Subject: How to determine if U++ is being utilized...
Posted by ptkacz on Thu, 02 Apr 2020 03:37:06 GMT
View Forum Message <> Reply to Message

Hi All,

I'm working on a library that will be made use of by a few different applications. For the applications that require a GUI, U++ will be utilized. In cases where no GUI is required, the standard C++ library will be made use of. While a non-GUI application will not require GUI visual elements, where needed, they may for example, still need need to process image data.

When developing code (I.e. classes or functions, etc) that makes use of U++'s framework, there are certain dependencies that will probably be inherited. The library becomes dependant on the U++ framework. If that same class or functions are to be reused outside of the U++ framework, they can't because of certain dependencies they are built around, dependencies no longer available.

Likewise, if code was first designed outside of the U++ framework, and later incorporated into a U++ program, the code would have to place nice in the U++ sandbox.

Is there a way for a library to determine or know if U++ is being made use of?

Peter

Subject: Re: How to determine if U++ is being utilized...
Posted by Novo on Thu, 02 Apr 2020 04:34:40 GMT
View Forum Message <> Reply to Message

You could check for an Upp-specific define. flagMT, for example.
#define flagMT // MT is now always on

Subject: Re: How to determine if U++ is being utilized...
Posted by Novo on Thu, 02 Apr 2020 22:44:23 GMT
View Forum Message <> Reply to Message

Another candidate is UPP
#ifdef UPP
...
#else
...
#endif

Subject: Re: How to determine if U++ is being utilized...
Posted by ptkacz on Sat, 04 Apr 2020 00:50:11 GMT
View Forum Message <> Reply to Message

Thanks Novo, to both your answers. I'm familiar with defining macros and making use of the logic directives (i.e. #if, etc). I was looking for something that could avoid having to manually make a macro definition, or prior inclusion, and then have to always make sure that things are defined or not when hopping from one application (project) to another.

After a little bit more digging on the web, it looks like C++17 has added to it for use with the marco pre-processor, a __has_include() function that checks if a header reference exists. Yesterday after updating my system, I tested out the following:

In TheIDE:

```
#if __has_include("CtrlLib/CtrlLib.h")
    CtrlLayout(*this, "U++ Window title");
#else
    CtrlLayout(*this, "U++ not here");
#endif
```

and even,

In Code::Blocks:

```
#if __has_include("iostream")
    cout << "iostream here." << endl;
#else
    cout << "iostream not here" << endl;
#endif

#if __has_include("CtrlLib/CtrlLib.h")
    cout << "U++ Window here." << endl;
#else
    cout << "U++ not here" << endl;
#endif
```

__has_include() worked in all circumstances, so it looks like I'll have to be making C++17 for just this one feature.

Some additional information on it can be found here,
https://en.cppreference.com/w/cpp/preprocessor/include

Again, thanks for your input!

Peter

Subject: Re: How to determine if U++ is being utilized...
Posted by Novo on Sat, 04 Apr 2020 04:34:01 GMT
View Forum Message <> Reply to Message

IMHO, you are over-complicating very simple things.
The whole idea of U++ is about turning very complicated things into very simple.
That is the point of U++. And you are trying to solve your problem in a completely opposite way.

Just my two cents.

Subject: Re: How to determine if U++ is being utilized...
Posted by Novo on Sat, 04 Apr 2020 04:53:28 GMT
View Forum Message <> Reply to Message

BTW, __has_include() will detect U++ even if you are not using it ...
It is enough to just add path to uppsrc to include paths.

Subject: Re: How to determine if U++ is being utilized...
Posted by Novo on Sat, 04 Apr 2020 13:20:14 GMT
View Forum Message <> Reply to Message

On second thought, __has_include() helps to get rid of configuration tools like CMake/configure.
This seems to be a nice addition to standard C++.
Thanks for pointing out.