Subject: umk to generate .o file
Posted by koldo on Fri, 31 Jan 2014 14:32:45 GMT
View Forum Message <> Reply to Message

Hello!

I would like to create a .o file alone with umk (with MinGW in Windows).

Do you know how to do it?

Subject: Re: umk to generate .o file
Posted by koldo on Sat, 01 Feb 2014 09:39:02 GMT
View Forum Message <> Reply to Message

This is a trick: Maybe doing a dummy executable that uses that compiles that .o, and copying it after compiling?

Subject: Re: umk to generate .o file
Posted by koldo on Sat, 01 Feb 2014 21:34:21 GMT
View Forum Message <> Reply to Message

Hello!

Trick done:
1. Compile dummy project
2. Get .a files. Add a "lib" to the names beginning so Core.a is libCore.a and so on
3. Include the .a files in gcc command, so it is added "-lz -lCore -lWeb"

4. However, gcc
complains:C:/MyProject/lib/libCore.a(z.o):z.cpp:(.text$_ZN3Upp4Zlib4FreeEv+0x14): undefined reference to `inflateEnd'
C:/MyProject/lib/libCore.a(z.o):z.cpp:(.text$_ZN3Upp4Zlib4FreeEv+0x22): undefined reference to `deflateEnd'
...
inflateEnd, deflateEnd and others are defined in z package, deflate.c file.

What could be the problem?

Subject: Re: umk to generate .o file
Posted by koldo on Sat, 01 Feb 2014 21:42:36 GMT
View Forum Message <> Reply to Message

Solved!

Just changing the order in gcc command to -lCore -lz -lWeb.

This way, gcc can compile a non U++ project that uses U++

For sure somebody knows how to do it better. If yes please post your way to do it.

Subject: Re: umk to generate .o file
Posted by dolik.rce on Sat, 01 Feb 2014 23:02:59 GMT
View Forum Message <> Reply to Message

Hi Koldo

I don't know what you're trying to achieve, but I can give you two simplyfiing tips

1) You can skip the renaming if you use the name of the static lib instead instead of -l option. E.g.: instead of -lCore just write Core.a. If that doesn't work, you might need to add path...

2) Determining correct order in which the libs must be entered on the command line can be hard (or impossible, if there is cyclic dependency:)). You can use startgroup/endgroup linker options to make gcc work with libs in any order:gcc -Wl,--startgroup z.a Core.a Web.a -Wl,--endgroup

If all you need to do is to mix U++ and non-U++ code, I can recommend you to use the universal makefile to take care of the U++ part (unless you need it on Windows). It is capable of compiling separate files or packages...

Best regards,
Honza

Subject: Re: umk to generate .o file
Posted by koldo on Sun, 02 Feb 2014 07:45:38 GMT
View Forum Message <> Reply to Message

Hello Honza

In this case the mix is:
- The main() is inside a non U++ .cpp file
- Other non U++ .cpp and .o files have to be linked
- Some of these .cpp files call U++ libs

Is it possible to use umk in this case?

Subject: Re: umk to generate .o file

Posted by dolik.rce on Sun, 02 Feb 2014 10:02:50 GMT
View Forum Message <> Reply to Message

koldo wrote on Sun, 02 February 2014 08:45Hello Honza

In this case the mix is:
- The main() is inside a non U++ .cpp file
- Other non U++ .cpp and .o files have to be linked
- Some of these .cpp files call U++ libs

Is it possible to use umk in this case?
I don't see any reason why it would be a problem. The only obstacle is that you'd have to provide .upp file. You could do this in theide or manually. Just create main package containing all the non-U++ code and make it depend on the U++ packages it uses. Both umk and theide don't really care if the code is U++ or not.

Of course if the non-U++ part already uses some build system, this could mean duplicating some work. In that case, it might (or might not) be better to try to incorporate it into the other build system using the dummy project approach that you described above...

Honza

Subject: Re: umk to generate .o file
Posted by koldo on Mon, 03 Feb 2014 10:21:04 GMT
View Forum Message <> Reply to Message

Hello Honza

Quote:I don't see any reason why it would be a problem. The only obstacle is that you'd have to provide .upp file. You could do this in theide or manually. Just create main package containing all the non-U++ code and make it depend on the U++ packages it uses. Both umk and theide don't really care if the code is U++ or not.You are right. I will generate the .upp file automatically and call umk with it. Thank you .