Subject: AsyncWork<Vector<T>> fails due to lack of copy-constructor
Posted by piotr5 on Mon, 06 Jan 2020 14:38:13 GMT
View Forum Message <> Reply to Message

I created AsyncWork<Vector<char>> and when I make use of the Get() member compiler complains Vector only has move-constructor.
is this a bug? CoWork.h reads:

```
template <class Ret>
class AsyncWork {
 template <class Ret2>
 struct Imp {
  CoWork co;
  Ret2   ret;
...
  const Ret2& Get()                       { return ret; }
 };
...
 Ret        Get()                       { ASSERT(imp); imp->co.Finish(); return imp->Get(); }
```

it's strange there is a 2nd template parameter for this class inside of AsyncWork.
sounds more like it was meant to be:

```
  Ret   ret;
...
  Ret2 Get()                       { return ret; }
```

and then use "Imp<const Ret&> imp;" in case Ret has copy-constructor?
or maybe rewrite to use if constexpr?
is there a speed-gain from using the copy-constructor?
I doubt there is one for Vector as return-type...

or did I make a mistake and AsyncWork is not for this kind of usage?
should I rewrite my code to use Thread?

Subject: Re: AsyncWork<Vector<T>> fails due to lack of copy-constructor
Posted by mirek on Mon, 06 Jan 2020 21:32:00 GMT
View Forum Message <> Reply to Message

piotr5 wrote on Mon, 06 January 2020 15:38
or did I make a mistake and AsyncWork is not for this kind of usage?
should I rewrite my code to use Thread?

Well, I am not sure what your usage is, but yes, in general, I think using AsyncWork directly is probably not going to have intended effect.

That said, I think it should work with Vector.

Can you test this little change for me (I am too lazy to setup proper testing code now...):

```
template <class Ret>
class AsyncWork {
...
 const Ret&        Get()                        { ASSERT(imp); imp->co.Finish(); return imp->Get(); }
```

Can you eventually post here a complete testcase?

Mirek

Subject: Re: AsyncWork<Vector<T>> fails due to lack of copy-constructor
Posted by piotr5 on Mon, 06 Jan 2020 23:14:51 GMT
View Forum Message <> Reply to Message

```
In file included from /home/p/upp/uppsrc/Core/Core.h:342,
                 from /home/p/upp/uppsrc/Draw/Draw.h:6,
                 from /home/p/upp/uppsrc/RichText/RichText.h:4,
                 from /home/p/upp/uppsrc/CtrlCore/CtrlCore.h:4,
                 from /home/p/upp/uppsrc/CtrlLib/CtrlLib.h:4,
                 from /home/p/MyApps/arrow2circle/arrow2circle.h:4,
                 from /home/p/MyApps/arrow2circle/main.cpp:1:
/home/p/upp/uppsrc/Core/CoWork.h: In instantiation of 'Ret2
Upp::AsyncWork<Ret>::Imp<Ret2>::Get() [with Ret2 = Upp::Vector<unsigned char>; Ret =
Upp::Vector<unsigned char>]':
/home/p/upp/uppsrc/Core/CoWork.h:195:99:   required from 'Ret Upp::AsyncWork<Ret>::Get()
[with Ret = Upp::Vector<unsigned char>]'
/home/p/MyApps/arrow2circle/arrow2circle.h:28:18:   required from here
/home/p/upp/uppsrc/Core/CoWork.h:173:50: error: use of deleted function 'constexpr
Upp::Vector<unsigned char>::Vector(const Upp::Vector<unsigned char>&)'
  173 |   Ret2 Get()                        { return ret; }
      |                                     ^~~~
```

similar if I add "pick(ret)" in place of "ret"

```
/home/p/upp/uppsrc/Core/CoWork.h: In instantiation of 'Ret2&
Upp::AsyncWork<Ret>::Imp<Ret2>::Get() [with Ret2 = Upp::Vector<unsigned char>; Ret =
Upp::Vector<unsigned char>]':
/home/p/upp/uppsrc/Core/CoWork.h:195:99:   required from 'Ret Upp::AsyncWork<Ret>::Get()
[with Ret = Upp::Vector<unsigned char>]'
/home/p/MyApps/arrow2circle/arrow2circle.h:28:18:   required from here
/home/p/upp/uppsrc/Core/CoWork.h:173:59: error: cannot bind non-const lvalue reference of type
'Upp::Vector<unsigned char>&' to an rvalue of type
'std::remove_reference<Upp::Vector<unsigned char>&>::type' {aka 'U
    pp::Vector<unsigned char>'}
```

my code is just simple file-processing, here a short version:

```
#include <Core/Core.h>
#include <array>
using namespace Upp;

struct Parse {
 using CLASSNAME=Parse;
 std::array<AsyncWork<Vector<byte> >,3 > parser;
 FindFile path;

 void init(const String& p) {path.Search(p); for(auto&& i:parser) {i.Cancel();SetWork(i);};}
 Vector<byte> Work(const String& file);
 void SetWork(AsyncWork<Vector<byte> >& where) {if(path&&path.IsFile()&&path.GetLength()>0)
{
  where.Do(THISFN(Work),path.GetPath());
  path.Next();
 };}
 void AddWork() { for(auto&& i:parser) if(i.IsFinished()) {
  auto out=i.Get();
  SetWork(i);
 }}
};

Vector<byte> Parse::Work(const String& file) {
 Vector<byte> out;
 return out;
}
CONSOLE_APP_MAIN
{
 Parse p;
 p.init(CommandLine()[1]);
}
```

this works with

```
  Ret Get()                    { return pick(ret); }
```

and also if I put "Ret&&" in place of "Ret" and use pick as above. using "std::forward<Ret2>(ret)" is a better idea here? (I compile with "-std=gnu++1z" and gcc-9.2.0)

Oh, and thanks for the hint. I'll try using CoWork to see if it's any better than my boilerplate...

Subject: Re: AsyncWork<Vector<T>> fails due to lack of copy-constructor
Posted by mirek on Tue, 07 Jan 2020 14:16:08 GMT
View Forum Message <> Reply to Message

OK, in trunk, AsyncWork is now able to get by here, but you need to call Pick instead of Get.

That said, still not 100% sure what is the goal, but if you wanted to process files in parallel, I have cooked up a little example for you for future reference:

```
#include <Core/Core.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 FindFile ff("c:/xxx/csv/*.csv");
 Mutex lock;
 int total_lines = 0;
 CoDo([&] {
  int lines = 0;
  for(;;) {
   String path;
   {
    Mutex::Lock __(lock);
    while(ff && !ff.IsFile())
     ff.Next();
    if(!ff) {
     total_lines += lines;
     return;
    }
    path = ff.GetPath();
    ff.Next();
    Cout() << "About to process " << path << "\n";
   }
   FileIn in(path);
   while(!in.IsEof()) {
    in.GetLine();
    lines++;
   }
  }
 });

 Cout() << "Total number of lines is " << total_lines << "\n";
}
```