Subject: Re: Most Common IDL Programming Errors
Posted by Vince Hradil on Wed, 09 Apr 2008 16:39:55 GMT
View Forum Message <> Reply to Message

On Apr 9, 10:26 am, ri...@crd.ge.com wrote:
>> What would be on your list?
>
> These are my top gotchas:
>
> 1) IDL silently truncates array operations to the shorter array.
>
> help, [1,2,3,4] - [1,2]
> [0, 0]
>
> (I wish there were a compile_opt switch to prohibit this!0
>
> 2) Where() returns a one-element array when there is only one match.
>
> help, where([1,2,3,4] EQ 4)
> <Expression> LONG = ARRAY[1]
>
> The combination (1) and (2) is deadly:
>
> array = [1, 2, 3, 4]
> j = where(array EQ max(array))
> array -= array[j]
> print, array
> -3
>
> My solution is to use a wrapper around where() which will return a
> scalar.
>
> 3) Operations which mix signed and unsigned integers can't be trusted.
>
> maxuint = 'ffff'xu
> help, maxuint
> MAXUINT UINT = 65535
> print, fix(1) GT maxuint
> 1
> print, long(1) GT maxuint
> 0
>
> These results can be understood using IDL's promotion rules for mixed
> expressions,
> but it's just not worth it. My solution: Never use unsigned integers.
>
> (Why would one use unsigned integers, you might ask? I've done a few
> bit-level

> simulations of digital hardware using IDL. It would be conceptually
> simpler to
> use unsigned integer variables to represent unsigned quantities or
> signed quantities
> where the sign bit is not the left-most bit.)

I just got stung with this one:
Changing a passed variable in a function, then assuming it has
changed:

```
function foo, bar
  bar = bar/2.0
  foobar = 7.0 * bar
  return, foobar
end

pro testit
  bar = 8.0
  fb = foo(bar)
  print, fb, bar
end
```
