# 序列化 Serialization

## 一、什么是序列化

序列化，就是通过“提供统一的保存与读取数据的接口”实现“远程参数传递”与“对象持久化”。

简单来说，就是提供“将类的对象A以某种格式存储为一段数据，并且允许从该数据还原对象A的方法”，实现“网络中的数据交换”与“文件的存储与读取”。

Java、C#等语言都对序列化提供了原生的支持。C++的“准标准库”boost也提供了序列化的实现，我们公司的序列化就是仿boost的序列化实现的。

## 二、FB\_Serialization

1. **意义**

简化保存与读取对象的过程，通过实现序列化接口可以将C++的对象以某种格式存到文件中，并且可以很方便地再从文件恢复这个对象。

1. **如何使用**
   1. 对想要被序列化的类实现序列化函数（有两种实现途径：侵入式和外置式），在函数中枚举类中需要被序列化的成员变量

class A

{

private:

string m\_strName;

int m\_nCount;

float\* m\_pfArray;

// 侵入式序列化

FBSerialization\_InsideDefine ()

{

FBSerialization\_Variable(“Name”, m\_strName);

FBSerialization\_Array (“FloatArray”, m\_pfArray, (unsigned&) m\_nCount);

}

}

// 非侵入式序列化

namespace FBSerialization

{

FBSerialization\_OutsideDefine(A)

{

FBSerialization\_Variable (“Name”, SER\_MVW(m\_strName));

FBSerialization\_ Array (“FloatArray”,

SER\_MVW(m\_pfArray),

(unsigned&) SER\_MVW(m\_nCount));

}

}

* 1. 创建Archive（档案，分为输入Archive与输出Archive，其中分别定义了load与save接口，它决定了boost序列化的输入输出格式）对象

A object;

// 输出档案

FILE\* pOutFile = fopen (“test.lua”, “wb”);

FBKernal\_Assert(pOutFile);

CFBSerialization\_OArchiveLua outLua(pOutFile);

// 输入档案

FILE\* pInFile = fopen (“test.lua”, “rb”);

FBKernal\_Assert(pInFile);

CFBSerialization\_IArchiveLua inLua(pInFile);

* 1. 使用Archive对类的实例执行序列化（输入or输出）

FBSerialization\_Execute (object, outLua, 1);

fclose(pOutFile);

FBSerialization\_Execute (object, inLua, 1);

fclose(pInFile);

* 1. 输出结果示例

Version = 1

{

Name = “Object A”,

FloatArray =

{

Count = 2,

[0] = 3.333,

[1] = 2.222,

}

}

* 1. 序列化指针前需要先注册指针所指向的实际类型

FBSerialization\_RegisterType (A);

FBSerialization\_RegisterTypeWithName (A \*, APointer);

1. **实现**

FB\_Serialization的实现主要包括下面几个模块：

* 1. 【Common模块】

提供了使用序列化的各种接口，定义了兼容侵入式与非侵入式序列化函数的访问类。

* 1. 【Archive模块】

定义了输入输出的文档格式，目前实现了lua格式与二进制格式的档案

* 1. 【Specialization（特化）模块】

定义了序列化模板函数Serialize的特化版本，提供对各种类型的序列化支持。如：C++的基本数据类型、STL容器、指针类型、D3DX类型等。

* 1. 【Pointer模块】

用于支持指针序列化，主要用于解决多态指针的序列化问题。

[序列化的基本工作流程](序列化.vsdx)