Subject: Re: Array of structures.
Posted by Ben Tupper on Tue, 16 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

"J.D. Smith" wrote:

> Nicolas Decoster wrote:
>>
>> Hi.
>>
>> I'd like to build an array of structure using one line. Something like
>> that:
>>
>> arr = [{sig:a, color:'red'}, {sig:b, color:'green'}]
>>
>> I don't want to use the replicate function and then fill the array one
>> by one:
>>
>> arr = replicate({myStruct, sig:a, color:'red'}, 2)
>> arr[0] = {myStruct, sig:a, color:'red'}
>> arr[1] = {myStruct, sig:b, color:'green'}
>>
>> Any suggestions ?
>>
>> Thanks.
>>
>> Nicolas.
>>
>> ps: In fact, I'm trying to find a way to handle lists of anything. In
>> the present case I want to pass as one argument a list of signals
>> (arrays of value) associated with a color. The list must be of any size,
>> we must be able to build it on the fly. Something like that : {{sig1
>> 'red'} {sig2 'green'}} in a Tcl-like syntax.
>
> Pointers are your salvation, but the initial request can be solved without them:
>
> arr=[{myStruct,sig:a,color:'red'},{myStruct,b,'green'}]
>
> The key being using a named array, fully filled in on the first element. If you
> don't want to use a named array, you're stuck with replicate.
>
> As for you p.s., I'd use lots o' pointers... an example in an object oriented
> design:
>
> pro myClass__define
> struct={COLSIG, $
> Color:'', $
> Signals:ptr_new()} ; pointer to a list of signal values

>
> struct={myClass, $
> colsigs:ptr_new(),$;pointer to list of structs of type COLSIG
> blah:0L, $
> ...}
> end
>
> So you have the following flexibility:
>
> 1. each colsig record can have any number of signals associated with a given
> color.
> 2. There can be as many such colsig color-signal pairings (structs) as you like.
>
> Simply make sure to use the appropriate "if ptr_valid(self.colsigs)" and "if
> ptr_valid((*self.colsigs)[1].Signals) to test for the validity of a given record
> before using it.... though see some prior postings about the advantages of
> splitting up such complicated dereferencing/indexing statements.
>
> You can easily find records for a given color like:
>
> wh=where((*self.colsigs).Color eq 'Green',cnt)
>
> just remember that structure field extraction and array indexing have higher
> precedence than pointer derefencing. This is confusing since the manual lists
> pointer derefence as second in Operator Precedence, just below parentheses. But
> "." and "[]" are not included on that list! So just remember: "*" is weak,
> comparatively speaking, which is why you can get away with things like:
>
> a={Foo,b:ptrarr(3)}
> c=*a.b[0]
>
> both the "." and the "[0]" are evaluated *before* the "*". This is why we had
> to use parentheses above: we wanted to take a subscript or a structure field of
> the thing *pointed to* by self.colsigs, so we had to empower our "*" operator
> with parentheses.
>

Hello,

I just wanted to thank you for always giving such detailed answers.

They are keepers!

Ben

--
Ben Tupper

Bigelow Laboratory for Ocean Science
tupper@seadas.bigelow.org

pemaquidriver@tidewater.net