Why you should never suspend a thread

devblogs.microsoft.com/oldnewthing/20031209-00

December 9, 2003

Raymond Chen

It's almost as bad as terminating a thread.

Instead of just answering a question, I'm going to ask you the questions and see if you can come up with the answers.

Consider the following program, in (gasp) C#:

```
using System.Threading;
using SC = System.Console;
class Program {
  public static void Main() {
    Thread t = new Thread(new ThreadStart(Program.worker));
    t.Start();
    SC.WriteLine("Press Enter to suspend");
    SC.ReadLine();
    t.Suspend();
    SC.WriteLine("Press Enter to resume");
    SC.ReadLine();
    t.Resume();
  }
  static void worker() {
    for (;;) SC.Write("{0}\r", System.DateTime.Now);
  }
}
```

When you run this program and hit Enter to suspend, the program hangs. But if you change the worker function to just "for(;;) {}" the program runs fine. Let's see if we can figure out why.

The worker thread spends nearly all its time calling System.Console.WriteLine, so when you call Thread.Suspend(), the worker thread is almost certainly inside the System.Console.WriteLine code.

Q: Is the System.Console.WriteLine method threadsafe?

Okay, I'll answer this one: Yes. I didn't even have to look at any documentation to figure this out. This program calls it from two different threads without any synchronization, so it had better be threadsafe or we would be in a lot of trouble already even before we get around to suspending the thread.

Q: How does one typically make an object threadsafe?

Q: What is the result of suspending a thread in the middle of a threadsafe operation?

Q: What happens if – subsequently – you try to access that same object (in this case, the console) from another thread?

These results are not specific to C#. The same logic applies to Win32 or any other threading model. In Win32, the process heap is a threadsafe object, and since it's hard to do very much in Win32 at all without accessing the heap, suspending a thread in Win32 has a very high chance of deadlocking your process.

So why is there even a SuspendThread function in the first place?

Debuggers use it to freeze all the threads in a process while you are debugging it. Debuggers can also use it to freeze all but one thread in a process, so you can focus on just one thread at a time. This doesn't create deadlocks in the debugger since the debugger is a separate process.

Raymond Chen

Follow