Subject: Problem breaking loop (with close button) in main thread
Posted by awksed on Tue, 18 Oct 2022 15:19:00 GMT
View Forum Message <> Reply to Message

Windows 7. U++ 16270. Multi-thread app.

I have a GuiAcquireMutex() function called from a main thread Paint() that calls

WaitForSingleObject(hMutex, dwShortTimeout) // dwShortTimeout = 1 second (full timeout is 60 seconds)

if the mutex is not acquired I call:

Ctrl::GuiSleep(1000) to allow other threads to run and if 60 seconds have not elapsed, jump back to WaitForSingleObject().

I wish to allow the close button to call Close() (that sets boolean g_bQuit) and allow the user close the app (breaking the mutex acquire loop).

Clicking the close button results in the windows message "... is not responding".

Adding Ctrl::ProcessEvents() causes a "WM_PAINT invoked ... while in Paint routine" error.

Is there some way in U++ to allow the close button to call Close() while the main thread is looping (like Windows PumpWaitingMessages())?

Code:

```
DWORD GuiAcquireMutex(HANDLE hMutex, DWORD dwTimeout)
{
 if(hMutex == INVALID_HANDLE_VALUE)
   return ERROR_INVALID_HANDLE;

 long long llNow;
 long long llStart       = GetMilliTime();
 long long llEnd         = llStart + (long long) dwTimeout;
 DWORD     dwShortTimeout = 1000;       // 1 second
 DWORD     dwLastError;
 DWORD     dwWaitResult;

Again:

 dwLastError  = 0; // Success
 dwWaitResult = WaitForSingleObject(hMutex, dwShortTimeout);

 switch(dwWaitResult)
  {
```

```
  case WAIT_FAILED:

      llNow = GetMilliTime();

      if(llNow < llEnd)
       {
        if(g_bQuit)
         {
          dwLastError = ERROR_COUNTER_TIMEOUT;

          SetLastError(dwLastError);
          return dwLastError;
         }

        Ctrl::GuiSleep(1000); // Never sleeps for 1000 ms (always returns immediately)
        goto Again;
       }

      dwLastError = GetLastError();
      break;

  case WAIT_TIMEOUT:

      llNow = GetMilliTime();

      if(llNow < llEnd)
       {
        if(g_bQuit)
         {
          dwLastError = ERROR_COUNTER_TIMEOUT;

          SetLastError(dwLastError);
          return dwLastError;
         }

        Ctrl::GuiSleep(1000); // Never sleeps for 1000 ms (always returns immediately)
        goto Again;
       }

      dwLastError = ERROR_COUNTER_TIMEOUT;

      SetLastError(dwLastError);
 }

 return dwLastError;
}
```

Thanks.

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by Lance on Tue, 18 Oct 2022 17:15:09 GMT
View Forum Message <> Reply to Message

Hi Awksed:

I think what you can do is make your wait/check function(check_again()) re-entrant and instead of Sleep in the main thread, send check_again to a PostCallback and let the GUI thread(main thread) to arrange it be recalled after (roughly)specified time duration and do the check again.

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by Lance on Tue, 18 Oct 2022 18:09:54 GMT
View Forum Message <> Reply to Message

On a second thought, your program needs the hMutex be acquired to continue its work. Then ProcessEvent() should be the way to go. Problem is, why is that ProcessEvent() failed you?

Can you try something like the following

```
DWORD GuiAcquireMutex(HANDLE hMutex, DWORD dwTimeout)
{
 if(hMutex == INVALID_HANDLE_VALUE)
  return ERROR_INVALID_HANDLE;

 DWORD     dwLastError = -1;

 while ( !g_bQuit && dwLastError != 0 )
 {
  dwLastError  = 0; // Success

  switch( WaitForSingleObject(hMutex, dwShortTimeout ) )
  {
  case WAIT_FAILED:
   Do_Set_LastError_Accordingly();
   break;

  case WAIT_TIMEOUT:
   dwLastError = ERROR_COUNTER_TIMEOUT;
```

```
   break;
  }
  // consider measure how long it takes ProcessEvents to finish,
  // if very quick, consider add a Sleep(500) or something like that
  // to avoid keeping CPU busy for nothing.
  topwin.ProcessEvents();
  if(TooSoon() )
  {
   Sleep (500);
  }
 }
 SetLastError (dwLastError) ;

 return dwLastError;
}
```

topwin is your TopWindow(or its derivative) object.

I am feeling your original code didnot prepare for the case when the WaitForSingleObject call actually succeeds. I might be wrong though. Above pseudo code will need to be tuned to reflect your true intention.

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by awksed on Tue, 18 Oct 2022 23:11:56 GMT
View Forum Message <> Reply to Message

Hi Lance,

Thanks for your reply.

Quote:
I am feeling your original code didnot prepare for the case when the WaitForSingleObject call actually succeeds

The function has worked perfectly for the last 5 years (about 5,000,000 calls) until the mutex was not acquired (which has happened only once - the other day).

GuiAcquireMutex() is called multiple times within a function (that calls many sub-functions - about 10,000 lines of code) called by Paint() to refresh the data required for Paint() to redraw 2 complex graphs. So PostCallback() would be rather difficult to implement.

I am probably being a little pedantic but I like my apps to fail gracefully rather than freeze for a minute before telling the user to restart it.

Thanks for your thoughts.

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by Lance on Wed, 19 Oct 2022 00:21:55 GMT
View Forum Message <> Reply to Message

Hi Awksed:

My bad. Your code goto again; only when WaitForSingleEvent fails. I was thinking in the context that it has been converted into a while loop.

Quote:
Is there some way in U++ to allow the close button to call Close() while the main thread is looping (like Windows PumpWaitingMessages())?

I would use ProcessEvents() in the loop. No idea how it will cause failure in your case. Maybe you can add a flag in Paint() to not call GuiAcquireMutex(..) again if there is a previous call not completed yet?

I would think the ProcessEvents should be inserted somewhere in your GuiAcquireMutex function (before jump back to again maybe), but it's somewhere else needs to be changed accordingly if problem arose because of it.

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by Lance on Wed, 19 Oct 2022 00:42:02 GMT
View Forum Message <> Reply to Message

In the loop (by goto, there is an implicit loop), if you call ProcessEvent, which gives chance for Paint() who will call this function one more time and stuck in the same [b]WaitForSingleEvent[b] failure, and it will call [b]ProcessEvents[b] in its waiting loop, ... it goes on until stack is exhausted. I might be wrong, but that's my gut's feeling why adding ProcessEvent doesn't work for you.

I would add some check like this:

```
DWORD GuiAcquireMutex(HANDLE hMutex, DWORD dwTimeout)
{
  static int inthecall;

  if( inthecall )
      return ERROR_IN_CALL; // define this value yourself
  ++inthecall;
  if(hMutex == INVALID_HANDLE_VALUE)
   return ERROR_INVALID_HANDLE;
```

```
long long llNow;
long long llStart       = GetMilliTime();
long long llEnd         = llStart + (long long) dwTimeout;
DWORD     dwShortTimeout = 1000;       // 1 second
DWORD     dwLastError;
DWORD     dwWaitResult;
  ....
```

Make sure you --inthecall; in all branches before leaving the function.

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by awksed on Wed, 19 Oct 2022 18:56:19 GMT
View Forum Message <> Reply to Message

Hi Lance,

As I said in my original post

Quote:
Adding Ctrl::ProcessEvents() causes a "WM_PAINT invoked ... while in Paint routine" error.

Looking at ProcessEvents() (it calls Windows PeekMessage) I tried PeekMessage with PM_REMOVE and also PM_NOREMOVE. This make things much worse (totally hung the app). Used as per Microsoft example it works ok for the first loops and subsequently loops continuously.

I guess the problem is calling ProcessEvents/PeekMessage from within Paint.

As the app not responding for a minute (showing the wait cursor) if a mutex is never acquired (while allowing the window to be dragged using the title bar) and then telling the user to restart the app is not so bad. Especially only once every 5 years .

So I shall give up and stop spinning my wheels on this one.

Thanks again for your efforts.

Cheers.

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by Lance on Wed, 19 Oct 2022 23:04:39 GMT
View Forum Message <> Reply to Message

Hi Awksed

I am sorry to hear that you have to satisfy with a compromise.

I manage to create the following test. It runs as expected on Linux, but would not work on Windows.

Regards,
Lance

Correction: No, it's not a problem with ProcessEvents(). After I change the thread lambda to not log on GUI but to debug output file, it runs like a charm. It's interesting. I probably should not modify GUI from within other than the GUI thread. Seems Linux+GTK are more tolerant to this kind of error.

```
 void Start(){
  static int cnt;
  if( cnt!= 0 )
   return;
  ++ cnt;
  stop_requested = false;
  log.Clear();
  badguy.Run([this]{
   LOG("Badguy running...\n"); // CHANGE THIS
   Mutex::Lock _(mut);
   LOG("Badguy now owns the Mutex.\n");  // CHANGE THIS
   while(!stop_requested && !IsShutdownThreads() ){
    Sleep(50);
   }
   LOG("Stop requested, Badguy exiting and releasing the Mutex...\n"); // AND THIS
  });

  // wait for badguy to acquire the mutex
  while(mut.TryEnter()){
   mut.Leave();
   ProcessEvents();
   Sleep(100);
  }

  log.Append("Trying to acquire the mutex...\n");
  GuiAcquireMutex();
  --cnt;
 }
```

File Attachments

1) DeadMutex.zip, downloaded 307 times

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by Oblivion on Thu, 20 Oct 2022 18:10:59 GMT
View Forum Message <> Reply to Message

Hi Lance,

Quote:
probably should not modify GUI from within other than the GUI thread.

You are allowed to modify GUI from within other threads. For this purpose, you need to use

a) EnterGuiMutex() & LeaveGuiMutex();
b) GuiLock, which is basicly the above functions wrapped in a class for convenience:

Rudimentary example:

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

using namespace Upp;

#define LAYOUTFILE <DeadMutex/DeadMutex.lay>
#include <CtrlCore/lay.h>

static std::atomic<bool> fin;

class DeadMutex : public WithDeadMutexLayout<TopWindow> {

 typedef DeadMutex CLASSNAME;

public:
 DeadMutex()
 {
  CtrlLayout(*this, "Window title");
  start.WhenPush = THISFN(Start);
  stop.WhenPush = [this]{ fin = true; };
  WhenClose = [this]{ ShutdownThreads(); Break(); };
 }

 void Start(){
  Thread().Run([=] {
   {
    GuiLock __; // EnterGuiMutex()
    log.Append("Badguy instance running & owned the gui mutex...\n");
    // LeaveGuiMutex();
   }
   while(!fin && !IsShutdownThreads() ){
    Sleep(50);
```

```
    }
    GuiLock __;
    log.Append("Stop requested, Badguy instance exiting...\n");
    fin = false;
   });
  }
};

GUI_APP_MAIN
{
 DeadMutex().Run();
}
```

There is also a GuiUnlock class which basically reverses the order of LeaveGuiMutex and EnterGuiMutex, so you can force the main thread to temporarily unlock its GUI lock.

What you are not allowed to do is, creating windows and prompts within other threads.

Best regards,
Oblivion

Subject: Re: Problem breaking loop (with close button) in main thread
Posted by Lance on Thu, 20 Oct 2022 19:49:01 GMT
View Forum Message <> Reply to Message

Great! Thank you very much, Oblivion!
