Why does my program crash when I throw an exception from an APC?

devblogs.microsoft.com/oldnewthing/20171012-00

October 12, 2017

Raymond Chen

Consider the following code fragment:

```
class thread_exit_exception
{
};

DWORD CALLBACK WorkerThreadProc(void* parameter)
{
  try {
    while (true) {
      SleepEx(INFINITE, TRUE); // alertable
    }
  } catch (thread_exit_exception& e) {
  }
  return 0;
}

void CALLBACK ExitWorkerThreadApcProc(ULONG_PTR parameter)
{
  throw thread_exit_exception();
}

void MakeTheWorkerThreadExit()
{
    QueueUserAPC(ExitWorkerThreadApcProc, WorkerThreadHandle, 0);
}
```

In this case, the worker thread just sits around waiting to be told to exit, but in the general case, it would be doing work and check in periodically to see if somebody told it to exit.

The problem is that this sample program crashes. Why?

Well, this is similar to the issue of throwing a C++ exception from a structured exception, because you're throwing an exception from an operating system callback. The C++ compiler isn't expecting that, and it might optimize out the `try` / `catch`, in which case the exception you throw from the APC goes unhandled and terminates the process.

But wait, even before you get to this point, you're already in trouble, because you're throwing an exception across frames, even though not all of the frames in between are in on the joke.

So this plan is double-broken.

The correct way to do this is to set a flag from the APC, and the worker thread checks this flag after every alertable wait. If the flag is set, then an APC wants the thread to exit, and the thread can exit on its own terms. This avoids raising exceptions across foreign frames: Instead of raising the exception, you merely set a variable that says "I'd really like to raise an exception, but I can't, so can you pretend that I raised an exception? Thanks."

Alternatively, you could have the alertable wait check the flag, and if the flag is set, then it does a `throw thread_exit_exception()`. With this design, the variable means "I'd really like to raise an exception, but I can't, so can you raise the exception when it's safe to do so? Thanks."

Raymond Chen

Follow