Subject: Re: How would you design a good copy/move semantics system?
Posted by Lance on Fri, 19 Dec 2014 23:13:20 GMT
View Forum Message <> Reply to Message

Hi cbpporter:

I am not sure if I understand you correctly, but it seems to be the way it is in C++11.

--each class can have a copy and a move optionally

true.

--copy works like the default copy constructor, except for classes where deep copies are needed most time you can use =default to provide a default copy constructor. If not, you supply a copy ctor body.

- move works pretty much the way it works in U++, but only destroys data that would be copied by a deep copy
move behavior is defined by the function body you supply. You may tailor it to behave exactly like what Upp does, you can do it otherwise.

--calling move on a class that does not have a move implemented will do a copy
that's the way it is.

```
#include <iostream>

using namespace std;

class C
{
public:
 C(){}
 C(const C& c){ cout<<"copy ctor of C invoked."<<endl;}
};

class CM
{
public:
 CM(){}
 CM(const CM& c){ cout<<"copy ctor of CM invoked."<<endl;}
 CM(CM&& c){ cout<<"move ctor of CM invoked."<<endl;}
};

C&& ReturnC(){ return C(); }
CM&& ReturnCM(){ return CM(); }

int main()
{
```

```
 C c=ReturnC();
 CM cm=ReturnCM();
}
```

--the rules apply based on class depth
basically true. copy ctor will always call base class copy ctor. Not so with move ctor. The case with move ctor is a little bit interesting.

```
#include <iostream>

using namespace std;

class CM
{
public:
 CM(){cout<<"CM default ctor invoked"<<endl;}
 CM(const CM& c){ cout<<"copy ctor of CM invoked."<<endl;}
 CM(CM&& c){ cout<<"move ctor of CM invoked."<<endl;}
};

class CMD : public CM
{
public:
 CMD(){cout<<"CMD default ctor invoked"<<endl;}
 CMD(const CMD& c){ cout<<"copy ctor of CMD invoked."<<endl;}
 CMD(CMD&& c){ cout<<"move ctor of CMD invoked."<<endl;}

};


int main()
{
 CMD cm;

 cout<<"-----------------------"<<endl;

 CMD cmd=std::move(cm);
}
```

Here is the result of running the above program:

```
CM default ctor invoked
CMD default ctor invoked
-----------------------
CM default ctor invoked
```

move ctor of CMD invoked.

This is not right. The right way to do it is like this:

```
#include <iostream>

using namespace std;

class CM
{
public:
 CM(){cout<<"CM default ctor invoked"<<endl;}
 CM(const CM& c){ cout<<"copy ctor of CM invoked."<<endl;}
 CM(CM&& c){ cout<<"move ctor of CM invoked."<<endl;}
};

class CMD : public CM
{
public:
 CMD(){cout<<"CMD default ctor invoked"<<endl;}
 CMD(const CMD& c){ cout<<"copy ctor of CMD invoked."<<endl;}
 CMD(CMD&& c):CM(std::move(c)){ cout<<"move ctor of CMD invoked."<<endl;}
};

int main()
{
 CMD cm;

 cout<<"------------------------"<<endl;

 CMD cmd=std::move(cm);
}
```

And here is the result of running it

```
CM default ctor invoked
CMD default ctor invoked
------------------------
move ctor of CM invoked.
move ctor of CMD invoked.
```

And if you are concerned with Upp containers, this post may be of interests to you:

http://www.ultimatepp.org/forums/index.php?t=msg&th=8448 &start=0&