

Re: AfxBeginThread on each user action X or re-use same thread (w

Re: AfxBeginThread on each user action X or re-use same thread (w

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2005-09/msg00458.html>

- *From:* "ultranet" <ultranet@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 12 Sep 2005 09:33:03 -0700
-

"William DePalo [MVP VC++] wrote:

```
> "ultranet" <ultranet@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
> <news:4AE37BF6-F877-4B55-8DB1-385A2B73BCA1@xxxxxxxxxxxxxxxxxxxx
>>> > But most importantly, will an unexpected termination, such as due to an
>>> > uncaught exception, be handled this way as well?
>>>
>>> If you are waiting on a thread handle, and the thread terminates due to
>>> an
>>> uncaught exception, then your wait will end with the exception.
>> I suppose this can be handled with a catch(...)?
>
> Well, yes but this catch-all catch clause will only sometimes catch Win32
> structured exceptions (e.g. access violations) usually in debug but not
> release builds. It is generally considered a defect in the compiler that it
> does that and it should be fixed in VS2005. If you really want to catch SEs
> such as access viloations you should use _set_se_translator() to translate
> SEs to C++ typed exceptions.
Yes, so catch(...) will catch all C++ exceptions, but things like access
violations will still cause the process to crash, unless they are also
handled w/ a _se_translator_function, or terminate.
```

So far i haven't changed:

```
CWinThread* pThread =
AfxBeginThread(
MyFunction, this, THREAD_PRIORITY_NORMAL);
for every occurence of that action. And i have a question: does pThread need
to be deleted at the end of MyFunction?
```

But i'll probably change this to a single thread, w/ the following:

```
DWORD exitCode;
GetExitCodeThread(yourThread, &exitCode);
if (exitCode == STILL_ACTIVE)
{
// thread still alive
}
and start a new thread if not STILL_ACTIVE. I hope i'll get exitCode of 0 if
the handle is no longer valid, cause the thread has terminated. Since
MyFunction currently doesn't throw, it could only be dead w/ process still
```

Re: AfxBeginThread on each user action X or re-use same thread (w

Re: AfxBeginThread on each user action X or re-use same thread (w

running when _se_translator_function, or terminate is added.

So i could add the code above, and it should always be alive as long as the process is alive. Then in the future i could add _se_translator_function, or terminate.

• **Follow-Ups:**

- ◆ **Re: AfxBeginThread on each user action X or re-use same thread (w**
◇ From: William DePalo [MVP VC++]

• **References:**

- ◆ **AfxBeginThread on each user action X or re-use same thread (w CEve**
◇ From: ultranet
 - ◆ **Re: AfxBeginThread on each user action X or re-use same thread (w CEve**
◇ From: William DePalo [MVP VC++]
 - ◆ **Re: AfxBeginThread on each user action X or re-use same thread (w**
◇ From: ultranet
 - ◆ **Re: AfxBeginThread on each user action X or re-use same thread (w**
◇ From: William DePalo [MVP VC++]
 - ◆ **Re: AfxBeginThread on each user action X or re-use same thread (w**
◇ From: ultranet
 - ◆ **Re: AfxBeginThread on each user action X or re-use same thread (w**
◇ From: William DePalo [MVP VC++]
- Prev by Date: **Re: which is method is better for database access!**
 - Next by Date: **Re: Proper place for interface comments: .h or .cpp?**
 - Previous by thread: **Re: AfxBeginThread on each user action X or re-use same thread (w**
 - Next by thread: **Re: AfxBeginThread on each user action X or re-use same thread (w**
 - Index(es):
 - ◆ **Date**
 - ◆ **Thread**