Subject: Re: How would you design a good copy/move semantics system?
Posted by Lance on Thu, 08 Jan 2015 00:16:58 GMT
View Forum Message <> Reply to Message

Hi cbpporter:

Happy New Year!

c++11 is here and will stay. The features you required are part of c++11, which means you can use it without any(almost) extra effort.

The code you quoted is like this: if base class have a copy constructor, derived class' move constructor will use the base class copy constructor to construct the base part of a derived object unless explicitly delegated to another ctor. Sounds very complicated, maybe it's easier to use an example:

```
struct base
{
   base() : buff(nullptr), buff_len(0u){}

   // copy ctor
   base(const base& b): buff_len(b.buff_len)
   {
      if(buff_len)
      {
          buff=new char[buff_len];
          memcpy(buff,b.buff,buff_len);
      }else
          buff=nullptr;
   }

   // move ctor, essentially do what Upp-pick is supposed to do
   base(base&& b):buff_len(b.buff_len), buff(b.buff)
   {
         b.buff_len=0u;
         b.buff=nullptr;
   }

private:
   char * buff;
   unsigned buff_len;
};

struct derived : public base
{
    derived(): i(0){} // will call base::default ctor to consturct base part of *this;
```

```
    derived(const derived& d) : i(d.i) {} // will call base::copy ctor to construct base part of *this;

    derived(derived&& d): i(d.i){} // you may expect base::move ctor to be called to construct base part of *this.
                // I do think the c++ committee should default to use base move ctor for derived move ctor.
                // unfortunately, this is not the case. you have to explicitly delegate the construction of
                // the base part of *this to base move ctor, with something like this:
                //
                //  derived(derived&& d) : base(std::move(d)), i(d.i){}
                //
                // this is the point I was trying to make.

private:
    int i;
};
```

HTH.

Lance
