Subject: Re: file_lines and expand path
Posted by James Kuyper on Fri, 04 Aug 2006 16:39:26 GMT
View Forum Message <> Reply to Message

FL wrote:
> Hi,
>
> I am implementing file_lines in FL and was wondering what to return when
> the input path expands to multiple files. The IDL reference guide says
> nothing. I have run IDL and got curious result:
>
> IDL> help, file_lines('*.pro')
> <Expression> LONG64 = 15
>
> I have 448 .pro files in the current directory, with a total of 10k lines.
> IDL seems to be randomly picking a single file and reporting its length
> (probably the first match in readdir()).
>
> What do you expect to get?
>
> Possibilities:
>
> 1. never expand (but why do we have NOEXPAND then?)

So that you can open files which have names containing characters that
happen to be wildcard characters.

Subject: Re: file_lines and expand path
Posted by Paul Van Delst[1] on Fri, 04 Aug 2006 18:04:00 GMT
View Forum Message <> Reply to Message

FL wrote:
> Hi,
>
> I am implementing file_lines in FL and was wondering what to return when
> the input path expands to multiple files. The IDL reference guide says
> nothing. I have run IDL and got curious result:
>
> IDL> help, file_lines('*.pro')
> <Expression> LONG64 = 15
>
> I have 448 .pro files in the current directory, with a total of 10k lines.
> IDL seems to be randomly picking a single file and reporting its length
> (probably the first match in readdir()).
>
> What do you expect to get?
>

> Possibilities:
>
> 1. never expand (but why do we have NOEXPAND then?)
> 2. give an error message if the expansion results in more than one file
> 3. return the total length of files
> 4. return the total length of files if a new keyword TOTAL was set
> 5. do as IDL, pick a single file silently

Well, if the routine allows for wildcard inputs, then I wouldn't expect *any* of the above.

What I would expect would be an array (in this case 448 elements in size) containing the lengths of *each* file that matched the wildcard expression, e.g.

IDL> help, file_lines('*.pro')
<Expression> LONG64 = Array[448]

Just because IDL does it wrong, doesn't mean you have to. :o)

paulv

p.s. Maybe FL should have the ability to run just like IDL. I.e. you set some sort of environment variable, say, FL_RUNLIKE_IDL=true, and it performs like IDL (i.e. giving the incorrect result (IMO) as in the above example). Set it to false, and FL behaves like a regular person would intuitively expect. :o)

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: file_lines and expand path
Posted by Jean H. on Fri, 04 Aug 2006 18:57:30 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
> FL wrote:
>
>> Hi,
>>
>> I am implementing file_lines in FL and was wondering what to return
>> when the input path expands to multiple files. The IDL reference guide
>> says nothing. I have run IDL and got curious result:
>>
>> IDL> help, file_lines('*.pro')
>> <Expression> LONG64 = 15
>>

>> I have 448 .pro files in the current directory, with a total of 10k
>> lines.
>> IDL seems to be randomly picking a single file and reporting its
>> length (probably the first match in readdir()).
>>
>> What do you expect to get?
>>
>> Possibilities:
>>
>> 1. never expand (but why do we have NOEXPAND then?)
>> 2. give an error message if the expansion results in more than one file
>> 3. return the total length of files
>> 4. return the total length of files if a new keyword TOTAL was set
>> 5. do as IDL, pick a single file silently
>
>
> Well, if the routine allows for wildcard inputs, then I wouldn't expect
> *any* of the above.
>
> What I would expect would be an array (in this case 448 elements in
> size) containing the lengths of *each* file that matched the wildcard
> expression, e.g.

what about a structure containing the array you said, and another one
containing the complete file name, so then it becomes fairly easy to
open the files (and the order in wich the files are read in the
directory is not a problem). This would allow you to read only files
that have more than X lines in a very convenient way!

Jean

>
> IDL> help, file_lines('*.pro')
> <Expression>   LONG64   = Array[448]
>
> Just because IDL does it wrong, doesn't mean you have to. :o)
>
> paulv
>
> p.s. Maybe FL should have the ability to run just like IDL. I.e. you set
> some sort of environment variable, say, FL_RUNLIKE_IDL=true, and it
> performs like IDL (i.e. giving the incorrect result (IMO) as in the
> above example). Set it to false, and FL behaves like a regular person
> would intuitively expect. :o)
>

Subject: Re: file_lines and expand path
Posted by Paul Van Delst[1] on Fri, 04 Aug 2006 19:35:15 GMT
View Forum Message <> Reply to Message

Jean H. wrote:
> Paul van Delst wrote:
>> FL wrote:
>>
>>> Hi,
>>>
>>> I am implementing file_lines in FL and was wondering what to return
>>> when the input path expands to multiple files. The IDL reference
>>> guide says nothing. I have run IDL and got curious result:
>>>
>>> IDL> help, file_lines('*.pro')
>>> <Expression> LONG64 = 15
>>>
>>> I have 448 .pro files in the current directory, with a total of 10k
>>> lines.
>>> IDL seems to be randomly picking a single file and reporting its
>>> length (probably the first match in readdir()).
>>>
>>> What do you expect to get?
>>>
>>> Possibilities:
>>>
>>> 1. never expand (but why do we have NOEXPAND then?)
>>> 2. give an error message if the expansion results in more than one file
>>> 3. return the total length of files
>>> 4. return the total length of files if a new keyword TOTAL was set
>>> 5. do as IDL, pick a single file silently
>>
>>
>> Well, if the routine allows for wildcard inputs, then I wouldn't
>> expect *any* of the above.
>>
>> What I would expect would be an array (in this case 448 elements in
>> size) containing the lengths of *each* file that matched the wildcard
>> expression, e.g.
>
> what about a structure containing the array you said, and another one
> containing the complete file name, so then it becomes fairly easy to
> open the files (and the order in wich the files are read in the
> directory is not a problem). This would allow you to read only files
> that have more than X lines in a very convenient way!

I think that would be marvelous (I've never used file_lines so maybe it does something like that already?)

However, to keep it more compatible with IDL, maybe rather than a structure it can have an optional output keyword argument, i.e.

```
IDL> fl=file_lines('*.pro',files=filenames)
IDL> help, fl, files
FL              LONG64    = Array[448]
FILENAMES       STRING    = Array[448]
```

?

Hang on a minute.... in the IDL file_lines documentation it states:

Return Value

Returns the number of lines of text contained within the specified file or files. If an array of file names is specified via the Path parameter, the return value is an array with the same number of elements as Path, with each element containing the number of lines in the corresponding file.

Arguments

Path
A scalar string or string array containing the names of the text files for which the number of lines is desired.

So, the IDL version of file_lines *should* return an array when multiple files are given. Does this mean IDL has a bug in file_lines() with regards to how it handles wildcard input? The words associated with the /NOEXPAND_PATH suggest that wildcard inputs are allowed (although it doesn't specifically say it).

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: file_lines and expand path
Posted by James Kuyper on Fri, 04 Aug 2006 21:48:38 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
...

> Hang on a minute.... in the IDL file_lines documentation it states:
>
>
> Return Value
> ------------
> Returns the number of lines of text contained within the specified file or files. If an
> array of file names is specified via the Path parameter, the return value is an array with

Note: "If an array of file names is specified ...".

> the same number of elements as Path, with each element containing the number of lines in
> the corresponding file.
>
> Arguments
> ---------
> Path
> A scalar string or string array containing the names of the text files for which the
> number of lines is desired.
>
>
> So, the IDL version of file_lines *should* return an array when multiple files are given.
> Does this mean IDL has a bug in file_lines() with regards to how it handles wildcard
> input? The words associated with the /NOEXPAND_PATH suggest that wildcard inputs are
> allowed (although it doesn't specifically say it).

I've tested it, and an array of strings works exactly as it should. I suspect that the possibility of a string argument containing wildcards that allow it to match multiple files was not even considered.

Issue: if the FL version returns an array when given a string argument with wildcards that match multiple files, how should it handle an array of string arguments, some or all of which have that same characteristic? Personally, I think it should return an array of line counts with the same length as the array of path names; anything else would make it very complicated for the the calling routine to match up inputs and outputs. I'd recommend using a total when multiple matches to a wildcard occur, because I can actually imagine contexts where that would be useful. I can't see anything useful about returning the length of a randomly member of the matching set of files.

Subject: Re: file_lines and expand path
Posted by Paul Van Delst[1] on Fri, 04 Aug 2006 22:50:15 GMT
View Forum Message <> Reply to Message

kuyper@wizard.net wrote:
> Paul van Delst wrote:
> ...

>> Hang on a minute.... in the IDL file_lines documentation it states:
>>
>>
>> Return Value
>> ------------
>> Returns the number of lines of text contained within the specified file or files. If an
>> array of file names is specified via the Path parameter, the return value is an array with
>
> Note: "If an array of file names is specified ...".
>
>> the same number of elements as Path, with each element containing the number of lines in
>> the corresponding file.
>>
>> Arguments
>> ---------
>> Path
>> A scalar string or string array containing the names of the text files for which the
>> number of lines is desired.
>>
>>
>> So, the IDL version of file_lines *should* return an array when multiple files are given.
>> Does this mean IDL has a bug in file_lines() with regards to how it handles wildcard
>> input? The words associated with the /NOEXPAND_PATH suggest that wildcard inputs are
>> allowed (although it doesn't specifically say it).
>
> I've tested it, and an array of strings works exactly as it should. I
> suspect that the possibility of a string argument containing wildcards
> that allow it to match multiple files was not even considered.

That's what I figured also. I deleted a paragraph of pontification about this before
sending my post off :o) Unfortunately, I revived it somewhat below (sorry).

> Issue: if the FL version returns an array when given a string argument
> with wildcards that match multiple files, how should it handle an array
> of string arguments, some or all of which have that same
> characteristic?

A string array input could imply /NOEXPAND, but I don;t think that's useful.

What if you want to do:
```
  n = file_lines(['*.pro','*.txt','*.inc'])
```
That seems perfectly reasonable to me. I would expect an array containing the lengths of
all the individual files to be returned. I.e. if you had 10 "*.pro", 20 "*.txt" and 5
"*.inc" files, I would expect an array of length 35 returned. If you want a 3-element
array where each element contains the total line count of each wildcard category, rather
than build that capability into the tool (say, via a /TOTAL keyword) you just do:
```
  n = [ total(file_lines('*.pro')),$
        total(file_lines('*.txt')),$
```

total(file_lines('*.inc'))]

What about if you did
  n = file_lines(['*.pro','*.txt','a*.pro'])
where files are repeated in the wildcard expansion? If I had 2 "a*.pro" files, I would expect an array of length 32 returned (where the "a*.pro" files had their lines counted twice). If a user tells the function to read the file twice then, by golly, the function should darn well read the file twice.

Each wildcard can be handled separately from the other. You could get fancy and check for duplicates to avoid reading a file twice, but a first cut could just read every file (including duplicates) in the list once the wildcards have been expanded.

> Personally, I think it should return an array of line
> counts with the same length as the array of path names; anything else
> would make it very complicated for the the calling routine to match up
> inputs and outputs. I'd recommend using a total when multiple matches
> to a wildcard occur, because I can actually imagine contexts where that
> would be useful.

True, but I think that the situation where you want an array of lengths by supplying a wildcard is more likely (in my Pauliverse at least :o)

Supplying a wildcard as input and getting a single number (however it's determined) doesn't make much sense to me at all. If you want the total number of lines then
  maxlines=total(file_lines('*.pro'))
seems more intuitive. As does,
   n_lines_of_code_I_wrote_today=total(file_lines(['*.pro','*.f 90','*.rb']))
If, in *nix, I can do
  wc -l *.pro *.f90 *.rb
and get the output I do, I don't see why file_lines can't do something similar. The current IDL behaviour is horribly low-rent.

Sorry for going on but my (very personal) opinion is that these sorts of things are more indicative of the quality of a product than the flash, whiz-bang features that are touted in the vendor literature (and I'm not being IDL specific here). When I see basic tools constructed shoddily, I no longer expect much from the fancy stuff. I sure trust the product a lot less.

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: file_lines and expand path
Posted by FL on Mon, 07 Aug 2006 10:46:01 GMT
View Forum Message <> Reply to Message

Hi guys,

thank you for the responses.

I will create an array containing the lengths of all the individual files
after expansion, and also add an optional FILES keyword to get back the
real filenames.

thanks,
lajos

Subject: Re: file_lines and expand path
Posted by greg michael on Mon, 07 Aug 2006 14:22:47 GMT
View Forum Message <> Reply to Message

This all seems a little upside-down to me. I'd sooner ask for the
filenames and then get the lengths than ask for lengths and get the
filenames thrown in. But I'm a file_lines newbie. Maybe you guys are
doing something much cleverer...

greg

Subject: Re: file_lines and expand path
Posted by Paul Van Delst[1] on Mon, 07 Aug 2006 18:27:12 GMT
View Forum Message <> Reply to Message

greg michael wrote:
> This all seems a little upside-down to me. I'd sooner ask for the
> filenames and then get the lengths than ask for lengths and get the
> filenames thrown in. But I'm a file_lines newbie. Maybe you guys are
> doing something much cleverer...

I agree with you (about getting the filenames and then the lengths). However, for good or
ill, the IDL version of file_lines "allows"[*] wildcards in its file list input. That
pretty much opens the box I reckon -- and, given that, my opinions were about seems the
most intuitive thing to do (to me at least.)

paulv

[*] One could argue, based on the documentation, that it doesn't allow wildcards. If so,
then wildcard input without the NOEXPAND keyword should throw an error. But that is not
what happens.

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545