
Subject: calling delete on pointer

Posted by [mtdew3q](#) on Tue, 10 Sep 2019 04:18:26 GMT

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

Hi all-

I am trying to figure out why the smart pointers are handy.
I called delete on a smart pointer and unless I specifically
set it to NULL after I call delete it doesn't return NULL
with the T * operator->() method.

I'd like my code to be able to detect that the smart pointer
is NULL.

I am not sure I understand the advantage if after I call delete
that it doesn't detect that delete has been called.

Here is the offending code:

Thanks for any cool help:

```
struct Foo: Pte<Foo> {  
  
    String s1;  
    Foo * f4;  
    Foo() {  
        s1 = String() << "Hello Jim";  
    }  
    String & myfunc () {  
        return s1;  
    }  
  
    Foo * operator-> () {  
        return f4;  
    }  
};
```

```
GUI_APP_MAIN  
{  
    Foo * f = new (Foo) ;  
  
    String str = f->myfunc();  
    delete f;  
  
    if (!f->operator->())  
        PromptOK("null");  
    else  
        PromptOK(str);  
}
```

Subject: Re: calling delete on pointer
Posted by [mirek](#) on Tue, 10 Sep 2019 06:39:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pte/Ptr are supposed to solve different problem:

```
Ptr<Foo> ptr;  
{  
    Foo x;  
    ptr = &x;  
}  
ASSERT(!ptr);
```

In other words, the mechanism is supposed to make Ptr NULL when the object that is pointed to by it is destroyed.

Mirek

Subject: Re: calling delete on pointer
Posted by [mtdew3q](#) on Tue, 10 Sep 2019 12:38:19 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi mirek,

Thanks!

Jim
