Subject: calling a user-named procedure
Posted by haskell on Mon, 13 Jan 1997 08:00:00 GMT
View Forum Message <> Reply to Message

hi,
i would like to be able to call a procedure whose name is given by the user during execution.
example:

```
pro stuff,filename
...
call/execute, filename       ;this is the line i cannot figure out
...
```

where filename is a string holding the name of a procedure (.pro file) input by the user e.g. from

```
pro junk
...
print,'input filename'
read,filename
stuff,filename  ;call procedure stuff with filename as the argument
...
```

i have tried different syntaxes of execute and call_procedure but to no avail.
either i am just not phrasing things correctly, or i am on the wrong track.
does anyone know if this can be done, and if so, how?

thanks,
eddie
--- ----------
When you are a Bear of Very Little Brain, and you Think of Things, you
find sometimes that a Thing which seemed very Thingish inside you is
quite different when it gets out into the open and has other people
looking at it.

- A.A. Milne, "The House At Pooh Corner"

Subject: Re: calling a user-named procedure
Posted by Peter Mason on Fri, 17 Jan 1997 08:00:00 GMT
View Forum Message <> Reply to Message

On 13 Jan 1997, eddie haskell wrote:
> i would like to be able to call a procedure whose name is given by the user
> during execution. example:
>
> pro stuff,filename
> ...
> call/execute, filename   ;this is the line i cannot figure out

```
> ...
>
> where filename is a string holding the name of a procedure (.pro file) input
> by the user  e.g. from
>
> pro junk
> ...
> print,'input filename'
> read,filename
> stuff,filename  ;call procedure stuff with filename as the argument
> ...
>
> i have tried different syntaxes of execute and call_procedure but to no avail.
> either i am just not phrasing things correctly, or i am on the wrong track.
> does anyone know if this can be done, and if so, how?
```

This shouldn't be giving you much trouble, as long as you don't include
a path-spec or a ".pro" (or ".sav") extension in your filename.
e.g.,

```
  t='xpalette' ;which really is $IDL_DIR/lib/xpalette.pro
  call_procedure,t
```

e.g. 2,

```
  t=''  ;initialise T as a string variable
  read,t,prom='Enter the name of a PROCEDURE (no path or extension) :'
  call_procedure,t
```

This is fine as long as your file is somewhere known to IDL, and as long
as you don't have both a .pro and a .sav file of the same routine (or multiple
copies here and there) and want to specify which one of them to call.
(Check out the docs on IDL's path, especially the bit about how the entire
directory tree is expanded under any path entry starting with "+".)

IDL seems to be "half-aware" of paths.   If you include a path spec, your
routine will get compiled/loaded by CALL_PROCEDURE/CALL_FUNCTION or
RESOLVE_ROUTINE, but IDL will crash straight afterwards.
CALL_PROCEDURE, CALL_FUNCTION and RESOLVE_ROUTINE don't appear to cope with
file extensions at all.
EXECUTE() won't give you any advantage here.

I think that these errors occur because - in all of the above routines -
sooner or later IDL tries to use the name you specify as the routine's
actual name, and so any '/', '\' and '.' characters in the name end up being
illegal.
I don't know of any way (apart from in a .RUN or .COMPILE executive
command at the "MAIN" level) that you can do just the compilation /
restoration of a routine specified with a path or extension.

Peter Mason