Subject: Re: majority voting
Posted by Allan Whiteford on Mon, 16 Feb 2009 12:11:05 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Allan Whiteford writes:
>
>> I remember the awful day when the result of "help,({a:[1]}).a" changed
>> and all my widget based code collapsed in a heap - I live in fear of
>> that day coming again so share your worries. It's possible that someone
>> will look at ++ and think they can split it across multiple cores - that
>> will be a bad day.
>
> I'm still trying to wrap my head around this whole
> issue, but I confess I don't understand the first thing
> about this paragraph. Do you think you could elaborate
> a bit, Allan?
>
> Thanks,
>
> David
>

David,

Previously (I dunno, sometime before IDL 5.something) if you extracted a one element array from a structure then it would return as a scalar rather than a array. This behaviour was clearly a bug but I had stacks of code which relied upon it. Namely stuff like:

widget_control,textwidget,get_value=myval ; myval 1-element strarr
myval = float(myval) ; myval 1-element fltarr
results = {myval:myval}

inside an event handler then much later...

myval=results.myval ; myval now magically a scalar

answer=myval+[1.,2.,3.]

"answer" would end up being a 3-element fltarr because myval had turned into a scalar. With the improvement answer now ends up as a 1 element fltarr. Plotting this isn't so useful!

I never actually knew that get_value from a text widget returned a 1-element array in the case of only one line of text because I never looked at the value explicitly and all subsequent testing meant that I never noticed the problem in my code.

"help,({a:[1]}).a" demonstrates the problem since it would return either:

<Expression>   INT   = Array[1]

or

<Expression>   INT   =   1

depending on your version of IDL.

I have a concern that someone at ITTVIS will one day decide that the behaviour that I was relying on in the ++ operator can be changed because they want to make is consistent with += or because they decide that if they don't treat repeated indices as they are just now they can parallelise it across multiple CPU cores.

I don't like it when an IDL expression can change in result just due to a version change of IDL.

Thanks,

Allan
