

## Re: atomicity of reads and writes in ntfs

---

*Source:*

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2006-09/msg00378.ht>

---

- *From:* Wingeezer <[Wingeezer@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:Wingeezer@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 26 Sep 2006 04:08:01 -0700
- 

Thanks a lot for your help, Don  
tony

"Don Burn" wrote:

No, I do not work for Microsoft. But I have seen filter driver designs that would impact this. If you want the official answer from Microsoft, ask on the NTFSD list at <http://www.osronline.com/> that is where the file system people including Microsoft hang out.

--  
Don Burn (MVP, Windows DDK)  
Windows 2k/XP/2k3 Filesystem and Driver Consulting  
<http://www.windrvr.com>  
Remove StopSpam from the email to reply

"Wingeezer" <[Wingeezer@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:Wingeezer@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)> wrote in message [news:8F4ECAA4-A22B-49B3-A2A4-59ED989AC234@xxxxxxxxxxxxxxxxxxxxx](mailto:news:8F4ECAA4-A22B-49B3-A2A4-59ED989AC234@xxxxxxxxxxxxxxxxxxxxx)

Thanks for the reply, Don. Forgive my ignorance but as you are responding in this managed newgroup, does this constitute an official Microsoft position on this question?

Reason I am asking is that the application I work on runs on many platforms, and I am the Windows platform integrator in the team. A colleague who is responsible for integration on IBM AIX has raised this with IBM and they say their operating kernel *\*does\** guarantee the atomicity of individual read/write operations. My boss then said to me "find out what the Microsoft position is on this" but my only route for developer support seems to be

Re: atomicity of reads and writes in ntfs

to  
ask in here.

Believe me I was very reluctant to post this question at all as my original response to my boss was the same as yours.

thanks,  
tony

"Don Burn" wrote:

You will never be able to rely on this, since even if this is the case for NTFS (not saying it is), it is undocumented behavior,. So all that needs to happen is a file system filter with a different model occurs on a machine, and your premise fails.

—  
Don Burn (MVP, Windows DDK)  
Windows 2k/XP/2k3 Filesystem and Driver Consulting  
<http://www.windrvr.com>  
Remove StopSpam from the email to reply

"Wingeezer" <Wingeezer@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>  
wrote in message  
[news:4B891B78-9C5A-47B7-B49B-6E40847D50DC@xxxxxxxxxxxxxxxxxxxx](mailto:news:4B891B78-9C5A-47B7-B49B-6E40847D50DC@xxxxxxxxxxxxxxxxxxxx)

I have a question about the serialisation or atomicity of performing read and write operations on NTFS files.

Suppose I write a 32k block of data to a file using the WriteFile api, and suppose at the same time I have another process using ReadFile to read the same 32k block. The two processes do not cooperate and therefore the

Re: atomicity of reads and writes in ntfs

read  
may  
occur before or after the write, but what I am  
interested in is  
whether  
the  
reading process can read a 'half written' 32k  
block.

So for example I have the writer program  
writing a 32k string  
111111.....  
then 222222.... then 3333333.... etc over and  
over. The reader program  
reads  
the 32k block and checks that the first and  
last characters are equal.  
It  
stops when it finds they are different which  
proves that the read has  
read  
some of the block prior to it being written  
and some after. Both  
programs  
use  
the ReadFile/WriteFile api calls with  
FILE\_FLAG\_NO\_BUFFERING.

What I have found in my testing is that this  
depends on the file  
fragmentation. If my target file is in one  
complete fragment then the  
reader  
never reports a difference in the block even  
up to blocks 6M in size.  
If  
however the file is in 2 or more fragments  
then the reader reports a  
difference even after a few hundred reads.

What I conclude from this is:

1. ntfs issues one physical i/o per file  
fragment.
2. A single physical i/o completes in an  
atomic fashion, ie. a single  
non-fragmented 6M write forces the reader  
to wait until the entire  
write  
is  
complete, and vice-versa.
3. A fragmented file results in multiple

Re: atomicity of reads and writes in ntfs

physical i/o's which then  
means  
that the ReadFile and WriteFile operations  
can become intermingled.  
4. By keeping my read and write sizes equal  
to or less than the volume  
cluster size, I should be able to guarantee the  
atomicity of  
individual  
read/write operations without having to have  
the programs lock each  
other  
manually with mutexes etc.

What I am looking for is some kind of  
official comment or endorsement  
of  
the  
above by someone who knows the internals,  
as some software I am  
working on  
is  
dependent on the truth of this behaviour.

Many thanks,  
tony