Subject: Building U++ for MinGW32
Posted by pcfreak on Mon, 30 Jun 2014 19:42:01 GMT
View Forum Message <> Reply to Message

Hello, I am new to U++ and I would like to use it with my MinGW build.
I am not using any offical precompiled gcc.
My current MinGW build looks like this:
Using built-in specs.
COLLECT_GCC=D:\Programme\msys\mingw\bin\gcc.exe
COLLECT_LTO_WRAPPER=d:/programme/msys/mingw/bin/../libexec/gcc/mingw32/4.8.1/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-4.8.1/configure --enable-languages=c,ada,c++,fortran,objc,obj-c++ --disable-sjlj-exceptions --disable-nls --disable-shared --enable-static --enable-fully-dynamic-string --enable-libgomp --enable-lto --with-dwarf2 --disable-win32-registry --enable-version-specific-runtime-libs --enable-bootstrap --build=mingw32 --enable-abi=32 --enable-checking=release --prefix=/mingw --with-mpfr=/mingw --with-gmp=/mingw --with-mpc=/mingw
Thread model: win32
gcc version 4.8.1 (GCC)

Here the questions:
- Which libraries do I need for U++?
- Is there any how-to on building U++ for MinGW (on MinGW, no cross)?

I checked the Makefiles and saw a couple of dependencies...
-lbz2 -lpthread -lrt -lz -lfreetype -lfontconfig -lexpat -lgtk-x11-2.0 -lgdk-x11-2.0 -latk-1.0 -lgdk_pixbuf-2.0 -lm -lpangocairo-1.0 -lfontconfig -lXext -lXrender -lXinerama -lXi -lXrandr -lXcursor -lXfixes -lpango-1.0 -lcairo -lX11 -lgobject-2.0 -lgmodule-2.0 -lglib-2.0 -lX11 -lXrender -lXft -lnotify -lpng

But I doubt that this will work out fine for MinGW (especially the X part). This is why I am curious how this was done so far.
I could not find any explanation on this topic in google or the forum so far and I hope someone here can help me out.

PS: I do not plan to use any official MinGW or TDM builds.

Subject: Re: Building U++ for MinGW32
Posted by mirek on Wed, 02 Jul 2014 09:35:49 GMT
View Forum Message <> Reply to Message

pcfreak wrote on Mon, 30 June 2014 21:42Hello, I am new to U++ and I would like to use it with my MinGW build.
I am not using any offical precompiled gcc.
My current MinGW build looks like this:

Using built-in specs.
COLLECT_GCC=D:\Programme\msys\mingw\bin\gcc.exe
COLLECT_LTO_WRAPPER=d:/programme/msys/mingw/bin/../libexec/gcc/mingw32/4.8.1/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-4.8.1/configure --enable-languages=c,ada,c++,fortran,objc,obj-c++ --disable-sjlj-exceptions --disable-nls --disable-shared --enable-static --enable-fully-dynamic-string --enable-libgomp --enable-lto --with-dwarf2 --disable-win32-registry --enable-version-specific-runtime-libs --enable-bootstrap --build=mingw32 --enable-abi=32 --enable-checking=release --prefix=/mingw --with-mpfr=/mingw --with-gmp=/mingw --with-mpc=/mingw
Thread model: win32
gcc version 4.8.1 (GCC)

Here the questions:
- Which libraries do I need for U++?
- Is there any how-to on building U++ for MinGW (on MinGW, no cross)?

I checked the Makefiles and saw a couple of dependencies...
-lbz2 -lpthread -lrt -lz -lfreetype -lfontconfig -lexpat -lgtk-x11-2.0 -lgdk-x11-2.0 -latk-1.0 -lgdk_pixbuf-2.0 -lm -lpangocairo-1.0 -lfontconfig -lXext -lXrender -lXinerama -lXi -lXrandr -lXcursor -lXfixes -lpango-1.0 -lcairo -lX11 -lgobject-2.0 -lgmodule-2.0 -lglib-2.0 -lX11 -lXrender -lXft -lnotify -lpng

Makefile is for POSIX based systems only.

I believe that you will need to have basic Win32 import libraries:

advapi32 comdlg32 comctl32 user32 gdi32

that perhaps should be all.

Now I am unsure whether you are trying "clean build from the scratch" with mingw from makefiles (which is not supported at the moment), or if you just want to use your mingw version with theide, eventually recompiling theide itself without your mingw - that should more or less work out of the box, as long as you have all these libs available.

Mirek

Subject: Re: Building U++ for MinGW32
Posted by pcfreak on Mon, 07 Jul 2014 03:53:40 GMT
View Forum Message <> Reply to Message

Hello,
thank you for the reply. I wanted to build U++ as library with my MinGW so that I can integrate it into my tool chain.

Too bad to hear that this is unsupported. Are there plans to support this?

Subject: Re: Building U++ for MinGW32
Posted by mirek on Thu, 10 Jul 2014 07:48:30 GMT
View Forum Message <> Reply to Message

pcfreak wrote on Mon, 07 July 2014 05:53Hello,
thank you for the reply. I wanted to build U++ as library with my MinGW so that I can integrate it into my tool chain.
Too bad to hear that this is unsupported. Are there plans to support this?

Now, the question is what you really mean by toolchain.

One of defining features of U++ is the transparent modularity system (assemblies, nests, packages), which needs its own build system - that is why theide exists. It is possible to use commandline version of this build system ("umk"). It is also certainly possible to forget about it and try to create 'normal libraries'.

There is one not really resolved issue with converting U++ into normal library. U++ is using "global constructor trick" to initialize its parts.

To explain by example: there are various raster graphics formats supported by U++, and the support for each format has its own module ("package", e.g. plugin/png). If your application needs to support jpeg, you add "plugin/jpg" into the project. Now there exists in U++ Image basic libraries function that is able to load image in any format. Obviously this function needs to have a list "format loaders", which means that "plugin/jpeg" has to register itself so that this function knows about it. The act of registering is done using global constructor magic, which in reality looks like (with the use of macro)

```
INITBLOCK {
 StreamRaster::Register<JPGRaster>();
}
```

- INITBLOCK is the code, that has to be run on the start of application, before 'main'.

So far so good. But if you put this as single thing into say "init_jpeg.cpp" file, create a library from several files and link with your application, linker will simply exclude "init_jpeg.o" because there are no references to its code from the application.

That is why U++ build system (besides of course many other things) recognizes ".icpp" extension, which means ".cpp code that always has to be included in final product". (Of course, U++ build system does much more than that too :)

Now, any effort to change U++ into library has to deal with this in some way. One possibility is to change the U++ library code, replace all INITBLOCKs with some sort "InitXXX" function and make it mandatory for client code to call them all. Perhaps not very user friendly, as there are now total

109 INITBLOCKs in base U++ now...

Alternative solution, implemented a couple of years ago, is a little addition to theide that automatically creates "init" files that look like this:

```
#ifndef _plugin_jpg_icpp_init_stub
#define _plugin_jpg_icpp_init_stub
#include "Draw/init"
#define BLITZ_INDEX__ F227b3ee1134af92a2493f791a66e2afe
#include "jpgreg.icpp"
#undef BLITZ_INDEX__
#endif
```

- they include all .icpp files in package and also init files of packages used by package.

The idea here is that your application would #include just top-level <init> files in its main .cpp, which would provide a safe way to call all INITBLOCKs.

I guess this is perhaps smart, but to my knowledge, nobody used it yet in practice.

Other than .icpp issue, I would say U++ as library might have the issue with extreme modularity it has. I am afraid that it will be a bit hard to decide which packages to group into what libraries. But I guess it is doable.

Subject: Re: Building U++ for MinGW32
Posted by pcfreak on Sun, 13 Jul 2014 10:54:48 GMT
View Forum Message <> Reply to Message

Thank you for the detailed explaination.
I am familier with this constructor magic and I now that there are some addions needed to force their linkage.
Yes it is doable, but I guess that would be a bit more affort than what I thought it would be.
I thought there was a way to easily pack everything together into a library file (.a for gcc, .lib for vc) and link that to the final executable.
My toolchain simple consists of Makefiles calling GCC's compiler and linker. Nothing more.

Subject: Re: Building U++ for MinGW32
Posted by mirek on Mon, 14 Jul 2014 11:18:58 GMT
View Forum Message <> Reply to Message

pcfreak wrote on Sun, 13 July 2014 12:54Thank you for the detailed explaination.
I am familier with this constructor magic and I now that there are some addions needed to force their linkage.

Yes it is doable, but I guess that would be a bit more affort than what I thought it would be.
I thought there was a way to easily pack everything together into a library file (.a for gcc, .lib for vc) and link that to the final executable.
My toolchain simple consists of Makefiles calling GCC's compiler and linker. Nothing more.

Well, OTOH, perhaps the issue can be taken from the different angle.... Once you decide how to "split" uppsrc into libraries, it would be relatively easy to produce single "InitXXX" per library.

Splitting into libraries is quite interesting task, though.

Perhaps:

"Core" (including all non-GUI plugins and everything that can run in console mode, including Draw)
"GUI" (including everything that requires GUI, CtrlCore, CtrlLib, RichText, Report....)
"SQLCommon"
then there should be single interface library per DB engine (PHSQL, MYSQL...)

I guess, for the first iteration, that would be enough. Specialities like Skylark or Turtle are not needed now...

Mirek
