Subject: Re: Bug in IDL's FILE_INFO function
Posted by R.Bauer on Tue, 04 Sep 2007 11:03:23 GMT
View Forum Message <> Reply to Message

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Ive not tried but did you have used that example on a nfs mounted devivce ?

I am asking because nfs does not know atomic append.

cheers
Reimar

Dave Wuertz wrote:
> I suspect FSTAT is executing FLUSH whereas FILE_INFO must not be. I
> doubt FSTAT is closing and re-opening, unless that is the only way to
> perform the flush.
>
> I know the Linux OS has a buffering mechanism, but my experience has
> been that it is very smart and efficient. For example, I've witnessed
> Linux holding some recently written data in a buffer until *either* the
> buffer gets filled *or* another application needs to read from the file
> the data still being held in the buffer. It's like Linux is smart
> enough to know that it doesn't *really* need to physically write small
> amounts of information (a configurable OS parameter) unless it really
> *has* to.
>
> There is no mention of flushing or buffering in the documentation for
> FILE_INFO and FSTAT and both are supposed to return the current size of
> an open file. The fact is they behave differently (at least on my IDL
> version and platform). Interestingly, the doc for FLUSH, however, states
> that *IDL* "uses buffered output for reasons of efficiency". I'm
> wondering if IDL has it's own buffering mechanism on top of Linux's,
> though I cannot imagine why it would need it.
>
> My only real purpose in this post is to point out that those functions
> behave differently. This behavior should be either documented or the
> code modified to to give the same result.
>
> -Dave Wuertz
>
> Jean H. said the following on 8/30/2007 4:10 PM:
>> FSTAT returns the info of an open file, while FILE_INFO returns the
>> info of a file, opened or not.
>>

>> When you write to a file, it probably goes through some buffers (not
>> sure of this / how)... for example, if one writes to a text file and
>> tries to read this file in another program before it is closed, then
>> you would see nothing in the file... though as soon as it has been
>> closed by IDL, you can access it.
>>
>> So it doesn't look surprising that the FILE_INFO returns the previous
>> size... the question is, does FSTAT close and re-open the file for
>> you? ... it appear so as after a call to it, you get the correct size!
>>
>> Jean
>>
>> Dave Wuertz wrote:
>>> Folks,
>>>
>>> I believe there's a bug in the FILE_INFO function. I am running IDL
>>> v6.4 on Linux.
>>>
>>> I'm writing a program that does a lot of file updates and it's
>>> necessary for me to get the current file size after an update to an
>>> open file. I decided to use FILE_INFO rather than FSTAT because I
>>> also must first check to see if the file exists. FILE_INFO can tell
>>> you if the file exists as well it's size in bytes. It's also newer
>>> than FSTAT, so I thought I'd just use FILE_INFO exclusively in my
>>> program.
>>>
>>> Well, things just weren't making sense, and I boiled it down to
>>> this. If I append a new record to a file and immediately check the
>>> file size with FILE_INFO it gives me the wrong size. It returns the
>>> size BEFORE the record was added. However, FSTAT will give the
>>> correct new size. And, once FSTAT has been called, then FILE_INFO
>>> knows about the new size. It's like FSTAT issues a FLUSH, because
>>> the only way FILE_INFO gives the correct size is if FLUSH (or FSTAT)
>>> is called first. This is fine, however there is no mention in the
>>> documentation that FLUSH must be called first.
>>>
>>> Below is some code to illustrate the problem:
>>>
>>>
>>> pro file_info_vs_fstat
>>>
>>> fname = 'test.txt'
>>> openw, lun, /get_lun, fname
>>> nrec = 3
>>> for i = 0, nrec-1 do begin
>>>
>>> print, 'Before writing record file_info.size, fstat.size:', $
>>> (file_info(fname)).size, (fstat(lun)).size,

>>> format='(a,1x,2i6)'
>>>
>>>   printf, lun, 'This is record number ', i
>>>
>>>   print, 'After writing record file_info.size, fstat.size:', $
>>>   (file_info(fname)).size, (fstat(lun)).size,
>>> format='(a,1x,2i6)'
>>>
>>>   print, ' ' ; print blank line for readability
>>>
>>> endfor
>>> free_lun, lun
>>>
>>> return
>>> end
>>> ;;;;;;;;;;;;;;; Run above procedure
>>> ;;
>>> IDL> file_info_vs_fstat Before writing record file_info.size,
>>> fstat.size: 0   0
>>> After writing record file_info.size, fstat.size: 0   31
>>>
>>> Before writing record file_info.size, fstat.size: 31   31
>>> After writing record file_info.size, fstat.size: 31   62
>>>
>>> Before writing record file_info.size, fstat.size: 62   62
>>> After writing record file_info.size, fstat.size: 62   93
>>>
>>> ;; ;;;;;;;;;;;;;;;;;;;;;;
>>>
>>>
>>> Now, if you replace the "After" print statement with the following
>>> one that simply
>>> reverses the order the two functions are called, you then get the
>>> correct result from
>>> the FILE_INFO function:
>>>
>>>   print, 'After writing record fstat.size, file_info.size:', $
>>>   (fstat(lun)).size, (file_info(fname)).size,
>>> format='(a,1x,2i6)'
>>>
>>> IDL> file_info_vs_fstat Before writing record file_info.size,
>>> fstat.size: 0   0
>>> After writing record fstat.size, file_info.size: 31   31
>>>
>>> Before writing record file_info.size, fstat.size: 31   31
>>> After writing record fstat.size, file_info.size: 62   62
>>>
>>> Before writing record file_info.size, fstat.size: 62   62

>>> After writing record fstat.size, file_info.size: 93   93
>>>
>>> ;;; ;;;;;;;;;;;;;;;;;;;;;;
>>>
>>>
>>> Ciao,
>>>
>>> -Dave Wuertz

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.5 (GNU/Linux)
Comment: Using GnuPG with SUSE - http://enigmail.mozdev.org

iD8DBQFG3Tt65aOc3Q9hk/kRAlTUAJoC6Q3RcmYXiydJgQcGu1noj697JwCg hOi9
cxuD/K5ROHSpqlNwf7sOCyo=
=KpDO
-----END PGP SIGNATURE-----