Subject: Map method bug?
Posted by penteado on Wed, 21 Jan 2015 23:48:24 GMT
View Forum Message <> Reply to Message

Hello,

I had been trying to debug some crazy results I got in a program, and I tracked it down to what appears to be a bug in the map method. It seems that if I pass to map a lambda function that takes a structure as a second argument, it either returns a wrong value, o IDL segfaults. Here is one example:

```
IDL> compile_opt idl2
IDL> s=replicate({a:0B},3) & s.a=bindgen(3)
IDL> lam1=lambda(x,y:typename(y))
IDL> l=[1,0,1]
IDL> foreach ll,l,il do print,il,ll,' ',lam1(ll,s)
           0       1 STRUCT
           1       0 STRUCT
           2       1 STRUCT
```

So far, so good. This was just to show that this simple lambda function works as expected, taking elements from l as input. But then:

```
IDL> l.map(lam1,s)
Segmentation fault (core dumped)
```

On some other cases, it did not segfault, it only gave me a wrong answer. Continuing with the structure s defined above:

```
IDL> l2=list([1,1,0],[0,1])
IDL> l2.map(lambda(x,y:x*y),-1)
[
   [-1, -1, 0],
   [0, -1]
]
IDL> lam2=lambda(x,y:(y[x]).a)
IDL> foreach ll,l2,il do print,lam2(ll,s)
  1   1   0
  0   1
```

All Ok so far. However, mapping lam2 into l2 gives nonsense:

```
IDL> l2.map(lam2,s)
[
   [0, 0, 0],
   [1, 1]
]
```

So far, I have only noticed this problem with 2-argument lambda functions, with the second argument being a structure. The second example looks like a bug to me - the first one must be a bug, since nothing done with just IDL code should ever cause a segfault.

Subject: Re: Map method bug?
Posted by chris_torrence@NOSPAM on Thu, 22 Jan 2015 01:54:08 GMT
View Forum Message <> Reply to Message

Hi Paulo,

I can confirm that this segfaults on IDL 8.4. I'll take a look and see if I can come up with a fix.

Thanks for reporting it!

-Chris
