Subject: Re: "ALOG2" ?
Posted by davidf on Sat, 03 Apr 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan (steinhh@ulrik.uio.no) writes:

> Nope, I don't have an IDL function, but it's not that
> difficult to write a DLM that interfaces to the C functions.

Now you are getting the idea, Stein Vidar. This is a MUCH
better solution than this wimpy solution by Martin Leuthi:

```
function alog2, x
  return, alog(x)/alog(2.0)
end
```

MUCH more billable and difficult! Are you looking for
a job? I like your style. :-)

Cheers,

David

P.S. In case there is any doubt whatsoever in anyone's
mind, I am just kidding Stein Vidar. As anyone who has
written any kind of program at all knows, it is damn
difficult sometimes to see the simple solutions first.
My first attempt to solve this problem was along the
lines of Amara's, only much, much worse. :-(

Thanks to Martin and Med Bennett for putting us right.
And to Stein Vidar for giving us another of his excellent
examples of well-written DLMs. :-)

--
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: "ALOG2" ?
Posted by steinhh on Sat, 03 Apr 1999 08:00:00 GMT
View Forum Message <> Reply to Message

> Does anyone have an IDL function to compute the logarithm
> base 2 of the elements of an array?

>
> I would like it to work like the the C function frexp() and
> the IEEE floating point function logb().

Nope, I don't have an IDL function, but it's not that
difficult to write a DLM that interfaces to the C functions.

Below is a dlm file and corresponding C file that makes
logb() and frexp() available as builtin IDL functions.
Extension to other, similar functions should be fairly
straightforward. I haven't checked the routines extensively,
but I think they work as expected.

Regards,

Stein Vidar

log2.dlm----------
```
# $Id: log2.dlm,v 1.1 1999/04/03 09:29:38 steinhh Exp steinhh $
MODULE LOG2
DESCRIPTION Base 2 log functions.
VERSION $Revision: 1.1 $
BUILD_DATE $Date: 1999/04/03 09:29:38 $
SOURCE S.V.H.HAUGAN
FUNCTION LOGB 1 1
FUNCTION FREXP 2 2
```
---

log2.c----------
```
#include <unistd.h>
#include <stdio.h>
#include <math.h>
#include "export.h"


#define NULL_VPTR ((IDL_VPTR) NULL)

/* Help function to create temporary variable of given type, with same
   dimensionality as the source */

void *result_var(IDL_VPTR src, int type, IDL_VPTR *res)
{
  /* Is it an array? */

  if (src->flags&IDL_V_ARR) {
    return (void*) IDL_MakeTempArray(type, src->value.arr->n_dim,
        src->value.arr->dim,
        IDL_ARR_INI_NOP,res);
```

```
  }

  /* Nope, scalar: */

  if ( (*res=IDL_Gettmp()) == NULL_VPTR ) {
    IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_LONGJMP,
  "Couldn't create temporary variable");
  }

  (*res)->type = type;
  return & (*res)->value.c;
}


IDL_VPTR LOGB(int argc, IDL_VPTR argv[])
{
  IDL_VPTR src,dst;

  float *f_src,*f_dst;
  double *d_src,*d_dst;

  IDL_MEMINT nn;

  src = argv[0];

  IDL_EXCLUDE_UNDEF(src);
  IDL_ENSURE_SIMPLE(src);
  IDL_EXCLUDE_COMPLEX(src);
  IDL_EXCLUDE_STRING(src);

  if (src->type != IDL_TYP_FLOAT && src->type != IDL_TYP_DOUBLE) {
    src = IDL_CvtDbl(1,argv); /* MIGHT make a temporary variable */
  }

  /* We can do operation "in place" if src is a temp. (but not const!) */
  /* Otherwise we need to make storage space for result                 */

  if (src->flags & IDL_V_TEMP && !(src->flags & IDL_V_CONST)) dst = src;
  else result_var(src,src->type,&dst);

  /* Get pointers to data spaces, and number of elements into nn */

  if (src->flags&IDL_V_ARR) {
    nn = dst->value.arr->n_elts;
    f_dst = (float*) dst->value.arr->data;
    f_src = (float*) src->value.arr->data;
    d_dst = (double*) dst->value.arr->data;
    d_src = (double*) src->value.arr->data;
```

```
  } else {
    nn=1;
    f_dst = &dst->value.f;
    f_src = &src->value.f;
    d_dst = &dst->value.d;
    d_src = &src->value.d;
  }

  if (src->type==IDL_TYP_FLOAT) while (nn--) *f_dst++ = logbf(*f_src++);
  else                          while (nn--) *d_dst++ = logb(*d_src++);

  return dst;
}

#define GETVARDATA(v,n) ((v->flags&IDL_V_ARR) \
    ? (*(n) = v->value.arr->n_elts, v->value.arr->data) \
    : (*(n) = 1, & v->value.c ) )

IDL_VPTR FREXP(int argc, IDL_VPTR argv[])
{
  IDL_VPTR src,dst,ds2,tmp;

  float *f_src,*f_dst;
  double *d_src,*d_dst;
  IDL_LONG *i_ds2;

  IDL_MEMINT nn;

  src = argv[0];

  IDL_EXCLUDE_UNDEF(src);
  IDL_ENSURE_SIMPLE(src);
  IDL_EXCLUDE_COMPLEX(src);
  IDL_EXCLUDE_STRING(src);

  if (src->type != IDL_TYP_FLOAT && src->type != IDL_TYP_DOUBLE) {
    src = IDL_CvtDbl(1,argv); /* MIGHT make a temporary variable */
  }

  /* We can do operation "in place" if src is a temp. (but not const!) */
  /* Otherwise we need to make storage space for result                 */

  if (src->flags & IDL_V_TEMP && !(src->flags & IDL_V_CONST)) dst = src;
  else result_var(src,src->type,&dst);


  /* Now, take care of second argument (output) */
```

```
  ds2 = argv[1];

  /* Avoid expressions - we check this before doing calculations, although */
  /* the IDL_VarCopy does it again later                                   */

  IDL_EXCLUDE_EXPR(ds2);

  /* Generate temporary variable with enough space & correct type */
  i_ds2 = (IDL_LONG*) result_var(src,IDL_TYP_LONG,&tmp);

  switch (src->type) {
  case IDL_TYP_FLOAT:
    f_src = (float*) GETVARDATA(src,&nn);
    f_dst = (float*) GETVARDATA(dst,&nn);
    while (nn--) *f_dst++ = frexpf(*f_src++,i_ds2++);
    break;
  case IDL_TYP_DOUBLE:
    d_src = (double*) GETVARDATA(src,&nn);
    d_dst = (double*) GETVARDATA(dst,&nn);
    while (nn--) *d_dst++ = frexp(*d_src++,i_ds2++);
    break;
  }

  /* Make sure we copy (i.e. move) the temporary variable to the output arg */
  /* This automatically frees any space occupied previously by output arg */
  IDL_VarCopy(tmp,ds2);

  return dst;
}


IDL_SYSFUN_DEF main_def[] =
  {{(IDL_FUN_RET) LOGB,"LOGB",  1, 1},
   {(IDL_FUN_RET) FREXP,"FREXP",  2, 2}
  };


int IDL_Load(void)
{
  return IDL_AddSystemRoutine(main_def,IDL_TRUE,2); /* Just add'em */
}
----------------------------------------
```

Subject: Re: "ALOG2" ?
Posted by Eric Vella on Mon, 05 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

How about simply ALOG2(X) = ALOG(X) / ALOG(2)
or approximately ALOG2(X) = ALOG(X) / 0.69315 ?

ALOG2 is the function you want (log base 2), ALOG is the standard logarithm (log base e) which IDL knows. Dust off an old math book on logarithms to see why this works.

Amara Graps wrote:

> Hi Folks,
>
> Does anyone have an IDL function to compute the logarithm
> base 2 of the elements of an array?
> ...

Subject: Re: "ALOG2" ?
Posted by steinhh on Tue, 06 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
[..about the tricky solution alog(x)/alog(2.0) instead
of writing a DLM...]
> MUCH more billable and difficult! Are you looking for
> a job? I like your style. :-)

Uh - blush :-)

One rule among my friends is "never try to explain when someone else has decided you've done a blunder - you're only gonna make it worse" :-)

det blir bare verre" :-)

But nevertheless, I would like to say in my "defence" that when such a question comes from Amara Graps, I don't expect the solution to be trivial!

And she did ask for something that works like the C functions logb and frexp, which don't actually compute the log in base 2!

logb() returns the integer part of log2 (but as a float/double, to enable signalling of +/-infinity), whilst frexp(a,i) sets i to logb(a)+1 and returns a/2^i...

Not that I expect anyone to go for this line of defence... :-)

But I did learn something valuable about writing DLM's: Forget
the IDL_EzCall() routine for processing parameters.

You're much better off doing whatever is necessary (like
ensuring the data is of the correct type/dimensionality etc)
"manually" through the IDL_ENSURE_xxx/IDL_EXCLUDE_xxx macros,
instead of spending an endless amount of time trying to figure
out exactly what goes on inside IDL_EzCall...

This is much like the CW_PDMENU discussion we had a while ago,
only worse. And do watch out for problems with recursive
routines if you're using the IDL_EzCall routine!!

And I just found out how to create a named variable inside
a DLM: Use the IDL_FindNamedVariable routine.

Well hidden functionality, if you ask me! Most users would
search for something along the lines of "IDL_Make..." or
"IDL_Get...." like I did for quite a while...

Stein Vidar
