
Subject: Re: What is the main difference between a script and a procedure?

Posted by [David Fanning](#) on Wed, 03 Oct 2007 13:40:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

mystea writes:

- > As far as I can tell, a script:
- > 1. Can't accept any arguments and can't take any extended loops.
- > 2. It can recognize any variable that exists in the current session
- > because it behaves just like a a list of commands in sequence.
- >
- > On the other hand, a procedure:
- > 1. Can accept arguments, but can't recognize any variables which exist
- > in current IDL session.

What you are calling a "script", most people call a "batch file". This is a way to execute a series of commands "as if" you were typing them at the IDL command line. Since this is just about the most limited way of using IDL, batch files are typically used infrequently.

More often people will put the same commands into a file and add an END statement at the end of the file. This file is now a "main-level program". It must be compiled before it can be executed. Normally the compile and execute is done with the .RUN executive command. The big advantage of main-level programs over batch files, is that you can include extended loops, etc. in a main-level program without all the shenanigans required to get a loop to work on the IDL command line.

As you become more sophisticated in your programming, you will eventually realize that having all your variables in one big pot is probably not such a great idea. (Especially if you tend to name all your variables "a" to avoid a lot of typing.) At that point, you might be interested in writing procedures and functions (just another term for "IDL commands") that do particular things for you, while at the same time, keeping their internal variables from contaminating your main-level working space.

IDL uses a "pass by reference" method of getting variables into and out of commands, so it is easy to write procedures and functions that change main-level variables, if that is your purpose. You do, in fact, have to pass the variables into the procedure or function via arguments or keywords, however, since all the "action" occurs on a level separate from the main level. (There are ways to access main-level variables

from within procedures and functions that don't involve passing the variables, but this is rarely done, and only by experienced programmers who REALLY know what they are doing and why they are doing it.)

- > However, I often run into the situation that I need a code which can
- > recognize variables in current session *as well as* taking arguments.
- > Is it possible to write such a code?

This is called "having your cake and eating it, too". It is as easy to do in IDL as it is in life. :-)

And, anyway, what could you possibly pass to a batch file that the batch file didn't already know about? The only thing you can pass are things that exist at the main IDL level, and the batch file already has access to all of that.

Cheers,

David

--

David Fanning, Ph.D.

Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: What is the main difference between a script and a procedure?

Posted by [JD Smith](#) on Thu, 04 Oct 2007 22:45:07 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Wed, 03 Oct 2007 07:40:49 -0600, David Fanning wrote:

> mystea writes:

>

>> As far as I can tell, a script:

>> 1. Can't accept any arguments and can't take any extended loops.

>> 2. It can recognize any variable that exists in the current session

>> because it behaves just like a list of commands in sequence.

>>

>> On the other hand, a procedure:

>> 1. Can accept arguments, but can't recognize any variables which exist

>> in current IDL session.

>

> What you are calling a "script", most people call a

> "batch file". This is a way to execute a series of

> commands "as if" you were typing them at the IDL

> command line. Since this is just about the most limited
> way of using IDL, batch files are typically used infrequently.

I use batch files as IDL equivalents of #include statements, to include frequently needed code, e.g. common block definitions, ala:

```
pro mypro,a,b,c
  @my_common
  my_common_var=2
end
```

So they aren't entirely useless.

JD

Subject: Re: What is the main difference between a script and a procedure?

Posted by [mystea](#) on Fri, 05 Oct 2007 09:08:41 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thank you very much, David.

I understand that when I know exactly what I want to do, writing a procedure is a better idea. However, in my case, I was trying to use batch files to save some keystrokes with plotting.

I was trying to make contour plots with different 2D arrays. These 2D arrays can all be generated by one single function "make2Darr", which takes the argument "q", where q is a real number from 0 to 1. I was trying to make contour plots with different qs and see what's going to happen.

The following is what I want to do when I try q=0, 0.5 and 0.8:

```
xaxis=5.0+0.2*findgen(200)
yaxis=3.0+0.1*findgen(200)
z00=make2Darr(0)
contour, z0, xaxis, yaxis, levels=[0], overplot=0
```

```
z05=make2Darr(0.5)
contour, z05, xaxis, yaxis, levels=[0], overplot=1
```

```
z08=make2Darr(0.8)
contour, z08, xaxis, yaxis, levels=[0], overplot=1
```

(This is a simplified version of what I was doing. I was having quite a few more contour keywords set)

I don't want to use a procedure because I do want to keep z00, z05 and z08 after I finished plotting. They contain useful information and I might want to shade_surf them or print them.

My own way of doing this task so far is:

```
<setup.pro>
.compile make2Darr.pro
xaxis=5.0+0.2*findgen(200)
yaxis=3.0+0.1*findgen(200)

<qContour.pro>
zq=make2Darr(q)
contour, zq, xaxis, yaxis, levels=[0], overplot=overplot
```

```
IDL>@setup
IDL>q=0
IDL>overplot=0
IDL>@qContour
IDL>z00=zq
IDL>q=0.5
IDL>overplot=1
IDL>@qContour
IDL>z05=zq
...
...
```

I wonder if there a better way to accomplish this task? In the ideal situation, I wish I could keyin something like:

```
"@qContour, 0.5, 1" to accomplish the task done by:
IDL>overplot=1
IDL>@qContour
IDL>z05=zq
```

Sincerely,

Gene

On Oct 3, 6:40 am, David Fanning <n...@dfanning.com> wrote:

```
> mystea writes:
>> As far as I can tell, a script:
>> 1. Can't accept any arguments and can't take any extended loops.
>> 2. It can recognize any variable that exists in the current session
>> because it behaves just like a a list of commands in sequence.
>
>> On the other hand, a procedure:
>> 1. Can accept arguments, but can't recognize any variables which exist
>> in current IDL session.
```

>

> What you are calling a "script", most people call a
> "batch file". This is a way to execute a series of
> commands "as if" you were typing them at the IDL
> command line. Since this is just about the most limited
> way of using IDL, batch files are typically used infrequently.

>

> More often people will put the same commands into
> a file and add an END statement at the end of the file.
> This file is now a "main-level program". It must be
> compiled before it can be executed. Normally the
> compile and execute is done with the .RUN executive
> command. The big advantage of main-level programs over
> batch files, is that you can include extended loops, etc.
> in a main-level program without all the shenanigans
> required to get a loop to work on the IDL command line.

>

> As you become more sophisticated in your programming, you
> will eventually realize that having all your variables
> in one big pot is probably not such a great idea. (Especially
> if you tend to name all your variables "a" to avoid a lot
> of typing.) At that point, you might be interested in writing
> procedures and functions (just another term for "IDL commands")
> that do particular things for you, while at the same time,
> keeping their internal variables from contaminating your
> main-level working space.

>

> IDL uses a "pass by reference" method of getting variables
> into and out of commands, so it is easy to write procedures
> and functions that change main-level variables, if that is
> your purpose. You do, in fact, have to pass the variables into
> the procedure or function via arguments or keywords, however,
> since all the "action" occurs on a level separate from the
> main level. (There are ways to access main-level variables
> from within procedures and functions that don't involve passing
> the variables, but this is rarely done, and only by experienced
> programmers who REALLY know what they are doing and why they are
> doing it.)

>

>> However, I often run into the situation that I need a code which can
>> recognize variables in current session *as well as* taking arguments.
>> Is it possible to write such a code?

>

> This is called "having your cake and eating it, too". It is
> as easy to do in IDL as it is in life. :-)

>

> And, anyway, what could you possibly pass to a batch file
> that the batch file didn't already know about? The only thing

> you can pass are things that exist at the main IDL level, and
> the batch file already has access to all of that.
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:<http://www.dfanning.com/>
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
