

Re: "Did not find leading dereferencer" – new findings to an old puzzle

## Re: "Did not find leading dereferencer" – new findings to an old puzzle

---

*Source:* <http://coding.derkeiler.com/Archive/Perl/comp.lang.perl.misc/2006-11/msg00896.html>

---

- *From:* "Ronny" <[ro.naldfi.scher@xxxxxxxxxx](mailto:ro.naldfi.scher@xxxxxxxxxx)>
  - *Date:* 13 Nov 2006 04:37:07 -0800
- 

Peter J. Holzer schrieb:

The use Switch was in the code only for historic reason, and I have removed it. I don't know of course if *\*this\** was sufficient to make the error go away: After all, the removal of the statement made the code a little bit smaller, so the problem might simply have disappeared for *\*this\** reason (removing a comment line instead of the use Switch would have had the same effect).

Possible, but unlikely. Unless you use another module which uses Text::Balanced, you aren't calling Text::Balanced any more and hence can't get any error messages from it any more.

Well, you are free to get a copy of my code to see what I'm using. Aside from the Switch, I'm using:

```
use Carp qw(croak cluck confess);
use File::Temp qw(mktemp);
use File::Basename;
```

I have no idea whether these modules use Text::Balanced, but note that even when I do a "use Switch", I never actually *\*write\** a switch statement.

Then I have the impression that the error message "comes from the Switch module and goes away when I remove it". I think this does not describe the situation. If at all, we can say that the Switch module somehow *\*masks\** the real error/warning message. This means: I *\*do\** have some construct which normally would yield an error (or warning) message, but instead I get the "leading dereferencer" error; and when I simplify the program (for example, by only removing a single comment line

Re: "Did not find leading dereferencer" – new findings to an old puzzle

Re: "Did not find leading dereferencer" – new findings to an old puzzle

out of a set of other comment lines, in a completely different part of the program), the "leading dereferencer" error disappears and instead I see the \*real\* error (or warning) message.

If you can explain this behaviour just by the fact that I have a "use Switch" at the beginning, without ever using a switch, I am curious to know how you do it...

I don't think it has anything to do with the size. The problem was probably that the Switch module needs to parse the source code and that uses a different grammar than perl (which is unavoidable, as the purpose of Switch is to change the grammar). So changing a character which shouldn't make a difference in Perl does make a difference in Perl+Switch.

I've had a look at Switch.pm .... this module doesn't even have a BEGIN section.

It definitely does not need to parse the source code of the whole program.

Instead it works by passing around code blocks when a switch "statement" is actually used in my program.

Then I've checked Text::Balanced. Looks similarly harmless (nothing be done

unless you actually \*use\* some of its features). The only \*probably\* suspicious

line is an overloading of the string conversion, i.e.

```
use overload "" => sub { "$_[0]->{error}, detected at offset  
$_[0]->{pos}" };
```

and the "leading dereferencer" is triggered when on code analysis, a variable

name was expected and the leading "\$" or "@" or "%" was not found.

Still I didn't

see anything which might cause this error to pop up when none of the features

in the "use" module are never actually called.

In addition, I would like to point out that I had the same error message a few

months ago in a different context. Following a recommendation in this newsgroup,

I found in that case that it helped to remove the prototype declaration from a

Re: "Did not find leading dereferencer" – new findings to an old puzzle

function definition, and the problem disappeared (i.e. I got the "real" error message displayed again, instead of the "dereferencer" one).

There is one point in which I agree with you: The error message is likely related to the fact that Text::Balanced is somehow included in the compilation – after all, that message is defined there and the Perl compiler is unlikely to invent this wording on its own. But I don't believe that this is due to a bug in Text::Balanced. At least to me, it is very hard to imagine that a bug in a module, which is only "use"d, but never "used" (pun intended), could turn a very valid error message at a completely different point in the source code, into a "Leading Dereferencer" one, unless we also assume a bug in the Perl compiler itself. This would require – if it is possible at all, which I doubt – such a deep "introspection" in its own code, that one would find a trace of it by glancing over the source code of Balanced.

Ronald

.