

Re: Why doesn't my thread terminate?

Source: <http://coding.derkeiler.com/Archive/Delphi/alt.comp.lang.borland-delphi/2008-05/msg00012.html>

- *From:* Rob Kennedy <me3@xxxxxxxxxxx>
 - *Date:* Fri, 02 May 2008 23:06:03 -0500
-

Prodigal Son wrote:

I'm trying to learn a thing or two about threads in Delphi, so I'm currently messing around with some examples and trying to create a few simple examples myself.

My current situation is the following: I have a form which has a button, and an editbox. I'd like to start a thread, and update the editbox at fixed intervals, from within the thread. But before I even got to that part, something strange happened.

Here are a few procedures from my unit:

```
-- Here I declare the thread, and start it
procedure TMainForm.Button9Click(Sender: TObject);
var
  thread : TMessageThread;
begin
  thread := TMessageThread.Create(true);
  thread.FreeOnTerminate := True;
  thread.OnTerminate := TMessageThreadTerminate;
  thread.Execute;
```

No. Never call Execute.

What you're doing is simply calling the TMessageThread object's Execute method in the context of the current thread. It's no different from calling any other method on any other regular class. Remember that TThread is just a class; there's nothing inherently magical about it that causes its methods to run in different OS threads.

It looks like you created the thread object suspended. A suspended thread does not execute until resumed, so when you're ready for the thread to run, call Resume on the object.

```
end;
```

```
-- After the thread has terminated, it should say so
procedure TMainForm.TMessageThreadTerminate(Sender: TObject);
begin
  ShowMessage('Terminated');
end;
```

Re: Why doesn't my thread terminate?

And it `_will_` say so, if the OS thread is ever given a chance to terminate. But you're not even letting it run. :)

```
-- Override for the execute procedure.  
procedure TMessageThread.Execute;  
var  
i: Integer;  
begin  
ShowMessage('Start');
```

No. Never call a VCL method from within a non-VCL thread. `ShowMessage` involves creating a `TForm` and calling all sorts of VCL methods that mustn't ever be called from secondary threads. Only do VCL stuff in the main VCL thread.

If a secondary thread needs to execute VCL stuff, then use the thread's `Synchronize` function.

```
for i := 0 to 1000 do  
begin  
Application.ProcessMessages;
```

No. There is only one `Application` object, and it receives all its messages in the main VCL thread since that's where it was created. (Message queues have thread affinity. If a thread has a message queue, it is completely separate from the message queues of any other threads unless you do some special setup, which you obviously haven't done, and which you definitely shouldn't do.)

Calling `Application.ProcessMessages` from a non-VCL thread is always wrong. When you do it, the `Application` object ends up processing messages from the current thread's queue instead of from the queue it expects to be reading from. Then it calls methods on itself and other objects that it expects are bound to the main thread, but now they're executing in an entirely different thread.

The only reason this appeared to work in your test was that you called the `Execute` method directly from your other code, so everything ended up executing in the main VCL thread anyway.

```
end;  
ShowMessage('Stop');  
Terminate;
```

You have the VCL source code. Go take a look at what the `Terminate` function does. The only thing that happens is that it sets the thread's `Terminated` property to `True`. The only reason for that is so your `Execute` method can check the `Terminated` property and know to stop executing.

What really signals a thread's termination is when the `Execute` method stops running. In your case, you achieve that simply by falling off the end of the procedure. Once the `Execute` method returns to its caller, other internal `TThread` code goes through the operations necessary to clean up the underlying OS thread and to execute the `OnTerminate` event handler.

Re: Why doesn't my thread terminate?

end;

Now, as soon as I click the button, a messagebox pops up (Start), followed by the stop messagebox a few instants later on. But I never get the "terminated" message!

This, I don't understand – I've specified the correct procedure (afaikt), and I've also tried to do a "DoTerminate" instead of a "Terminate", but to no avail.

Can anybody please explain to me what's going on?

Thank you very much!

PS.

--
Rob

.