Subject: Re: Global variables and command line
Posted by steinhh on Tue, 21 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

J.D.Smith wrote:

[..snip..]

> At the risk of being too arcane here, I think Daniel's point is quite a
> valid one, and his problem less easily solved than it might at first
> glance seem. Sometimes, one wants to access variables from the $MAIN$
> level in a routine, and sometimes one wants to put variables created
> inside a routine into $MAIN$. You might wonder what circumstance could
> exist that would require this functionality. I would simply then direct
> you to RSI's own Insight, in which after getting in you have the option
> of "Select Data to Import"..."IDL Variables"... with a list of $MAIN$
> level variables displayed. Now clearly the folks at RSI have convenient
> access to the names and data locations of the $MAIN$ level variables,
> and I think they should give us mere ordinary users that access also.

[..snip..]

> I am henceforth considering submitting an enhancement request for a
> function that will provide the names and data locations (e.g. through
> pointers) of $MAIN$ level variables inside routines, and the ability to
> create $MAIN$ level variables from within routines, in a safe form.

[..snip..]

> Perhaps RSI is trying to protect us from ourselves here, realizing that
> with the problems people have with variable locality already, breaking
> it slightly might serve to confuse even more. But they really shouldn't
> underestimate our ability to harness and control whatever new power
> comes our way. And besides, if they get to do it, then so should we.

I agree.

In an attempt to nominate myself for the 'IDL Hacker of the year'
award :-) I dug into something that I'd seen while programming RPC
calls to IDL ... and ended up wasting a days work on this thing that
doesn't seem to work, even though I think it *should* work;

Below is a set of C routines that need to be compiled and linked into
shareable libraries (and remember to include the "libidl" shareable
library in the linking... on my alpha that's done by adding e.g.,
"-L$IDL_DIR/bin/bin.alpha -lidl" to the linker statement). The
easiest way to find the (other) correct compile/link flags is
to modify the $IDL_DIR/external/sharelib/Makefile to include your

file.

The getmain() function is added as a System Routine (very neat - at
least I learned *this* from my little exercise!), but it doesn't
work as expected:

```
IDL> print,CALL_EXTERNAL('gmain.so','addmain')  ;; Install routine
           1
IDL> print,getmain('ADF')
% GETMAIN: Couldn't find variable     ; As expected
% Execution halted at:  $MAIN$
IDL> adf=1
IDL> print,getmain('ADF')
Main name ADF, at 140033a58
Main variable type/flag: 2 0
Copying:
Returning: 2 2
       1                        ; So far, so good...
IDL> .r
- pro test
- print,getmain('ADF')
- end
% Compiled module: TEST.
IDL> test
Main name ADF, at 140033c98        ; <- address is wrong, but not NULL!
Main variable type/flag: 0 0
Copying:
Returning: 0 2
% PRINT: Variable is undefined: <UNDEFINED>.
% Execution halted at:  TEST                2 /dev/tty
%                      $MAIN$
```

So, does anyone know what's going wrong here...? Does it behave like
this for all platforms...?

Regards,
Stein Vidar

C source follows (void for non-nerds.. :-)

```
 ------------------------------------------------------------ --gmain.c
#include <stdio.h>
#include "export.h"
#include <strings.h>

/*
   Initialize with:

   print,CALL_EXTERNAL('gmain.so','addmain')
```

```
   then, you *should* be able to get the value of a non-dynamic
   main-level variable with:

   print,getmain('<VARNAME>')

   Alas, this only works from the $MAIN$ scope - which is a bit
   silly....

   */

#define NULL_VPTR ((IDL_VPTR) NULL)

IDL_VPTR getmain(int argc, IDL_VPTR argv[])
{
  IDL_VPTR main_variable;
  IDL_VPTR retv;
  IDL_VPTR main_name;

  retv = IDL_Gettmp();

  main_name = argv[0];

  IDL_ENSURE_STRING(main_name);
  IDL_ENSURE_SCALAR(main_name);

  main_variable = IDL_GetVarAddr(main_name->value.str.s);

  if (main_variable == NULL_VPTR) {
    IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_LONGJMP,"Couldn't find variable");
  }

  printf("Main name %s, at %lx\n\r",main_name->value.str.s,main_variable);
  printf("Main variable type/flag: %d %d\n\r",(int)main_variable->type,
    (int)main_variable->flags);

  if ( ! (main_variable->flags & IDL_V_DYNAMIC) ) {
    fprintf(stderr,"Copying: \n\r");
    bcopy(main_variable,retv,sizeof(*retv));
    retv->flags |= IDL_V_TEMP;
  }

  fprintf(stderr,"Returning: %d %d\n\r",retv->type,retv->flags);

  return retv;
}
```

```
IDL_SYSFUN_DEF main_def[] = { {(IDL_FUN_RET) getmain, "GETMAIN", 1, 1} };

IDL_LONG addmain(int argc,char *argv[])
{
  int tmp;

  /* I haven't the faintest idea whether or not this statement is required,
     or whether it's completely out of line... */

  tmp=IDL_Init(0,&argc,argv);

  /* This one was neat, though: */

  IDL_AddSystemRoutine(main_def,IDL_TRUE,1); /* Just add getmain... */

  return tmp;
}
 ------------------------------------------------------------ ----------
```