Subject: Global variable vs common block
Posted by marc schellens[1] on Thu, 16 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

hello everybody,

I have a very large program which uses common blocks (sorry David),
nowadays I don't like them any more (never liked them but lately I
discovered the DEFSYSV command :-). I used for the whole program just
one common block holding one structure (all the global data).
It was easy to change the suff to use a ('!named') global variable, and
at a first look everything seem to work fine.
So my question is: Is there something I have not thought about/do not
know were I can run into a limitation with this?
Wanna know this before I spent more time on the prog and eventually have
to change it back.

thanks,
:-) marc

Subject: Re: Global Variable
Posted by Martin Downing on Mon, 01 Oct 2001 09:40:36 GMT
View Forum Message <> Reply to Message

First comment is to only use global variables sparingly, sometimes they are
appropriate but try not to rely on them to make programming simpler.
I know of three ways, the SYSTEM variables are fully global, and may be user
defined, COMMON BLOCK variables are shared between those routines which
include a
declaration for the common block, these are very suitable for those programs
where you cant find a way of writing your code without creating global
variables, but do not want the variables to be seen by any old routine.

system variable eg:

IDL>filepaths = {tag:"FILEPATHS", bin:bin)
IDL>defsysv, '!fpaths',filepaths ; a variable for local paths

now you can use this variable from anywhere in your code as:

IDL> print, !fpaths.bin
d:\martin\bin\

For common block variables you need a block identifier then a list of
variables in the block. Note that the order is all that is relevent and that
the first declaration defines the size of the block: eg

```
pro foo
   COMMON   FOO_COMMON_VARIABLES,  a,b,c,d
   a = "first"
   b="second"
   c="third"
   d="fourth"
   print_foo
end

pro print_foo
   ; note later calls do not have to decare all block variables
   COMMON   FOO_COMMON_VARIABLES,  d,c,var3
   print, d
   print, c,
   print, var3
end
```

the result of calling this procedure is demonstrates that there is no association between variable names inthe common block just the variable order.

```
IDL> foo
first
second
third
```

Lastly there is a *really* nice way of passing around all the 'global' variables you need in your widget programs using get/set_uvalue which. This is my favourite tip learnt from David's books so I shall leave him to describe. In essence you call:

```
WIDGET_CONTROL, base, SET_UVALUE=info_structure
WIDGET_CONTROL, event.top, GET_UVALUE=info_structure
```

cheers

Martin

Martin Downing,
Clinical Research Physicist,
Orthopaedic RSA Research Centre,
Woodend Hospital, Aberdeen, AB15 6LS.

news:3BB82DB7.265E2670@ehma.upc.es...

> Anybody know how can I define al global variable?.
>
> Thanx.

>
> --
> +- -+

> | |
> | http://campus.uab.es/~2034008 |
> | |

> | (http://www.upc.es/ehma/gmh) |
> +- -+
>
>
>

Subject: Re: Global Variable
Posted by R.Bauer on Mon, 01 Oct 2001 09:53:21 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
> First comment is to only use global variables sparingly, sometimes they are
> appropriate but try not to rely on them to make programming simpler.
> I know of three ways, the SYSTEM variables are fully global, and may be user
> defined, COMMON BLOCK variables are shared between those routines which
> include a
> declaration for the common block, these are very suitable for those programs
> where you cant find a way of writing your code without creating global
> variables, but do not want the variables to be seen by any old routine.
>
> system variable eg:
> ------------
> IDL>filepaths = {tag:"FILEPATHS", bin:bin)
> IDL>defsysv, '!fpaths',filepaths ; a variable for local paths
>
> now you can use this variable from anywhere in your code as:
>
> IDL> print, !fpaths.bin
> d:\martin\bin\
> ------------
>
> For common block variables you need a block identifier then a list of
> variables in the block. Note that the order is all that is relevent and that
> the first declaration defines the size of the block: eg
>
> pro foo
>   COMMON   FOO_COMMON_VARIABLES, a,b,c,d
>   a = "first"

> b="second"
> c="third"
> d="fourth"
> print_foo
> end
>
> pro print_foo
> ; note later calls do not have to decare all block variables
> COMMON FOO_COMMON_VARIABLES, d,c,var3
> print, d
> print, c,
> print, var3
> end
>
> the result of calling this procedure is demonstrates that there is no
> association between variable names inthe common block just the variable
> order.
> IDL> foo
> first
> second
> third
>
> ----------
> Lastly there is a *really* nice way of passing around all the 'global'
> variables you need in your widget programs using get/set_uvalue which. This
> is my favourite tip learnt from David's books so I shall leave him to
> describe. In essence you call:
> WIDGET_CONTROL, base, SET_UVALUE=info_structure
> WIDGET_CONTROL, event.top, GET_UVALUE=info_structure
>
> cheers
>

The fourth way is to use Pointers

```
value=10
ptr=ptr_new(value)

help, *ptr
```

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-1)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg1/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

http://www.fz-juelich.de/zb/text/publikation/juel3786.html
========== ======

read something about linux / windows
http://www.suse.de/de/news/hotnews/MS.html

Subject: Re: Global Variable
Posted by on Mon, 01 Oct 2001 10:46:05 GMT
View Forum Message <> Reply to Message

Reimar Bauer wrote:

> Martin Downing wrote:
>>
>> First comment is to only use global variables sparingly, sometimes they are
>> appropriate but try not to rely on them to make programming simpler.
>> I know of three ways, the SYSTEM variables are fully global, and may be user
>> defined, COMMON BLOCK variables are shared between those routines which
>> include a
>> declaration for the common block, these are very suitable for those programs
>> where you cant find a way of writing your code without creating global
>> variables, but do not want the variables to be seen by any old routine.
>>
>> system variable eg:
>> ---
>> IDL>filepaths = {tag:"FILEPATHS", bin:bin)
>> IDL>defsysv, '!fpaths',filepaths ; a variable for local paths
>>
>> now you can use this variable from anywhere in your code as:
>>
>> IDL> print, !fpaths.bin
>> d:\martin\bin\
>> ---
>>
>> For common block variables you need a block identifier then a list of
>> variables in the block. Note that the order is all that is relevent and that
>> the first declaration defines the size of the block: eg
>>

```
>>  pro foo
>>     COMMON   FOO_COMMON_VARIABLES,  a,b,c,d
>>     a = "first"
>>     b="second"
>>     c="third"
>>     d="fourth"
>>     print_foo
>>  end
>>
>>  pro print_foo
>>     ; note later calls do not have to decare all block variables
>>     COMMON   FOO_COMMON_VARIABLES,  d,c,var3
>>     print, d
>>     print, c,
>>     print, var3
>>  end
>>
>>  the result of calling this procedure is demonstrates that there is no
>>  association between variable names inthe common block just the variable
>>  order.
>>  IDL> foo
>>  first
>>  second
>>  third
>>
>>  ------------------------------
>>  Lastly there is a *really* nice way of passing around all the 'global'
>>  variables you need in your widget programs using get/set_uvalue which. This
>>  is my favourite tip learnt from David's books so I shall leave him to
>>  describe. In essence you call:
>>  WIDGET_CONTROL, base, SET_UVALUE=info_structure
>>  WIDGET_CONTROL, event.top, GET_UVALUE=info_structure
>>
>>  cheers
>>
>
>  The fourth way is to use Pointers
>
>  value=10
>  ptr=ptr_new(value)
>
>  help, *ptr
>
>  Reimar
>
>  --
>  Reimar Bauer
>
```

> Institut fuer Stratosphaerische Chemie (ICG-1)
> Forschungszentrum Juelich
> email: R.Bauer@fz-juelich.de
> http://www.fz-juelich.de/icg/icg1/
> ========== ======
> a IDL library at ForschungsZentrum Juelich
> http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml
>
> http://www.fz-juelich.de/zb/text/publikation/juel3786.html
> ========== ======
>
> read something about linux / windows
> http://www.suse.de/de/news/hotnews/MS.html

Ok, thanx.
I prefer the de defsysv method.

--
+- -+
| Miguel Angel Cï¿½rdoba cordoba@ehma.upc.es |
| |
| http://campus.uab.es/~2034008 |
| |
| Grup de Modelitzaciï¿½ Hidrometeorolï¿½gica (UPC) |
| (http://www.upc.es/ehma/gmh) |
+- -+

Subject: Re: Global Variable
Posted by Martin Downing on Mon, 01 Oct 2001 11:14:13 GMT
View Forum Message <> Reply to Message

"Reimar Bauer" <r.bauer@fz-juelich.de> wrote in message
news:3BB83D11.5A58FC00@fz-juelich.de...
>
> The fourth way is to use Pointers
>
>
> value=10
> ptr=ptr_new(value)
>
> help, *ptr
>
> Reimar
>

Reimar,

If this is to be a global variable, do you have a cunning method of locating
and identifying this variable off the heap?

Martin

Subject: Re: Global Variable
Posted by R.Bauer on Mon, 01 Oct 2001 12:04:59 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
> "Reimar Bauer" <r.bauer@fz-juelich.de> wrote in message
> news:3BB83D11.5A58FC00@fz-juelich.de...
>>
>> The fourth way is to use Pointers
>>
>>
>> value=10
>> ptr=ptr_new(value)
>>
>> help, *ptr
>>
>> Reimar
>>
>
> Reimar,
>
> If this is to be a global variable, do you have a cunning method of locating
> and identifying this variable off the heap?
>
> Martin

- locating is be done by ptr_valid()

- identifing is a bit more complicated.

At the moment I am only able to identify named structures as pointers.
We are using them as global variable in some of our widgets with
external communication
between some of our widgets.

http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_ html/dbase/download/get_pointer.tar.gz

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-1)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg1/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

http://www.fz-juelich.de/zb/text/publikation/juel3786.html
========== ======

read something about linux / windows
http://www.suse.de/de/news/hotnews/MS.html

Subject: Re: Global Variable
Posted by Pavel A. Romashkin on Mon, 01 Oct 2001 15:33:30 GMT
View Forum Message <> Reply to Message

Martin,
The following URL points to a tiny file that does what you want.
http://spot.colorado.edu/~romashki/idl/single_set.sav
You can see basic description by typing
SINGLE_SET, /HELP
To set a global variable, use
SINGLE_SET, 'The_Name', My_State[, /NO_COPY]
To read it, use
My_State = SINGLE_GET('The_Name'[, /NO_COPY])
To kill it:
SINGLE_KILL, 'The_Name'
There is no limit on the number of named global variables.
If you did not keep track of global pointers using GET_HANDLE then
HEAP_GC will be able to kill them, as they exist as dangling references.
In this case, call to SINGLE_GET will re-create the named global
variable, but, of course, not its contents.
However, they are accessible via SINGLE_GET at any time.
Cheers,
Pavel
Disclaimer: I do not support using global variables. Provided procedure was
only written as an alternative to using COMMON blocks for those rarest
cases when widget programs are absolutely independent but benefit from
being aware of each other.

Martin Downing wrote:

>
> Reimar,
>
> If this is to be a global variable, do you have a cunning method of locating
> and identifying this variable off the heap?
>
> Martin
