
Subject: Re: C++11

Posted by [Lance](#) on Sun, 02 Dec 2012 17:47:32 GMT

[View Forum Message](#) <> [Reply to Message](#)

And any object can be `std::move()`'ed to become moveable, no matter it's a temporary or not.

```
#include <iostream>
#include <string>
#include <algorithm>
#include <new>
```

```
using namespace std;
```

```
struct S
{
    S(){}
    S(const S& rhs) : text(rhs.text){}
    S(S&& rhs){
        struct T{ char _[sizeof(S)]; };
        std::swap(reinterpret_cast<T*>(*this),
            reinterpret_cast<T*>(rhs)
        );
    }
}
```

```
S& operator=(const S& rhs)
{
    this->~S();
    return *new(this)S(rhs);
}
```

```
S& operator=(S&& rhs)
{
    this->~S();
    // std::move() is necessary even when rhs is declared as temporary
    return *new(this)S(move(rhs));
}
```

```
string text;
```

```
};
ostream& operator<<(ostream& os, const S& s)
{
    return os<<s.text<<endl;
}
```

S global;

```
int main()
{
    global.text="This is the global text";
    cout<<global;

    S a;
    a.text="This is object a";

    //
    //
    S b(a);
    cout<<"after S b(a), now <b> is:"<<b<<"and <a> is:"<<a;

    S c(std::move(global)); // use move constructor
    cout<<"after S c=std::move(global), now <c> is:"<<c<<"and <global> is:"<<global;

}
```
