Subject: ONE and assignement
Posted by mdelfede on Sat, 28 Jul 2018 07:48:30 GMT
View Forum Message <> Reply to Message

Hi,

I'm right that One<> is missing those operators?

```
template <class TT>
void       operator=(One<TT>&d) = delete;

template <class TT>
void       operator=(One<TT> const &d) = delete;
```

In an application I've noticed that if I do :

```
One<String> A = new String("pippo");
One<String> B;
B = A;
Cout() << "*A = " << *A << "\n";
Cout() << "*b = " << *B << "\n";
```

The internal pointer is copied, NOT picked, and NO COMPILER ERROR IS GENERATED, so when A and B get destroyed the pointer is
double freed and I get a violation access error.
Adding the above operators brings a compiler error, which is the correct behaviour, IMHO.

Ciao

Max

Subject: Re: ONE and assignement
Posted by mirek on Sat, 28 Jul 2018 09:35:13 GMT
View Forum Message <> Reply to Message

Thanks, correct, fixed (also with Any and Bits)

Subject: Re: ONE and assignement
Posted by mdelfede on Sat, 28 Jul 2018 12:46:40 GMT
View Forum Message <> Reply to Message

Thank you!

BTW, I've seen that One<> is also missing DeepCopyNew()... when I use operator <<= to get a deep copy of contained object I get a compiler error.

Subject: Re: ONE and assignement
Posted by mirek on Sat, 28 Jul 2018 15:06:22 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sat, 28 July 2018 14:46Thank you!

BTW, I've seen that One<> is also missing DeepCopyNew()... when I use operator <<= to get a deep copy of contained object I get a compiler error.

Uhm, that is leftover from pre c++11 Core. DeepCopyNew was means to allow polymorphic copy, but I do not remember anybody ever using it. For current core it is replaced (in Array) with simple clone (non-polymorphic), so that is how I have fixed it here...

Mirek

Subject: Re: ONE and assignement
Posted by mdelfede on Sun, 29 Jul 2018 08:22:36 GMT
View Forum Message <> Reply to Message

So the only way to deep copy an One<> is to use:

```
One<AClass> A = new AClass(1, 2, 3);
One<AClass> B;

B = new AClass(*A, 1);
```

Assuming that AClass has a deep copy constructor, right ?
I'm missing the polymorfic deep copy...
Btw, also the clone() complains about missing DeepCopyNew.

Subject: Re: ONE and assignement
Posted by mirek on Sun, 29 Jul 2018 08:35:28 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sun, 29 July 2018 10:22So the only way to deep copy an One<> is to use:

Btw, also the clone() complains about missing DeepCopyNew.

Now that is really weird, as "DeepCopyNew" identifier does not exist in U++ sources...

Subject: Re: ONE and assignement
Posted by mdelfede on Sun, 29 Jul 2018 09:18:19 GMT
View Forum Message <> Reply to Message

Core/Other.h, line 69, in my tree, which is up-to-date with svn repo

---

Subject: Re: ONE and assignement
Posted by mirek on Sun, 29 Jul 2018 18:21:00 GMT
View Forum Message <> Reply to Message

Oops, sorry, forgot to commit.

Mirek
