

Re: Using CreateFile() in a CreateThread()

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.gdi/2005-06/msg00115.html>

- *From:* "Jason Doucette" <www.jasondoucette.com>
 - *Date:* Fri, 17 Jun 2005 21:06:37 -0300
-

Josh,

> Well, it makes no sense to me why the OS would have to check the folder
> information about other files when handed a direct path to the data file
> opened, but I guess that's one of the penalties you pay for the File
> system.

Just think what **you** would have to do, if given a path and filename, to find the file in question (assume that the files aren't sorted on the disk, either). Essentially, the OS **does** have to search for the file. When you find the data, the entire file isn't necessarily stored contiguously, either. When you start writing to a file, the system has to find (i.e. search for) an empty place on the disk. Perhaps several times for a single file, depending on its size. I know this explanation is highly un-technical, but I am just trying to give you a sense of why a path+filename isn't a direct link to where the data resides.

> The crash is probably because the time for creation of a new thread is
> constant at an x-constant interval (determined by the rate of the data
> collection vehicle). As driving continues, thread execution becomes
> slower to a point of y+n, once that exceeds x, we have a more threads
> being created in one set time period than finishing, causing a thread
> overflow and subsequent crash.

I was wondering if the threads were being generated at some constant rate, but I didn't ask, because I couldn't think of a reason why they would be. But, yes, this is definitely the cause of the crash...

> Other than using subfolders, is there any way to work around the bug of
> the
> increased load times on larger folders?

Ok, my advice would be to first fix the crash problem with the threads first. You should not rely on the HD being fast enough. The program should be able to handle this, regardless of what happens. What if another application starts accessing the disk, or if it is being written to over a network that gets bogged down? The application should be ok with this.

Re: Using CreateFile() in a CreateThread()

To answer your question, the best way is to shove everything into one really large file, and let your program handle the process of accessing different sections of this file. In other words, *you* do the handling of where the data gets stored. Right now, you are relying on the OS to do this, and are unhappy with it. Since you know the manner in which you access your own data, you likely you could code a better solution.

Other than creating one large file, you could store a bunch of files in a bunch of different directories. This would be unadvisable, but it can work. It will be slower than what I have described above.

I hope this helps...

--

Jason Doucette

<http://www.jasondoucette.com/>

<http://www.sawtoothdistortion.com/>

- **Follow-Ups:**

- ◆ **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
 - ◇ From: Josh McFarlane

- **References:**

- ◆ **[Using CreateFile\(\) in a CreateThread\(\)](#)**
 - ◇ From: Josh McFarlane
- ◆ **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
 - ◇ From: Christian Kaiser
- ◆ **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
 - ◇ From: Josh McFarlane
- ◆ **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
 - ◇ From: Jason Doucette
- ◆ **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
 - ◇ From: Josh McFarlane

- Prev by Date: **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
- Next by Date: **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
- Previous by thread: **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
- Next by thread: **[Re: Using CreateFile\(\) in a CreateThread\(\)](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**