boost::serialization M.A.N.I.A.

written by Nidhoggr

●はじめに

会誌ほど出だしが書き辛いものもないですね。どうもほげ※1です。
今年はなんか技術的な話をしてくれと言われたような言われてないような感じだったので技術的な話をすることにしました。なんというかそろそろおいらも結構な上級生であるわけですから、それなりに**変態的な**利用価値のある話をします。
とはいえそんなに難しい話でもないので気楽に見てもらえればよいかと思います。
（というか触りしか説明してないしね・・・）

●シリアライズとは？

boost※2とか云々の話はとりあえずおいとくとして、そもそも**「シリアライズ」**という言葉すら聞いたことのない人もいるかと思います。
簡単に言ってしまえばクラスや構造体をファイルに書きだせるように、保持してる変数だとか配列だとかポインタが参照してる値だとかを含めてまとめて一本のバイト列として扱えるように変換することです。変換後のデータはパケットに切って送信したり、ファイルに保存したり出来ます。
「シリアライズ」でググれば一発で出てくるので、Wikipedia※3だとかe-Words※4だとかを見れば詳しい説明が載ってるので興味がある人は見ておくといいでしょう。逆にこの一本のバイト列を解析して各々の変数に復元する作業のことを**「デシリアライズ」**といいます。
ここでは両方合わせて**「シリアライズ」**もしくは**「シリアル化」**などと呼ぶことにします。

●なんかゲームに使え無そうなんだけど？

いまいち直感的な説明ではないので、ゲームのどこに使うのかと思うかもしれませんが、実はこれが結構色んなところに使える概念であったりします。
「クラスを固めてファイルに書き出す」という作業はつまるところ**「クラスの現在の状態をそのまま保存することが可能になる」**ということです。この概念はシステムやキャラクタのデータセーブはもとより、シーンの保存、リプレイデータの作成、その他外部ツールでの利用など様々なものに利用可能です。例えばシーンの保存などは、クイックセーブ、クイックロード※5などにも利用可能ですし、リプレイでは基底状態をシリアル化し、そこからの変化差分を保存していけばデータの管理も楽です。工夫次第ではもっと色々なことに使えるのではないかと思います。

●クラスをシリアライズをする上での問題点

そんな便利なシリアライズですが、いざ自力で実装するとなるとかなり大変です。
というのも、対象のクラスが継承したり継承されてたり抽象クラスだったりポインタ持ってたり相互参照してたりする場合、書込みや復元がキャストやポインタ追跡等で大変になるからです。
単純にポインタも持たない構造体ならそのままfwriteでファイルに書き込んでしまえばいいのですが、ポインタを持っている場合などはその示すアドレスの先のデータも書き込まなければいけません。一つ二つしか含まれていない場合はそんなに大変ではないですが、これが複数だったりクラスのポインタだったりすると激しく面倒なことになってくることは容易に想像出来ることだと思います。保持してるものがクラスのポインタだった場合などはデシリアライズするときにnewしてやる必要があります。派生クラスなどの場合はキャストだとかが必要になったりもします。
正直面倒でやってられません。

●最強のシリアライズクラス「boost::serialization」

そこで登場するのが boost::serialization です。MFC にも似たようなクラス※6が存在します。このクラスは対象のクラスをシリアル化し、ファイルに書き出したり読み出したりしてくれるものです。書き出すクラスは勿論ポインタ保持するメンバを持っていても良いし、派生クラスでもいいしデフォルトコンストラクタがあってもなくてもその辺りの煩わしい部分を全て吸収して書き出してくれます。しかも読み込むときもちゃんと参照先のクラスとか new してくれるし、派生クラスだったりしてもキャストし直してくれるし、循環参照してても問題なく復元してくれます。正直凄過ぎる。
ちなみに使用するためには boost のフルインストールが必要です。
インストール方法などは解説サイトに任せるとして、簡単にソースコードを紹介しておきます。

【適当なクラス CHoge】

```
#include <string>

class CHoge
{
private:
		std::string c;
};
```

【Choge をシリアル化】

```
#include <string>
//boost::serialization を使う上でおまじない的な感じ
#define BOOST_LIB_NAME boost_serialization
#include <boost/config/auto_link.hpp>
//今回は std::string を使ってるので
#include <boost/serialization/string.hpp>
#include <boost/archive/text_oarchive.hpp>

class CHoge
{
private:
		std::string c;

		//追加分。ここは private でないとだめな点に注意
		friend class boost::serialization::access;
		template<class Archive>
		void serialize(Archive& ar, const unsigned int version)
		{
			ar & c;
		}

};
```

とりあえず簡単に解説を。

重要なところは**追加された serialize テンプレートメソッドは private でないといけない**というところです。あとは追加でインクルードされているファイルにも注意。使用する std のコンテナによってはインクルードファイルが変わったりします。
今回は std::string を使ったので string.hpp を読み込んでいます。

次にこれを書き込んだり読み込んだりしてみます。

【書込み】

```
#include "CHoge.h"
#include <fstream>
#include <boost/archive/text_oarchive.hpp>

int main()
{
	CHoge hoge;

	std::ofstream ofs("output.txt");
	boost::archive::text_oarchive oa(ofs);

	oa << (const CHoge&)hoge;

	return 0;
}
```

【読み込み】

```
#include "CHoge.h"
#include <fstream>
#include <boost/archive/text_iarchive.hpp>

int main()
{
	CHoge hoge;

	std::ifstream ifs("output.txt");
	boost::archive::text_iarchive ia(ifs);

	ia >> hoge;

	return 0;
}
```

こんな感じで読んでもらえればいいかと思われます。
正直何にも考えないで定型文として使っていいと思います。
ちなみに読み書きメソッドをシリアル化するクラスに置いても多分動作します※7。

●もっと色々あるけどここらへんで

正直語ると物凄く長くなるのでこの辺で終わりです。まあ触りだけのつもりだったので・・・
もっと詳しく知りたい人は筆者に直接聞いたり、ググったりしてみるときっと一杯出てきます。

参考 URL：http://hw001.gate01.com/eggplant/tcf/cpp/boost_serialization.html

ちなみにこのライブラリは STL とかのコンテナを併用したときに最も力を発揮します。
STL も結構便利なライブラリなので（っていうか C++標準なんだけど）知らない人は双方使ってみるといいかもしれません。

●オマケ：用語説明など

※1：BIOS とか X とか作ってます。今回も中身の無い会誌ですみません。

※2：C++の準標準ライブラリと言われているものです。頑張って作ってる団体がいます。
正規表現の Regex などを筆頭に他言語にあったりする便利な機能が C++でも使えます。

※3：言わずと知れた Web で見れるフリー百科辞典です。
ブリタニカに匹敵する勢いとも言われたりもしてますが本当なのかどうか・・・

※4：IT 関連に特化した用語辞典です。
プログラミングに関係するところには簡単なコードがあってくれると嬉しいのですが。

※5：ゲームエミュレータ等ではよく実装されてる機能ですね。多分似たようなことをしています。

※6：CArchive というクラスです。筆者は使ったことがないのでよくわかりませんが・・・

※7：あんまりお勧めできないかもですね。boost::serialization は既存のクラスを固めるようなものなので専用クラスとか作らない方がいいような気がします。