Subject: creating DLLs HOWTO ??
Posted by kohait00 on Thu, 19 Nov 2009 10:56:12 GMT
View Forum Message <> Reply to Message

Hi there

i was searching the forum up and down for some kind of practical info on how to build a DLL project, which i.e exports some classes (same as you can do with visual studio) with the current U++ releases. And that using both MINGW and MSC9 (current win SDK proposed during install). There is no/little quite usable info around this issue. So this is a try to put together what i found out so far, this definitely should be xtended and maybe put somewhere in docs..

So far my steps taken are the following:

Asume you have an u++ application, that wants to loadtime bind to a dll, exorting some functionality classes, also created in u++ (the fact that is is difficult and why, like mirek stated, should be cleared somehow, moreover, the fact, that you actually include double code, i.e Core in the dll and Core in the application is ignored for now)

general setup: U++ 1679, Windows 2003 SDK, like stated during install, mingw 4.4.0 (should u++ update to current?) and MSC9

//creating a dll

1) create a core console, creating also the header name it after your dll
2) remove the CONSOLE_APP_MAIN stuff in the main cpp file
3) setup the configuration of package to additionally have DLL flag
4) make your code, your classes and so on.
5) build your dll project, both MINGW and MSC will produce a .dll file (MSC *also* should produce a .lib file, but doesnt, thats why it later wont work, WHY?)

//creating the app project that will use the dll
1) create a ctrllib (with or without main window)
2) in your app header #include <YourDllPackage/YourDllPackage.h>, this should make the dll headers and includes available, together with the class description (take care of really having declaration and implementation in your dll well seperated)
3) in package organizer, for your app package add a library dependencie with rightcklick/new library

4) compile with MINGW: for that purpose you either have to adjust the PATH or buildmethods PATH stuff to make the previously compiled dll available during link time, i simply compied it to my mingw/lib for short try
5) run your application: copy the dll also in the executable folder of your app (build/open output folder), this should work.

4) compile with MSC9: now this is the problem: linker states it wants to have the .lib file which was not generated when compiling the dll. if it had, one needed to copy it somehpw in the package

folder of the app package, and also in the output folder, from where the app would run.

i admit, the path stuff and so on is subject to optimize, was just a short shot, but it should work anyway.

DEPENDENCY WALKER: the mingw compiled dll had the exported functions form all the used packages (Core, etc), the MSC dll had NO exported symbols at all

any ideas on how to do that? this is espacially neccessaru when one wants to keep one's code portable and usable on windows and linux..

thanks for help

PS: attached is a TestDll and a TestDllApp package, demonstrating the goal

File Attachments

1) TestDll.zip, downloaded 378 times