Don’t forget, the fourth parameter to ReadFile and WriteFile is sometimes mandatory

devblogs.microsoft.com/oldnewthing/20130404-00

April 4, 2013

Raymond Chen

The `ReadFile` and `WriteFile` functions have a parameter called `lpNumberOfByteRead`, which is documented as

```
  __out_opt LPDWORD lpNumberOfBytesRead,
// or
  __out_opt LPDWORD lpNumberOfBytesWritten,
```

“Cool,” you think. “That parameter is optional, and I can safely pass `NULL`.”

> My program runs fine if standard output is a console, but if I redirect standard output, then it crashes on the `WriteFile` call. I verified that the handle is valid.
>
> ```
> int __cdecl main(int, char **)
> {
>   // error checking removed for expository purposes
>   HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
>   WriteFile(hStdOut, “hello”, 5, NULL, NULL);
>   return 0;
> }
> ```
>
> The crash occurs inside the `WriteFile` function trying to write to a null pointer.

But you need to read further in the documentation for `WriteFile`:

> ***lpNumberOfBytesWritten* [out, optional]**
> A pointer to the variable that receives the number of bytes written when using a synchronous *hFile* parameter. **WriteFile** sets this value to zero before doing any work or error checking. Use **NULL** for this parameter if this is an asynchronous operation to avoid potentially erroneous results.
>
> This parameter can be **NULL** only when the *lpOverlapped* parameter is not **NULL**.

That second paragraph is the catch: The parameter is sometimes optional and sometimes mandatory. The annotation language used in the function head is not expressive enough to say, “Sometimes optional, sometimes mandatory,” so it chooses the weakest annotation (“optional”) so as not to generate false positives when run through static code analysis tools.

With the benefit of hindsight, the functions probably should have been split into pairs, one for use with an `OVERLAPPED` structure and one without. That way, one version of the function would have a mandatory `lpNumberOfBytesWritten` parameter and no `lpOverlapped` parameter at all; the other would have a mandatory `lpOverlapped` parameter and no `lpNumberOfBytesWritten` parameter at all.

The crash trying to write to a null pointer is consistent with the remark in the documentation that the `lpNumberOfBytesWritten` is set to zero before any work is performed. As for why the code runs okay if output is not redirected: Appearing to succeed is a valid form of undefined behavior. It appears that when the output handle is a console, the rule about `lpNumberOfBytesWritten` is not consistently enforced.

At least for now.

Raymond Chen

Follow