Why can't I delete a file immediately after terminating the process that has the file open?

devblogs.microsoft.com/oldnewthing/20120329-00

March 29, 2012

Raymond Chen

A customer discovered a bug where terminating a process did not close the handles that the process had open, resulting in their emergency cleanup code not working:

```
TerminateProcess(processHandle, EXITCODE_TERMINATED);
DeleteFile(someFile);
```

Their workaround was to insert a call to `WaitForSingleObject(processHandle, 500)` before deleting the file. The customer wanted to know whether they discovered a bug in `TerminateProcess`, and they were concerned that their workaround could add up to a half second to their cleanup code, during which the end user is sitting there waiting for everything to clean up.

As MSDN notes,

> **TerminateProcess** initiates termination and returns immediately. This stops execution of all threads within the process and requests cancellation of all pending I/O. The terminated process cannot exit until all pending I/O has been completed or canceled.

(Emphasis mine.)

Termination is begun, but the function does not wait for termination to complete. Sometimes a thread gets stuck because a device driver has gotten wedged (or the driver doesn't support cancellation).

To know when the handles are closed, wait on the process handle, because the process handle is not signaled until process termination is complete. If you are concerned that this can take too long, you can do like the customer suggested and wait with a timeout. Of course, if the timeout expires, then you have to decide what to do next. You can't delete the file, since it's still open, but maybe you can log an error diagnostic and let the user know why things are taking so long to clean up, and maybe add the file to a list of files to clean up the next time the program starts up.

Raymond Chen

Follow