Subject: calling IDL from outside (Python or shell)
Posted by LC's No-Spam Newsread on Wed, 04 Dec 2002 18:04:25 GMT
View Forum Message <> Reply to Message

We have the need to have a Python application of our own call a quite complex IDL procedure supplied by a colleague elsewhere. At the moment he supplies its procedure as a couple of SAV files to be RESTOREd (no sources). One file contains all his procedures, and another all his variables and data (a lot).

To use his procedure essentially one does (in interactive IDL) :

```
1  @initialization_file       ; does the restore's and initializes
                              ; some variables to default
2  xxx = yyy                  ; set some custom variables
3  pass1,zzzz,tttt            ; call a procedure to load data into
                              ; a structure
4  pass2,uuuu,vvvv            ; call the main procedure
```

Our idea is that the python environment prepares a data file, and then invokes commands 3 and 4 into an existing IDL session (in which commands 1 and 2 have already been executed)

We have set up this using a named pipe as communication channel. In IDL we start with a "goinbatch,pipename", where goinbatch.pro is a silly procedure which reads a string from the pipe and does an "execute(inputstring)".

The communication mechanism works nicely on generic commands. We tested it using an echo command from an Unix shell redirected into the pipe, and verified that IDL takes commands and executes them. Of course we cannot send to the pipe command 1 as "@file", but we shall send each individual command.

The communication mechanism works also from the python end.

In both cases however we get an error on the 4th command (when we run from a pipe, be it from a shell or from python) which we do not get when we run interactively (pass2 cannot open a file ... we have no control or knowledge of the source, so we think that some variable is altered when one runs from the pipe).

Note that we cannot simply write all command groups 1 to 4 into a file and call it as "idl filename" or "idl << filename". The reason is that command 1 restore's are very bulky, and constitute a large (time) overhead. We want to execute 1 & 2 only once, and 3 & 4 each time we have to process a dataset.

Is there anything intrinsically wrong in having IDL running a
goinbatch.pro which is essentially

```
inputloop=1
while inputloop eq 1 do begin
 readf,1,inputstring
 print,"batch> ",inputstring
 if (inputstring ne 'ZZ') then begin
   inputstatus=execute(inputstring)
 endif else begin
   inputloop=0
 endelse
endwhile
```

The inputstring can be anything, namely a RESTORE statement, an
assignment xx=yy or the invocation of a procedure name,arg1,arg2

We looked into using call_procedure, but that does not apply because it
cannot handle plain assignment statements, it requires parsing of the
string into procedure and arguments, and also it won't work. I thought
we could put command groups 1 and 2 in an init.pro, commands 3 and 4 in
a do_your_job.pro (and write it each time) and then just pass on the
pipe : init (once) and do_your_job (all other times) ... but in such
case one has the following chain :

```
 shell
   IDL
     procedure goinbatch (modified)
         procedure init (restores all variables then exits and they
                         are lost)
         procedure do_your_job
```

While in the current (execute) case one should have

```
 shell
   IDL
     procedure goinbatch
         any command executed at this level
```

But apparently it is not so.

We are trying to get the code of the IDL procedure from our colleague,
but in the meanwhile we would like to know if there is anything
intrinsically wrong in the execute approach, or any pitfall to be aware

--
 ---------- ----------
nospam@ifctr.mi.cnr.it is a newsreading account used by more persons to

avoid unwanted spam. Any mail returning to this address will be rejected.
Users can disclose their e-mail address in the article if they wish so.