

---

Subject: Re: Oddball Event Handling (Longer than it Ought to Be)  
Posted by [Craig Markwardt](#) on Sun, 31 Dec 2000 15:45:33 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Hi David--

Thanks for the cool description of you project.

Now, prepare to be lightly toasted :-)

```
> FUNCTION FindTLB, startID
>
> ; This function traces up the widget hierarchy to find the top-level base.
>
> FORWARD_FUNCTION FindTLB
> parent = Widget_Info(startID, /Parent)
> IF parent EQ 0 THEN RETURN, startID ELSE parent = FindTLB(parent)
> RETURN, parent
> END
```

I have no problem with recursion. In this case however it's not really needed. For the book I'm reading, this is known as tail recursion I believe, which is often easily optimized. I admit recursion may help you conceptualize what's going on though. Wouldn't the following code do the same thing?

```
parent = startid
while widget_info(parent, /parent) NE 0 do $
  parent = widget_info(parent, /parent)
```

Craig

--

-----  
Craig B. Markwardt, Ph.D.      EMAIL: [craigmnet@cow.physics.wisc.edu](mailto:craigmnet@cow.physics.wisc.edu)  
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response  
-----

---

Subject: Re: Oddball Event Handling (Longer than it Ought to Be)  
Posted by [davidf](#) on Sun, 31 Dec 2000 19:01:38 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Craig Markwardt ([craigmnet@cow.physics.wisc.edu](mailto:craigmnet@cow.physics.wisc.edu)) writes:

```
> Thanks for the cool description of you project.
>
```

```
> Now, prepare to be lightly toasted :-)
>
>> FUNCTION FindTLB, startID
>>
>> ; This function traces up the widget hierarchy to find the top-level base.
>>
>> FORWARD_FUNCTION FindTLB
>> parent = Widget_Info(startID, /Parent)
>> IF parent EQ 0 THEN RETURN, startID ELSE parent = FindTLB(parent)
>> RETURN, parent
>> END
>
> I have no problem with recursion. In this case however it's not
> really needed. For the book I'm types, this is known as tail
> recursion I believe, which is often easily optimized. I admit
> recursion may help you conceptualize what's going on though. Wouldn't
> the following code do the same thing?
>
> parent = startid
> while widget_info(parent, /parent) NE 0 do $
>   parent = widget_info(parent, /parent)
```

Oh, sure, it would \*work\*. But how you gonna give something like that away? :-)

Apparently I didn't make it clear that I wasn't looking for criticism of my exciting new program, but I have to admit I fooled around for a few minutes trying to get a WHILE loop to work. But after becoming confused I just wandered around in the wilderness for a while, making a change here, and another change there for no apparent rational reason (you know, how you do when you are improvising) and all of a sudden, BLAM, something worked.

In my personal programming myth, if something works it is clearly the most highly optimised solution. But thanks for your suggestion. :-)

Cheers,

David

--

David Fanning, Ph.D.  
Fanning Software Consulting  
Phone: 970-221-0438 E-Mail: davidf@dfanning.com  
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

---

Subject: Re: Oddball Event Handling (Longer than it Ought to Be)  
Posted by [John-David T. Smith](#) on Tue, 02 Jan 2001 17:22:44 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Craig Markwardt wrote:

```
>
> Hi David--
>
> Thanks for the cool description of you project.
>
> Now, prepare to be lightly toasted :-)
>
>> FUNCTION FindTLB, startID
>>
>> ; This function traces up the widget hierarchy to find the top-level base.
>>
>> FORWARD_FUNCTION FindTLB
>> parent = Widget_Info(startID, /Parent)
>> IF parent EQ 0 THEN RETURN, startID ELSE parent = FindTLB(parent)
>> RETURN, parent
>> END
>
> I have no problem with recursion. In this case however it's not
> really needed. For the book I'm writing, this is known as tail
> recursion I believe, which is often easily optimized. I admit
> recursion may help you conceptualize what's going on though. Wouldn't
> the following code do the same thing?
>
> parent = startid
> while widget_info(parent, /parent) NE 0 do $
>   parent = widget_info(parent, /parent)
>
```

To defend the concept of recursion, you need a full tree walk. I found this lying among many other scraps, written several years ago. It goes the other direction, starting with a TLB (or any widget for that matter), and compiling a list of all widgets in the hierarchy beneath it. You can do it without recursion (as you can any algorithm), but it's ugly.

It's depth first (which, David, means that the recursive function call occurs before the part which looks "sideways" at a given tree level, so you descend all the way to the "leaf nodes" before unraveling the recursive stack). You can also easily make it breadth first search, so that siblings are discovered before children, grandchildren, etc. It's all in the ordering.

JD

```
;; descend hierarchy of tlb .. return entire tree's widget_ids in 'list'  
pro treedesc, curin, list  
  cur=curin ;ensure local copy of current widget  
  if cur ne 0 then begin  
    if n_elements(list) eq 0 then list=cur else list=[list,cur]  
    cur=widget_info(cur,/CHILD) ;descend one level  
    while cur ne 0 do begin ; find siblings... descend their subtrees  
      treedesc,cur,list ; recurs on sibling  
      cur=widget_info(cur,/SIBLING)  
    endwhile  
  endif  
end
```

## File Attachments

---

1) [treedesc.pro](#), downloaded 129 times

---