Subject: Re: Passing keywords to DLM
Posted by Haje Korth on Mon, 09 Aug 2004 13:20:07 GMT
View Forum Message <> Reply to Message

JG,

I use the lines below to do keyword checking and it works great. I recommend
Ronn Kling's book, it will get you up to speed quickly.

Haje

```
typedef struct {

IDL_KW_RESULT_FIRST_FIELD;

int degree;

} KW_RESULT;


static IDL_KW_PAR kw_pars[] = { IDL_KW_FAST_SCAN,

{"DEGREE",IDL_TYP_LONG,1,IDL_KW_ZERO,0,IDL_KW_OFFSETOF(degree)},

{NULL}

};


KW_RESULT kw;

IDL_KWProcessByOffset(argc,argv,argk,kw_pars,0,1,&kw);
```

"JGG" <jargoogle@explore4life.com> wrote in message
news:4ed7753c.0408081143.38ece3f5@posting.google.com...
> Hello,
>
> I'm adapting Henry Chapman's DLM (itself modified from an earlier
> version) for calling the FFTW v3 fast fourier transform library.
>
> Henry's dlm:
>
http://groups.google.com/groups?as_umsgid=b8f3f671.031213215
8.5cda69f4@posting.google.com

```
>
> I'm working on an Ultra Sparc 5 running Solaris 7 using IDL version
> 5.6 and Solaris compilers.  So far, so good, except that only 2 of 7
> keywords are recognized.  Of the keywords, (X, Y, Z, MEDIUM, HI,
> OVERWRITE, and CLEANUP), only MEDIUM and HI are recognized.
>
> All are treated as read only input boolean keywords.  In the kw_pars[]
> array, they all look something like:
>
>     {"X", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>         IDL_KW_OFFSETOF(x)},
>
> My IDL wrapper syntax looks something like:
>
> dataOut = fftdft3D_IDL(dataIn, direction, [x=x, y=y, z=z,
> medium=medium, $
>                     hi=hi, overwrite=overwrite, cleanup=cleanup])
>
> The following perform as expected:
> dataOut = fftdft3D_IDL(findgen(16,16,16), -1)
> dataOut = fftdft3D_IDL(findgen(16,16,16), -1, /medium)
> dataOut = fftdft3D_IDL(findgen(16,16,16), -1, /hi)
>
> While no other option using the keywords works:
> dataOut = fftdft3D_IDL(findgen(16,16,16), -1, /x)
> % Loaded DLM: FFTDFT3D_IDL.
> % Keyword X not allowed in call to: FFTDFT3D_IDL
> % Execution halted at: $MAIN$
>
> Now, I thought that perhaps there was an ambiguity with single letter
> keywords, but that doesn't explain why overwrite and cleanup are not
> recognized.  They generate the same "not allowed" message.
>
> Yet the keywords that are "not allowed" are all syntactically the same
> as far as the c-code declarations of kw_pars[] and KW_RESULT are
> concerned.
>
> The DLM:
> MODULE fftdft3D_IDL
> DESCRIPTION 3-dimensional Complex fftw (v.3)
> VERSION $Revision: 1.0 $
> BUILD_DATE $Date: Thu Aug  5 09:19:05 MDT 2004$
> SOURCE J. Roberts
> FUNCTION fftdft3D_IDL    1 2 KEYWORDS
>
> Any ideas?
>
> Stumped,
```

> JG.

Subject: Re: Passing keywords to DLM
Posted by jargoogle on Mon, 09 Aug 2004 18:34:37 GMT
View Forum Message <> Reply to Message

Haje,

"Haje Korth" <haje.korth@jhuapl.edu> wrote in message
news:<cf7tm7$k8a$1@aplcore.jhuapl.edu>...

> JG,

>

> I use the lines below to do keyword checking and it works great. I recommend

> Ronn Kling's book, it will get you up to speed quickly.

>

> Haje

Thanks. I'll have to look up Kling's book. I use nearly the same
syntax as you indicated in your email, except that I have 7 keywords
instead of 1. The code compiles without error messages and functions
correctly with no keywords. With keywords, the code accepts only 2 of
the 7 defined and reports that the other 5 keywords are "not allowed"
in the call to the function.

If none of them worked, I'd at least look for a syntactical error.
But since 2 of the 7 are recognized and all are declared exactly the
same way, I'm at a loss.

Here are my declaration statements (keywords medium and hi function
correctly while the others cause "keyword not allowed" errors to be
issued by IDL. See my previous posting.):

```
  typedef struct {
    IDL_KW_RESULT_FIRST_FIELD;
    IDL_LONG x;
    IDL_LONG y;
    IDL_LONG z;
    IDL_LONG medium;
    IDL_LONG hi;
    IDL_LONG overwrite;
```

```
     IDL_LONG cleanup;
  } KW_RESULT;
  static IDL_KW_PAR kw_pars[] = {
     IDL_KW_FAST_SCAN,
     {"X", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
        IDL_KW_OFFSETOF(x)},
     {"Y", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
        IDL_KW_OFFSETOF(y)},
     {"Z", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
        IDL_KW_OFFSETOF(z)},
     {"HI", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
        IDL_KW_OFFSETOF(hi)},
     {"OVERWRITE", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
        IDL_KW_OFFSETOF(overwrite) },
     {"MEDIUM", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
        IDL_KW_OFFSETOF(medium) },
     {"CLEANUP", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
        IDL_KW_OFFSETOF(cleanup) },
     {NULL}
  };
  KW_RESULT kw;

  (void) IDL_KWProcessByOffset(argc, argv, argk, kw_pars,
                              (IDL_VPTR *) 0, 1, &kw);
```

Maybe I shouldn't use IDL_KW_VALUE? I adopted the same syntax here as
in the example given in the 5.6 manual as well as Henry Chapman's
similar DLM.

The DLM is included in my previous post.

Still stumped,
JGG.

Subject: Re: Passing keywords to DLM
Posted by Karl Schultz on Mon, 09 Aug 2004 19:35:12 GMT
View Forum Message <> Reply to Message

"JGG" <jargoogle@explore4life.com> wrote in message
news:4ed7753c.0408091034.5d054741@posting.google.com...
> Haje,
>
> "Haje Korth" <haje.korth@jhuapl.edu> wrote in message
news:<cf7tm7$k8a$1@aplcore.jhuapl.edu>...
>
>> JG,
>

>>
>
>> I use the lines below to do keyword checking and it works great. I
recommend
>
>> Ronn Kling's book, it will get you up to speed quickly.
>
>>
>
>> Haje
>
>
>   Thanks. I'll have to look up Kling's book. I use nearly the same
> syntax as you indicated in your email, except that I have 7 keywords
> instead of 1. The code compiles without error messages and functions
> correctly with no keywords. With keywords, the code accepts only 2 of
> the 7 defined and reports that the other 5 keywords are "not allowed"
> in the call to the function.
>
>   If none of them worked, I'd at least look for a syntactical error.
> But since 2 of the 7 are recognized and all are declared exactly the
> same way, I'm at a loss.
>
> Here are my declaration statements (keywords medium and hi function
> correctly while the others cause "keyword not allowed" errors to be
> issued by IDL. See my previous posting.):
>

```
>    typedef struct {
>       IDL_KW_RESULT_FIRST_FIELD;
>       IDL_LONG x;
>       IDL_LONG y;
>       IDL_LONG z;
>       IDL_LONG medium;
>       IDL_LONG hi;
>       IDL_LONG overwrite;
>       IDL_LONG cleanup;
>    } KW_RESULT;
>    static IDL_KW_PAR kw_pars[] = {
>       IDL_KW_FAST_SCAN,
>       {"X", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>          IDL_KW_OFFSETOF(x)},
>       {"Y", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>          IDL_KW_OFFSETOF(y)},
>       {"Z", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>          IDL_KW_OFFSETOF(z)},
>       {"HI", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>          IDL_KW_OFFSETOF(hi)},
>       {"OVERWRITE", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>          IDL_KW_OFFSETOF(overwrite) },
>       {"MEDIUM", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>          IDL_KW_OFFSETOF(medium) },
>       {"CLEANUP", IDL_TYP_LONG, 1, IDL_KW_ZERO|IDL_KW_VALUE|15, 0,
>          IDL_KW_OFFSETOF(cleanup) },
>       {NULL}
>    };
>    KW_RESULT kw;
>
>    (void) IDL_KWProcessByOffset(argc, argv, argk, kw_pars,
>                              (IDL_VPTR *) 0, 1, &kw);
>
```
> Maybe I shouldn't use IDL_KW_VALUE? I adopted the same syntax here as
> in the example given in the 5.6 manual as well as Henry Chapman's
> similar DLM.
>
> The DLM is included in my previous post.
>
> Still stumped,
> JGG.

Perhaps you just need to sort your list of keywords. The IDL docs say:

The IDL_KW_PAR struct provides the basic specification for keyword
processing.

The IDL_KWProcessByOffset() function is passed a null-terminated array of
these

structures. IDL_KW_PAR structures specify which keywords a routine accepts,
the

attributes required of them, and the kinds of processing that should be done
to them.

IDL_KW_PAR structures must be defined in lexical order according to the
value of

the keyword field.

Karl