
Subject: DeepCopyOption do_clone inconsistency?
Posted by [kohait00](#) on Tue, 14 May 2019 08:16:52 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hey guys

I've been away for quite some years (due to work changes) but never forgot about U++, and recently decided to give it new spin and revive some 'old' and never finished projects. I practically slept through the whole C++11 refactoring of U++ and need to do a fresh start.. good for me, I get to dive in the code again :p yeah!!

Speaking of which: I stumbled across the following

Topt.h

```
template <class T, class B = EmptyClass>
class WithClone : public B {
public:
    friend T do_clone(const T& src) { T c(src, 1); return c; }
};
```

```
template <class T, class B = EmptyClass>
class DeepCopyOption : public B {
public:
#ifdef DEPRECATED
    friend T& operator<=<=(T& dest, const T& src)
    { if(&dest != &src) { (&dest)->~T(); ::new(&dest) T(src, 1); } return dest; }
#endif
    friend T do_clone(const T& src) { T c(src, 1); return c; }
};
```

```
template <class T, class B = EmptyClass>
class MoveableAndDeepCopyOption : public B {
    friend void AssertMoveable0(T *) {}
#ifdef DEPRECATED
    friend T& operator<=<=(T& dest, const T& src)
    { if(&dest != &src) { (&dest)->~T(); ::new(&dest) T(src, 1); } return dest; }
#endif
    friend T clone(const T& src) { T c(src, 1); return c; } <<<< SHOULDN'T THIS BE do_clone?
};
```

the changes were introduced back in 2016 in this commit

<https://github.com/ultimatepp/mirror/commit/f501894b10b42d9b1a35950089818607d98c4d4b>

If I get it right, the do_clone is the final function that is addressed by the clone() (pick counterpart).

do_clone could be 'reimplemented' with a specific template instantiation.
clone is used throughout the code in the containers as 'higher level' clone function that maps to do_clone.

Nevertheless, MoveableAndDeepCopyOption is using clone(), instead of do_clone() like the other classes.

Isn't that it should be uniformly either clone() or do_clone() for all of them?

My bet: do_clone() should be used in MoveableAndDeepCopyOption as well.. as it maps to the Deep copy constructor instead of the default copy constructor.

Can anyone point out what the do_clone() is about if I am wrong and missing something?

PS: I try to get through bazaar and get some of those old packages of mine to work again. Many have been ported by others, thanks guys :)

Subject: Re: DeepCopyOption do_clone inconsistency?

Posted by [mirek](#) on Tue, 14 May 2019 13:07:42 GMT

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

kohait00 wrote on Tue, 14 May 2019 10:16Hey guys

I've been away for quite some years (due to work changes) but never forgot about U++, and recently decided to give it new spin and revive some 'old' and never finished projects.
I practically slept through the whole C++11 refactoring of U++ and need to do a fresh start.. good for me, I get to dive in the code again :p yeah!!

Speaking of which: I stumbled across the following

Topt.h

```
template <class T, class B = EmptyClass>
class WithClone : public B {
public:
    friend T do_clone(const T& src) { T c(src, 1); return c; }
};
```

```
template <class T, class B = EmptyClass>
class DeepCopyOption : public B {
public:
#ifdef DEPRECATED
    friend T& operator<=<=(T& dest, const T& src)
    { if(&dest != &src) { (&dest)->~T(); ::new(&dest) T(src, 1); } return dest; }
#endif
    friend T do_clone(const T& src) { T c(src, 1); return c; }
};
```

```

template <class T, class B = EmptyClass>
class MoveableAndDeepCopyOption : public B {
    friend void AssertMoveable0(T *) {}
#ifdef DEPRECATED
    friend T& operator<=<=(T& dest, const T& src)
    { if(&dest != &src) { (&dest)->~T(); ::new(&dest) T(src, 1); } return dest; }
#endif
    friend T clone(const T& src) { T c(src, 1); return c; } <<<< SHOULDN'T THIS BE do_clone?
};

```

the changes were introduced back in 2016 in this commit

<https://github.com/ultimatepp/mirror/commit/f501894b10b42d9b1a35950089818607d98c4d4b>

If I get it right, the do_clone is the final function that is addressed by the the clone() (pick counterpart).

do_clone could be 'reimplemented' with a specific template instantiation.

clone is used throughout the code in the containers as 'higher level' clone function that maps to do_clone.

Nevertheless, MoveableAndDeepCopyOption is using clone(), instead of do_clone() like the other classes.

Isn't that it should be uniformly either clone() or do_clone() for all of them?

My bet: do_clone() should be used in MoveableAndDeepCopyOption as well.. as it maps to the Deep copy constructor instead of the default copy constructor.

Can anyone point out what the do_clone() is about if I am wrong and missing something?

PS: I try to get through bazaar and get some of those old packages of mine to work again. Many have been ported by others, thanks guys :)

Welcome back.

Looks like it is doing the same thing, just in slightly different way. I will recheck this soon and perhaps will try to remove do_clone.

Mirek

Subject: Re: DeepCopyOption do_clone inconsistency?
 Posted by [kohait00](#) on Mon, 09 Mar 2020 07:46:43 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi mirek

any news about the issue here?

DeepCopyOption
MoveableAndDeepCopyOption

are essentially identical, or so i'd expect
hence should have both do_clone() IMHO..
I don't see why they should be different.

this enables me to only specialize template<> do_clone instead of both when dealing with some
clone types and wanting to provide the same end interface for DeepCopyOption and
MoveableAndDeepCopyOption..

Subject: Re: DeepCopyOption do_clone inconsistency?

Posted by [mirek](#) on Mon, 09 Mar 2020 14:52:05 GMT

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

Thanks for reminding me; after investigating the issue, I have removed do_clone (what is was
doing is now doing generic variant of clone).

Mirek
