Subject: Re: quirk End of file encountered before end of program
Posted by lecacheux.alain on Fri, 12 Dec 2014 14:38:53 GMT
View Forum Message <> Reply to Message

On Friday, December 12, 2014 2:38:52 PM UTC+1, LC's No-Spam Newsreading account wrote:
> On Thu, 11 Dec 2014, Jim P wrote:
>
>>> The code should look something like this
>>>
>>> pro xxx.pro,ARGUMENTS
>>>   openw,2,'temporary.temp'
>>>   printf,2,'structure=create_struct($'
>>> CASE STATEMENT FILLING THE STRUCTURE DEFINITION in the temp file
>>> printf,2,')'
>>> close,2
>>> ; and instantiate it
>>>   @ temporary.temp
>>> return
>>> end
>
>> I suspect this isn't behaving as you wish it to behave, due to a
>> misinterpretation of how "@" is used.
>>
>> The "@" in your code is interpreted at compile time, not at run time.
>
> I did not realize that.
>
> The original code I had (dating to several years ago, before anonymous
> structures were introduced) used in fact a much more complex mechanism.
>
> It wrote the structure definition (without use of CREATE_STRUCT which
> did not exist) to the temporary.temp file
>
> then wrote a xxxtmpnnn.pro file with nnn increasing each time, which
> contained the invocation of @temporary.temp (and passed back the
> created structure as argument)
>
> then built a string with the invocation of xxxtmpnnn.pro
> and used execute to invoke it
>
> When I did now some simple tests, I thought that the above arrangement
> was become unnecessary now with create_struct and anonymous structures.
> I wanted to simplify it and hoped it were NOW possible.
>
>
> Apparently I cannot do execute('@temporary.temp')
>
> Is there any limitation to the string length in execute(string) ?

>
> I used a temporary file because the structure definition is rather long,
> and is constructed in steps (driven from a data file) ... for legibility
> temporary.temp has the statement on several lines (terminated by the $
> continuation marker but last one)
>
> If there is no limitation I could concatenate the pieces of the
> statement instead of writing them to temporary.temp and just execute the
> resulting string.

if you do not want to initialize your structure directly in the main procedure (by x = {...}), you could create an external function 'fun.pro' like:

```
 function fun
   x = {...}
   return, x
 end
```

and call it, from the main procedure by x = call_function('fun').

alx.
