Does the thread pool have different handle access privileges? Why am I getting ERROR_INVALID_HANDLE?

devblogs.microsoft.com/oldnewthing/20160129-00

January 29, 2016

Raymond Chen

A customer was observing strange behavior in their application with handles and the thread pool.

We have a service that spawns a child process, and under certain conditions, we need to terminate that child process. If I try to terminate the process immediately upon the condition being met, then everything works. But now we want to wait a little while before terminating the child process. To do that, we create a thread pool timer and terminate the process from the thread pool.

Here's the code that runs when we detect that the condition is met. No errors are detected except where noted.

```
PTP_TIMER timerTask = CreateThreadpoolTimer(
                            DelayTerminate,
                            static_cast<PVOID>(ProcHandle),
                            &m_CallBackEnviron);

if (NULL == timerTask) { ... }

// Set the timer to fire after a little while.

ulDueTime.QuadPart = (ULONGLONG)(-TimeOutIn100Nanoseconds);
FileDueTime.dwHighDateTime = ulDueTime.HighPart;
FileDueTime.dwLowDateTime = ulDueTime.LowPart;

SetThreadpoolTimer(timerTask, &FileDueTime, 0, 0);

// if we set the debugging flag, then the TerminateProcess call succeeds.
if (DebuggingFlag) {
  if (!TerminateProcess(ProcHandle, 1)) { ... }
}
```

Here is our callback function:

```
VOID
CALLBACK
DelayTerminate(
    PTP_CALLBACK_INSTANCE Instance,
    PVOID                 Parameter,
    PTP_TIMER             Timer
    )
{
  // This call to TerminateProcess fails
  if (!TerminateProcess((HANDLE)Parameter, 1)) {
    Log(GetLastError()); // ERROR_INVALID_HANDLE
  }
  CloseThreadpoolTimer(Timer);
}
```

Does the thread pool thread run with different access privileges from the main thread?

> We verified that the handle is the same in the main thread and in the callback. It is our understanding that `DuplicateHandle` is not needed to share handles between threads of a single process. Is there some other special thing that has to be done in order to share the handle between threads?

I asked, "Is it possible that somebody closed the handle in the meantime?" After all, if the problem were due to access, then you would expect the error to be `ERROR_ACCESS_DENIED`. Since the error is `ERROR_INVALID_HANDLE`, the most likely reason is, um, an invalid handle.

A clue that something strange is going on is the `static_cast<PVOID>(ProcHandle)`. This suggests that `ProcHandle` is not itself a `HANDLE`, but is rather some sort of RAII class that manages a process handle and which has an implicit conversion to `HANDLE`. (Because if `ProcHandle` were a `HANDLE`, then you wouldn't need to cast it to `PVOID`.)

The customer eventually wrote back,

> Yes, that was it. We found that the handle was being closed before the thread pool tried to use it. Thanks.

Raymond Chen

Follow