Subject: CHECK macro
Posted by exolon on Fri, 27 Oct 2006 01:26:33 GMT
View Forum Message <> Reply to Message

Hi, I was wondering if anyone can explain the purpose of the CHECK macro defined in Core/Diag.h? Is it an assertion macro?

It's a very generic name, and since macros seem to have no scope or namespaces (I don't know much about preprocessor stuff since I don't use it much), it conflicts in my case with the unit testing library I use (UnitTest++). This means I either have to rename the macro in the testing library which doesn't seem right, or rename the one in Diag.h and wherever it's used. This is what I've done before, but I don't like having to do it every time I move to the next version of UPP. It causes trouble when I include Core/Core.h to use the String class, for example.

Does anyone have any ideas about handling this more easily?

Subject: Re: CHECK macro
Posted by mirek on Fri, 27 Oct 2006 01:41:46 GMT
View Forum Message <> Reply to Message

It is similar to ASSERT, but expression gets evaluated even in release mode (therefore you do not lose desired side effects - e.g. you can use CHECK to test the return code of some routine. If you would do it with ASSERT, routine would not get called in release mode).

I guess the simple thing to do is #undef CHECK before including your unit testing stuff.

Mirek

P.S.: Unit testing is on my radar for future U++ development, perhaps you could share your opinions in develeopment forum...

Subject: Re: CHECK macro
Posted by exolon on Fri, 27 Oct 2006 02:05:12 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 27 October 2006 02:41It is similar to ASSERT, but expression gets evaluated even in release mode (therefore you do not lose desired side effects - e.g. you can use CHECK to test the return code of some routine. If you would do it with ASSERT, routine would not get called in release mode).
So it's a sort of design-by-contract "defensive programming" thing?

luzr wrote on Fri, 27 October 2006 02:41I guess the simple thing to do is #undef CHECK before including your unit testing stuff.
Well, doing this caused a lot of errors since CHECK was redefined and some code in Core is trying to use it. It's not a big deal, I made a terrible sed script to do some of the replacements. And

if, as you say, unit testing is a future consideration for U++ (a very good thing!!), I'll participate as much as my limited experience allows on that topic.

Subject: Re: CHECK macro
Posted by mirek on Fri, 27 Oct 2006 06:46:05 GMT
View Forum Message <> Reply to Message

Oh, I was thinking about something like:

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

#undef CHECK

#include <unittest.h>
```

should not cause any problems with U++....

Mirek

Subject: Re: CHECK macro
Posted by exolon on Fri, 27 Oct 2006 21:11:01 GMT
View Forum Message <> Reply to Message

Hmm, you're right Mirek, that seems to work fine now. I was sure I tried this before and it caused many errors, so I renamed it in uppsrc instead.

However, with upp610-dev2 undefining CHECK before the unittest library does indeed work fine.

```
// trivial example
#include <Palindrome/Palindrome.h>
#undef CHECK
#include <UnitTest++.h>

TEST(CanInstantiate) {
 String test = "abc";
 Palindrome palindrome(test);
}

TEST(FalseForNonPalindrome) {
 String test = "abc";
 Palindrome palindrome(test);
 CHECK_EQUAL(palindrome.isPalindrome(), false);
}
```

```
TEST(TrueForPalindrome) {
 String test = "amanaplanacanalpanama";
 Palindrome palindrome(test);
 CHECK_EQUAL(palindrome.isPalindrome(), true);
}

int main(int, char **) {
 return UnitTest::RunAllTests();
}
```
Thanks, that saves me trouble

Subject: Re: CHECK macro
Posted by mirek on Sat, 28 Oct 2006 14:35:22 GMT
View Forum Message <> Reply to Message

exolon wrote on Fri, 27 October 2006 17:11Hmm, you're right Mirek, that seems to work fine now. I was sure I tried this before and it caused many errors, so I renamed it in uppsrc instead.

I bet you tried #undef before #include of U++ headers

Mirek
