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.