Subject: Re: IDLVM and execute
Posted by Russell[1] on Sun, 22 Jan 2012 15:30:32 GMT
View Forum Message <> Reply to Message

On Jan 20, 3:36 pm, TonyLanz <tolan...@gmail.com> wrote:
> Hi all,
> I realize this has been asked on the group before. I'm hoping someone
> might have a "simple" workaround in IDLVM to what would be easily
> accomplished using the execute function in IDL proper. I have a widget
> program that contains a compound widget field that allows the user to
> define an equation for manipulating a set of four variables. For the
> sake of argument let's say var1, var2, var3, and var4. In this
> compound widget field they can define what math they want to do,
> they're restricted to addition, subtraction, multiplication and
> division.
>
> so they could for example enter
>
> var1/var2
> or
> var1-var2*0.2
> or
> var1+var2+var3
>
> etc., you get the idea. Now in IDL I can just get the value from the
> widget and simply do something like
>
> s='result='+cwidget_value(0)
> status=execute(s)
> print,result
>
> of course in IDLVM I'm not allowed to use EXECUTE()
>
> Anyone have any suggestions for implementing this simply? I know in
> the past there was discussion of writing code to break down the string
> and pass the pieces to the right operators or functions (using
> call_function()) to assemble the result.
>
> Tony

Yes, like all things, there are several ways to skin this cat. The simplest way is to write the string to a file, compile that file, then call it as a function (and presumably delete it after you're finished). So if you have the command as a string, such as "result = var1 + var2*0.2" then just write it to a file with the appropriate declarations and closings. So, you'd execute:

;save the data:

```
vars=[var1,var2]

;write the file:
openw,lun,'file.pro',/get_lun
printf,lun,'function file,vars'
printf,lun,'return,vars(0)+vars(1)*0.2'
printf,lun,'end'
close,lun & free_lun,lun

;compile the file
resolve_routine,'file.pro',/is_function

;call the function:
result=call_function('file',var1,var2)

;clean up
file_delete,'file.pro',/allow_non
```

now the rub here is that you need to know how many variables there are to store in that temporary variable. Hopefully, there's some way for you to loop over that.

The alternative way, would be to parse the command using the usual order of operations rules. Then process that parsed things using pre-defined functions for the four arithmetic operations. Such as:

```
t=multiply(v1,v2)
t=divide(v1,v2)
```

etc. WHere they have the obvious "innards". Of course, this can get very tedious, but is very useful. If you go down this path, I recommend using the byte typecasting to find the major operators: t=byte(cmd).

Good Luck
Russell

PS, the reason this is so complicated, is that they don't want you doing this.

Subject: Re: IDLVM and execute
Posted by Michael Galloy on Sun, 22 Jan 2012 19:59:18 GMT
View Forum Message <> Reply to Message

On 1/22/12 8:30 am, Russell wrote:

> On Jan 20, 3:36 pm, TonyLanz<tolan...@gmail.com> wrote:
>> Hi all,
>> I realize this has been asked on the group before. I'm hoping someone
>> might have a "simple" workaround in IDLVM to what would be easily
>> accomplished using the execute function in IDL proper. I have a widget
>> program that contains a compound widget field that allows the user to
>> define an equation for manipulating a set of four variables. For the
>> sake of argument let's say var1, var2, var3, and var4. In this
>> compound widget field they can define what math they want to do,
>> they're restricted to addition, subtraction, multiplication and
>> division.
>>
>> so they could for example enter
>>
>> var1/var2
>> or
>> var1-var2*0.2
>> or
>> var1+var2+var3
>>
>> etc., you get the idea. Now in IDL I can just get the value from the
>> widget and simply do something like
>>
>> s='result='+cwidget_value(0)
>> status=execute(s)
>> print,result
>>
>> of course in IDLVM I'm not allowed to use EXECUTE()
>>
>> Anyone have any suggestions for implementing this simply? I know in
>> the past there was discussion of writing code to break down the string
>> and pass the pieces to the right operators or functions (using
>> call_function()) to assemble the result.
>>
>> Tony
>
> Yes, like all things, there are several ways to skin this cat. The
> simplest way is to write the string to a file, compile that file, then
> call it as a function (and presumably delete it after you're
> finished). So if you have the command as a string, such as "result =
> var1 + var2*0.2" then just write it to a file with the appropriate
> declarations and closings. So, you'd execute:
>
> ;save the data:
> vars=[var1,var2]
>
> ;write the file:
> openw,lun,'file.pro',/get_lun

> printf,lun,'function file,vars'
> printf,lun,'return,vars(0)+vars(1)*0.2'
> printf,lun,'end'
> close,lun& free_lun,lun
>
> ;compile the file
> resolve_routine,'file.pro',/is_function
>
> ;call the function:
> result=call_function('file',var1,var2)
>
> ;clean up
> file_delete,'file.pro',/allow_non
>
>
> now the rub here is that you need to know how many variables there are
> to store in that temporary variable. Hopefully, there's some way for
> you to loop over that.

You can't compile a file from the VM, so I think you are stuck actually
parsing the expression.

> The alternative way, would be to parse the command using the usual
> order of operations rules. Then process that parsed things using pre-
> defined functions for the four arithmetic operations. Such as:
>
> t=multiply(v1,v2)
> t=divide(v1,v2)
>
> etc. WHere they have the obvious "innards". Of course, this can get
> very tedious, but is very useful. If you go down this path, I
> recommend using the byte typecasting to find the major operators:
> t=byte(cmd).
>
>
> Good Luck
> Russell
>
> PS, the reason this is so complicated, is that they don't want you
> doing this.

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL, A Guide to Learning IDL: http://modernidl.idldev.com
Research Mathematician
Tech-X Corporation

Subject: Re: IDLVM and execute
Posted by greg.addr on Mon, 23 Jan 2012 12:57:36 GMT
View Forum Message <> Reply to Message

Yes, I've wanted to do this before. I suppose the reason execute() is not allowed is to prevent you from writing a wrapper of IDL interpreter for the VM. If this is so, perhaps it could be possible for Exelis to provide a reduced version of execute which would simply evaluate an arithmetic expression. For my purposes - and yours, too, it appears - it would be enough if only +-*/^ were allowed (although probably sqrt, alog10, exp, sin... and a bunch of other built-in fns could sensibly be included, too): I don't think you could build anything interpreter-like from those. A user-defined function is a useful thing to have in a scientific data analysis language...

Greg

Subject: Re: IDLVM and execute
Posted by Reimar Bauer on Mon, 23 Jan 2012 16:17:42 GMT
View Forum Message <> Reply to Message

On Jan 23, 1:57 pm, greg.a...@googlemail.com wrote:
> Yes, I've wanted to do this before. I suppose the reason execute() is not allowed is to prevent you from writing a wrapper of IDL interpreter for the VM. If this is so, perhaps it could be possible for Exelis to provide a reduced version of execute which would simply evaluate an arithmetic expression. For my purposes - and yours, too, it appears - it would be enough if only +-*/^ were allowed (although probably sqrt, alog10, exp, sin... and a bunch of other built-in fns could sensibly be included, too): I don't think you could build anything interpreter-like from those. A user-defined function is a useful thing to have in a scientific data analysis language...
>
> Greg

If you read this article you find what you need
http://cow.physics.wisc.edu/~craigm/idl/down/routine_names.t xt

Reimar

Subject: Re: IDLVM and execute
Posted by Michael Galloy on Wed, 25 Jan 2012 00:25:26 GMT
View Forum Message <> Reply to Message

This problem has been a problem for me before so I wrote a simple
arithmetic parser:

docs: http://docs.idldev.com/idllib/analysis/mg_evalexpr.html
source: http://docs.idldev.com/idllib/analysis/mg_evalexpr.pro

Evaluates a mathematical expression using the basic arithmetic operators

+, -, *, /, and ^ along with parentheses for grouping and simple function calls of a single variable.

This routine does not use `EXECUTE`, so it is safe to use in the Virtual Machine.

For example, simple arithmetic expressions can be evaluated::

```
IDL> print, mg_evalexpr('1 + 2 + 3', error=error), error
                   6       0
```

Note that the `ERROR` keyword returns whether there was an error in evaluating the expression. Expressions can also take variables, if their values are provided via a structure or hash-like object::

```
IDL> print, mg_evalexpr('exp(i * pi)', { pi: !dpi, i: complex(0, 1) })
(      -1.0000000,   1.2246468e-16)
```

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL, A Guide to Learning IDL: http://modernidl.idldev.com
Research Mathematician
Tech-X Corporation
