

Re: Question about FILE_OBJECT

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2006-07/msg00314.htm>

- *From:* "Doron Holan [MS]" <doronh@xxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 15 Jul 2006 17:46:51 -0700
-

instead of allocating/freeing the context in FsContext2 on the IOCTL calls, create a context in create and freed in close and have this context contain the list of contexts that you allocate on an IOCTL basis. this way, the structure of your data is localized per file object and can track better.

d

—

Please do not send e-mail directly to this alias. this alias is for newsgroup purposes only.

This posting is provided "AS IS" with no warranties, and confers no rights.

"John S" <JohnS@xxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:8599DBA8-6ECC-468E-9AEC-35D5D76F9543@xxxxxxxxxxxxxxxx

Ok so if i understand correctly the PFILE_OBJECT ptr for that handle will always be the same. I.E. when subsequent ioctl's are made.

In my actual code the context object stored in FsContext2 is allocated and stored on an ioctl that comes after the IRP_MJ_CREATE and can be freed on IRP_MJ_CLOSE or prior to that via another ioctl code that my driver handles.

My major concern was that somehow the location of the PFILE_OBJECT would change and my validation of the context object would fail even though the context object actually belonged to that file object.

Just an FYI the reason im storing the ptr of the PFILE_OBJECT in the context object is because i know its possible for the system to re-use that memory location (after i have freed the original context object) when i allocate another context object. So just making sure the value pointed to by FsContext2 is in my table only lets me know that the memory is valid, not that it actually belongs to the PFILE_OBJECT. By also checking to see that stored PFILE_OBJECT ptr in my context object matches the PFILE_OBJECT of the

Re: Question about FILE_OBJECT

current irp then i am certain that the context object is not only valid memory, but that the context object is actually associated w/ that PFILE_OBJECT.

"Doron Holan [MS]" wrote:

the use of FsContext and FsContext2 are dictated by each driver in the stack. you must make sure that no other driver (like NDIS itself) is using these fields. Once you know that you can use either field, you can set it and you own own. you don't have to validate it unless you are freeing it before IRP_MJ_CLOSE, you can blindly use the pointer b/c no one else knows about it. Every i/o that is sent on that handle will have the same PFILE_OBJECT pointer and the same pointer value you put into FsContext(2).

d

Please do not send e-mail directly to this alias. this alias is for newsgroup purposes only.

This posting is provided "AS IS" with no warranties, and confers no rights.

"John S" <John S@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:0EFAF252-02EC-4550-BD0E-0B2030CD5435@xxxxxxxxxxxxxxxxxxxx

Ok, here is the scenario. I have an ndis intermetiate driver. Basically a new handle to my intermdiate driver is opened for each nic adapter that a user application wants to talk to.

On the open call (IRP_MJ_CREATE) I allocate a context structure and add it to a global list. I also store the pointer of the allocated context memory in IoGetCurrentIrpStackLocation(pIrp)->FileObject->FsContext2; I also want to save the pointer of the FileObject to a member in the the context structure I allocated. Note that i will never attempt to read or

Re: Question about FILE_OBJECT

write
to
the location i will just be using it later for validation.

Now, on subsequent irp's instead of accessing the context
memory at
FileObject->FsContext2 object directly, I want to use
FsContext2 to
match
the
context structure in my global list (to assure that its valid).
When
i
find i match i also want to make sure the FileObject pointer i
stored
in
the
context memory matches the pointer to FileObject of the
current irp.

So my question is.. Will a FILE_OBJECT change memory
locations? I.E.
whence a file object is established, is it always found at that
location,
or can the os re-allocate and copy it to another memory
location.

I know that this is probably a ridiculous amount of
verification but
the
situation is a little more complicated than stated above, it's
just
easier
to explain this way.
John.