Re: Use of TerminateThread Function

Source:
http://www.tech−archive.net/Archive/WindowsCE/microsoft.public.windowsce.platbuilder/2008−12/msg00034.html

- *From*: Raj <pandeyraj79@xxxxxxxxx>
- *Date*: Mon, 1 Dec 2008 22:57:59 −0800 (PST)

On Dec 1, 9:44 pm, "Michel Verhagen (eMVP)" <mic...@xxxxxxxxxx> wrote:

> The preferred way to exit is of course graceful. TerminateThread doesn't
> clean up any resources allocated by the thread it's terminating and so
> using TerminateThread could result in memory leaks.
>
> Your while loop seems a bit strange... Why not use an event? That is
> much more efficient; it won't use up any CPU ticks until the event is
> set, and it will allow you to exit immediately instead of having to wait
> up to 1000 ms before the thread exits:

```
DWORD WINAPI MyThread(LPVOID lpParameter)
{
  HANDLE hEvent = (HANDLE)lpParameter;

  LPBYTE buffer = new BYTE[100];
  for (;;)
  {
     if (WAIT_OBJECT_0 == WaitForSingleObject(hEvent, INFINITE))
     {
        // Check if event was set by application exit code
        if (GetEventData(hEvent))
            break;
        // Do normal operation
        memcpy(buffer, "lalalalala", 10);
     }
     else
        break;
  }

  // Clean up any resources you allocated in this thread
  delete []buffer;

  return 0;

}
```

> Your application would create the thread and the event like this:

```
HANDLE hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
SetEventData(hEvent, FALSE);  // Indicate normal operation
HANDLE hThread = CreateThread(NULL, 0, MyThread, (LPVOID)hEvent, 0, NULL);
```

In this example I'm using Set/GetEventData to set/determine if this is a normal event or an "exit" event.

To kick the thread to do it's normal operation you'd:

```
SetEvent(hEvent);
```

To stop the thread gracefully you'd execute the following code:

```
SetEventData(hEvent, TRUE);  // Indicate thread needs to exit
SetEvent(hEvent);

// Give the thread 1 second to exit gracefully, otherwise terminate
if (WAIT_TIMEOUT == WaitForSingleObject(hThread, 1000))
    TerminateThread(hThread, 1);

// And clean up
CloseHandle(hEvent);
CloseHandle(hThread);
```

Good code will never execute the TerminateThread line! If you have to terminate a thread it's time for some code review...

(Note that it's also not good practice to assume LPVOID can hold a HANDLE, but for simplicity sake I've used it in this example.)

PS. I did not compile this code!

Good luck,

Michel Verhagen, eMVP
Check out my blog:http://GuruCE.com/blog

GuruCE
Microsoft Embedded Partner
http://GuruCE.com
Consultancy, training and development services.

Raj wrote:

> Hi All,
>
> I am working on an app in WM 6.1 and there is a thread that is sleeping in a while loop until some condition is met.

If I satisfy the condition and then exit out of the application it is working fine for me but if I don't the exit is giving me prefetch abort while unloading the DLL.

When I am explicitly terminating the thread before unloading the DLL in some DeInit function it is working fine but otherwise don't.

I have one question is it advisable to use TerminateThread.I know the code the thread will be executing when I will terminate the thread.

So basically it is the while loop that is causing the prefetch abort when I am exiting out without satisfying the condition.

the loop is very simple.

```
while(x==y)
{
  Sleep(1000);
}
```

where y is default behavior.

Appreciate any help.

Raj

Thanks Michel,C.L.

The code is same both for the application(*.EXE) and the DLL (Control Panel Applet (*.CPL)).

I am not sure Michel,when the application exits I mean the EXE it just exits gracefully but when the application(Control Panel Applet *.CPL (In turn Dll)) exits it just gives the prefetch abort because the

thread is in this while loop.

I tried the event options but it didn't work may be I need to review the code again and give it another try.

Anyways Michel really appreciated your help.

Raj

.