Subject: Re: About storing references and pointers to callbacks.
Posted by mirek on Sun, 25 Jun 2017 19:34:16 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sun, 25 June 2017 21:13Hello,

I'd like to ask you a question. Here is the problem I need to solve:

I need to store pointers of complex objects, say, Streams, to callbacks (e.g. for deferred/async file reads and writes) so that I can access them only when I need them.
The culprit is that I don't want the caller function (or callback) to own those objects. Knowing their current state -whether they are destroyed or existing- to proceed or to halt is sufficient.

I know that simply passing pointers is dangerous, since the life time of objects can vary and not be strictly determined especially on complex applications.
Now, I know C++11 and above versions of C++ standard have std::shared_ptr and std::weak_ptr suitable for this purpose.
Also U++ has something similar: Ptr and Pte.

What would be the U++ way to handle these situations?

```
MyClass::MyAsyncReadMethod(Stream& s)
{
	Stream *p = &s; // This is a bad, very bad practice!

	vector_containing_callbacks.Add() << [=] {

		p->GetLine() // !! might eat cats!

		  ?? // How should I proceed?
		    // Should I use shared_ptr && weak_ptr?
		    // Or A Stream derivative with Ptr && Pte?
		    // or another alternative?
	};
}
```

Any suggestions or ideas will be much appreciated. Thanks.

Bestt regards,
Oblivion

It all depends on context, which you do not provide. However, in general, I would guess in similar situations destroying stream before MyClass does not make much sense from user perspective. So you can go along with it, even maybe with [=, &s] and just put something like "Stream must

exist through the duration of MyClass (or until calling some methods that removes it from MyClass).