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