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 233 times