2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)
IPCSIT vol. 53 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V53.No.2.94

Object Automatic Serialization and De-serialization in C++

Hua Chen[+] and Yi-ning Luo

College of Computer Science, Sichuan University

Chengdu, China

Abstract—With the rapid development of computer networks, data sharing brings great convenience to people. However, how to transmit variety of data in the network and preservation of data persistence becomes one of the most concerned issues. To solve such problems, we propose a data object serialization and de-serialization method. There are many serialization and de-serialization methods in C# and JAVA, such as Binary Serialization, SOAP Serialization, XML Serialization. This article analysis a method which uses Binary Formatter in C++ to realize object automatic serialization and de-serialization, and applied to Air Traffic Control System and has achieved good effect.

Keywords-serialization; de-serialization; Binary Serialization; Air Traffic Control

1. Introduction

Serialization is a process which transfers the object's state information into forms which can be stored or transmitted in the network. During serialization, object writes its current state to a temporary or persistent storage, which is widely used in the distributed system. After that, we can read from the storage area or de-serialize the state of the object, re-create the object. To serialization relative is the de-serialization, refers to the process which transfer the storage or transmission of the stream data in the network into the object state information [1]. Combination of these two processes, we can easily implement data storage and transmission in the network.

The purpose of Serialization: 1)Stored in some form of persistence of custom objects, in order to store the custom objects in a relational database, of course, can be stored on disk files, XML files, binary files, etc, that can make the data not lost even after power. 2)A custom object is passed from one application space to another [2]. 3)The custom object can be transmitted in a variety of networks, so that it achieves a distributed system object shared. Since in the network it can only be transformed in a string or binary format, so, in order to transfer the custom object, the object must be serialized into a string or binary format.

C # and JAVA have achieved the object serialization and de-serialization, and have a good reflection, from the sequence of the data stream we can revert object directly. And, C #.NET Framework provides automatic serialization architecture can handle object graph and a circular reference [3]. The JAVA serialization which usually does not need the custom code to save and restore object state is very simple. JAVA object serialization is not only an object's data can be saved, but also the preservation of recursive object reference data of each object.

C++, as an object-oriented programming language, its encapsulation, inheritance, polymorphism features are used by most programmers, therefore, implementation of C++ object serialization and de-serialization is even more important. However, C++ object is not inherited from a single root, and supports multiple inheritances, which increases the difficulty of object serialization.

[+] Corresponding author.
E-mail address: 578129801@qq.com

2. Implement Serialization and De-Serialization In C++

2.1 Principles of serialization de-serialization in C++

For the multiple inheritance class, each parent class must be serialized in some way, and the same rules need to comply when de-serialize. At the same time, C++ serialization needs to provide some assistance mechanism to solve the problem of object de-serialization. We can design a signature for each class, when serialize we should save the object signature, and then save the data of object itself; when de-serialize, we generate object instance based on object signature, then use the data after signature to initialize the instance. Serialization and de-serialization run around a run-time class list (the following class, we will give more details in the next part, here just for description of principle):

```
struct RUNTIME_CLASS_ITEM
{
//class signature
string classname;
// create a function pointer to class object
CNetBaseObj (*pfuncCreateObj)();
// point to the next run-time class object
 struct RUNTIME_CLASS_ITEM *pnext;
// head of the list
 static struct RUNTIME_CLASS_ITEM *pfirst;
 };
```

The importance of automatic serialization and de-serialization is to construct such a table automatically when class declaration. After we have this table, when de-serialization traverse this table under class signature, find the function pointer which will create the class object, and then create the object (null object), then call out the Unserialize to de-serialize the object (fill data).

Automatically construct the run-time list is to use the principle that C++ global object will automatically call the constructor function to initialize, and continue to add data to the list.

```
struct RUNTIME_CLASS_INIT
{
   RUNTIME_CLASS_INIT(string classname, CNetBaseObj (*pfuncCreateObj)())
{
RUNTIME_CLASS_ITEM *pnew = new RUNTIME_CLASS_ITEM;
pnew->classname = classname;
pnew->pfuncCreateObj = pfuncCreateObj;
pnew->pnext = pfirst;
pfirst = pnew;
}
};
```

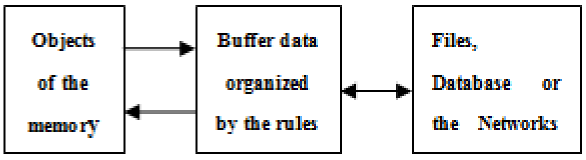
The class which needs serialization and de-serialization can complete automatic construction using two macros in the declaration:

```
// The class will be called finally when it states
#define DECLARE_RUNTIME_CLASS() \
  static CNetBaseObj* CreateObj();
  // It will be called In the realization of class
  #define IMPLEMENT_RUNTIME_CLASS(classname)\
  RUNTIME_CLASS_INIT _init_##classname(classname, &classname::CreateObj);
  CNetBaseObj *classname::CreateObj()
  {
  return new classname;
  }
  For example：
   //Statement
   Class CNetPlanChgInfo:public CNetBaseOj
```

```
 {
 ……
 DECLARE_RUNTIME_CLASS()
 };
//Implementation
IMPLEMENT_RUNTIME_CLASS(CNetPlanChgInfo)
CNetPlanChgInfo:: CNetPlanChgInfo()
{
……
}
……
```

It will automatically add one into the running class list after the calling of the IMPLEMENT_RUNTIME_CLASS in the implementation. The function which creates this class object is also achieved automatically in the macro calling.

Extended Application: It can also add the version information into the list item and will be serialized to the data stream, so that it can determine whether the class version is changed when it is de-serialized to ensure that de-serialization will not go wrong. Figure 1 is a brief diagram of the object serialization process.



Objects of the memory
Buffer data organized by the rules
Files, Database or the Networks

Fig.1. The process of the serialization

According to the principle, the method of the binary serialization and de-serialization in C# is improved so that it achieves the method of object automatic serialization and de-serialization in C++.

2.2 The interface of the serialization method

It will create a binary stream interface and a formatter instance when it is serialized in the binary formatter. The binary stream interface is defined as a CNetBaseObj class which is the parent class of all serialize and de-serialize data-class in the Air Traffic Control System, including the Serialize method and the de-serialize method. All the classes which need to have the serialized function have to implement this interface, which can make all the data classes better to be organized and achieved. The BeginSerialize method will generate the header information when it is serialized and it will be called in the first sentence of the Serialization function of all the classes which require serialization function. The BeginUnserialize method will read the header information when it is de-serializing, which will be called in the first sentence of the Unserialize function of all the classes which require de-serialization function.

The implementation of CNetBaseObj class is as follows (only several more important functions declaration)

```
  class CNetBaseObj
   {
   public:
virtual uint32 Serialize( net_archive &ar );
virtual uint32 Unserialize( net_archive &ar );
void BeginSerialize( net_archive &ar );
void BeginUnserialize( net_archive &ar );
   };
```

2.3 The serialization of the basic data type

The parameters of several important functions in the CNetBaseObj class is the form of the document serialization class net_archive, which achieves a simple structure(object) serialization method and uses the function template to support the most data types and the custom structures, also supports the serialization of the array. In the net_archive class, the operator “<<” and “>>” are overloaded to store and read non-array data in the network, while the array (pointer) is used by wr_array and rd_array to write into the array (pointer) and read data to the array (pointer) so that it achieves the package of the cache. The implementation of several more important functions of the net_archive class are as follows:

```
        class net_archive
         {
             public：
// The following function is used to obtain the cache length of net_archive object
       inline uint32 get_buf_size( )
                       {
                                return buf_size;
                 };
 // The following function is used to obtain the used cache length of net_archive object
              inline uint32 get_buf_used( ) const
                    {
                                    return data_size;
                 };
              inline uint32 get_buf_size( )
                       {
                                return buf_size;
                 };
//to abtain net_archivethe length of the object cache has been used
              inline uint32 get_buf_used( ) const
                   {
                                  return data_size;
               };
//here only list the overloaded function declaration about storing and reading string data
net_archive& operator <<(const std::string& strValue);
net_archive& operator >>(std::string& strValue);
// document (serialization) class,write data（non-array),when the cache is not enough, it can be grew.
template <typename T>
inline net_archive &operator <<( const T a_val );
// document (serialization) class,read data（non-array),when the cache is not,assertion error will be occurred.
template <typename T>
inline net_archive &operator >>( T &a_val );
// document (serialization) class,write array（pointer),when the cache is not enough, it can be grew.
template <typename T>
inline net_archive & wr_array( const T a_array[], const uint32 array_size );
// document (serialization) class,read data to the array（pointer),when the cache is not,assertion error will be occurred.
template <typename T>
inline net_archive & rd_array( T a_array[], const uint32 array_size );
         };
```

3. Example Applications in Distributed Systems

Air Traffic Control（ATC)System has become a networked, distributed complex real-time control system. In this system object serialization and de-serialization operations is very important. In the data transmission process, it is not as a top-level object directly (for many complex data we need to set its data structure reasonably), The data type transfered on the network does not need derived from the CNetBaseObj class, according to this principle set all the package structure definition.

The changing package problem on the flight plan is an important problem in ATC system, and if you are free to change the flight plan will cause very serious consequences. For the changing package issues, we briefly describe the application of the serialization and de-serialization and prove its availability.

```
Class declaration:
class CNetPlanChgInfo:public CNetBaseObj
class method realization：
```

```
virtual int Serialize(net_archive &ar)
{
      BeginSerialize(ar);
      ar << szMID;
      int size = vecContent.size();
      ar << size;
      for(int i = 0; i < size; i++)
      {
          vecContent[i].Serialize(ar);
       }
       return ar.get_buf_used();
 }
 virtual int Unserialize(net_archive &ar)
 {
       BeginUnserialize(ar);
       ar >> szMID;
       int size = 0;
       ar >> size;
       for(int i = 0; i < size; i++)
       {
          T t;
          t.Unserialize(ar);
          vecContent.push_back(t);
        }
       return ar.get_buf_used();
  }
```

In Serialize member function above, BeginSerialize(ar) function generate header information firstiy,and use overloaded operator “<<” to serialize the String data and the data in the vecContent container, last get_buf_used() function returns the length of the object cache has been used. But in Unserialize member function, BeginUnserialize(ar) function used to read header information from the network. And use overloaded operator “>>” to de-serialize the stream data in the network.The two member functions complete the serialization and de-serialization function of the changing package class CNetPlanChgInfo object on flight plan, so the controller can modify or read the data which stored to the changing package class CNetPlanChgInfo on flight plan, and the data of CnetPlanChgInfo on each interface are consistent.

4. Conclusion

Serialization refers to save the state of an object to a file, so that it is easy to later objects recovery or for remote transmission. In this paper, it realizes the object automatic serialization and de-serialization in C++, and achieved the purpose of data persistence saved in the network by C++ language. The system which the author of the paper participate in have completed the test, and will be put into application soon. The test proves that the system is running well for object serialization and de-serialization side.

5. Acknowledgment

This paper is funded by the National Natural Science Foundation Project (60832011).

6. References

[1] Junjie Hou, Dissecting MFC[M], Jie Hou, Translation, the second edition, Hua Zhong University of Science and Technology Press, 2001.

[2] Liqun Gao, Jiawen Yu,Junsong Ding, Serialization and De-serialization methods of analysis and application in .NET Framework [C]. Microcomputer Applications, 2007,29（11)

[3] C#.NET Entity class serialization methods Why serialization. http://pcajax.javaeye.com/blog/564919

[4] Deeply Discussion C# Serialization and De-serialization. http://www.360doc.com/content/09/1201/15/60849_10139374.shtml,2009, 5

[5] Serialization. http://new-restart.javaeye.com/blog/485369, 2009, 5

[6] Chenguang Hu, Jiefeng Yan, Zhengdong Gao, Bin Hu, Rui Xu,A class Serialization and De-serialization Framework[C]. Computer Knowledge and Technology[J], ,2009,5（24)