Subject: 2023.2
Posted by mirek on Tue, 24 Oct 2023 09:17:09 GMT
View Forum Message <> Reply to Message

I consider now 2023.2 released...

https://sourceforge.net/p/upp/news/2023/10/u-20232/
https://sourceforge.net/projects/upp/files/upp/2023.2/

Subject: Re: 2023.2
Posted by Tom1 on Tue, 24 Oct 2023 12:27:40 GMT
View Forum Message <> Reply to Message

Thanks again for your hard work!

Best regards,

Tom

Subject: Re: 2023.2
Posted by jjacksonRIAB on Tue, 24 Oct 2023 15:41:21 GMT
View Forum Message <> Reply to Message

Great job and thank you, Mirek.

Subject: Re: 2023.2
Posted by zsolt on Tue, 24 Oct 2023 21:19:10 GMT
View Forum Message <> Reply to Message

Thank you, Mirek!

Subject: Re: 2023.2
Posted by Klugier on Tue, 24 Oct 2023 22:00:17 GMT
View Forum Message <> Reply to Message

Thank you Mirek and good job! As previously in addition ot SourceForge release I created release on GitHub:
- https://github.com/ultimatepp/ultimatepp/releases/tag/2023.2

Subject: Re: 2023.2
Posted by BetoValle on Wed, 25 Oct 2023 02:42:43 GMT
View Forum Message <> Reply to Message

Hi,

https:// www.ultimatepp.org/forums/index.php?t=msg&th=12133&s tart=0&

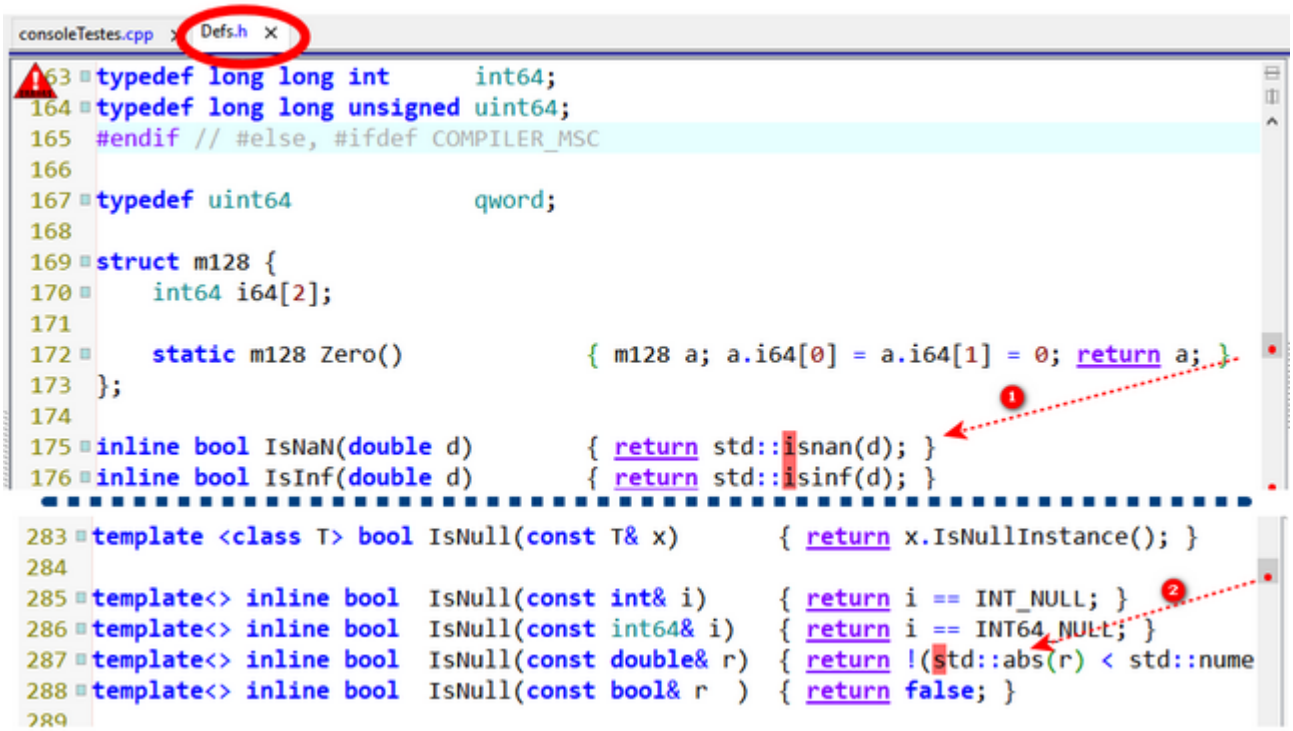
I don't know what your criteria is for previously reported bugs. I just noticed that the above occurrence continues.

Despite the visual flag being presented as an error, all routines involving the projects I compiled did not have problems with this current version.

Thanks!

File Attachments

1) 2023-10-24_232957.png, downloaded 259 times



Subject: Re: 2023.2
Posted by mirek on Wed, 25 Oct 2023 07:47:44 GMT
View Forum Message <> Reply to Message

Klugier wrote on Wed, 25 October 2023 00:00Thank you Mirek and good job! As previously in addition ot SourceForge release I created release on GitHub:
- https://github.com/ultimatepp/ultimatepp/releases/tag/2023.2

Thank you!

Mirek

Subject: Re: 2023.2
Posted by Tom1 on Tue, 14 Nov 2023 14:30:59 GMT
View Forum Message <> Reply to Message

Hi,

One of my application windows lost context menu (and timer) with U++ 2023.2. I was able to track the breakdown to U++ git commit 608b204.

Here are the details:

In one of my windows (in a multi window app) deeper in the hierarchy MenuBar::Execute(THISBACK(ContextMenu)) runs a context menu once, but refuses to start again until I have closed and reopened that window again. It gets stuck in Win32Wnd.cpp in the while loop:

```
bool Ctrl::ProcessEvents(bool *quit)
{
 ASSERT_(IsMainThread(), "ProcessEvents can only run in the main thread");
 if(ProcessEvent(quit)) {
  while(ProcessEvent(quit) && (!LoopCtrl || LoopCtrl->InLoop())); // LoopCtrl-MF 071008
  SweepMkImageCache();
  return true;
 }
 SweepMkImageCache();
 return false;
}
```

It seems that "LoopCtrl->InLoop()" never returns false, even after closing the popup menu. I tried to follow deeper, but my ability to understand CtrlCore proved insufficient.

Another, related issue is with TimeCallback that does not work at all in that same window either. Not even before using the context menu.

Unfortunately, this is a very complex program and I have not yet been able to build a separate test case to demonstrate the issue. Simple test cases do not reproduce this issue.

Best regards,

Tom

Subject: Re: 2023.2

Posted by mirek on Tue, 14 Nov 2023 14:48:44 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 14 November 2023 15:30
Another, related issue is with TimeCallback that does not work at all in that same window either. Not even before using the context menu.

Unfortunately, this is a very complex program and I have not yet been able to build a separate test case to demonstrate the issue. Simple test cases do not reproduce this issue.

MenuBar is doing some things through callbacks, so I think if we can make timer work for that window, it might fix the menu as well.

Anything special about TimeCallback or window? Can you paste TimeCallback related source snippets from the app?

Mirek

Subject: Re: 2023.2
Posted by Tom1 on Tue, 14 Nov 2023 15:00:25 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I removed the TimeCallback timer entirely to test, but it does not have any effect to the MenuBar issue...

But yes, this was there:

```
class MyWindow : public WithMyWindowLayout<TopWindow> {
 StatusBar    status;
...
 TimeCallback   timer;
 String    remotely_updated_string;

 void refreshStatus(){ // timer runs this at 100 ms intervals
  if(!IsForeground()) return;
  status = remotely_updated_string;
 }

...
 MyWindow(){
...
  timer.Set(-100,[&](){ refreshStatus(); });
...
 }
```

```
 ~MyWindow(){
...
  timer.Kill();
...
 }

   }

};
```

Best regards,

Tom

Subject: Re: 2023.2
Posted by Tom1 on Tue, 14 Nov 2023 15:07:15 GMT
View Forum Message <> Reply to Message

And more. MyWindow is constructed with new. I:

```
 MyWindow *mywindow;
...
 mywindow=new MyWindow(&mywindow);
 mywindow->OpenMain();
```

At a GUI_APP_MAIN my main window is started like this:

```
  One<MyMainWindow> mainwindow;
  mainwindow.Create();
  mainwindow->OpenMain();

  if(cmd.GetCount()) mainwindow->OpenFile(cmd[0]);
  Ctrl::EventLoop();
```

// Tom

Subject: Re: 2023.2
Posted by mirek on Tue, 14 Nov 2023 16:17:07 GMT
View Forum Message <> Reply to Message

Just quick note first:

```
 ~MyWindow(){
...
  timer.Kill();
...
 }
```

Not necessarry - TimerCallback destructor "kills itself". And no timer event can happen during destruction process.

Anyway, try

```
void refreshStatus(){ // timer runs this at 100 ms intervals
DDUMP(IsForeground());
  if(!IsForeground()) return;
  status = remotely_updated_string;
 }
```

Subject: Re: 2023.2
Posted by Tom1 on Tue, 14 Nov 2023 16:49:26 GMT
View Forum Message <> Reply to Message

Hi,

RDUMP/DDUMP does not print anything to log, so we're not getting any timer events at all here.

BTW: My app uses a whole lot of timers all around to do all kinds of background update tasks. Could this contribute to the problem?

Also, thanks for the tip: I can clean off the timer.Kill();s then.

BR, Tom

Subject: Re: 2023.2
Posted by mirek on Tue, 14 Nov 2023 22:48:07 GMT
View Forum Message <> Reply to Message

timer.Set(-100,[&](){ refreshStatus(); });

[&] is unusual here (and pretty error prone with timer). Normal is

timer.Set(-100, [=] { refreshStatus(); });

Now I do not frankly see a reason why [&] should not work in this very case, but can you try [=] anyway?

Also, make sure that all other timer (and GUI in general) lambdas are with [=]....

You should only use [&] if you are sure that lambda is invoked immediately (e.g. with CoDo).

Mirek

Subject: Re: 2023.2
Posted by koldo on Wed, 15 Nov 2023 07:38:47 GMT
View Forum Message <> Reply to Message

Hi Mirek

These sentences are very interesting.
Quote:Also, make sure that all other timer (and GUI in general) lambdas are with [=]....
You should only use [&] if you are sure that lambda is invoked immediately (e.g. with CoDo).
I almost understand why you say that but, for didactic reasons, could you explain why to include capture by value rather than by reference. Apparently, by reference is more efficient and every time the lambda function is called, it finds the classes in their current state, not in the initial state.

Subject: Re: 2023.2
Posted by mirek on Wed, 15 Nov 2023 09:41:30 GMT
View Forum Message <> Reply to Message

koldo wrote on Wed, 15 November 2023 08:38Hi Mirek

These sentences are very interesting.
Quote:Also, make sure that all other timer (and GUI in general) lambdas are with [=]....
You should only use [&] if you are sure that lambda is invoked immediately (e.g. with CoDo).
I almost understand why you say that but, for didactic reasons, could you explain why to include capture by value rather than by reference.

```
struct MyApp ... {
   Button btn;
   String a;
   ...
};

void MyApp::MyApp()
{
    String b;
    btn << [&] { a = b; };
}
```

This creates a dangling reference to b. In fact, if the lambda execution is not 'immediate' (before next statement basically), in most of GUI cases it makes only sense to work with member variables (e.g. 'a' in this case). But then, [=] in fact captures 'this' and that is likely MORE efficient than capturing all member variables individually.

Note that capturing this by [=] was deemed confusing by C++ comitee and they tried to improve on it with [=, this], unfortunately currently it is hard to make code compatible both ways.

Mirek

Subject: Re: 2023.2
Posted by Tom1 on Wed, 15 Nov 2023 10:14:33 GMT
View Forum Message <> Reply to Message

Hi,

It does not make difference if I use [=] here. However, I would not dare to point to local temporary variables with timer anyway. Instead, my timer sets are mostly of type timer.Set(THISBACK(UpdateRoutine));In that case the reference is all about using 'this'.

INTERESTINGLY: The entire timer system freezes when I open the secondary window. Ctrl::TimerProc() does not get called at all until I close that secondary window.

UPDATE: While the secondary window is open, there is not a single WM_TIMER coming into Ctrl::UtilityProc(). The execution of thread is not frozen within TimerProc() though. Windows is just no longer sending WM_TIMER messages in.

Best regards,

Tom

Subject: Re: 2023.2

Posted by mirek on Wed, 15 Nov 2023 11:17:51 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 15 November 2023 11:14Hi,

It does not make difference if I use [=] here. However, I would not dare to point to local temporary variables with timer anyway. Instead, my timer sets are mostly of type timer.Set(THISBACK(UpdateRoutine));In that case the reference is all about using 'this'.

INTERESTINGLY: The entire timer system freezes when I open the secondary window. Ctrl::TimerProc() does not get called at all until I close that secondary window.

UPDATE: While the secondary window is open, there is not a single WM_TIMER coming into Ctrl::UtilityProc(). The execution of thread is not frozen within TimerProc() though. Windows is just no longer sending WM_TIMER messages in.

Best regards,

Tom

Without seeing the whole logic of your app: Timer only runs when GUI is "idle". Any chance the secondory window is doing something like processing events itself?

Subject: Re: 2023.2
Posted by Tom1 on Wed, 15 Nov 2023 12:07:53 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I found a solution:

```
 virtual void Paint(Draw& draw){

  ...

  // scaleview->Refresh();
 }
```

I have a 'scaleview' control under the window. The window's Paint() calls scaleview->Refresh(); After this the timer is frozen until the window is closed. Removing this call to Refresh() makes everything work again.

The funny thing (if any in this case) is that this code is about 10-15 years old and has worked without any problem until now. (The scaleview gets painted anyway automatically, so this call was probably just causing some recursive feeding of more paint requests. So, I'm glad its gone now. Strange thing is that it was not visible on CPU loading...)

Anyway, Mirek, thanks for your help on this issue.

Best regards,

Tom

Subject: Re: 2023.2
Posted by koldo on Thu, 16 Nov 2023 12:03:05 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 15 November 2023 10:41koldo wrote on Wed, 15 November 2023 08:38Hi Mirek

These sentences are very interesting.
Quote:Also, make sure that all other timer (and GUI in general) lambdas are with [=]....
You should only use [&] if you are sure that lambda is invoked immediately (e.g. with CoDo).
I almost understand why you say that but, for didactic reasons, could you explain why to include capture by value rather than by reference.

```
struct MyApp ... {
   Button btn;
   String a;
   ...
};

void MyApp::MyApp()
{
    String b;
    btn << [&] { a = b; };
}
```

This creates a dangling reference to b. In fact, if the lambda execution is not 'immediate' (before next statement basically), in most of GUI cases it makes only sense to work with member variables (e.g. 'a' in this case). But then, [=] in fact captures 'this' and that is likely MORE efficient than capturing all member variables individually.

Note that capturing this by [=] was deemed confusing by C++ comittee and they tried to improve on it with [=, this], unfortunately currently it is hard to make code compatible both ways.

Mirek

Thank you Mirek for answering this off-topic question.
Additional information about this can be found here.
I didn't know this:Quote:[=] in fact captures 'this'

In summary, especially in classes, when we sometimes put a [&], we should really have to put [this], and in some cases a [=].
