Subject: Re: bug in CoWork since C++11
Posted by mirek on Tue, 09 Aug 2016 09:43:51 GMT
View Forum Message <> Reply to Message

crydev wrote on Tue, 09 August 2016 11:36mirek wrote on Tue, 09 August 2016 11:15Well, the one possible explanation is that your CoWork is global (or static) variable. Is that so?

My CoWork instance is not a global variable. It is a variable as member of a class and it is not a pointer. It is a singleton class, though. I construct it with a private constructor and a GetInstance method.

```
class X
{
private:
    CoWork mThreadPool;
    X();
public:
    static X* GetInstance()
    {
        static X instance;
        return &instance;
    };
}
```

I see. It is static then.

I believe what happens here is that first, CoWork instance is constructed. Then you schedule the work, which creates global static thread pool. On app exit, pool is therefore destructed BEFORE destructor of CoWork, which would normally performed the remaining work (by calling Finish).

Can you try to call GetInstance()->Finish() at the end of APP_MAIN to prove this hypothesis? (if true, I will then start thinking if resolving this situation is worth the trouble or rather mentioning in docs...

Mirek