Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 26 Sep 2007 17:26:25 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 26 September 2007 12:44luzr wrote on Wed, 26 September 2007 18:25sergei wrote on Wed, 26 September 2007 09:13I'd love a unique ID generator, but that wouldn't be portable.

Current implmentation IS FULLY PORTABLE. What makes it unportable is SCU. You cannot simply include everything into a single file and expect it to work - that is not 100% C/C++ compatible.

Quote:
P.S. I personally dislike the idea of INITBLOCK. Such code would be difficult to debug in case something goes wrong, since it executes before main. And it's scattered in the source.

That is the point. Module intialization is placed into the module. No action is required by client besides adding the module.

Mirek

I was talking about your BLITZ_INDEX__ suggestion - that would work only with BLITZ.

Why? You can use it in any SCU system. Actually, it just makes U++ sources more usable with SCU approach.

Quote:
But once it's made to work, it can be maintained relatively easily.

Once you create some release system, it can be as easily maintained as it is.

Quote:
Just tell me if it's fine to change the INITBLOCKs to INITBLOCK_s like I did, or I'll have to find another way around.

Well, I wanted to tell you that I am not going to avoid this feature (I mean, unnamed INITBLOCK) only because of SCU experiment. Means, I do not have a problem with it, but expect to fix these for each release...

Quote:
There is no easy way to go through these initializations while debugging, one by one.

Maybe not one by one, but breakpoints still work....

Quote:
I'm probably unclear, so here's my idea: create a global array of pointers to functions (get void, return void) - like 1000 cells, and an index saying how many pointers are actually stored. In INITBLOCKs, instead of doing something, the only action would be to append the initialization function to the global array (and increment index). That way, when main is started, you'll have an array with pointers to functions you have to call to initialize everything.

Actually, that is exactly how global constructors (and therefore INITBLOCKs) are implemented

Quote:
Add a for loop there, that would initialize things. The pro is that you then can debug these initializations by stepping in during the loop. You'll also be able to do something before the initializations, in case that will ever be necessary (who knows, charset setup or something). Additionally, INITBLOCK could look like INITBLOCK(MyPkgInitFunc), and as long as the initialization
function name is meaningful, it probably would be unique.

Yes, great. So instead of calling initializations directly, we will have an array of functions (global constructors). This array will be performed by C++ runtime to create another array of functions and that one will be performed in main?

Also, if you need to do something more, you simply cannot use INITBLOCK.

Mirek