Subject: U++ 2019.1.rc4 released
Posted by mirek on Mon, 15 Apr 2019 06:23:05 GMT
View Forum Message <> Reply to Message

Hi,

after a ministorm of bugs and fixes, we are now at rc4.

https://sourceforge.net/projects/upp/files/upp/2019.1.rc4/

Thinks fixed and added:

- Support for Visual C++ / Build Tools 2019
- MacOS version does not require crypto.dylib installed to run theide
- Fixed issue in theide help (multiplying entries for topics with TOC)
- Mingw (and, in fact GCC and CLANG) now support "SO" compilation (compiles packages to .dll/.so/.dylib, this helps with linking speed)

Thanks for testing!

Mirek

Subject: Re: U++ 2019.1.rc4 released
Posted by koldo on Mon, 15 Apr 2019 06:54:16 GMT
View Forum Message <> Reply to Message

Thank you all for your great job :) :)

Subject: Re: U++ 2019.1.rc4 released
Posted by Tom1 on Tue, 16 Apr 2019 06:39:58 GMT
View Forum Message <> Reply to Message

Hi,

I have now tested rc4 (on Windows 10 1809 / 64-bit with my own application code) and everything seems to work fine. :)

Thanks and best regards,

Tom

Subject: Re: U++ 2019.1.rc4 released
Posted by cbpporter on Wed, 17 Apr 2019 12:48:00 GMT

View Forum Message <> Reply to Message

Will test it ASAP. Takes a couple of days to merge all my forks into the main release...

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Wed, 17 Apr 2019 19:31:55 GMT
View Forum Message <> Reply to Message

I was investigating a crash with TheIDE and discovered a bunch of uninitialized memory reads. Please check an attached file.

File Attachments

1) vg.log, downloaded 322 times

Subject: Re: U++ 2019.1.rc4 released
Posted by forlano on Wed, 17 Apr 2019 21:43:08 GMT
View Forum Message <> Reply to Message

Hi Mirek,

the Mac OS version seems to work properly.
Thanks,
Luigi

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Wed, 17 Apr 2019 22:22:16 GMT
View Forum Message <> Reply to Message

I'm getting a random crash with TheIDE (every second time I launch it) with the call stack below. (Full rebuild)

```
#0  0x0000555555879007 in SkipGdbInfo(Upp::CParser&) (p=...) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/ide/Debuggers/GdbData.cpp:8
#1  0x000055555587a7d8 in DataClean(Upp::CParser&) (p=...) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/ide/Debuggers/GdbData.cpp:63
#2  0x000055555587ac13 in DataClean(char const*) (s=<optimized out>) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/ide/Debuggers/GdbData.cpp:79
#3  0x000055555587fac2 in Gdb::TryAuto(Upp::Index<Upp::String>&, Upp::String const&)
(this=0x7fffde4e4030, tested=..., exp=...)
    at /home/ssg/dvlp/cpp/upp/git/uppsrc/Core/String.h:313
#4  0x000055555587fd69 in Gdb::Autos() (this=0x7fffde4e4030) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/ide/Debuggers/GdbData.cpp:172
#5  0x0000555555883dd9 in Gdb::Cmdp(char const*, bool, bool)
    (this=0x7fffde4e4030, cmdline=<optimized out>, fr=<optimized out>, setframe=<optimized
out>)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/ide/Debuggers/Gdb.cpp:292
#6  0x000055555588470a in Gdb::Step(char const*) (this=0x7fffde4e4030, cmd=0x555555f18f84
"next")
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/ide/Debuggers/Gdb.cpp:453
#7  0x00005555558fcb7d in Upp::Function<void ()>::operator()() const (this=<optimized out>) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Function.h:76
#8  0x00005555558fcb7d in Upp::Bar::ScanKeys::Do(unsigned int) (k=131015,
this=0x7fffffffcad10)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlLib/Bar.cpp:509
#9  0x00005555558fcb7d in Upp::Bar::Scan(Upp::Function<void (Upp::Bar&)>, unsigned int)
(proc=..., key=131015)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlLib/Bar.cpp:520
#10 0x00005555558fd26f in Upp::TopSubMenuItem::HotKey(unsigned int) (this=0x7ffff3a05080,
key=131015) at /usr/include/c++/8/bits/atomic_base.h:299
#11 0x0000555555bf1d15 in Upp::Ctrl::HotKey(unsigned int) (this=<optimized out>, key=131015)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/CtrlCore.h:1113
#12 0x0000555555bf1d15 in Upp::Ctrl::HotKey(unsigned int) (this=this@entry=0x7fffffffccb70,
key=key@entry=131015)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/CtrlCore.h:1113
#13 0x00005555559191f2 in Upp::MenuBar::HotKey(unsigned int) (this=0x7fffffffccb70,
key=131015)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlLib/MenuBar.cpp:433
#14 0x0000555555bf1d15 in Upp::Ctrl::HotKey(unsigned int) (this=<optimized out>, key=131015)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/CtrlCore.h:1113
#15 0x0000555555bf1d15 in Upp::Ctrl::HotKey(unsigned int) (this=<optimized out>, key=131015)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/CtrlCore.h:1113
#16 0x0000555555bf5010 in Upp::Ctrl::DispatchKey(unsigned int, int) (keycode=131015,
count=1) at /home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Ptr.h:43
#17 0x0000555555c0b945 in Upp::Ctrl::Proc() (this=<optimized out>) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/GtkEvent.cpp:485
#18 0x0000555555c0c78f in Upp::Ctrl::ProcessEvent0(bool*, bool) (quit=0x0, fetch=<optimized
out>)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/GtkEvent.cpp:557
#19 0x0000555555c0cd21 in Upp::Ctrl::ProcessEvents0(bool*, bool) (fetch=fetch@entry=true,
quit=0x0)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/GtkEvent.cpp:583
#20 0x0000555555c0d50d in Upp::Ctrl::ProcessEvents(bool*) (quit=0x0) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/GtkEvent.cpp:631
#21 0x0000555555c0d50d in Upp::Ctrl::EventLoop(Upp::Ctrl*) (ctrl=0x7fffffffcb380) at
/home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/GtkEvent.cpp:631
#22 0x0000555555c15a71 in Upp::TopWindow::Run(bool) (this=this@entry=0x7fffffffcb380,
appmodal=appmodal@entry=false)
   at /home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlCore/TopWindow.cpp:324
#23 0x0000555555811f1e in GuiMainFn_() () at
/home/ssg/dvlp/cpp/upp/git/uppsrc/ide/main.cpp:305
#24 0x0000555555710d08 in main(int, char**, char const**) (argc=1, argv=0x7fffffffdef8,
envptr=0x55555628b2f0)
```

```
    at /home/ssg/dvlp/cpp/upp/git/uppsrc/ide/main.cpp:120
```

It is most likely related to uninitialized memory reads or corrupted memory.

Subject: Re: U++ 2019.1.rc4 released
Posted by mirek on Thu, 18 Apr 2019 05:59:32 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 18 April 2019 00:22I'm getting a random crash with TheIDE (every second time I launch it) with the call stack below. (Full rebuild)

Thanks. It would be really helpful to provide info about host platform and steps to reproduce.

From the call stack it looks like like crash in Gdb interface. Does it happen when trying to debug something?

Mirek

Subject: Re: U++ 2019.1.rc4 released
Posted by mirek on Thu, 18 Apr 2019 06:03:13 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 18 April 2019 00:22I'm getting a random crash with TheIDE (every second time I launch it) with the call stack below. (Full rebuild)
It is most likely related to uninitialized memory reads or corrupted memory.

Based on stack dump, hopefully fixed in trunk. Please test.

Mirek

Subject: Re: U++ 2019.1.rc4 released
Posted by cbpporter on Thu, 18 Apr 2019 08:17:46 GMT
View Forum Message <> Reply to Message

Weirdly I got a compilation error in some code that hasn't been touched in years related to variable shadowing...

Something must have changed in the way the C++ compiler is invoked.

Not a real problem though, it is better to not shadow so I renamed the variables.

My projects are made out of multiple executable and usually I have multiple TheIDEs running (3+), but this new release has a blazing fast "open main package".

I tested the command line stuff. No issues.

Next I'll test GUI.

Is there a a way to force dark mode under Windows? I have code editor colors for dark mode, but they always looked out of place with light UI.

Subject: Re: U++ 2019.1.rc4 released
Posted by Tom1 on Thu, 18 Apr 2019 08:30:20 GMT
View Forum Message <> Reply to Message

Hi cbpporter,

Dark theme is enabled in Windows 10 using Settings > Colors > Choose your default app mode > Dark

Then restart TheIDE.

Best regards,

Tom

Subject: Re: U++ 2019.1.rc4 released
Posted by mirek on Thu, 18 Apr 2019 08:53:13 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 17 April 2019 21:31I was investigating a crash with TheIDE and discovered a bunch of uninitialized memory reads. Please check an attached file.

Should be all fixed in the trunk.

All of it was innocent things like Enabling/Disableing random buttons in theide toolbar during initialization, then it got fixed by second SetBar call even before theide is open.

That said, it is definitely better if theide runs valgrind clean, so I from now on I will do valgrind checks before each release. Thank you.

Mirek

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Thu, 18 Apr 2019 20:20:33 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 18 April 2019 02:03Novo wrote on Thu, 18 April 2019 00:22I'm getting a random crash with TheIDE (every second time I launch it) with the call stack below. (Full rebuild) It is most likely related to uninitialized memory reads or corrupted memory.

Based on stack dump, hopefully fixed in trunk. Please test.

Mirek
Thank you! So far everything seems to work fine.

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Thu, 18 Apr 2019 22:32:57 GMT

View Forum Message <> Reply to Message

mirek wrote on Thu, 18 April 2019 04:53Novo wrote on Wed, 17 April 2019 21:31I was investigating a crash with TheIDE and discovered a bunch of uninitialized memory reads. Please check an attached file.

Should be all fixed in the trunk.

All of it was innocent things like Enabling/Disableing random buttons in theide toolbar during initialization, then it got fixed by second SetBar call even before theide is open.

That said, it is definitely better if theide runs valgrind clean, so I from now on I will do valgrind checks before each release. Thank you.

Mirek
Thank you! Below are two more fixes which make TheIDE 100% valgrind-clean.
(I fixed/added two constructors)
 CppItem() : access(), kind(), at(), virt(), decla(), lvalue(), isptr(), filetype(), impl(), file(), line(), qualify(true) {}

struct PPItem {
 PPItem() : type(), segment_id() {}

These changes won't make TheIDE any slower.
This is C++98 initialization style. I didn't do that C++11 way for compatibility reason.

Subject: Re: U++ 2019.1.rc4 released
Posted by mirek on Fri, 19 Apr 2019 10:27:49 GMT

View Forum Message <> Reply to Message

Novo wrote on Fri, 19 April 2019 00:32mirek wrote on Thu, 18 April 2019 04:53Novo wrote on Wed, 17 April 2019 21:31I was investigating a crash with TheIDE and discovered a bunch of uninitialized memory reads. Please check an attached file.

Should be all fixed in the trunk.

All of it was innocent things like Enabling/Disableing random buttons in theide toolbar during initialization, then it got fixed by second SetBar call even before theide is open.

That said, it is definitely better if theide runs valgrind clean, so I from now on I will do valgrind checks before each release. Thank you.

Mirek
Thank you! Below are two more fixes which make TheIDE 100% valgrind-clean.

Did it trigger any valgrind errors? If so, which ones?

Quote:
(I fixed/added two constructors)
 CppItem() : access(), kind(), at(), virt(), decla(), lvalue(), isptr(), filetype(), impl(), file(), line(), qualify(true) {}

struct PPItem {
 PPItem() : type(), segment_id() {}

These changes won't make TheIDE any slower.

Probably not noticeably, but it is matter of principle: Do I need from now on to initialize variables that are not used in particular branch of code?

Quote:
This is C++98 initialization style. I didn't do that C++11 way for compatibility reason.

Why? :)

Mirek

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Fri, 19 Apr 2019 14:43:01 GMT

View Forum Message <> Reply to Message

mirek wrote on Fri, 19 April 2019 06:27
Did it trigger any valgrind errors? If so, which ones?

Yes, they did. Please check the attached file.

mirek wrote on Fri, 19 April 2019 06:27
Quote:
(I fixed/added two constructors)

CppItem() : access(), kind(), at(), virt(), decla(), lvalue(), isptr(), filetype(), impl(), file(), line(), qualify(true) {}

struct PPItem {
 PPItem() : type(), segment_id() {}

These changes won't make TheIDE any slower.

Probably not noticeably, but it is matter of principle: Do I need from now on to initialize variables that are not used in particular branch of code?

Well, valgrind and sanitizers is very often the only way to figure out an origin of a problem because sometimes this can be a bug with a compiler but the code itself. An when I see a lot of garbage in the output I just want to stop using the code because it is unprofessional. It is not that hard to fix all errors reported by valgrind and sanitizers.

In the second case both members are used, I believe.
In the first case I initialized everything because I didn't have time to figure out which one exactly is used uninitialized.

mirek wrote on Fri, 19 April 2019 06:27
Quote:
This is C++98 initialization style. I didn't do that C++11 way for compatibility reason.

Why? :)

Old habit.

File Attachments

1) vg.log.02, downloaded 233 times

Subject: Re: U++ 2019.1.rc4 released
Posted by cbpporter on Fri, 19 Apr 2019 14:53:50 GMT
View Forum Message <> Reply to Message

It would be great if "Output mode/Target file override" would be made package specific :).

Currently, you open package A, compile with "Target file override" into the desired location, open up a random package from examples, compile and your old executable from another project is overwritten.

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Fri, 19 Apr 2019 22:33:19 GMT

View Forum Message <> Reply to Message

I do not know what is wrong with the GUI_APP_MAIN, but I'm getting 1.2 Mb of complains from valgrind. Please check attached file.

File Attachments

1) vg.log, downloaded 232 times

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Fri, 19 Apr 2019 22:51:44 GMT
View Forum Message <> Reply to Message

I do not know who is responsible for this crash (libfontconfig.so.1+0xdb67 or /uppsrc/Draw/Font.cpp:34:10), but I cannot even get close to my own code.
Memory Sanitizer:

```
Uninitialized bytes in __interceptor_strlen at offset 0 inside [0x7010000008a0, 11)
==30845==WARNING: MemorySanitizer: use-of-uninitialized-value
    #0 0x7f3bb6714097  (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0xb097)
    #1 0x7f3bb6716baa in FcConfigFilename (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0xdbaa)
    #2 0x7f3bb672f607  (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0x26607)
    #3 0x7f3bb6721be3  (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0x18be3)
    #4 0x7f3bb6721e45  (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0x18e45)
    #5 0x7f3bb6714736  (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0xb736)
    #6 0x7f3bb6721f05 in FcInitBringUptoDate
(/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0x18f05)
    #7 0x7f3bb6724be9 in FcFontList (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0x1bbe9)
    #8 0x292ed45 in Upp::GetAllFacesSys()
/home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/FontFc.cpp:236:18
    #9 0x292e354 in Upp::Font::FaceList() /home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:34:10
    #10 0x292fc97 in Upp::sInitFonts() /home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:42:2
    #11 0x292fd68 in Upp::s__sF0_46_fn() /home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:47:2
    #12 0x12bb799 in Upp::Callinit::Callinit(void (*)(), char const*, int)
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Defs.h:176:83
    #13 0x469944 in __cxx_global_var_init.4
/home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:46:1
    #14 0x469d5c in _GLOBAL__sub_I__blitz.cpp
/home/ssg/dvlp/cpp/upp/git/out/MyApps/Draw/CLANGcpp17msan.Debug.Debug_Full.Gui.Shared.
Usemalloc/$blitz.cpp
    #15 0x3dfd874 in __libc_csu_init
(/home/ssg/dvlp/cpp/upp/git/out/MyApps/CLANGcpp17msan.Debug.Debug_Full.Gui.Shared.Use
malloc/OpenCorpora+0x3dfd874)
    #16 0x7f3bb47a0029 in __libc_start_main
/build/glibc-B9XfQf/glibc-2.28/csu/../csu/libc-start.c:264:6
    #17 0x471839 in _start
(/home/ssg/dvlp/cpp/upp/git/out/MyApps/CLANGcpp17msan.Debug.Debug_Full.Gui.Shared.Use
malloc/OpenCorpora+0x471839)
```

```
  Uninitialized value was created by a heap allocation
    #0 0x47adfc in __interceptor_malloc
(/home/ssg/dvlp/cpp/upp/git/out/MyApps/CLANGcpp17msan.Debug.Debug_Full.Gui.Shared.Use
malloc/OpenCorpora+0x47adfc)
    #1 0x7f3bb6716b67 in FcConfigFilename (/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0xdb67)
    #2 0x292e354 in Upp::Font::FaceList() /home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:34:10
    #3 0x292fc97 in Upp::sInitFonts() /home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:42:2
    #4 0x292fd68 in Upp::s__sF0_46_fn() /home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:47:2
    #5 0x12bb799 in Upp::Callinit::Callinit(void (*)(), char const*, int)
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Defs.h:176:83
    #6 0x469944 in __cxx_global_var_init.4 /home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/Font.cpp:46:1
    #7 0x469d5c in _GLOBAL__sub_I__blitz.cpp
/home/ssg/dvlp/cpp/upp/git/out/MyApps/Draw/CLANGcpp17msan.Debug.Debug_Full.Gui.Shared.
Usemalloc/$blitz.cpp
    #8 0x3dfd874 in __libc_csu_init
(/home/ssg/dvlp/cpp/upp/git/out/MyApps/CLANGcpp17msan.Debug.Debug_Full.Gui.Shared.Use
malloc/OpenCorpora+0x3dfd874)

SUMMARY: MemorySanitizer: use-of-uninitialized-value
(/usr/lib/x86_64-linux-gnu/libfontconfig.so.1+0xb097)
Exiting
```

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Fri, 19 Apr 2019 22:57:13 GMT

View Forum Message <> Reply to Message

```
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/lheap.cpp:46:18: runtime error: index -1 out of bounds for
type 'Upp::word [77]'
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Hash.cpp:307:33: runtime error: load of misaligned
address 0x0000027ed2a3 for type 'const unsigned int', which requires 4 byte alignment
0x0000027ed2a3: note: pointer points here
 74  2e 68 00 28 64 65 66 61  75 6c 74 29 00 28 70 65  72 20 64 65 66 65 63 74  65 29 00 28 76
c3 bd
             ^
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Ops.h:3:51: runtime error: load of misaligned address
0x00000288de89 for type 'const Upp::dword' (aka 'const unsigned int'), which requires 4 byte
alignment
0x00000288de89: note: pointer points here
 61 6e 73  2d 73 65 72 69 66 00 41  72 69 61 6c 00 e6 96 b0  e5 ae 8b e4 bd 93 00 53  69 6d 53
75 6e
             ^
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Ops.h:7:55: runtime error: store to misaligned address
0x7f1e14817182 for type 'Upp::dword' (aka 'unsigned int'), which requires 4 byte alignment
0x7f1e14817182: note: pointer points here
 65 65  46 72 65 65 46 72 65 65  46 72 65 65 46 72 65 65  46 72 65 65 46 72 65 65  46 72 65 65
46 72
```

```
              ^
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Ops.h:2:51: runtime error: load of misaligned address
0x7f1e102b4699 for type 'const Upp::word' (aka 'const unsigned short'), which requires 2 byte
alignment
0x7f1e102b4699: note: pointer points here
 42 00 00  00 0d 00 0d 00 01 00 01  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 b2 b2
be ff
              ^
/home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/RescaleFilter.cpp:70:32: runtime error: left shift of
negative value -5
/home/ssg/dvlp/cpp/upp/git/uppsrc/Draw/RescaleFilter.cpp:88:31: runtime error: left shift of
negative value -5
/home/ssg/dvlp/cpp/upp/git/uppsrc/CtrlLib/ChGtk0.cpp:478:16: runtime error: index 30 out of
bounds for type 'GdkColor [5]'
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Other.h:294:71: runtime error: downcast of address
0x7ffe8dc05f98 which does not point to an object of type 'Upp::DisplayPopup'
0x7ffe8dc05f98: note: object is of type 'Upp::Ctrl'
 00 00 00 80  20 bd 85 02 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00
00 00
              ^~~~~~~~~~~~~~~~~~~~~~~
              vptr for 'Upp::Ctrl'
/usr/bin/xmessage
Segmentation fault (core dumped)
```

This time this is UB Sanitizer ... :)

Subject: Re: U++ 2019.1.rc4 released
Posted by mirek on Fri, 19 Apr 2019 23:06:43 GMT
View Forum Message <> Reply to Message

Will check all those too, but

Ops.h:3:51: runtime error: load of misaligned address

(and stores)

- that one is impossible to fix, it is meant to be.

Note #ifdef CPU_UNALIGNED before it...

Mirek

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Sat, 20 Apr 2019 02:46:21 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 19 April 2019 19:06Will check all those too, but

Ops.h:3:51: runtime error: load of misaligned address

(and stores)

- that one is impossible to fix, it is meant to be.

Note #ifdef CPU_UNALIGNED before it...

Mirek

Do you mind adding code below to Defs.h and marking PeekXXle and PokeXXle with NO_UBSAN ?

```
// ***** Sanitizer-related
#if defined(__has_feature)

#if __has_feature(address_sanitizer)
  #define NO_ASAN __attribute__((no_sanitize_address))
#else
  #define NO_ASAN
#endif

#if __has_feature(memory_sanitizer)
  #define NO_MSAN __attribute__((no_sanitize_memory))
#else
  #define NO_MSAN
#endif

#else

// gcc doesn't have __has_feature
#if defined (__GNUC__)
  #define NO_ASAN __attribute__((no_sanitize_address))
#else
  #define NO_ASAN
#endif

  // gcc doesn't support memory sanitizer.
  #define NO_MSAN
  // gcc doesn't support safe-stack sanitizer.
  // gcc doesn't support shadow-call-stack sanitizer.

#endif

#if defined(__clang__) || defined (__GNUC__)
  #define NO_UBSAN __attribute__((no_sanitize("undefined")))
#else
```

```
  #define NO_UBSAN
#endif
```

Thank you in advance.

Subject: Re: U++ 2019.1.rc4 released

Posted by mirek on Mon, 22 Apr 2019 09:03:49 GMT

View Forum Message <> Reply to Message

Novo wrote on Sat, 20 April 2019 00:33I do not know what is wrong with the GUI_APP_MAIN, but I'm getting 1.2 Mb of complains from valgrind. Please check attached file.

Looks like something that affects menu item enabling is not initialized in OC class...

Mirek

Subject: Re: U++ 2019.1.rc4 released

Posted by mirek on Mon, 22 Apr 2019 09:08:29 GMT

View Forum Message <> Reply to Message

"__interceptor_malloc" - It is very unlikely that U++ called that malloc (normally we do not allocated memory via malloc).

Subject: Re: U++ 2019.1.rc4 released

Posted by mirek on Tue, 23 Apr 2019 07:53:19 GMT

View Forum Message <> Reply to Message

TheIDE now seems to run 'ubsan clean' (with GCC), but I really have mixed feeling about santizing bools:

```
struct Foo { bool a, b; };

Foo a, b;

a = b;
```

Above code fails with ubsan, as a / b are uninitialized. In practice this means you have to initialize all bool member variables if the struct is to be copied even if code logic absolutely does not need it....

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Tue, 23 Apr 2019 13:05:01 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 22 April 2019 05:08"__interceptor_malloc" - It is very unlikely that U++ called that malloc (normally we do not allocated memory via malloc).
I'm using standard allocator with static build + shared libraries. Otherwise it won't link.
"All shared" will link fine with U++ allocator, if I remember correctly.

Subject: Re: U++ 2019.1.rc4 released
Posted by Novo on Tue, 23 Apr 2019 13:33:59 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 23 April 2019 03:53TheIDE now seems to run 'ubsan clean' (with GCC), but I really have mixed feeling about santizing bools:

```
struct Foo { bool a, b; };

Foo a, b;

a = b;
```

Above code fails with ubsan, as a / b are uninitialized. In practice this means you have to initialize all bool member variables if the struct is to be copied even if code logic absolutely does not need it....
I'm personally not getting any messages with ubsan + gcc + C++17 + Debug in this case. Everything is clean.
I'm getting them with ubsan + clang + C++17 + Debug.

```
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Hash.cpp:308:33: runtime error: load of misaligned address 0x0000008c646e for type 'const unsigned int', which requires 4 byte alignment
0x0000008c646e: note: pointer points here
 41 47 45 00 45 6d  70 74 79 20 64 72 69 76  65 00 44 72 69 76 65 20  76 61 63 c3 ad 6f 00 4c  65 65
             ^
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Ops.h:7:63: runtime error: store to misaligned address 0x7f7ca993b222 for type 'Upp::dword' (aka 'unsigned int'), which requires 4 byte alignment
0x7f7ca993b222: note: pointer points here
 65 65  46 72 65 65 46 72 65 65  46 72 65 65 46 72 65 65  46 72 65 65 46 72 65 65  46 72 65 65  46 72
              ^
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Ops.h:3:59: runtime error: load of misaligned address 0x7f7ca993b222 for type 'const Upp::dword' (aka 'const unsigned int'), which requires 4 byte alignment
0x7f7ca993b222: note: pointer points here
 35 00  46 72 fe 4c 6c 56 65 65  46 72 65 65 46 72 65 65  46 72 65 65 46 72 65 65  46 72 65 65
```

```
46 72
          ^
```

It doesn't look like NOUBSAN is defined for Clang.

I'm also getting a bunch of warnings with Clang + C++17.

```
In file included from /home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Core.h:280:
/home/ssg/dvlp/cpp/upp/git/uppsrc/Core/Stream.h:94:141: warning: add explicit braces to avoid dangling else [-Wdangling-else]
        void      Put(const void *data, int size)  { ASSERT(size >= 0); if(size) if(ptr + size <= wrlim) { memcpy(ptr, data, size);
     ptr += size; } else _Put(data, size); }
```

One of the reasons why I'm using Clang on Linux is because it has the best support for sanitizers.

I attached my project and build methods for clang and gcc.

File Attachments

1) test_ubsan.zip, downloaded 212 times

Subject: Re: U++ 2019.1.rc4 released

Posted by Novo on Tue, 23 Apr 2019 13:36:26 GMT

View Forum Message <> Reply to Message

mirek wrote on Tue, 23 April 2019 03:53TheIDE now seems to run 'ubsan clean' (with GCC)

Thank you very much!