

## Re: generic type, tree

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2008-05/msg02364.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2008-05/msg02364.html)

---

- *From:* David Resnick <[Indresnick@xxxxxxxx](mailto:Indresnick@xxxxxxxx)>
  - *Date:* Mon, 19 May 2008 07:06:53 -0700 (PDT)
- 

On May 18, 4:54 pm, [j...@xxxxxxxx](mailto:j...@xxxxxxxx) (Jens Thoms Toerring) wrote:

Eligiusz Narutowicz <[eligiuszdotn...@xxxxxxxx](mailto:eligiuszdotn...@xxxxxxxx)> wrote:

Antoninus Twink <[nos...@xxxxxxxx](mailto:nos...@xxxxxxxx)> writes:

On 18 May 2008 at 10:46, Jens Thoms Toerring wrote:

[alternative...@xxxxxxxx](mailto:alternative...@xxxxxxxx) wrote:

```
int v = 1;
node *n =
malloc(sizeof(node));
n->value = &v;
```

Here you have a bad problem. What you store in the 'value' member is a pointer to a variable that vanishes the moment you return from this function.

You're right, but perhaps it will also be helpful to mention to the OP that tracking down such bugs is very easy using the valgrind tool, which will pick up invalid memory accesses like this. (This is just in case the OP isn't a preternaturally talented programmer like all the clc regulars, who as we know have no need for debugging tools besides their eyes.)

If the OP is using Linux (or Windows too I think) then splint is also quite good. I got the link from this group I think.

Re: generic type, tree

While splint indeed gives you some hint at what's going wrong (alas in a way that probably will make it hard for the OP to grasp) using valgrind seems to me like looking for your misplaced keys with a microscope. All it tells you is something about an invalid read in the affiche() function, rather likely just reinforcing the OP in the wrong believe that there would be something wrong with this function and not helping at all in finding the real reason for the problem. I would recommend to use valgrind (or similar memory debuggers) only as a last resort and instead to learn to avoid such mistakes since valgrind can only detect the results of those mistakes, not the mistakes themselves.

While I don't often agree with Mr. Twink, but I think that (assuming the OP is using Linux, of which I can see absolutely no hint in his post) using valgrind early and often is a good thing. Even on programs that seem to work. I added it to our large regression suite a while back at work and found a number of defects that way that had not been otherwise noticed. They were of the class of things (e.g. off by one errors causing invalid memory reads/writes) that can cause difficult to diagnose at the customer site memory corruptions, but often aren't detected in unit testing... I'm thus a big fan of using valgrind (and in the past purify) at any stage of development...

I also agree with your point to some degree, that being able to find these things by eye is a good thing. A competent code inspector should be able to catch return of automatic memory (or putting it into a struct that is returned).

-David

.