Subject: Re: using an unlnown number of keywords
Posted by John-David T. Smith on Mon, 22 Oct 2001 20:32:14 GMT
View Forum Message <> Reply to Message

```
Sean Raffuse wrote:
>
> Liam E. Gumley <Liam.Gumley@ssec.wisc.edu> wrote in message
> news:3BD469CE.7B6483C6@ssec.wisc.edu...
>> Sean Raffuse wrote:
>>>
>>> David Fanning <david@dfanning.com> wrote in message
>>> news:MPG.163a5b3f51d8f2ea989719@news.frii.com...
>>>> Sean Raffuse (sean@me.wustl.edu) writes:
>>>>
>>>> > I'm trying to write a more or less universal procedure that takes a
> data
>>>> > structure and prints the variables out one by one one a line in an
> ascii
>>>> > file.  The problem is that the structure could have any number of
>>> variables.
>>>> > I want to do this:
>>>> >
>>>> > printf, lun, structure.var1, structure.var2, structure.var3, . . .,
>>>> > structure.varN
>>>> >
>>>> > If I know N, is there a way to do this without resorting to N cases?
>>>>
>>>> How about something like this:
>>>>
>>>>   numTags = N_Tags(structure)
>>>>   PrintF, lun, structure, Format='(' + StrTrim(numTags,2) + 'F10.2)'
>>>>
>>> This would work fine if I didn't have different data types.  Strings (of
>>> various lengths) may be mixed with floats and integers.  Any other ideas
> out
>>> there in IDL expert land??
>>
>> Why not use default formatting? For example:
>>
>> IDL> a = {var1:indgen(30), var2:['a', 'b', 'c'], var3:findgen(9)}
>> IDL> print, a
>> {      0       1       2       3       4       5       6       7
>>        8       9      10      11      12      13      14      15
>>       16      17      18      19      20      21      22      23
>>       24      25      26      27      28      29
>> a b c
>>       0.00000      1.00000      2.00000      3.00000      4.00000
>>       5.00000      6.00000      7.00000      8.00000
>>  }
>>
>>  Each variable starts on a new line in the printed output.
>
>  The problem is that I am trying to format the data into columns:
>
>  var1    var2    var3    . . .  varN
>  var1    var2    var3    . . .  varN
>  var1    var2    var3    . . .  varN
>  var1    var2    var3    . . .  varN
>
```

Hmm:

```
s=string(FORMAT='(A)',a)
print,s,FORMAT='('+strtrim(n_elements(s),2)+'(A,2X))'
```

but of course you won't be able to recover the existing structure this way. I.e. you won't know "var2" is a three element vector.

JD