Subject: Win32 nightly builds now using mingw
Posted by mirek on Wed, 06 Apr 2016 08:55:18 GMT
View Forum Message <> Reply to Message

In recent weeks, in the preparation of move to C++11, I have reworked nightly builds infrastructure to use mingw-w64 5.3.0 to build theide. (instead of MSC9).

After a period of instability, all seems to work fine now (I had to tweak -O options etc...).

Also, mingw-w64 5.3.0 is now bundled...

I hope that everything works fine, but testing and comments is welcome.

Mirek

Subject: Re: Win32 nightly builds now using mingw
Posted by koldo on Thu, 07 Apr 2016 07:30:12 GMT
View Forum Message <> Reply to Message

Excellent!

Now U++ is fully portable. When running TheIDE the first time, it searches and auto installs available MSC compilers and in case of "toolchain download", it also installs its own MinGW. Compiling and running Examples with no problem. My actual U++ installation is unaffected. And TheIDE has been compiled with MinGW! Just to say something, maybe it is necessary in U++ download page to say new Windows users to just run theide.exe or theide32.exe for 32 bits Windows version.

Thank you Mirek.

Subject: Re: Win32 nightly builds now using mingw
Posted by cbpporter on Thu, 07 Apr 2016 08:25:56 GMT
View Forum Message <> Reply to Message

Nice!

I've been trying to integrate more and more GCC in my stuff, but it is not that easy.

This is not related to U++, but me doing funky stuff on program terminate, but often with GCC I have a weird problem: the program crashes on the atexit stage, where I do a pre-global destructor cleanup. And the weird part is that this only happens on a fresh compile after a clean. I compile in optimized mode, crashes. Recompile without optimal mode, no crash. After this, no mater how many times I compile in optimized mode, no more crash and the problem goes away for hours or until I shut down everything and clean the build folder.

This happens on Windows only, and anyway I can't Valgrind, since it is in optimized mode.

But without the cleanup, GCC looks like it works well and I try to compile once a week.

---

Subject: Re: Win32 nightly builds now using mingw
Posted by mirek on Thu, 07 Apr 2016 20:48:01 GMT
View Forum Message <> Reply to Message

I have found that -O3 mode does not work with mingw. I suspect linker issue....

---

Subject: Re: Win32 nightly builds now using mingw
Posted by cbpporter on Thu, 07 Apr 2016 21:39:45 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 07 April 2016 23:48I have found that -O3 mode does not work with mingw. I suspect linker issue....
So it is not just me then?

Anyway, I might be able to circumvent the problem.

What I'm doing in the atexit method is basically leak detection. How did you manage to do leak detection in U++, since such code needs to run after global destructors, right?

---

Subject: Re: Win32 nightly builds now using mingw
Posted by mirek on Fri, 08 Apr 2016 07:04:21 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 07 April 2016 23:39mirek wrote on Thu, 07 April 2016 23:48I have found that -O3 mode does not work with mingw. I suspect linker issue....
So it is not just me then?

Anyway, I might be able to circumvent the problem.

Well, this one can be "circumvented" by using -O2 instead  (It seems that the speed is similar).

Quote:
What I'm doing in the atexit method is basically leak detection. How did you manage to do leak detection in U++, since such code needs to run after global destructors, right?

Check the code  It is at the end of heapdbg.cpp and near then end of Core.h.

MSC and GCC have some means to achieve it. If that does not work, universal approach is to make it the first (or likely first) constructor (Core.h 337).

Mirek

Subject: Re: Win32 nightly builds now using mingw
Posted by cbpporter on Mon, 11 Apr 2016 09:31:50 GMT
View Forum Message <> Reply to Message

Thank you!

The GCC solution seems more elegant, but I need to try both.

But this is hell to test. I have exactly one clean compile to catch the error, because the second compile works, as long as I change the source code, by adding space somewhere. Sometimes. It is very random.

Subject: Re: Win32 nightly builds now using mingw
Posted by mirek on Mon, 11 Apr 2016 12:01:02 GMT
View Forum Message <> Reply to Message

That sounds BLITZ related. Sometimes, BLITZ can hide error.

Subject: Re: Win32 nightly builds now using mingw
Posted by cbpporter on Mon, 11 Apr 2016 14:34:41 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 11 April 2016 15:01That sounds BLITZ related. Sometimes, BLITZ can hide error.
As I said, very random. But I think I'm finally detecting the pattern: external build method modifies source file time, but content is unchanged, I load up TheIDE, build mode is MSC (not sure if build is needed or not), I switch to GCC and it crashes. I think it doesn't rebuild something. I need a bit of more testing, but I think this pattern is 100% reproducible. Apparently, the atexit code is not the source of the trouble, but I still feel like your solution is better than my hack. Neither is -03 the problem since I checked and TheIDE pushes for -O2.

Subject: Re: Win32 nightly builds now using mingw
Posted by cbpporter on Mon, 16 May 2016 11:27:06 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 07 April 2016 11:25Nice!

I've been trying to integrate more and more GCC in my stuff, but it is not that easy.

This is not related to U++, but me doing funky stuff on program terminate, but often with GCC I have a weird problem: the program crashes on the atexit stage, where I do a pre-global destructor cleanup. And the weird part is that this only happens on a fresh compile after a clean. I compile in optimized mode, crashes. Recompile without optimal mode, no crash. After this, no mater how many times I compile in optimized mode, no more crash and the problem goes away for hours or until I shut down everything and clean the build folder.

This happens on Windows only, and anyway I can't Valgrind, since it is in optimized mode.

But without the cleanup, GCC looks like it works well and I try to compile once a week.
There are two problems here.

First, a TheIDE related linking error. If you change the build method from MSC it can happen.

But sometimes it persisted no matter what.

But I finally found something: removing a single inline from a function fixed this. It appears that maybe TDM has some problems with too aggressive inline-ing. The following code crashes:

```
inline void _free(void** p) {
 if (*p) {
  STDLIB_GC--;
  free(*p);
 }
 *p = 0;
}
```

Do you guys see anything wrong with this code? It crashes when freeing some members from global variables.

---