Subject: U++ as .lib
Posted by mirek on Thu, 20 Sep 2007 13:07:49 GMT
View Forum Message <> Reply to Message

I would like to discuss all options and problem for releasing U++ as .lib.

My take (Sergei will disagree, but that is what discussion is for:):

Release simple .lib files for MSVC and MINGW (later perhaps package them for Dev-C++ and CodeBlocks).

Release files are just that - release files.

Debug files would be without debug info with assert or perhaps with lines info only to reduce the size.

It would have to come with at least part of sources too.

I will add "file edit mode" to theide so that users of other environments have a chance to edit .iml / .lay and perhaps even .tpp.

I would use theide and some custom program to build libs... Actually, I think I can extend our current release code ("MakeInstall") to generate all .libs for Win32 as part of standard release process...

(OK, the only funny part about all this is that .lib package will be longer

Hm, an idea: As it seems logical to ship theide with it anyway (as .lay and .iml editor), may the libraries could be built after the installation using theide?

Mirek

Subject: Re: U++ as .lib
Posted by sergei on Thu, 20 Sep 2007 13:28:42 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 20 September 2007 15:07I would like to discuss all options and problem for releasing U++ as .lib.

My take (Sergei will disagree, but that is what discussion is for:):

Release simple .lib files for MSVC and MINGW (later perhaps package them for Dev-C++ and CodeBlocks).

Release files are just that - release files.

Debug files would be without debug info with assert or perhaps with lines info only to reduce the size.

It would have to come with at least part of sources too.

I will add "file edit mode" to theide so that users of other environments have a chance to edit .iml / .lay and perhaps even .tpp.

I would use theide and some custom program to build libs... Actually, I think I can extend our current release code ("MakeInstall") to generate all .libs for Win32 as part of standard release process...

(OK, the only funny part about all this is that .lib package will be longer

Hm, an idea: As it seems logical to ship theide with it anyway (as .lay and .iml editor), may the libraries could be built after the installation using theide?

Mirek

Actually, I mostly agree. Without a lib for release mode, penalty would be either huge build times (5-10 mins easily) or moderate EXE size increase (due to SCU). However, my idea was to provide an easy way for users to build the lib. That's a common way in open-source cross-platform projects, and it would remove the need to maintain up-to-date built libs for several compilers. There might be a compiler we're not aware of / don't support, and yet works with U++.

File edit mode would be great - just associate all these files with TheIDE.

IMHO debug libs aren't necessary - better to just use the sources (especially if you intend to make changes to U++). + what's the size of a debug lib, if release lib is 10MB? But that's just my opinion

---

Subject: Re: U++ as .lib
Posted by mirek on Thu, 20 Sep 2007 15:00:51 GMT
View Forum Message <> Reply to Message

sergei wrote on Thu, 20 September 2007 09:28
However, my idea was to provide an easy way for users to build the lib. That's a common way in open-source cross-platform projects, and it would remove the need to maintain up-to-date built libs for several compilers.

The common way is to use make (and perhaps some scripts).

There is little problem doing that. There is even makefile generator in theide.

Mirek

Subject: Re: U++ as .lib
Posted by sergei on Sat, 22 Sep 2007 15:44:28 GMT
View Forum Message <> Reply to Message

I tried to build U++ as a lib in MSVC8. The regular way, no SCU, no precompiled headers.

Results:
Debug - 76MB / 10 mins
Release - 353MB / 8 mins

While build times are very good, resulting size is quite a concern. Either I did something wrong, or lib in MSVC is absolutely unusable. I'll retry in Code::Blocks, and see if I get 400MB/10MB sizes I got before. If so, lib in MSVC probably won't work...

Subject: Re: U++ as .lib
Posted by sergei on Sat, 22 Sep 2007 20:22:38 GMT
View Forum Message <> Reply to Message

Bad news I've rebuilt with Code::Blocks/MinGW, and here are the results:

Debug: 548MB / 132 mins
Release: 13MB / 112 mins

That's similar to my previous results (size increase is probably due to increased number of working packages). I have to conclude that MSVC indeed builds incredibly huge release libs... This might be related to the compiler/linker bug on release, but this time I didn't use SCU, and it worked.

Subject: Re: U++ as .lib
Posted by cbpporter on Sat, 22 Sep 2007 22:04:16 GMT
View Forum Message <> Reply to Message

An how much time does it take to compile and link a test app with these libs. On my computer it takes 2-4 seconds in debug mode.

Subject: Re: U++ as .lib

Posted by sergei on Sat, 22 Sep 2007 23:43:17 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sun, 23 September 2007 00:04An how much time does it take to compile and link a test app with these libs. On my computer it takes 2-4 seconds in debug mode.

You also built U++ libs?
Hmm... since I've built the libs already, why not test

I've used the same Animated Hello example.

MSVC / debug / 76MB lib : 2.2MB / 10 sec
MSVC / release / 353MB lib : 544KB / 30 sec
MinGW / debug / 548MB lib : 7.3MB / 17 sec
MinGW / release / 13MB lib : 1MB / 8 sec

MSVC / debug / scu : 3.9MB / 32 sec
MSVC / release / scu : doesn't work... (stupid compiler/linker bug)
MinGW / debug / scu : 11MB / 1:34
MinGW / release / scu : 3.2MB / 4:00

I'll leave the conslusions to you

P.S. MinGW is usually slower and creates larget exes than MSVC8, and MSVC8 is usually slower and creates larger exes than MSVC71. So MinGW isn't such a good compiler . Any free/opensource alternatives?

Subject: Re: U++ as .lib
Posted by cbpporter on Sun, 23 Sep 2007 08:19:38 GMT
View Forum Message <> Reply to Message

My conclusion is that it's not worth it. These times are quite awful (unless you are using a really old computer, which I hope is not the case). And anyway I wouldn't download such huge libs when I can compile them from source.

But if we could get U++ in a state in which it can be recompiled in TheIDE and other more traditional environments (make, and I hope cmake), than that certainly would give more choice. And I also would like to get my hands on a new stable version, this one has a lot of bug and I'm already maintaining a list of bug fixes even though I barely started using U++.

And I think it would be very useful to have an article on exactly how BLITZ works, what do you have to do to achieve it and why is it faster. Maybe other projects could benefit from it (like at my work place where the slightest modification in a file requires a 1-2 minute rebuild).

Subject: Re: U++ as .lib
Posted by mirek on Sun, 23 Sep 2007 09:04:43 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sun, 23 September 2007 04:19
And I think it would be very useful to have an article on exactly how BLITZ works, what do you have to do to achieve it and why is it faster. Maybe other projects could benefit from it
(like at my work place where the slightest modification in a file requires a 1-2 minute rebuild).

Hm, in ToDo for eons.

OK, so quick BLITZ overview:

First, BLITZ processes packages (not the whole program) - each package can have a single BLITZ block.

Only .cpp including files with guards (#ifdef H... #define H) can qualify to be part of BLITZ block. Alternatively, you can force inclusion by #pragma BLITZ_APPROVE (also for header) or exlusion by #pragma BLITZ_PROHIBIT.

AND only files older than one hour qualify for BLITZ block. (Because you do not want files you work on to be in BLITZ block).

Also, the whole package can be excluded based on .upp settings.

Then files are scanned for any #defines, these are undefined at the end of file. BLITZ block is in fact a file generated into output directory that include all BLITZ approved files and gets compiled instead (it is named $blitz.cpp, you can check the output directory for details).

Now a dirty trick:

```
#ifdef  flagBLITZ
#define MK__s          MK__s_(COMBINE(BLITZ_INDEX__, __LINE__))
#else
#define MK__s          MK__s_(__LINE__)
#endif
```

Blitz block defines BLITZ_INDEX__ for each file, in order to give a library code chance to define unique static variable names...

OK, I believe that is all about the BLITZ magic:)

Mirek

Subject: Re: U++ as .lib
Posted by mirek on Sun, 23 Sep 2007 12:11:46 GMT
View Forum Message <> Reply to Message

sergei wrote on Sat, 22 September 2007 19:43cbpporter wrote on Sun, 23 September 2007 00:04An how much time does it take to compile and link a test app with these libs. On my computer it takes 2-4 seconds in debug mode.

You also built U++ libs?
Hmm... since I've built the libs already, why not test

I've used the same Animated Hello example.

MSVC / debug / 76MB lib : 2.2MB / 10 sec
MSVC / release / 353MB lib : 544KB / 30 sec
MinGW / debug / 548MB lib : 7.3MB / 17 sec
MinGW / release / 13MB lib : 1MB / 8 sec

MSVC / debug / scu : 3.9MB / 32 sec
MSVC / release / scu : doesn't work... (stupid compiler/linker bug)
MinGW / debug / scu : 11MB / 1:34
MinGW / release / scu : 3.2MB / 4:00

I'll leave the conslusions to you

Well, my conclusion is "avoid SCU and pursue reasonable .lib approach".

Mind you, things do not need to be perfect at this stage... Right now the goal is to open a way for people that are not ready (yet to use theide to play with U++.

Could you try debug .lib without full debug info? (Only basic info and ASSERTs).

Mirek

Subject: Re: U++ as .lib
Posted by sergei on Sun, 23 Sep 2007 14:46:24 GMT
View Forum Message <> Reply to Message

I might have a very old computer, depending on what you consider an old one. I'm on P4 - 3.0 GHz, Cedar Mill (duh, the last one before Core Duo), FSB 800 MHz, 2GB RAM - CL5, 667 MHz. Old WinXP SP2 setup.

As I said you can recompile the libs rather easily. 20 mins for debug+release in MSVC. The huge times in MinGW are because I didn't use precompiled headers. If I did it would be about 20 mins too. Long times for MSVC/release and MinGW/debug with the libs are because the libs are huge.

For MSVC that's rather surprising, shouldn't release be small...

If you want to compare my times to your computer, here are my times for UWord:

TheIDE / 708dev2b / MinGW (bundled) / debug : 15.1MB / 1:24
TheIDE / 708dev2b / MinGW (bundled) / optimal : 2.2MB / 3:04

SCU / lib times aren't that awful now, right? Remember I'm always posting full rebuild times, which are probably only necessary for the first build.

I could try debug libs without full info (that means flagDEBUG but no flagDEBUG_FULL, right?). Yet I still favor SCU more than debug lib (step in, quick modify and rebuild).

Subject: Re: U++ as .lib
Posted by mirek on Sun, 23 Sep 2007 17:04:22 GMT
View Forum Message <> Reply to Message

sergei wrote on Sun, 23 September 2007 10:46
If you want to compare my times to your computer, here are my times for UWord:

TheIDE / 708dev2b / MinGW (bundled) / debug : 15.1MB / 1:24

0:26

E4300 @ 2.7Ghz, 2 GB RAM

Anyway, this is with new MinGW as compared to 708dev2b (4.x GCC is faster).

BTW, fastest time with MSC7.1 is 16s. That is quite acceptable for full U++ GUI rebuild

Mirek

Subject: Re: U++ as .lib
Posted by sergei on Sun, 23 Sep 2007 17:30:44 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 23 September 2007 19:04sergei wrote on Sun, 23 September 2007 10:46
If you want to compare my times to your computer, here are my times for UWord:

TheIDE / 708dev2b / MinGW (bundled) / debug : 15.1MB / 1:24

0:26

E4300 @ 2.7Ghz, 2 GB RAM

Anyway, this is with new MinGW as compared to 708dev2b (4.x GCC is faster).

BTW, fastest time with MSC7.1 is 16s. That is quite acceptable for full U++ GUI rebuild

Mirek

In other words my comp is slow
That means that you could, like, divide my times by 3

I just realized that flagDEBUG_FULL makes no difference (it's really not used). I've disabled PDB, now debug is MSVC is 26MB (vs 78MB). Still too large. But I've found the reason for 353MB release lib - link time whole program optimization (enabled by default). Without it the size 26MB (again, too large).

Subject: Re: U++ as .lib
Posted by mirek on Sun, 23 Sep 2007 17:55:45 GMT
View Forum Message <> Reply to Message

26MB does not need to too large - try to zip it...

Anyway, check the flags you are using for debug info...

Mirek

Subject: Re: U++ as .lib
Posted by sergei on Sun, 23 Sep 2007 18:05:17 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 23 September 2007 19:5526MB does not need to too large - try to zip it...

Anyway, check the flags you are using for debug info...

Mirek

I use no debug info. There's PDB, C7, ... I chose no debugging symbols.

ZIP is 6MB. OK, not large. But there would be one for debug, one for release, + maybe a separate set for multithreaded. And what about SQL? Include/exclude (currently excluded)?

Subject: Re: U++ as .lib
Posted by mirek on Sun, 23 Sep 2007 19:34:57 GMT
View Forum Message <> Reply to Message

sergei wrote on Sun, 23 September 2007 14:05luzr wrote on Sun, 23 September 2007 19:5526MB does not need to too large - try to zip it...

Anyway, check the flags you are using for debug info...

Mirek

I use no debug info. There's PDB, C7, ... I chose no debugging symbols.

ZIP is 6MB. OK, not large. But there would be one for debug, one for release, + maybe a separate set for multithreaded. And what about SQL? Include/exclude (currently excluded)?

Well, yes. That is why it is so much complex problem... Told you at the beginning, right?

I mean, I really would like to have .lib versions. The trouble is that I do not really know how to do that

Looks like a simple job, but in reality, there is a lot to be solved.

Mirek

Subject: Re: U++ as .lib
Posted by sergei on Sun, 23 Sep 2007 19:59:25 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 23 September 2007 21:34sergei wrote on Sun, 23 September 2007 14:05luzr wrote on Sun, 23 September 2007 19:5526MB does not need to too large - try to zip it...

Anyway, check the flags you are using for debug info...

Mirek

I use no debug info. There's PDB, C7, ... I chose no debugging symbols.

ZIP is 6MB. OK, not large. But there would be one for debug, one for release, + maybe a separate set for multithreaded. And what about SQL? Include/exclude (currently excluded)?

Well, yes. That is why it is so much complex problem... Told you at the beginning, right?

I mean, I really would like to have .lib versions. The trouble is that I do not really know how to do

that

Looks like a simple job, but in reality, there is a lot to be solved.

Mirek

That's why SCU is cool - no libs required

But really, why not let the users build the libs by themselves? It shouldn't take more than half an hour, worst case. Plus they'll select whatever packages they want, multithreaded or not, if they want SQL they'll have SQL installed so it will work. Two builds - debug and release, and no extra large downloads.

Subject: Re: U++ as .lib
Posted by mirek on Sun, 23 Sep 2007 20:21:40 GMT
View Forum Message <> Reply to Message

Yes! Using makefiles?

I guess the real problem is that there are so many options and no really good solutions

Mirek

Subject: Re: U++ as .lib
Posted by sergei on Sun, 23 Sep 2007 21:12:50 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 23 September 2007 22:21Yes! Using makefiles?

I guess the real problem is that there are so many options and no really good solutions

Mirek

AFAIK makefiles are compiler/system dependent. There is such a thing as bakefiles, but I've never used them (actually I did, when I tried to build VCF, but it didn't work).

Many options - yes. Good solution? Well, IMHO what I've posted in the other thread is fine. You just have a folder (UppLib), tell the user to use whatever compiler he wants, just add all files in the folder to a static lib project and build. Problem would appear only if the user has a compiler but doesn't have an IDE. But then, use TheIDE

This solution is flexible - a premade makefile would probably include a preset set of packages, or would be tricky to code to enable options. But with a folder of source, user could modify pkggen.txt, select whatever packages he wants, use the required flags with pkggen.exe and get a folder of sources HE WANTS. I might be reinventing the bakefiles/makefiles wheel, but this works.

I understand that you really want libs, but I wouldn't discard SCU for debug. It proved to be extremely useful when I was modifying U++ source (those tiny bugfixes). TheIDE debugger didn't work as expected, but with MSVC + SCU I was able to fix something and test it within half a minute. Guess I'm in the minority (people who modify U++ would prefer TheIDE), but still... It could be marked "for advanced users"

Subject: Re: U++ as .lib
Posted by mr_ped on Mon, 24 Sep 2007 09:54:26 GMT
View Forum Message <> Reply to Message

makefiles are firstly make-dependent.
The system/compiler is issue which can be solved by creating some configuration script and a modular universal makefile. (but it's not a simple thing to do)
Common linux way of building binaries is "configure && make && make install", i.e. the first step is to set up makefile for current system, and to switch on/off modules as you wish it.

So actually you really are reinventing makefiles.

What platforms do you want libs for?
So far mingw+linux+OSX/X11 can work with same (nasty) universal autoconf+makefile.
The other one is needed for MSC.

pkggen.exe looks to me less portable. Some makefile+platforms guru would do this very likely in shorter time.

Subject: Re: U++ as .lib
Posted by cbpporter on Mon, 24 Sep 2007 10:14:44 GMT
View Forum Message <> Reply to Message

With something like CMake we could get autogenerated makefiles and MSVC projects. It is reasonably cross platform and a lot easier to use than makefiles.

Subject: Re: U++ as .lib
Posted by sergei on Mon, 24 Sep 2007 11:00:04 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Mon, 24 September 2007 11:54makefiles are firstly make-dependent.
The system/compiler is issue which can be solved by creating some configuration script and a modular universal makefile. (but it's not a simple thing to do)
Common linux way of building binaries is "configure && make && make install", i.e. the first step is to set up makefile for current system, and to switch on/off modules as you wish it.

So actually you really are reinventing makefiles.

What platforms do you want libs for?
So far mingw+linux+OSX/X11 can work with same (nasty) universal autoconf+makefile.
The other one is needed for MSC.

pkggen.exe looks to me less portable. Some makefile+platforms guru would do this very likely in shorter time.

pkggen.exe is written in U++, source attached. So I believe it should be possible to precompile it as binary for the main platforms.

I agree that CMake or some other make system, if cross-platform/compiler and working, would be a better solution. Yet the configure/make script will look rather ugly, right? Plus, pkggen scans .upp sources for file lists and dependencies. E.g. unless a package is added/removed, nothing has to be modified, neither pkggen.exe nor pkggen.txt. With CMake/configure, I think any modification to a package (add/remove file/dependency) will have to be represented in the script.

The solution IMHO should be a "minimal maintenance" one - one that would allow working on U++ with little regard to the libs. Unless CMake/configure can handle change of filelist, they don't really qualify. A combination of them + .upp scanner could work, though.

Note: I never used CMake/configure, so I might be wrong.

Subject: Re: U++ as .lib
Posted by mr_ped on Mon, 24 Sep 2007 16:35:15 GMT
View Forum Message <> Reply to Message

Good point about .upp files.

IMHO processing the .upp files directly trough (GNU) Makefile string manipulation functions may be possible (but pushing it to the limit - if possible).
Usually pretty often things like perl scripts are used, but that's adding another dependency, and I don't like such idea.
Under common posix/linux shell environment there's always good old sed and similar, but I have no idea how to process such things under bare windows.

Subject: Re: U++ as .lib
Posted by sergei on Mon, 24 Sep 2007 17:57:48 GMT

mr_ped wrote on Mon, 24 September 2007 18:35Good point about .upp files.

IMHO processing the .upp files directly trough (GNU) Makefile string manipulation functions may be possible (but pushing it to the limit - if possible).
Usually pretty often things like perl scripts are used, but that's adding another dependency, and I don't like such idea.
Under common posix/linux shell environment there's always good old sed and similar, but I have no idea how to process such things under bare windows.

The only thing definitely present on Windows is .bat (batches). But processing strings in batches is true hell (there is replace, but no way to tokenize), and is probably very slow too. There are also .vbs (VB scripts), but these could be disabled due to security risks (high likeness of them being viruses). Other scripts would either require some dependency or have to be compiled to exe - not much better than a dedicated C++ program...

OTOH, why not use a C/C++ program for Windows (amd maybe OSX), and use scripts for Linux/Posix (AFAIK it's next to impossible to precompile a binary to support all distros)? Or write something in plain C/C++, to be compileable with GCC anywhere?

Edit: Thinking of it, pkggen uses the U++ package headers it creates. So, if it's run the first time on Windows like pkggen.exe LINUX POSIX (or whatever other flags are required), the resulting source tree should be all that is required to compile pkggen from source on Linux.