Subject: Using U++ with existing code
Posted by TRNG98 on Thu, 04 Feb 2010 13:47:21 GMT
View Forum Message <> Reply to Message

I have an existing C program that do all the work I need.
Can I connect this with an U++ application, best would compile them together? I need very little integration, some static global variables would do. Can absolutely not translate the C program to U++.

//
Best regards
Bo

Subject: Re: Using U++ with existing code
Posted by cbpporter on Thu, 04 Feb 2010 14:35:41 GMT
View Forum Message <> Reply to Message

Well U++ is normal C++ so the short answer is yes. Long answer is that there are a few steps that are necessary.

First step would be to compile your C program the way you do now, but replace the C compiler with C++ compiler. You might get a few easy to fix compilation errors, unless your program is C99. After you program compiles, create a package you should be done.

A few more extra steps that could be necessary depending on your sources are adjusting your include paths and deactivating BLITZ for the package that contains your former C sources, but in most cases you won't have to do this.

Subject: Re: Using U++ with existing code
Posted by dolik.rce on Thu, 04 Feb 2010 14:46:47 GMT
View Forum Message <> Reply to Message

TRNG98 wrote on Thu, 04 February 2010 14:47I have an existing C program that do all the work I need.
Can I connect this with an U++ application, best would compile them together? I need very little integration, some static global variables would do. Can absolutely not translate the C program to U++.

//
Best regards
Bo

Hi Bo!
Since C is a subset of C++ and U++ is still just C++, there is technically no problem in using them

together. Also TheIDE is aware of the difference between .c and .cpp files, so if you just add your existing files into the project, they should be compiled and linked correctly. A little work will be needed to bind the two parts (U++ and C) correctly, but that depends a lot on what do you want to do.

Typical case I can imagine would be writing U++ GUI for C console program. In that case, you would just write the GUI part, calling existing functions from the original sources (including necessary headers of course). Commenting out the original main() or renaming it is probably the only big change you would have to do in original sources in this case. This is just an example, if you supply more info, we can probably give you better hints

If you want to see a practical example, you can have a look into package uppsrc/plugin/ndisasm. That is package that includes C sources of libndisasm to provide disassembler used in TheIDE. It's probably not the best example (it won't compile as standalone app anymore, not sure why), but it should give you an idea about mixing U++ and C. Actually, most of the packages in uppsrc/plugin are using existing C code, so just have a look in that direction

Best regards,
Honza

Subject: Re: Using U++ with existing code
Posted by koldo on Thu, 04 Feb 2010 15:12:17 GMT
View Forum Message <> Reply to Message

Hello TRNG98

And additionally to the last comments, do not forget to add a:

```
extern "C" {
    Declarations of C functions called from C++
}
```

in the .cpp files where functions in .c are called.

Subject: Re: Using U++ with existing code
Posted by TRNG98 on Thu, 04 Feb 2010 19:37:32 GMT
View Forum Message <> Reply to Message

Thank you for your help -- I'll hack around a while --
The C is multi-threaded and run some API functions (not Win API:s).
I don't want to rewrite it.

// Bo

Subject: Re: Using U++ with existing code
Posted by mirek on Fri, 05 Feb 2010 09:17:28 GMT
View Forum Message <> Reply to Message

TRNG98 wrote on Thu, 04 February 2010 14:37Thank you for your help -- I'll hack around a while --
The C is multi-threaded and run some API functions (not Win API:s).
I don't want to rewrite it.

// Bo

Multithreaded: There is one thing that you should know: U++ uses own memory allocator that need some support in threads - basically, as allocator keeps per-thread cache, this cache has to be released when thread finishes.

For you, in practice, this should not be a problem, as long as you do not use new/delete in threads of your C application. But new/delete is (likely to be) used if actually call any U++ code from any non-main thread!

Simple way how to fix this is to specify "USEMALLOC" flag - that kicks out our memory allocator and uses plain malloc/free from the C library (but performance of U++ drops - IMO not an issue if you app depends on malloc/free anyway).

Best solution is to use U++ Thread class for threading. Good solution is to release the per-thread cache at the end of your threads (by calling MemoryFreeThread()

Mirek