Subject: Re: More LIST weirdness?
Posted by David Fanning on Tue, 14 Dec 2010 20:05:32 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> Based on the other thread about list access errors, does the following make sense to anyone:
>
> IDL> q = list()
> IDL> help, q
> Q LIST <ID=1 NELEMENTS=0>
> IDL> help, q[0]
> <Expression> UNDEFINED = !NULL
> IDL> help, q[1]
> <Expression> UNDEFINED = !NULL
> IDL> help, q[2]
> <Expression> UNDEFINED = !NULL
> IDL> x=q[5]
> IDL> help, x
> X UNDEFINED = !NULL
>
> ??

Actually, it does. :-)

Given that lists must be built internally using pointers,
I would say this makes perfect sense to me. I just don't
want to explain to anyone right now. :-)

Cheers,

David

P.S. I'm not saying I'm in favor of inconsistent behavior
or anything like that. I'm just saying that in a weird way,
this does make sense to me. Pointers created with
ALLOCATE_HEAP sort of thing. :-(

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: More LIST weirdness?
Posted by penteado on Tue, 14 Dec 2010 20:06:55 GMT
View Forum Message <> Reply to Message

On Dec 14, 5:55 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
> IDL> q = list()
> IDL> help, q
> Q LIST <ID=1 NELEMENTS=0>
> IDL> help, q[0]
> <Expression> UNDEFINED = !NULL
> IDL> help, q[1]
> <Expression> UNDEFINED = !NULL
> IDL> help, q[2]
> <Expression> UNDEFINED = !NULL
> IDL> x=q[5]
> IDL> help, x
> X UNDEFINED = !NULL
>
> ??
>
> I was expecting the "index out of range" type of error. Or e.g.
>
> IDL> y=q[*]
> % Illegal subscript range.
> % Error occurred at: LIST::_OVERLOADBRACKETSRIGHTSIDE
> % $MAIN$
> % Execution halted at: $MAIN$
>
> Now I'm confused. Removing an item causes the "LIST::REMOVE: Index is out of range" error.
Why wouldn't simply accessing
> an invalid index do a similar thing? The behaviour seems inconsistent. Or am I?

That did surprise me. I was expecting it to throw an error, just as
hash does for an inexistent key. By the way, assigning to an
inexistent list element does throw an error, as it should.

So to me there seems to be a bug in hash::remove() and in
list::_overloadbracketsrightside(), as both should be throwing errors
instead of silently returning !null. The reason is, as I wrote in the
other thread, that a list/hash having !null elements is not the same
as one having no elements (or just not having the element specified by
the index/key) .That is the whole reason for the length argument in
list::init(), and hash::init() assigning !null to elements when only
keys are passed.

Subject: Re: More LIST weirdness?
Posted by penteado on Tue, 14 Dec 2010 20:09:59 GMT

View Forum Message <> Reply to Message

On Dec 14, 6:06 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> So to me there seems to be a bug in hash::remove() and in
> list::_overloadbracketsrightside(), as both should be throwing errors
> instead of silently returning !null. The reason is, as I wrote in the
> other thread, that a list/hash having !null elements is not the same
> as one having no elements (or just not having the element specified by
> the index/key) .That is the whole reason for the length argument in
> list::init(), and hash::init() assigning !null to elements when only
> keys are passed.

In fact, when IDL 8 was still in the Tech Previews, I suggested having
hash return !null for non-existing keys, but others convinced me of
these reasons why it should throw an error.

Subject: Re: More LIST weirdness?
Posted by David Fanning on Tue, 14 Dec 2010 20:13:42 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> Based on the other thread about list access errors, does the following make sense to anyone:

These kinds of errors with HASH and LIST strike me as
the same kinds of errors you make when you write a book.
By the time you get to the end of the project, you don't
really remember what is was you were thinking when you
started the project. Probably you have learned something
and your ideas have changed, anyway. You have to go
back and start all over again to make the damn thing
coherent.

These errors are probably the result of poor editors,
rather than poor programmers. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")
