

---

Subject: passing functions as arguments in IDL?  
Posted by [Rick Baer](#) on Tue, 01 Feb 2000 08:00:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Is there any way to pass functions as arguments in IDL? I would like to be able to do something like this:

```
function f1, x
  return, x*x
end
```

```
pro tf1, func, arg
  return, func(arg)
end
```

```
IDL> print, tf1, f1, 4
16
```

Thanks!

Rick Baer  
Hewlett-Packard Laboratories

---

---

Subject: Re: passing functions as arguments in IDL?  
Posted by [Martin Schultz](#) on Tue, 29 Feb 2000 08:00:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Benno Puetz wrote:

```
>
> David Fanning wrote:
>
>> Rick Baer (baer@hpl.hp.com) writes:
>>
>>> Is there any way to pass functions as arguments in IDL? I would
>>> like to be able to do something like this:
>>> ...
>>> IDL> print, tf1, f1, 4
>>
>> You can do this:
>>
>> function f1, x
>>   return, x*x
>> end
>>
>> function tf1, func, arg
>>   IF Size(func, /Type) NE 7 THEN BEGIN
```

```

>>     Print, 'String argument required'
>>     RETURN, -1
>> ENDIF
>>     retVal = Call_Function(func, arg)
>>     return, retVal
>> end
>>
>> IDL> print, tf1('f1', 4)
>>
>> Cheers,
>>
>> David
>> --
>
> Would it also be possible to pass argument lists of variable length,
> depending on the function to be called?
>
> I've tried the _EXTRA keyword but to no avail ...
>

```

Only if you don't exceed the number of arguments the underlying function can take AND (in case you pass fewer than the maximum number of arguments) if the function is written well enough to cope with an incomplete set of arguments.

Please find attached a program named loop.pro which illustrates this. Loop.pro is a wrapper routine that allows to call functions which accept only scalars with arrays as first argument.

Good luck,  
Martin

```

--
[[ Dr. Martin Schultz  Max-Planck-Institut fuer Meteorologie  [[
[[      Bundesstr. 55, 20146 Hamburg      [[
[[      phone: +49 40 41173-308      [[
[[      fax: +49 40 41173-298      [[
[[ martin.schultz@dkrz.de      [[
[[-----
; $Id: loop.pro,v 1.10 1999/01/22 20:12:17 mgs Stab $
;-----
;+
; NAME:

```

```

; LOOP
;
;
; PURPOSE:
;   This routine provides a wrapper for function calls that accept
;   only scalars so that they can operate on arrays.
;
;
; CATEGORY:
;   serious stuff
;
;
; CALLING SEQUENCE:
;   result = LOOP(name,arg,p1,p2,p3,p4)
;
;
; INPUTS:
;   NAME --> the name of the function (string)
;
;   ARG --> the argument (array)
;
;   P1 .. P4 --> optional function parameters
;
;
; KEYWORD PARAMETERS:
;   unfortunately none. Would be nice if _EXTRA would work.
;
;
; OUTPUTS:
;   a result *vector* with the same number of elements as arg.
;
;
; SUBROUTINES:
;
;
; REQUIREMENTS:
;
;
; NOTES:
;
;
; EXAMPLE:
;   a=[0.05,0.01,0.001]
;   print,loop("chisqr_cvf",a,17)
;
;
;   IDL prints:
;   27.5871   33.4087   40.7903
;
;
; MODIFICATION HISTORY:
;   mgs, 05 Dec 1997: VERSION 1.00
;
;
;
; Copyright (C) 1997, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,

```

```
; please contact the author to arrange payment.  
; Bugs and comments should be directed to mgs@io.harvard.edu  
; with subject "IDL routine loop"  
;-----
```

```
function loop,name,arg,p1,p2,p3,p4
```

```
    on_error,2 ; return to caller
```

```
    result = fltarr(n_elements(arg))
```

```
; print,n_elements(p1),n_elements(p2),n_elements(p3),n_elements(p4)  
; call function with number of parameters supplied  
; not very elegant but safe.  
; (would probably look nicer with EXECUTE, but isn't that slower ?)
```

```
if (n_elements(p4) gt 0) then begin  
    for i=0,n_elements(arg)-1 do $  
        result(i) = call_function(name,arg(i),p1,p2,p3,p4)  
    return,result  
endif
```

```
if (n_elements(p3) gt 0) then begin  
    for i=0,n_elements(arg)-1 do $  
        result(i) = call_function(name,arg(i),p1,p2,p3)  
    return,result  
endif
```

```
if (n_elements(p2) gt 0) then begin  
    for i=0,n_elements(arg)-1 do $  
        result(i) = call_function(name,arg(i),p1,p2)  
    return,result  
endif
```

```
if (n_elements(p1) gt 0) then begin  
    for i=0,n_elements(arg)-1 do $  
        result(i) = call_function(name,arg(i),p1)  
    return,result  
endif
```

```
for i=0,n_elements(arg)-1 do $  
    result(i) = call_function(name,arg(i))  
return,result
```

end

## File Attachments

---

1) [loop.pro](#), downloaded 123 times

---

---

Subject: Re: passing functions as arguments in IDL?  
Posted by [davidf](#) on Tue, 29 Feb 2000 08:00:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Benno Puetz ([puetz@mpipsykl.mpg.de](mailto:puetz@mpipsykl.mpg.de)) writes:

```
> David Fanning wrote:
>
>> Rick Baer (baer@hpl.hp.com) writes:
>>
>>> Is there any way to pass functions as arguments in IDL? I would
>>> like to be able to do something like this:
>>> ...
>>> IDL> print, tf1, f1, 4
>>
>> You can do this:
>>
>> function f1, x
>>   return, x*x
>> end
>>
>> function tf1, func, arg
>>   IF Size(func, /Type) NE 7 THEN BEGIN
>>     Print, 'String argument required'
>>     RETURN, -1
>>   ENDIF
>>   retVal = Call_Function(func, arg)
>>   return, retVal
>> end
>>
>> IDL> print, tf1('f1', 4)
>>
>> Cheers,
>>
>> David
>> --
>
> Would it also be possible to pass argument lists of variable length,
> depending on the function to be called?
>
> I've tried the _EXTRA keyword but to no avail ...
```

If those arguments are \*keywords\* I should imagine \_Extra will work: :-)

```
function f1, x, Multi=multi
  if n_elements(multi) eq 0 then multi=2
  return, x*multi
end

function tf1, func, arg, _Extra=extra
IF Size(func, /Type) NE 7 THEN BEGIN
  Print, 'String argument required'
  RETURN, -1
ENDIF
retVal = Call_Function(func, arg, _Extra=extra)
return, retVal
end
```

```
IDL> print, tf1('f1', 4, Multi=3)
12
IDL> print, tf1('f1', 4, Multi=6)
24
```

Cheers,

David

--

David Fanning, Ph.D.  
Fanning Software Consulting  
Phone: 970-221-0438 E-Mail: davidf@dfanning.com  
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>  
Toll-Free IDL Book Orders: 1-888-461-0155

---

Subject: Re: passing functions as arguments in IDL?  
Posted by [Benno Puetz](#) on Tue, 29 Feb 2000 08:00:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

David Fanning wrote:

```
> Rick Baer (baer@hpl.hp.com) writes:
>
>> Is there any way to pass functions as arguments in IDL? I would
>> like to be able to do something like this:
>> ...
>> IDL> print, tf1, f1, 4
>
> You can do this:
```

```
>
> function f1, x
>   return, x*x
> end
>
> function tf1, func, arg
>   IF Size(func, /Type) NE 7 THEN BEGIN
>     Print, 'String argument required'
>     RETURN, -1
>   ENDIF
>   retVal = Call_Function(func, arg)
>   return, retVal
> end
>
> IDL> print, tf1('f1', 4)
>
> Cheers,
>
> David
> --
```

Would it also be possible to pass argument lists of variable length, depending on the function to be called?

I've tried the `_EXTRA` keyword but to no avail ...

```
--
Benno Puetz
Kernspintomographie
Max-Planck-Institut f. Psychiatrie      Tel.: +49-89-30622-413
Kraepelinstr. 10                      Fax : +49-89-30622-520
80804 Muenchen, Germany
```

---