Subject: 2019.2 rc5
Posted by mirek on Mon, 21 Oct 2019 09:57:03 GMT
View Forum Message <> Reply to Message

Here we go.

https://sourceforge.net/projects/upp/files/upp/2019.2/

(pre)release: I will announce on sf.net after a couple of days without serious bug reports...

EDIT: OK, it is now rc5...

Core

Allocator huge block handling optimized, memory consumption improved in specific cases
Index refactored to reduce sizeof and improve performance in specific cases
Introduced workaround for Mingw TLS performance issue
pick/clone semantics refined in several places
Random algorithm changed to xoshiro256**
Uuid generation optimized
Fast U++ allocator is now used internally where possible even if standard allocator us used for new/delete (with USEMALLOC flag)
pick/clone refinements
Value::Compare optimized for standard types

Core/SSH:

A synchronization issue is fixed with socket waits.
3 new fingerprint methods are added (md5, sha1, sha256).
Password change request and event is implemented.
authentiaction-less connection is now possible.
SshShell can now set terminal modes.

Graphics

SVG parser improvements
GLCtrl new method ExecuteGL to execute code with correct GL context
Font metrics optimized for MingW with TLS optimization workaround

CtrlLib

ArrayCtrl sorting improved
CtrlMapper small helper class
DropList supports Add with initializer_list

FileSelButton WhenSelected Event
FileSel has now more sorting options, PreSelect now supports Save As too

Ide

Console output now supports search
Font settings 'set to defaults' button
Debugger now has arrow buttons to change frame without dropping the list
New GUI patch tool
GDB debugger frontend now has memory tab
Icon designer export to .png now exports uhd/dark variants too
Main config dialog now can drag/drop (or move) lines
Run options dialog got new buttons to insert file path / directory path into commandline args
pkg-config support

Subject: Re: 2019.2 (pre)released
Posted by Oblivion on Mon, 21 Oct 2019 10:47:55 GMT
View Forum Message <> Reply to Message

Thank you for your efforts!

As a side note, Core/SSH package is also updated:

- A synchronization issue is fixed with socket waits.
- 3 new fingerprint methods are added (md5, sha1, sha256).
- Password change request and event is implemented.
- authentiaction-less connection is now possible.
- SshShell can now set terminal modes.
- Docs are updated accordingly.

Also

There is a minor annoyance with Glib2: I get a lot of deprecation messages for glib2 (2.62) types:

/usr/include/glib-2.0/glib/gtypes.h:549:26: bilgi: declared here
  549 | typedef struct _GTimeVal GTimeVal
GLIB_DEPRECATED_TYPE_IN_2_62_FOR(GDateTime);

Nothing big, but bloats the error console on up-to-date setups.

Best regards,

Subject: Re: 2019.2 (pre)released
Posted by mirek on Mon, 21 Oct 2019 10:51:16 GMT
View Forum Message <> Reply to Message

Updated; i will ignore glib warning for now, as I am working on gtk3 for the next release.

Mirek

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Mon, 21 Oct 2019 13:40:59 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks! I have been waiting for this release...

Unfortunately, once I now have it and started to migrate to it, I stumbled across a problem: My application rendering charts crashes after a few hundred or a few thousand times after rendering an image (well, a very LARGE IMAGE of 9880 x 10758 pixels) with BufferPainter. I discovered this in x64 release mode and reproduced in debug mode for the backtrace below.

Platform: Windows 10 Pro x64, MSBT19x64, latest upp -- the release 2019.2 (13648).

Here's the backtrace:

Upp::Dbl_Unlink<Upp::BlkHeader_<256> >(x=1b7948c3840->{ prev=ff4b6644ff4b6644, next=ff526e48ff4b6644, prev_size=26180, size=65355, free=1, last=1, heap=ff4b6644ff4b6644 }) at C:\upp-13648\uppsrc\Core\Util.h 251
Upp::BlkHeader_<256>::UnlinkFree() at C:\upp-13648\uppsrc\Core\HeapImp.h 48
Upp::BlkHeap<Upp::Heap::LargeHeapDetail,256>::Free(h=1b7948c5040->{ prev=dcc, next=1b7c89aaee0, prev_size=24, size=15, free=0, last=1, heap=1b78f94d590 }) at C:\upp-13648\uppsrc\Core\HeapImp.h 243
Upp::Heap::LFree(ptr=1b7948c5050) at C:\upp-13648\uppsrc\Core\lheap.cpp 151
Upp::Heap::Free(ptr=1b7948c5050) at C:\upp-13648\uppsrc\Core\sheap.cpp 248
Upp::MemoryFree_(ptr=1b7948c5050) at C:\upp-13648\uppsrc\Core\sheap.cpp 363
Upp::MemoryFree(ptr=1b7948c5070) at C:\upp-13648\uppsrc\Core\heapdbg.cpp 145
Upp::Rasterizer::AddCells(y=1993, n=1) at C:\upp-13648\uppsrc\Painter\Rasterizer.cpp 97
Upp::Rasterizer::LineRaw(x1=0, y1=510208, x2=0, y2=0) at C:\upp-13648\uppsrc\Painter\Rasterizer.cpp 275
Upp::Rasterizer::CvLine(x1=0, y1=1993, x2=0, y2=0) at C:\upp-13648\uppsrc\Painter\RasterizerClip.cpp 11

Upp::Rasterizer::LineClip(x1=-51570.6378196279, y1=1993, x2=-53122.6291349367, y2=0) at C:\upp-13648\uppsrc\Painter\RasterizerClip.cpp 41
Upp::Rasterizer::Line(p=39d391ce78->{ x=-63112.5752626844, y=-12828.6559571624 }) at C:\upp-13648\uppsrc\Painter\RasterizerClip.cpp 100
Upp::BufferPainter::RenderPathSegments(g=1b7915676e0->{ }, path=1b7948293a0->{ vector=1b7917e71e0 "", items=120 'x', alloc=156 }, attr=1b791567630->{ mtx={ x={ x=1, y=0 }, y={ x=0, y=1 }, t={ x=0, y=0 } }, miter_limit=4, opacity=1, dash=0, evenodd=1, join=3, cap=0, invert=0 }, tolerance=0.3) at C:\upp-13648\uppsrc\Painter\Render.cpp 80
Upp::BufferPainter::RenderPath(width=-1, ss=39d391d6e0->{ ptr=1b791801460 }, color=39d391d6b0->{ b=0, g=0, r=0, a=0 }) at C:\upp-13648\uppsrc\Painter\Render.cpp 312
Upp::BufferPainter::RenderImage(width=-1, image=1b79482e358->{ data=1b79482e260 }, transsrc=39d391d860->{ x={ x=6.58393520168497, y=5.12775868972685 }, y={ x=-5.12705203855568, y=6.58484552575728 }, t={ x=-63112.5752626844, y=-12828.6559571624 } }, flags=128) at C:\upp-13648\uppsrc\Painter\Image.cpp 275
Upp::BufferPainter::FillOp(image=1b79482e358->{ data=1b79482e260 }, transsrc=39d391d860->{ x={ x=6.58393520168497, y=5.12775868972685 }, y={ x=-5.12705203855568, y=6.58484552575728 }, t={ x=-63112.5752626844, y=-12828.6559571624 } }, flags=128) at C:\upp-13648\uppsrc\Painter\Image.cpp 284
Upp::Painter::Fill(image=1b79482e358->{ data=1b79482e260 }, transsrc=39d391d860->{ x={ x=6.58393520168497, y=5.12775868972685 }, y={ x=-5.12705203855568, y=6.58484552575728 }, t={ x=-63112.5752626844, y=-12828.6559571624 } }, flags=128) at C:\upp-13648\uppsrc\Painter\Painter.hpp 88
ChartPainter::PaintIMAGE2D(painter=1b791567468->{ }, object=1b79482e310->{ xform={ x={ x=0.671614273585535, y=0.523072573168923 }, y={ x=0.523000489073552, y=-0.671707133952751 }, t={ x=-1781644.69061699, y=7673915.55762473 } }, image={ data=1b79482e260 }, flags=1573104, selected=0, approved=0, highlight=0, linestyle=15, no_rotate=0, hidden=0, bold=0, italic=, .. }, optimize=1) at C:\Users\tom\D5\program52\ChartCtrl\ChartPainter.h 1086
ChartPainter::PaintBranch0(painter=1b791567468->{ }, branch=1b79482d320->{ visible=1, name={ len=45 '-', s=1b79482acc8 "S:\\maps\\kkj\\ImagemGoogleEarthKK".., chr=1b79482d328

qptr=1b79482acc8, v=1b79482d328, w=1b79482d328, q=1b79482d328 }, layerclass=0, fillstyle=1, fillcolor=, .. }, optimize=1) at C:\Users\tom\D5\program52\ChartCtrl\ChartPainter.h 1214
ChartPainter::PaintBranch(painter=1b791567468->{ }, branch=1b79482d320->{ visible=1, name={ len=45 '-', s=1b79482acc8 "S:\\maps\\kkj\\ImagemGoogleEarthKK".., chr=1b79482d328

qptr=1b79482acc8, v=1b79482d328, w=1b79482d328, q=1b79482d328 }, layerclass=0, fillstyle=1, fillcolor=, .. }, optimize=1) at C:\Users\tom\D5\program52\ChartCtrl\ChartPainter.h 1237
ChartPainter::Paint(ib=1b791567318->{ kind=0, size={ cx=3068, cy=1993 }, pixels={ ptr=1b7c9901070 }, hotspot={ x=0, y=0 }, spot2={ x=0, y=0 }, dots={ cx=7392, cy=4802 }, resolution=-1 }, charts=1b79155c4c8->{ vector={ vector=1b791554e60, items=1, alloc=3 } }) at C:\Users\tom\D5\program52\ChartCtrl\ChartPainter.h 1335
ChartView::Render() at C:\Users\tom\D5\program52\ChartCtrl\ChartCtrl.h 1803
ChartView::MouseMove(p=39d391e610->{ x=1191, y=843 }, keyflags=4194304) at C:\Users\tom\D5\program52\ChartCtrl\ChartCtrl.h 2357
Upp::Ctrl::MouseEvent(event=32 ' ', p=39d391e798->{ x=1191, y=843 }, zdelta=0, keyflags=4194304) at C:\upp-13648\uppsrc\CtrlCore\CtrlMouse.cpp 148
Upp::Ctrl::MouseEvent0(event=32 ' ', p=39d391e870->{ x=1191, y=843 }, zdelta=0,

keyflags=4194304) at C:\upp-13648\uppsrc\CtrlCore\CtrlMouse.cpp 92
Upp::Ctrl::MouseEventH(event=32 ' ', p=39d391e9c0->{ x=1191, y=843 }, zdelta=0,
keyflags=4194304) at C:\upp-13648\uppsrc\CtrlCore\CtrlMouse.cpp 110
Upp::Ctrl::MEvent0(e=32 ' ', p=39d391eaf8->{ x=1191, y=843 }, zd=0) at
C:\upp-13648\uppsrc\CtrlCore\CtrlMouse.cpp 337
Upp::Ctrl::DispatchMouseEvent(e=32 ' ', p=39d391ef68->{ x=1963, y=843 }, zd=0) at
C:\upp-13648\uppsrc\CtrlCore\CtrlMouse.cpp 602
Upp::Ctrl::DispatchMouse(e=32 ' ', p=39d391efd0->{ x=1963, y=843 }, zd=0) at
C:\upp-13648\uppsrc\CtrlCore\CtrlMouse.cpp 586
Upp::Ctrl::DoMouse(e=32 ' ', p=39d391f368->{ x=1963, y=843 }, zd=0) at
C:\upp-13648\uppsrc\CtrlCore\Win32Wnd.cpp 568
Upp::Ctrl::WindowProc(message=512, wParam=2, lParam=55248811) at
C:\upp-13648\uppsrc\CtrlCore\Win32Proc.cpp 210
Upp::TopWindow::WindowProc(message=512, wParam=2, lParam=55248811) at
C:\upp-13648\uppsrc\CtrlCore\TopWin32.cpp 70
Upp::Ctrl::WndProc(hWnd=5c0954->{ unused=?? }, message=512, wParam=2,
lParam=55248811) at C:\upp-13648\uppsrc\CtrlCore\Win32Wnd.cpp 671
CallWindowProcW()
DispatchMessageW()
Upp::Ctrl::sProcessMSG(msg=39d391f990->{ hwnd=5c0954, message=512, wParam=2,
lParam=55248811, time=542712750, pt={ x=1963, y=888 } }) at
C:\upp-13648\uppsrc\CtrlCore\Win32Wnd.cpp 768
Upp::Ctrl::ProcessEvent(quit=39d391fa20->0) at C:\upp-13648\uppsrc\CtrlCore\Win32Wnd.cpp
791
Upp::Ctrl::ProcessEvents(quit=39d391fa20->0) at C:\upp-13648\uppsrc\CtrlCore\Win32Wnd.cpp
806
Upp::Ctrl::EventLoop(ctrl=0) at C:\upp-13648\uppsrc\CtrlCore\Win32Wnd.cpp 839
GuiMainFn_() at C:\Users\tom\x\y\z\main.cpp 111
Upp::AppExecute__(app=7ff6b4c516e7) at C:\upp-13648\uppsrc\Core\App.cpp 343
WinMain(hInstance=7ff6b4c20000->{ unused=9460301 }, __formal=0, lpCmdLine=1b78f952343
"", nCmdShow=10) at C:\Users\tom\x\y\z\main.cpp 17
invoke_main()
__scrt_common_main_seh()
__scrt_common_main()
WinMainCRTStartup()
BaseThreadInitThunk()
RtlUserThreadStart()

Thanks and best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Mon, 21 Oct 2019 14:08:50 GMT
View Forum Message <> Reply to Message

Mirek,

Now I went back and encountered similar issues on U++ 13068 using MSBT17x64 too. (The reason I have not discovered this problem before, is probably that I have not had this large images to render until now.)

This time it crashed while my program called SetSurface(), but stack trace showed crash again in Upp::Dbl_Unlink(). Is this possibly an allocator issue, or could I have a very strange bug in my code? Debugger does not report any memory leaks if I exit the application before it crashes.

Best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by mirek on Mon, 21 Oct 2019 14:35:16 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 21 October 2019 16:08Mirek,

Now I went back and encountered similar issues on U++ 13068 using MSBT17x64 too. (The reason I have not discovered this problem before, is probably that I have not had this large images to render until now.)

This time it crashed while my program called SetSurface(), but stack trace showed crash again in Upp::Dbl_Unlink(). Is this possibly an allocator issue, or could I have a very strange bug in my code? Debugger does not report any memory leaks if I exit the application before it crashes.

Best regards,

Tom

Well, that is weird indeed as 13068 is old allocator... So maybe, just maybe it is not allocator problem after all.

Can you post trace for 13068 too please?

Mirek

Subject: Re: 2019.2 (pre)released
Posted by mirek on Mon, 21 Oct 2019 15:02:50 GMT
View Forum Message <> Reply to Message

If it would be possible to catch again the crash through AddCells (in 2019.2), could you switch in debugger to that frame and tell me the value the debugger reports for "a.alloc" ?

Mirek

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Mon, 21 Oct 2019 16:05:13 GMT
View Forum Message <> Reply to Message

OK. I will try this tomorrow.

Thanks,

Tom

Subject: Re: 2019.2 (pre)released
Posted by mirek on Tue, 22 Oct 2019 07:18:53 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 21 October 2019 17:02If it would be possible to catch again the crash through AddCells (in 2019.2), could you switch in debugger to that frame and tell me the value the debugger reports for "a.alloc" ?

Mirek

Also.. could you please use "Copy backtrace of all threads" for both U++ versions? (I would like to know what other threads are doing...)

Subject: Re: 2019.2 (pre)released
Posted by mirek on Tue, 22 Oct 2019 07:46:39 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Mon, 21 October 2019 12:47
There is a minor annoyance with Glib2: I get a lot of deprecation messages for glib2 (2.62) types:

/usr/include/glib-2.0/glib/gtypes.h:549:26: bilgi: declared here
  549 | typedef struct _GTimeVal GTimeVal
GLIB_DEPRECATED_TYPE_IN_2_62_FOR(GDateTime);

Nothing big, but bloats the error console on up-to-date setups.

OK, so as we have to investigate issue reported by Tom error anyway, we might try to fix this as well.

Unfortunately, it seems like internal inconsistency of gtk2 dev libraries (AFAIK, I do not even use GTimeVal anywhere). Unfortunately, this error does not happend on any of distros I have available, so it is a bit hard for me to fix, but I guess we should fix it by specifying max gtk2 version somehow, e.g. -DGLIB_VERSION_MAX_ALLOWED=GLIB_VERSION_2_60 in compiler flags or perhaps even before gtk/glib include (that is prefered IMO) or by -Wno-deprecated-warnings in compiler flags.

https://tecnocode.co.uk/2019/08/24/gtimeval-deprecation-in-g lib-2-61-2/

Mirek
https://github.com/clearlinux/clr-installer/pull/545

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 08:49:15 GMT
View Forum Message <> Reply to Message

Hi,

I think I'm getting closer to a solution:

First, I think it is not the allocator as USEMALLOC crashes too. So, I guess this is good news.

Anyway, it seems clear is that the heap gets corrupted somehow.

The issue disappears if I use:

BufferPainter bpainter(ib);

in my rendering routine instead of:

bpainter.Create(ib);

along with the class wide BufferPainter.

--

Resizing the window very easily causes the crash. Resizing the window also resizes/re-allocates the ImageBuffer used for rendering. This imagebuffer is associated with the BufferPainter through Create(ib). Could this be the real issue? If so, how can I go around it?

I will now get some MT backtraces to show...

Best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 09:03:14 GMT
View Forum Message <> Reply to Message

Mirek,

Here's one backtrace for 2019.2. They are different for each crash.

Thread ID: 0xfa0

Upp::Dbl_Unlink<Upp::BlkHeader_<256> >(x=1eed9c1f840->{ prev=ff182412ff0f1b0c, next=ff15210dff182412, prev_size=5387, size=65291, free=1, last=1, heap=ff0f1b0cff091408 })
Upp::BlkHeader_<256>::UnlinkFree()
Upp::BlkHeap<Upp::Heap::LargeHeapDetail,256>::Free(h=1eed9c20340->{ prev=6cc, next=1ee90d31050, prev_size=11, size=8, free=0, last=0, heap=1eed4fed460 })
Upp::Heap::LFree(ptr=1eed9c20350)
Upp::Heap::Free(ptr=1eed9c20350)
Upp::MemoryFree_(ptr=1eed9c20350)
Upp::MemoryFree(ptr=1eed9c20370)
Upp::Vector<Upp::Ctrl::MoveCtrl>::RawFree(ptr=1eed9c20370->{ ctrl={ prec=1eed6a40090 }, from={ left=2297, top=1995, right=2325, bottom=2035 }, to={ left=3812, top=1995, right=3840, bottom=2035 } })
Upp::Vector<Upp::Ctrl::MoveCtrl>::Free()
Upp::Vector<Upp::Ctrl::MoveCtrl>::Clear()
Upp::AMap<Upp::Ctrl *,Upp::Ctrl::MoveCtrl,Upp::Vector<Upp::Ctrl::MoveCtrl> >::Clear()
Upp::Ctrl::SyncMoves()
Upp::Ctrl::WndProc(hWnd=3b04fe->{ unused=?? }, message=3, wParam=0, lParam=2949120)
CallWindowProcW()
SendMessageW()
GetWindowTextW()
MapWindowPoints()
IsCompositionActive()
IsCompositionActive()
GetWindowTextW()
Upp::Ctrl::WindowProc(message=71 'G', wParam=0, lParam=592725532528)
Upp::TopWindow::WindowProc(message=71 'G', wParam=0, lParam=592725532528)
Upp::Ctrl::WndProc(hWnd=3b04fe->{ unused=?? }, message=71 'G', wParam=0, lParam=592725532528)
CallWindowProcW()
DispatchMessageW()
MBToWCSEx()
KiUserCallbackDispatcher()
NtUserMessageCall()
GetWindowTextW()
MapWindowPoints()
IsCompositionActive()
Ordinal96()

IsCompositionActive()
IsCompositionActive()
GetWindowTextW()
Upp::Ctrl::WindowProc(message=274, wParam=61488, lParam=14944857)
Upp::TopWindow::WindowProc(message=274, wParam=61488, lParam=14944857)
Upp::Ctrl::WndProc(hWnd=3b04fe->{ unused=?? }, message=274, wParam=61488,
lParam=14944857)
CallWindowProcW()
DispatchMessageW()
IsWindowVisible()
KiUserCallbackDispatcher()
NtUserMessageCall()
GetWindowTextW()
MapWindowPoints()
IsCompositionActive()
Ordinal96()
IsCompositionActive()
IsCompositionActive()
GetWindowTextW()
Upp::Ctrl::WindowProc(message=161, wParam=9, lParam=14944857)
Upp::TopWindow::WindowProc(message=161, wParam=9, lParam=14944857)
Upp::Ctrl::WndProc(hWnd=3b04fe->{ unused=?? }, message=161, wParam=9,
lParam=14944857)
CallWindowProcW()
DispatchMessageW()
Upp::Ctrl::sProcessMSG(msg=8a0131fc00->{ hwnd=3b04fe, message=161, wParam=9,
lParam=14944857, time=613160312, pt={ x=2649, y=228 } })
Upp::Ctrl::ProcessEvent(quit=8a0131fc90->0)
Upp::Ctrl::ProcessEvents(quit=8a0131fc90->0)
Upp::Ctrl::EventLoop(ctrl=0)
GuiMainFn_()
Upp::AppExecute__(app=7ff6b05416e7)
WinMain(hInstance=7ff6b0510000->{ unused=9460301 }, __formal=0, lpCmdLine=1eed4ff2211
"", nCmdShow=10)
invoke_main()
__scrt_common_main_seh()
__scrt_common_main()
WinMainCRTStartup()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1db4

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0xad8

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0xe44

ZwDelayExecution()
SleepEx()
LicenseEnforcement::t1cb()
<lambda_2eac692c55e4d4ef341436c9b465fd37>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_2eac692c55e4d4ef341436c9b465fd37>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6a53660)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6a53660)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed5088ff0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0xec0

ZwDelayExecution()
SleepEx()
LicenseEnforcement::t2cb()
<lambda_2eac692c55e4d4ef341436c9b465fd37>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_2eac692c55e4d4ef341436c9b465fd37>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6a53720)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6a53720)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed5089190)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x146c

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x27d0

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x220

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=0)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3460)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3460)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed8920a90)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x300

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=2)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac32e0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac32e0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed8924ab0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x28c4

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=1)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac33a0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac33a0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed8924f40)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x2a44

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=3)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac34a0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac34a0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed891cfb0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0xc54

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,

LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=5)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3760)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3760)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed891d110)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1abc

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=4)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac36a0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac36a0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed891bb30)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1fbc

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=6)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3820)

invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3820)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed8980bc0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x21c

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=7)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac38e0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac38e0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed898fe60)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x182c

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=8)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac39a0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac39a0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed899eb70)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1a68

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=9)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3a60)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3a60)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed899f460)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1974

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=10)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3b20)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3b20)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed89ada40)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1f68

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=11)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()

Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3be0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3be0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed89bbe10)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x23cc

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=12)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3ca0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3ca0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed9c6eb60)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x2b4c

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=13)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3d60)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3d60)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed89bc760)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x7f8

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=14)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3e20)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3e20)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed89bd680)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1e88

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=15)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3ee0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3ee0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed9cbab60)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x1e78

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,

LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=16)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6ac3fa0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6ac3fa0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed89bff30)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x54c

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff6b21420c8->{ section={ DebugInfo=1eed50d7820,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=17)
<lambda_ae0911e3bb62843b7fb2857684178ebc>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_ae0911e3bb62843b7fb2857684178ebc>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=1eed6afb0a0)
invoke_thread_procedure(procedure=7ff6b0a69670, context=1eed6afb0a0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=1eed9cd8030)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 0x87c

NtUserGetMessage()
GetMessageA()
Upp::Ctrl::Win32OverwatchThread(__formal=0)
BaseThreadInitThunk()
RtlUserThreadStart()

Next I will go for the earlier version...

Best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 09:10:20 GMT
View Forum Message <> Reply to Message

Hi,

Here's the backtrace for 13068:

Thread ID: 149c3f0

Upp::Dbl_Unlink<Upp::Heap::DLink>(x=14fc259ef30->{ next=ff182616ff152412, prev=ff1a2616ff172212 })
Upp::Heap::DLink::Unlink()
Upp::Heap::DivideBlock(b=14fc259ef30->{ next=ff182616ff152412, prev=ff1a2616ff172212 }, size=4016)
Upp::Heap::TryLAlloc(ii=38 '&', size=4016)
Upp::Heap::LAlloc(size=72cf1d1a0->4016)
Upp::Heap::AllocSz(sz=72cf1d1a0->4016)
Upp::MemoryAlloc_(sz=4016)
Upp::MemoryAlloc(size=3752)
operator new[](size=3752)
Upp::Buffer<Upp::RGBA>::Alloc(size=938)
Upp::RasterEncoder::Create(sz=72cf1d2d8->{ cx=938, cy=2 })
Upp::RasterEncoder::Create(sz=72cf1d330->{ cx=938, cy=2 }, pal_raster=72cf1d510->{ })
Upp::Rescale(tgt=72cf1d560->{ size={ cx=938, cy=2 }, dots={ cx=0, cy=0 }, hotspot={ x=0, y=0 }, line=7ff78094ff18, scanline={ ptr=0?? }, line_bytes=-1011478400, h={ ptr=0 }, palette={ ptr=0 }, palette_cv={ ptr=0 }, format={ type=9, rmask=7, gmask=2128677031, bmask=32759, rpos=2, gpos=1, bpos=0, apos=3 } }, tsz=72cf1d548->{ cx=938, cy=2 }, src=72cf1d510->{ }, src_rc=72cf1d8d8->{ left=2, top=0, right=18, bottom=2 }, progress=72cf1d540->{ ptr=0 })
Upp::Rescale(src=72cf1d8e8->{ data=14ffa1433a0 }, sz=72cf1d690->{ cx=938, cy=2 }, src_rc=72cf1d8d8->{ left=2, top=0, right=18, bottom=2 }, progress=72cf1d688->{ ptr=0 })
Upp::ChImageMaker::Make()
Upp::scImageMaker::Make(object=14fc3b610a0->{ data=0 })
Upp::LRUCache<Upp::Image,Upp::String>::Get(m=72cf1d808->{ })
Upp::MakeImage__(m=72cf1d8c8->{ }, paintonly=0)
Upp::MakeImage(m=72cf1d8c8->{ })
Upp::ChDraw(w=72cf1eec0->{ }, x=2, y=0, cx=938, cy=2, img=72cf1d958->{ data=14ffa1433a0 }, src=72cf1dcb8->{ left=2, top=0, right=18, bottom=2 })
Upp::StdChLookFn(w=72cf1eec0->{ }, r=72cf1e008->{ left=0, top=0, right=942, bottom=1590 },

q=72cf1df98 }, svo=7ff78069cfe0 }, op=2)
Upp::sChOp(w=72cf1eec0->{ }, r=72cf1e008->{ left=0, top=0, right=942, bottom=1590 },

q=72cf1df98 }, svo=7ff78069cfe0 }, op=2)
Upp::ChPaintEdge(w=72cf1eec0->{ }, r=72cf1e008->{ left=0, top=0, right=942, bottom=1590 },

q=72cf1df98 }, svo=7ff78069cfe0 })
Upp::LookFrame::FramePaint(w=72cf1eec0->{ }, r=72cf1e008->{ left=0, top=0, right=942, bottom=1590 })
Upp::Ctrl::CtrlPaint(w=72cf1eec0->{ pageSize={ cx=3840, cy=2160 }, nativeSize={ cx=3840, cy=2160 }, nativeDpi={ cx=192, cy=192 }, palette=0, color16=0, is_mono=0, native=0, actual_offset_bak={ x=754052888, y=7 }, cloff={ vector={ vector=14fc25317a0, items=8, alloc=8 } }, drawingclip={ left=0, top=0, right=942, bottom=1590 }, .. }, clip=72cf1e578->{ left=0, top=0, right=942, bottom=1993 })
Upp::Ctrl::CtrlPaint(w=72cf1eec0->{ pageSize={ cx=3840, cy=2160 }, nativeSize={ cx=3840, cy=2160 }, nativeDpi={ cx=192, cy=192 }, palette=0, color16=0, is_mono=0, native=0, actual_offset_bak={ x=754052888, y=7 }, cloff={ vector={ vector=14fc25317a0, items=8, alloc=8 } }, drawingclip={ left=0, top=0, right=942, bottom=1590 }, .. }, clip=72cf1e898->{ left=0, top=0, right=3840, bottom=1993 })
Upp::Ctrl::CtrlPaint(w=72cf1eec0->{ pageSize={ cx=3840, cy=2160 }, nativeSize={ cx=3840, cy=2160 }, nativeDpi={ cx=192, cy=192 }, palette=0, color16=0, is_mono=0, native=0, actual_offset_bak={ x=754052888, y=7 }, cloff={ vector={ vector=14fc25317a0, items=8, alloc=8 } }, drawingclip={ left=0, top=0, right=942, bottom=1590 }, .. }, clip=72cf1ebb8->{ left=0, top=0, right=3840, bottom=1993 })
Upp::Ctrl::CtrlPaint(w=72cf1eec0->{ pageSize={ cx=3840, cy=2160 }, nativeSize={ cx=3840, cy=2160 }, nativeDpi={ cx=192, cy=192 }, palette=0, color16=0, is_mono=0, native=0, actual_offset_bak={ x=754052888, y=7 }, cloff={ vector={ vector=14fc25317a0, items=8, alloc=8 } }, drawingclip={ left=0, top=0, right=942, bottom=1590 }, .. }, clip=72cf1f500->{ left=0, top=0, right=3840, bottom=2035 })
Upp::Ctrl::UpdateArea0(draw=72cf1f590->{ pageSize={ cx=3840, cy=2160 }, nativeSize={ cx=3840, cy=2160 }, nativeDpi={ cx=192, cy=192 }, palette=0, color16=0, is_mono=0, native=0, actual_offset_bak={ x=2128683405, y=32759 }, cloff={ vector={ vector=0, items=0, alloc=0 } }, drawingclip={ left=-1073741823, top=-1073741823, right=1073741823, .. }, .. }, clip=72cf1f500->{ left=0, top=0, right=3840, bottom=2035 }, backpaint=2)
Upp::Ctrl::UpdateArea(draw=72cf1f590->{ pageSize={ cx=3840, cy=2160 }, nativeSize={ cx=3840, cy=2160 }, nativeDpi={ cx=192, cy=192 }, palette=0, color16=0, is_mono=0, native=0, actual_offset_bak={ x=2128683405, y=32759 }, cloff={ vector={ vector=0, items=0, alloc=0 } }, drawingclip={ left=-1073741823, top=-1073741823, right=1073741823, .. }, .. }, clip=72cf1f500->{ left=0, top=0, right=3840, bottom=2035 })
Upp::Ctrl::WindowProc(message=15, wParam=0, lParam=0)
Upp::TopWindow::WindowProc(message=15, wParam=0, lParam=0)
Upp::Ctrl::WndProc(hWnd=4104d6->{ unused=?? }, message=15, wParam=0, lParam=0)
CallWindowProcW()
DispatchMessageW()
IsWindowVisible()
KiUserCallbackDispatcher()
NtUserDispatchMessage()
DispatchMessageW()
Upp::Ctrl::sProcessMSG(msg=72cf1fbc0->{ hwnd=4104d6, message=15, wParam=0, lParam=0,

time=613725437, pt={ x=947, y=913 } })
Upp::Ctrl::ProcessEvent(quit=72cf1fc50->0)
Upp::Ctrl::ProcessEvents(quit=72cf1fc50->0)
Upp::Ctrl::EventLoop(ctrl=0)
GuiMainFn_()
Upp::AppExecute__(app=7ff77ed11c6e)
WinMain(hInstance=7ff77ece0000->{ unused=9460301 }, __formal=0, lpCmdLine=14fc0921b43
"", nCmdShow=10)
invoke_main()
__scrt_common_main_seh()
__scrt_common_main()
WinMainCRTStartup()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

NtWaitForMultipleObjects()
WaitForMultipleObjectsEx()
Ordinal154()
Ordinal154()
Ordinal154()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwDelayExecution()
SleepEx()
LicenseEnforcement::t1cb()

<lambda_2eac692c55e4d4ef341436c9b465fd37>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_2eac692c55e4d4ef341436c9b465fd37>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc26e27e0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc26e27e0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc09b7940)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwDelayExecution()
SleepEx()
LicenseEnforcement::t2cb()
<lambda_2eac692c55e4d4ef341436c9b465fd37>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_2eac692c55e4d4ef341436c9b465fd37>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc26e27a0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc26e27a0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc09b79d0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForWorkViaWorkerFactory()
RtlInitializeResource()
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()

Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=0)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751660)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751660)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc38fc8e0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=2)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751620)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751620)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc3902970)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=1)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()

Upp::sThreadRoutine(arg=14fc2751720)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751720)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc390eec0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=3)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751860)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751860)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc39024e0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=4)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751920)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751920)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc3910b00)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=7)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751b60)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751b60)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc394b760)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=5)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc27519e0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc27519e0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc38f90e0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=6)

<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751aa0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751aa0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc394b440)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=8)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751c20)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751c20)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc3983cd0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=9)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751ce0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751ce0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc3983fd0)
BaseThreadInitThunk()

RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=10)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751da0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751da0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc3984820)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=11)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751e60)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751e60)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc39a0cc0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()

Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=12)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751f20)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751f20)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc4b8cc00)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=13)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc2751fe0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc2751fe0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc4b8cf10)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=14)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()

Upp::sThreadRoutine(arg=14fc3ac10e0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc3ac10e0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc39a58a0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=15)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc3ac11a0)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc3ac11a0)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc4bb6d60)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=16)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc3ac1260)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc3ac1260)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc39a7fb0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

ZwWaitForAlertByThreadId()
RtlSleepConditionVariableCS()
SleepConditionVariableCS()
Upp::ConditionVariable::Wait(m=7ff780942fa8->{ section={ DebugInfo=14ff3c2cf60,
LockCount=-1, RecursionCount=0, OwningThread=0, LockSemaphore=ffffffffffffffff,
SpinCount=33556432 }, mti=0 })
Upp::CoWork::Pool::ThreadRun(tno=17)
<lambda_dd03c9ba596f2864e900bd67a2cd4ace>::operator()()
Upp::Function<void __cdecl(void)>::Wrapper<<lambda_dd03c9ba596f2864e900bd67a2cd4ace>
>::Execute()
Upp::Function<void __cdecl(void)>::operator()()
Upp::sThreadRoutine(arg=14fc3ac1320)
invoke_thread_procedure(procedure=7ff77f21b130, context=14fc3ac1320)
thread_start<unsigned int (__cdecl*)(void * __ptr64)>(parameter=14fc39a9bb0)
BaseThreadInitThunk()
RtlUserThreadStart()

Thread ID: 149c3f0

NtUserGetMessage()
GetMessageA()
Upp::Ctrl::Win32OverwatchThread(__formal=0)
BaseThreadInitThunk()
RtlUserThreadStart()

Best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 09:18:16 GMT
View Forum Message <> Reply to Message

Mirek,

I do not seem to be able to crash through AddCells(). Tried many times.

BR,

Tom

Subject: Re: 2019.2 (pre)released
Posted by mirek on Tue, 22 Oct 2019 09:37:39 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 22 October 2019 10:49

BufferPainter bpainter(ib);

in my rendering routine instead of:

bpainter.Create(ib);

along with the class wide BufferPainter.

ib is class wide too?

(Just to be in sync, 'class wide' means it exists for more than one Paint invocations, correct?)

Mirek

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 09:47:45 GMT
View Forum Message <> Reply to Message

They both (ib and bpainter) are actually members of different classes, but both classes are continuously alive throughout the life of the window.

BR,

Tom

EDIT: And yes, the same ib is used in bpainter many times over. Before each use, bpainter.Create(ib); is called. (Meanwhile, the ib may have been resized/reallocated between the calls.)

Subject: Re: 2019.2 (pre)released
Posted by mirek on Tue, 22 Oct 2019 09:51:04 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 22 October 2019 11:47They both (ib and bpainter) are actually members of different classes, but both classes are continuously alive throughout the life of the window.

BR,

Tom

Would it be possible to test with temporary ib? Just to get some clues...

Also, obvious reason, is not it possible that bpainter gets used (any method called) between ib being resized and Create called? (As in it would be referencing old data of ImageBuffer).

Is Finish called properly?

Mirek

Subject: Re: 2019.2 (pre)released
Posted by mirek on Tue, 22 Oct 2019 09:55:07 GMT
View Forum Message <> Reply to Message

Idea to test:

Start the program.

After all is 'stable', place breakpoint at place where you resize ib (probably by ib.Create ?).

Resize, breapoint hits. Remove it and place breakpoint at bpainter.Create and also into BufferPainter::RenderPath. Continue and check that RenderPath is not called before bpainter.Create...

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 10:20:01 GMT
View Forum Message <> Reply to Message

Hi,

Thanks! I will look into your suggestions next.

Meanwhile, here's a "crashable" testcase:

```
#include <CtrlLib/CtrlLib.h>
#include <Painter/Painter.h>

using namespace Upp;

class BufferPainterCreateCrash: public TopWindow{
 public:
```

```
  ImageBuffer ib;
  BufferPainter bpainter;
  Image largeimage;

  BufferPainterCreateCrash(){
   Sizeable().MaximizeBox().MinimizeBox();
   bpainter.PreClipDashed();

   ImageBuffer lib(10000,10000);
   BufferPainter bp(lib);
   bp.Clear(LtGray());
   largeimage=lib;
  }

  void Paint(Draw &draw){
   SetSurface(draw,Rect(ib.GetSize()),ib,ib.GetSize(),Point(0,0));
  }

  void Render(Point p){
   // Use local bpainter below instead of bpainter.Create to avoid crash:
   //BufferPainter bpainter(ib);
   //bpainter.PreClipDashed();

   bpainter.Create(ib);
   bpainter.EvenOdd();
   bpainter.Co();

   bpainter.Clear(White());

   bpainter.Rectangle(0,0,GetSize().cx,GetSize().cy);
   bpainter.Fill(largeimage);
   int fh=StdFont().GetHeight();

   bpainter.Text(10,fh,Format("Cursor now at %`, %`",p.x,p.y),StdFont()).Fill(Black());
   bpainter.Text(10,2*fh,"Grab the window by the edge and resize wildly. Expect
crash...",StdFont()).Fill(Black());
   bpainter.Finish();
   // Fast updates
   if(GetTopCtrl()&&GetTopCtrl()->GetHWND()){
    ViewDraw draw(this);
    SetSurface(draw,Rect(ib.GetSize()),ib,ib.GetSize(),Point(0,0));
   }

   // For slow / delayed updates use Refresh() instead:
   // Refresh();
  }

  void Layout(){
```

```
   ib.Create(GetSize());
   Render(Point(0,0));
  }

  void MouseMove(Point p, dword keyflags){
   Render(p);
  }
};


GUI_APP_MAIN
{
 BufferPainterCreateCrash().Execute();

}
```

Best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 10:37:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 22 October 2019 12:55Idea to test:

Start the program.

After all is 'stable', place breakpoint at place where you resize ib (probably by ib.Create ?).

Resize, breapoint hits. Remove it and place breakpoint at bpainter.Create and also into BufferPainter::RenderPath. Continue and check that RenderPath is not called before bpainter.Create...

It takes quite a few repeats until it crashes, so I cannot get it to crash this way in near future. However, if crash is not required: This entered Create first and only thereafter to RenderPath. If GUI is single threaded, I do not think it should do it any other way.

The testcase is logically pretty close to what I'm doing here, so let's focus on that for a while as it crashes very similarly, shall we?

Best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by mirek on Tue, 22 Oct 2019 10:42:16 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 22 October 2019 12:37
The testcase is logically pretty close to what I'm doing here, so let's focus on that for a while as it crashes very similarly, shall we?

Best regards,

Tom

Definitely, I have not hoped for testcase, that is perfect. Thank you.

Mirek

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 10:51:21 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 22 October 2019 12:51Tom1 wrote on Tue, 22 October 2019 11:47They both (ib and bpainter) are actually members of different classes, but both classes are continuously alive throughout the life of the window.

BR,

Tom

Would it be possible to test with temporary ib? Just to get some clues...

Also, obvious reason, is not it possible that bpainter gets used (any method called) between ib being resized and Create called? (As in it would be referencing old data of ImageBuffer).

Is Finish called properly?

Mirek

Temporary ib would change the rendering in a big way as it is my base for getting Paint and Render separated.

While that would be a sensible explanation for what is happening here, I do not think it is possible for bpainter to be used when ib is out-of-order after resizing. As you can see from the testcase, Layout doing ib.Create calls Render doing bpainter.Create(ib) immediately.

Best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by Oblivion on Tue, 22 Oct 2019 10:55:07 GMT
View Forum Message <> Reply to Message

Quote:
OK, so as we have to investigate issue reported by Tom error anyway, we might try to fix this as well.

Unfortunately, it seems like internal inconsistency of gtk2 dev libraries (AFAIK, I do not even use GTimeVal anywhere). Unfortunately, this error does not happend on any of distros I have available, so it is a bit hard for me to fix, but I guess we should fix it by specifying max gtk2 version somehow, e.g. -DGLIB_VERSION_MAX_ALLOWED=GLIB_VERSION_2_60 in compiler flags or perhaps even before gtk/glib include (that is prefered IMO) or by -Wno-deprecated-warnings in compiler flags.

https://tecnocode.co.uk/2019/08/24/gtimeval-deprecation-in-g lib-2-61-2/

Mirek
https://github.com/clearlinux/clr-installer/pull/545

My workaround was to suppress the message in CtrlCore/Gtk.h
This should be qualified though. IMO the auppression should be disabled in release mode.

```
#ifdef __GNUC__
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wdeprecated-declarations"
#endif

#include <gtk/gtk.h>
#include <gdk/gdkkeysyms.h>

#ifdef __GNUC__
#pragma GCC diagnostic pop
#endif
```

Best regards,
Oblivion

Subject: Re: 2019.2 (pre)released

Posted by mirek on Tue, 22 Oct 2019 11:51:45 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 22 October 2019 12:20Hi,

Thanks! I will look into your suggestions next.

Meanwhile, here's a "crashable" testcase:

Hopefully fixed - does not crash in trunk anymore.

Can you please check the trunk in your application?

Mirek

Subject: Re: 2019.2 (pre)released
Posted by mirek on Tue, 22 Oct 2019 12:22:42 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Tue, 22 October 2019 12:55Quote:
OK, so as we have to investigate issue reported by Tom error anyway, we might try to fix this as well.

Unfortunately, it seems like internal inconsistency of gtk2 dev libraries (AFAIK, I do not even use GTimeVal anywhere). Unfortunately, this error does not happend on any of distros I have available, so it is a bit hard for me to fix, but I guess we should fix it by specifying max gtk2 version somehow, e.g. -DGLIB_VERSION_MAX_ALLOWED=GLIB_VERSION_2_60 in compiler flags or perhaps even before gtk/glib include (that is prefered IMO) or by -Wno-deprecated-warnings in compiler flags.

https://tecnocode.co.uk/2019/08/24/gtimeval-deprecation-in-g lib-2-61-2/

Mirek
https://github.com/clearlinux/clr-installer/pull/545

My workaround was to suppress the message in CtrlCore/Gtk.h
This should be qualified though. IMO the auppression should be disabled in release mode.

```
#ifdef __GNUC__
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wdeprecated-declarations"
#endif
```

```
#include <gtk/gtk.h>
#include <gdk/gdkkeysyms.h>

#ifdef __GNUC__
#pragma GCC diagnostic pop
#endif
```

Best regards,
Oblivion

Thank you, applied. I am not much worried about release here... This is temporary solution until the next release anyway.

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Tue, 22 Oct 2019 12:33:23 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks for your prompt action on this. Now I have already left the office for today, but I will test it as soon as I get back there tomorrow.

Thanks and best regards,

Tom

Subject: Re: 2019.2 (pre)released
Posted by Tom1 on Wed, 23 Oct 2019 06:34:44 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 22 October 2019 14:51Tom1 wrote on Tue, 22 October 2019 12:20Hi,

Thanks! I will look into your suggestions next.

Meanwhile, here's a "crashable" testcase:

Hopefully fixed - does not crash in trunk anymore.

Can you please check the trunk in your application?

Mirek

Thanks Mirek! The fix works perfectly.

Best regards,

Tom
