Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 26 Sep 2007 16:44:29 GMT
View Forum Message <> Reply to Message

luzr 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. INITBLOCK without BLITZ, is indeed portable without SCU. I agree that SCU isn't C/C++ compatible. But once it's made to work, it can be maintained relatively easily. + If code works with SCU you don't have to worry about naming if you wish to transfer some code from one file to another. But why argue? 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.

I'm not against the idea that the only action required is adding the module, I actually like that. What I dislike is the code scattered among constructors. There is no easy way to go through these initializations while debugging, one by one.

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. 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. MyPkgInitFunc would contain all the register stuff, and it could be debugged. Con is that you don't know how many INITBLOCKs will be, so you'll have to statically allocate a big enough array of function pointers.