
Subject: Code before Thread.Run() nor executed
Posted by [rafiwui](#) on Mon, 14 Aug 2017 11:23:02 GMT
[View Forum Message](#) <> [Reply to Message](#)

I have these methods:

```
void StartGeneration()
{
    // Disable buttons
    buttonBar.generate.Enable(false);
    buttonBar.next.Enable(false);
    buttonBar.previous.Enable(false);

    // Switch page
    displayLayouts[buttonBar.pageIndex].Show(false);
    generationProgress.Show(true);

    // Setup main progress bar
    generationProgress.progressMain.Set(0, 3);

    // Start thread
    generationThread.Run(THISBACK(Generate));
    //Thread::Start(THISBACK(Generate));

    return;
}
```

It is called from a button click event.

My problem is that nothing but the Run()-method of the thread is executed or at least there is now visual change in my application but there should be a layout change.

Also it is kinda not started as a thread because the application "freezes" while the thread is running.

Any ideas how I can fix this?

EDIT: Because I was asked for in this thread: Here is an example application to reproduce the problem.

File Attachments

1) [Bugtester.7z](#), downloaded 366 times

Subject: Re: Code before Thread.Run() nor executed
Posted by [Oblivion](#) on Tue, 15 Aug 2017 06:05:05 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Daniel,

Again this information is not really enough. :)

What does Generate() do, and how does it?

If I may presume it is manipulating the UI, then there are rules your thread needs to obey.

It is really hard to say anything based on this code snippet.

Best regards,

Oblivion

Subject: Re: Code before Thread.Run() nor executed

Posted by [rafiwui](#) on Tue, 15 Aug 2017 07:07:24 GMT

[View Forum Message](#) <> [Reply to Message](#)

Sorry. I thought it would be enough :?

Added an valid example as zipped folder for you ;)

Subject: Re: Code before Thread.Run() nor executed

Posted by [Oblivion](#) on Tue, 15 Aug 2017 16:45:56 GMT

[View Forum Message](#) <> [Reply to Message](#)

As I suspected, you are calling GUI elements/methods from a worker thread directly. You need to use PostCallback, and make sure the thread terminates.

Here you go:

```
TestLayout::TestLayout()
{
    CtrlLayout(*this);
    button1 << THISBACK(OnClickButton);
}
```

```
void TestLayout::OnClickButton()
{
    WhenThreadStart();
}
```

```
TestLayout2::TestLayout2()
{
    CtrlLayout(*this);
    progress.Set(0, 1);
}
```

```
TestWindow::TestWindow()
{
    SetRect(0, 0, Zx(400), Zy(200));
}
```

```

Add(testLayout);
Add(testLayout2);
testLayout.Show(true);
testLayout2.Show(false);
testLayout.button1.WhenAction = THISBACK(StartProgress); // button1's WhenAction callback
was not defined.
}

```

```

void TestWindow::StartProgress()
{
    testLayout.Show(false);
    testLayout2.Show(true);
    testLayout2.progress.Set(0, 10);
    Thread().Run(THISBACK(Progress));
}

```

```

void TestWindow::Progress()
{
    // This thread code could be written in a much better way, but I'm simply giving you the idea :).
    for(int i = 0; i < 10; i++)
    {
        if(Thread::IsShutdownThreads())
            break;
        Sleep(1000);
        PostCallback([=]{
            // Never directly call GUI elements. If you want to do fancy GUI stuff with
            // threads, use PostCallback.
            testLayout2.progress++;

        });
    }
    PostCallback([=]{
        // Let us roll-back;
        testLayout2.Hide();
        testLayout.Show();
    });
}

```

```

TestWindow::~TestWindow()
{
    // Make sure that running threads terminate.
    if(Thread::GetCount())
        Thread::ShutdownThreads();
}

```

Best regards,

Oblivion

Subject: Re: Code before Thread.Run() nor executed
Posted by [rafiwui](#) on Wed, 16 Aug 2017 07:43:36 GMT
[View Forum Message](#) <> [Reply to Message](#)

I changed the code to your code but still no layout change. So the main problem persists.
Any idea why?

Subject: Re: Code before Thread.Run() nor executed
Posted by [mirek](#) on Sat, 26 Aug 2017 11:16:08 GMT
[View Forum Message](#) <> [Reply to Message](#)

Oblivion wrote on Tue, 15 August 2017 18:45As I suspected, you are calling GUI elements/methods from a worker thread directly. You need to use PostCallback, and make sure the thread terminates.

Actually, for some time now many methods can be called in non-main thread, with GuiLock. Exception are window creation and event loop related things (the reason there is still the same: Windows associates event loop to the thread that created the window).

Subject: Re: Code before Thread.Run() nor executed
Posted by [mirek](#) on Sat, 26 Aug 2017 11:16:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

rafiwui wrote on Wed, 16 August 2017 09:43I changed the code to your code but still no layout change. So the main problem persists.
Any idea why?

Can you post a package here?

Mirek

Subject: Re: Code before Thread.Run() nor executed
Posted by [rafiwui](#) on Mon, 28 Aug 2017 06:38:17 GMT
[View Forum Message](#) <> [Reply to Message](#)

mirek wrote on Sat, 26 August 2017 13:16
Can you post a package here?

Here you go ;)

File Attachments

1) [Bugtester.7z](#), downloaded 373 times

Subject: Re: Code before Thread.Run() nor executed

Posted by [mirek](#) on Mon, 28 Aug 2017 07:27:55 GMT

[View Forum Message](#) <> [Reply to Message](#)

There is only a small typo: You have assigned StartProgress to TestLayout instead of button1:

```
TestWindow::TestWindow()
{
    SetRect(0, 0, Zx(400), Zy(200));
    Add(testLayout);
    Add(testLayout2);
    testLayout.Show(true);
    testLayout2.Show(false);
    testLayout.button1.WhenAction = THISBACK(StartProgress);
}
```

Here are some further hints:

```
TestWindow::TestWindow()
{
    SetRect(0, 0, Zx(400), Zy(200));
    Add(testLayout);
    Add(testLayout2);
    testLayout.Show(true);
    testLayout2.Show(false);
    testLayout.button1 << [=] {
        testLayout.Show(false);
        testLayout2.Show(true);
        testLayout2.progress.Set(0, 100);
        Thread::Start([=] {
            for(int i = 0; i < 100; i++) {
                if(Thread::IsShutdownThreads())
                    break;
                Sleep(100);
                GuiLock __;
                testLayout2.progress++;
            }
            GuiLock __;
            testLayout.Show(true);
            testLayout2.Show(false);
        });
    }
```

```
};  
}
```

Subject: Re: Code before Thread.Run() nor executed
Posted by [rafiwui](#) on Mon, 28 Aug 2017 07:39:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quote:There is only a small typo: You have assigned StartProgress to TestLayout instead of button1:
... :|
Thanks for finding that stupid mistake :lol:

Subject: Re: Code before Thread.Run() nor executed
Posted by [rafiwui](#) on Mon, 28 Aug 2017 08:51:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

How exactly does GuiLock work? Because when I tried to implement it in my application it doesn't work. Is there a documentation?

EDIT:

This is part of the code I use:

```
void MainWindow::Generate()  
{  
    // Generate  
    GuiLock __;  
    progress.progressMainLabel.SetText(t_("(0/3) Generating first..."));  
    if(!GenerateFirst())  
    {  
        AbortGeneration(1);  
        return;  
    }  
    progress.progressMain++;  
    progress.progressMainLabel.SetText(t_("(1/3) Generating second..."));  
    if(!Second())  
    {  
        AbortGeneration(2);  
        return;  
    }  
    progress.progressMain++;  
    progress.progressMainLabel.SetText(t_("(2/3) Generating third..."));  
    if(!GenerateThird())  
    {  
        AbortGeneration(3);  
        return;  
    }  
}
```

```
}  
progress.progressMain++;  
progress.progressMainLabel.SetText(t_("3/3) Done"));  
}
```

Instead of showing every step it just generates all three and at the end updates the gui. So what do I have to change that every step it is updated?

Subject: Re: Code before Thread.Run() not executed
Posted by [mirek](#) on Tue, 29 Aug 2017 15:20:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

rafiwui wrote on Mon, 28 August 2017 10:51 How exactly does GuiLock work? Because when I tried to implement it in my application it doesn't work. Is there a documentation?

EDIT:

This is part of the code I use:

```
void MainWindow::Generate()  
{  
    // Generate  
    GuiLock __;  
    progress.progressMainLabel.SetText(t_("0/3) Generating first..."));  
    if(!GenerateFirst())  
    {  
        AbortGeneration(1);  
        return;  
    }  
    progress.progressMain++;  
    progress.progressMainLabel.SetText(t_("1/3) Generating second..."));  
    if(!Second())  
    {  
        AbortGeneration(2);  
        return;  
    }  
    progress.progressMain++;  
    progress.progressMainLabel.SetText(t_("2/3) Generating third..."));  
    if(!GenerateThird())  
    {  
        AbortGeneration(3);  
        return;  
    }  
    progress.progressMain++;  
    progress.progressMainLabel.SetText(t_("3/3) Done"));  
}
```

Instead of showing every step it just generates all three and at the end updates the gui. So what do I have to change that every step it is updated?

That is because GUI is blocked while GuiLock is active...

GuiLock is "guard class" - it locks GUI mutex in constructor and releases it in destructor. While that mutex is locked, GUI is stopped and you can meddle with variables that GUI accesses (without using PostCallback).

In above function, you lock GUI at the start of function and unlock it at the end. That is why the GUI is only updated when Generate ends...

Subject: Re: Code before Thread.Run() nor executed
Posted by [rafiwui](#) on Tue, 29 Aug 2017 16:21:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

And how can I unlock it inside the function?

Subject: Re: Code before Thread.Run() nor executed
Posted by [mirek](#) on Tue, 29 Aug 2017 18:26:46 GMT
[View Forum Message](#) <> [Reply to Message](#)

rafiwui wrote on Tue, 29 August 2017 18:21 And how can I unlock it inside the function?

Put it into the block.

```
{  
    GuiLock ___;  
    do something  
} // destructor unlocks
```

Alternatively, GuiLock is really simple:

```
struct GuiLock {  
    GuiLock() { EnterGuiMutex(); }  
    ~GuiLock() { LeaveGuiMutex(); }  
};
```

You can call EnterGuiMutex() / LeaveGuiMutex() directly too...

BTW, note that "___" is my way of saying "name does not matter".

Mirek

Subject: Re: Code before Thread.Run() nor executed
Posted by [rafiwui](#) on Tue, 29 Aug 2017 18:30:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

mirek wrote on Tue, 29 August 2017 20:26
You can call EnterGuiMutex() / LeaveGuiMutex() directly too...

Ok thanks.

mirek wrote on Tue, 29 August 2017 20:26
BTW, note that "___" is my way of saying "name does not matter".

Yeah I thought so :d
