Subject: So many preprocessor defines for platform !
Posted by jibe on Tue, 09 Apr 2013 12:56:58 GMT
View Forum Message <> Reply to Message

Hi,

Trying (for the first time) to develop something for various platforms, I get very confused about what to use to have platform specific parts of code.

Those preprocessor directives can be found (maybe more... I just grep'ed the upp folder) :

#ifdef BSD
#ifdef POSIX
#ifdef SOLARIS
#ifdef WIN32
#ifdef WIN64
#ifdef WINDOWS
#ifdef WINVER
#ifdef WIN_SHM_BASE
#ifdef flagPOSIX

#ifdef PLATFORM_BSD
#ifdef PLATFORM_FREEBSD
#ifdef PLATFORM_LINUX
#ifdef PLATFORM_OSX11
#ifdef PLATFORM_POSIX
#ifdef PLATFORM_SOLARIS
#ifdef PLATFORM_WIN32
#ifdef PLATFORM_WINCE
#ifdef PLATFORM_X11

#ifdef _WIN32
#ifdef _WIN32_WCE
#ifdef _WIN32_WINNT
#ifdef _WIN64
#ifdef _WINDOWS

#ifdef __WIN32__
#ifdef __WIN64__
#ifdef __linux__

Could somebody explain (or point to some doc or explanation) why so many, what are the differences and which one must be used in which case ?

ie : how do I choose between #ifdef WIN32, #ifdef PLATFORM_WIN32, #ifdef _WIN32 or #ifdef __WIN32__ ?

And when I #define one of them, are the others automatically defined or undefined ? That is, if I #define PLATFORM_LINUX, will be __linux__, POSIX, flagPOSIX and PLATFORM_POSIX defined, and all ones regarding WINDOWS, OSX, BSD etc. undefined ?

Subject: Re: So many preprocessor defines for platform !
Posted by jibe on Thu, 11 Apr 2013 13:33:44 GMT
View Forum Message <> Reply to Message

Hi,

I cannot believe that nobody knows !

Is my question stupid ? (just say why...)

Did I missed some important doc ? (at least, say "RTFM" or "STFW", or better, please point me to this doc that I was unable to find)

Please, could those of you making multi-platform applications at least tell me what they use ?

Thanks.

Subject: Re: So many preprocessor defines for platform !
Posted by Zbych on Thu, 11 Apr 2013 14:20:59 GMT
View Forum Message <> Reply to Message

My guess is that different libraries (plugins) use different definitions and that's why there is plenty of them.

For example:
_WIN32 - png, sqlite, bz2
WIN32 - z lib, png, tiff
__WIN32__ - png, tiff
__linux__ - png

and so on.

I think you should use PLATFORM_* definitions in your upp code.

```
void MyMultiplatformFunction(...)
{

common code
```

#ifdef PLATFORM_LINUX

code specific to linux only

#elif defined(PLATFORM_POSIX)

code specific to posix systems (linux, bsd, solaris, osx?)

#elif defined(PLATFORM_WIN64)

code specific to windows 64-bit only

#elif defined(PLATFORM_WIN32)

code specific to windows 32/64

#endif

common code

}

Subject: Re: So many preprocessor defines for platform !
Posted by dolik.rce on Thu, 11 Apr 2013 14:25:09 GMT
View Forum Message <> Reply to Message

Hi jibe,

I was hoping someone more informed than me will answer, but it seems it's up to me I didn't check all the things I'm saying here, so take it just as "educated guess"

First of all, the list you posted can be divided into two parts:
a) macros defined by U++ (PLATFORM_*, WIN32, BSD, POSIX etc.)
b) macros defined by the compiler or other libraries (__linux__, __WIN64__, _WIN32)

You should probably be ok if you only used the first group. The PLATFORM_* macros are quite generic and should satisfy most of your needs. I think the rest (WIN32, BSD, POSIX, ...) are meant to be used by U++ internally. Good place to start looking for how all the flags are defined and used should be Core/config.h.

Also note that some of the macros have overlapping meanings, e.g. PLATFORM_POSIX is defined together with PLATFORM_BSD on BSD and with PLATFORM_LINUX on Linux. It always depends on what exactly you need to check, but with basic knowledge about the target platforms, you can ussualy guess easily which one should be used.

Best regards,

Honza

Subject: Re: So many preprocessor defines for platform !
Posted by jibe on Thu, 11 Apr 2013 20:53:27 GMT
View Forum Message <> Reply to Message

Hi,

Thanks for your replies !

Core/config.h is interresting, but just confirms my worries : for example, PLATFORM_WINCE is not used (when it is in some libraries/examples and could be necessary in some applications), same for PLATFORM_X11, PLATFORM_WIN64 is defined for MS compiler, not for GNUC compiler...

Maybe, I worry too much about not very important things. But I'm developping an application that will run on various computers, at least LINUX, Windows XP, 7, probably 8 and Windows CE. And it will be very difficult to compile on various computers, I should prefer to cross-compile all on my linux computer.

So, I need to be sure that all will be correctly defined, and know how to do the right definitions.

Subject: Re: So many preprocessor defines for platform !
Posted by dolik.rce on Fri, 12 Apr 2013 06:21:39 GMT
View Forum Message <> Reply to Message

It all depends on what exactly you need to do, i.e. what platform specific code you need to write. U++ in itself is ready to be used in multiplatform applications. And if you plan to use some low level stuff, which can be very specific, you can always use directly the macros defined by the compiler (e.g. _WIN32 in MSVC or __linux for GCC on linux).

Also, you mention compiling on various systems vs. cross-compiling. I think these days the best choice is to use virtualbox, wmware or some other virtualization tool and just create virtual machines for all the target systems. In some cases you don't even have to go as far using virtualization, e.g. windows builds can be easily done using Wine + windows version of TheIDE.

Honza

Subject: Re: So many preprocessor defines for platform !
Posted by jibe on Fri, 12 Apr 2013 07:14:30 GMT
View Forum Message <> Reply to Message

Hi,

dolik.rce wrote on Fri, 12 April 2013 08:21It all depends on what exactly you need to do, i.e. what platform specific code you need to write. U++ in itself is ready to be used in multiplatform applications. And if you plan to use some low level stuff, which can be very specific, you can always use directly the macros defined by the compiler (e.g. _WIN32 in MSVC or __linux for GCC on linux).
Ok, so there is no more complete list than Core/config.h ? Well, I'll have to look for the ones I need... and to complete the C++ build flags page !

dolik.rce wrote on Fri, 12 April 2013 08:21I think these days the best choice is to use virtualbox, wmware or some other virtualization tool
Yes, it's surely the best way when you have to compile and test. I was thinking to use cross-compiling because I'll have all the systems to test, and it was to avoid to install UPP on each one, and make changes on various machines that must be centralized at the end...

But my question was more especially about the flags to use for every specific part.

Well, I'll study all that and do some testing. Thank for your help