable custom\_table = stackQueryTable();  result.name = custom\_table["name"].get<std::string>();  result.age = custom\_table["age"];  result.drunken = custom\_table["drunken"];  result.position[0] = custom\_table["position"][1];  result.position[1] = custom\_table["position"][2];  result.position[2] = custom\_table["position"][3];  }  stackRestore();  return result;  } |

将结构体序列化并写入LuaTable：

|  |
| --- |
| template<> void LuaTableNode::set<CustomType>(const CustomType &value) {  LuaTable custom\_table = stackCreateLuaTable();  // set the fields of the custom type  custom\_table["name"] = value.name;  custom\_table["age"] = value.age;  custom\_table["drunken"] = value.drunken;  custom\_table["position"][1] = value.position[0];  custom\_table["position"][2] = value.position[1];  custom\_table["position"][3] = value.position[2];  } |