

Re: handling errors through catch

Source: <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2006-04/msg01342.html>

- *From:* Neil Madden <nem@xxxxxxxxxxxxxx>
 - *Date:* Fri, 28 Apr 2006 14:40:45 GMT
-

maura.monville@xxxxxxxxxx wrote:

So far I've bothered you all trying to eliminate the folloeing error

```
% set num 20000000000
20000000000
```

```
set num_pnt [expr $num-int($num)]
integer value too large to represent
```

Since the value returned, witten in a point-info window, is correct I wonder whether it would be simpler just suppress such an error message. To this purpose I was thinking of replacing the following error-causing statement:

```
set num_pnt [expr $num-int($num)]
with
catch {set num_pnt [expr $num-int($num)]}
```

Is it going to fail or work ?

Well, that depends on what code following that statement expects. If the following code expects that \$num_pnt exists and contains a floating-point number value, then it will fail, as in the error case num_pnt will not be set to anything. e.g.

```
set num $veryLargeNumber
catch { set num_pnt [expr $num-int($num)] }
....
incr num_pnt ;# this will now fail as num_pnt doesn't exist
```

The catch command returns a code which says whether an error occurred or not, so you can use this to determine what to do:

```
if {[catch {expr {$num-int($num)}} num_pnt]} {
# an error occurred
set num_pnt $someDefaultValue
} else {
# No error, continue as normal
}
```

Re: handling errors through catch

Firstly, note that you can supply a variable to the catch command (catch script ?varName?). If the script completes with no errors, then this variable is set to the result of the script. i.e. "set num_pnt [expr]" is equivalent to "catch { expr ... } num_pnt" for the non-error case. If an error is raised, however, then catch returns 1 and sets the variable to a descriptive error message.

Just catching the error usually isn't enough, though, as future code may depend on the result. So you need to handle the error in some manner. The appropriate way to handle the error depends on what you are trying to do. It might be appropriate to set num_pnt to some default value in this case, or you may want to exit from the procedure, or raise a different error. Silently catching and dealing with the error here might get the code to work, but it might be the wrong place to put the fix. If some other code is calling this procedure with incorrect values, then it is likely that the bug is in that code, and trying to deal with it here just leads to more problems.

You say that the "value returned [...] is correct" despite the error, but I suspect that this isn't the case. If an error is really being generated then the procedure won't return a value at all -- the error will cause it to exit. Perhaps the value you see in the "point-info" window is from a previous calculation, or from some other part of the code?

-- Neil

.