Knowing just enough about debugging IRPs to chase the problem out of the I/O stack

devblogs.microsoft.com/oldnewthing/20171207-00

December 7, 2017

Raymond Chen

One of my colleagues was running a tool that wound up stuck on `FlushFileBuffers`. Since this was a hang in the I/O stack, a kernel dump is more useful.

I used the `!irp` debugger command to look at the I/O request that got stuck:

```
0: kd> !irp 0xffffab0c`fced9340 1

Irp is active with 2 stacks 2 is current (= 0xffffab0cfced9458)
 No Mdl: No System Buffer: Thread ffffab0d15731080:  Irp stack trace.
     cmd  flg cl Device   File     Completion-Context
 [N/A(0), N/A(0)]
            0  0 00000000 00000000 00000000-00000000

                        Args: 00000000 00000000 00000000 00000000
>[IRP_MJ_FLUSH_BUFFERS(9), N/A(0)]
            0  1 ffffab0cdf855060 ffffab0ce2c6eef0 00000000-00000000    pending
               \FileSystem\Npfs
                        Args: 00000000 00000000 00000000 00000000
```

I don't know what any of this means, but somebody else did.

The file system is `Npfs`, which is the "named pipe" file system. This means that the code is trying to flush a named pipe, and the process on the other end of the pipe is not responding.

With the help of debugger documentation I dumped the file object:

```
0: kd> !fileobj ffffab0ce2c6eef0

\contoso44268

Device Object: 0xffffab0cdf855060   \FileSystem\Npfs
Vpb is NULL

Flags:  0x40082
        Synchronous IO
        Named Pipe
        Handle Created

File Object is currently busy and has 1 waiters.

FsContext: 0xffffe30b60eefe70   FsContext2: 0xffffe30b23b593d3
Private Cache Map: 0x00000001
CurrentByteOffset: 0
```

I don't know what any of this means either, but the name of the named pipe is apparently `contoso44268`.

We provided this information to the owner of the tool, and they recognized it as a named pipe they use to communicate between the tool and a helper process, and the helper process in turn satisfies the pipe request by contacting a Web service.

The owner of the tool requested some diagnostic logs to figure out why the named pipe got stuck. But that's not the point of the story today. The point here is just being able to chase the stuck I/O out of kernel mode back into an application so the forward progress can be made.

Bonus reading: More on debugging the I/O system:

Raymond Chen

Follow