Subject: Re: Attempt to subscript FILNAM with I is out of range....
Posted by JD Smith on Fri, 16 Aug 2002 19:43:02 GMT
View Forum Message <> Reply to Message

On Fri, 16 Aug 2002 01:05:11 -0700, Timm Weitkamp wrote:

```
>  Kolbjorn,
>
>  Following Craig's suggestion is definitely the easiest (and cleanest)
>  way, but I think his script "run_myfile.pro" should be extended by one
>  line that will make all the difference:
>
>    .run myfile.pro   ; compile the routine(s) contained in myfile.pro
>    myfile            ; run procedure myfile (if that's what is in
>    myfile.pro) exit              ; quit IDL
>
>  Don't let yourself confuse by the fact that file naming conventions in
>  IDL are such that the ".pro" extension is used for both batch scripts
>  and files containing procedures and/or functions, or by the fact that
>  the .RUN statement only *compiles* the routines in its argument file,
>  but does *not run* them. This is why the additional 2nd line in the
>  batch file above is needed, which must have seemed too obvious to Craig
>  so he didn't mention it.
```

Or he meant for myfile.pro to contain a MAIN-level routine, i.e. one like:

```
do_some_stuff
for i=0,1000 do begin
 do_something_else
endfor
end
```

This doesn't have the batch file no-multi-line-block restriction, and only differs from a real procedure in that no routine definition is included (but notice the mandatory closing END statement), and, obviously, no arguments are allowed. When you ".run" a main level routine, it compiles and runs at the same time. In fact it is re-compiled each time -- one advantage to MAIN-level routines is you don't have to compile then run for each change:

```
IDL> .run myplot

<change some plot parameters>

IDL> .run myplot ; new plot
```

They're good for one-off scripts like plotting sequences.

The only good uses of batch files I can think of:

1. Including small definitions, especially common blocks:

```
pro my_pro
  @my_common_block
end
```

in my_common_block:

```
common my_common_block,blarh,foo,test
```

Why is this a nice way to include common blocks? If it's used throughout many different files, redifining it becomes an awful chore. But if it's just included from a single file, all changes (e.g. adding a variable) occur in one place. And no, using the abbreviated form:

```
common my_common_block
```

doesn't always work. For this to succeed you need one point of entrance where the common block can be defined... not always practical.

2. Because they're the default for scripts mentioned on the command line (though SAVE files can also be used, as was demonstrated). Not really an advantage, just a necessity.

JD