Montana Tech Library
Digital Commons @ Montana Tech

Computer Science & Software Engineering | Faculty Scholarship

10-2015

Ban 'Naked' Braces!

A. Frank Ackerman, Ph.D.
Montana Tech of the University of Montana

Follow this and additional works at: http://digitalcommons.mtech.edu/sw_engr


Part of the Engineering Education Commons, and the Software Engineering Commons

Recommended Citation
Ackerman, F. (2015). Ban "naked" braces. Communications of the ACM, 58(10). doi:10.1145/2816943

This Letter to the Editor is brought to you for free and open access by the Faculty Scholarship at Digital Commons @ Montana Tech. It has been accepted for inclusion in Computer Science & Software Engineering by an authorized administrator of Digital Commons @ Montana Tech. For more information, please contact ccote@mtech.edu.

Call for Nominations
for ACM General Election

The ACM Nominating Committee is preparing to nominate candidates for the officers of ACM: **President**, **Vice-President**, **Secretary/Treasurer**; and five **Members at Large**.

Suggestions for candidates are solicited. Names should be sent by **November 5, 2015** to the Nominating Committee Chair, c/o Pat Ryan, Chief Operating Officer, ACM, 2 Penn Plaza, Suite 701, New York, NY 10121-0701, USA.

With each recommendation, please include background information and names of individuals the Nominating Committee can contact for additional information if necessary.

Vinton G. Cerf is the Chair of the Nominating Committee, and the members are Michel Beaudouin-Lafon, Jennifer Chayes, P.J. Narayanan, and Douglas Terry.

Association for Computing Machinery

DOI:10.1145/2816943

Ban 'Naked' Braces!

One fine business afternoon early in 1990, when we still used wires and microwave towers to make phone calls, and almost all long-distance calls went through big AT&T switches, one of the 100 or so 4ESS switches that handled U.S. long-distance traffic at the time hit a glitch and executed some untested recovery code. The switch went down briefly. No biggie, since traffic automatically took other routes, but in the process the initial switch that hit the glitch dragged its neighboring switches down, and the process cascaded across the country, as all the switches that handled long-distance traffic began to repeatedly crash and auto-recover. The result was that hardly any public telephone customer in the U.S. could make a long-distance phone call that afternoon, along with millions of dollars of time-sensitive business lost.

AT&T tried to contain the damage by rebooting the misbehaving switches, but as soon as a switch was brought back up, a neighboring switch would tell it to go down. The engineers at AT&T's R&D arm, Bell Labs, who wrote the switch programs, were called in, and, by the end of the day, network normality was restored by reducing the network message load.

An investigation was launched immediately, and after digging through a few hundred lines of code, word-of-mouth within Bell Labs was that the culprit was a closing brace (}) that terminated a selection construct—but the wrong one. The lawyers at Bell Labs quickly claimed such a lapse of human frailty could never be avoided entirely, and so dodged any potential lawsuits.

The lawyers were right; the intrinsic nature of software is such that the total absence of bugs is never guaranteed. But the simple practice of tagging all closing braces (or `end` in some languages) with a brief comment that indicates which construct they are closing would go far toward eliminating such an error; for example, instead of just writing '}' all by its naked self, write }//for, or }//if, or whatever.

Tagging construct terminators can be done without changing existing compilers, and since such construct terminators usually appear on a line of code by themselves, the structure of the code is not affected. All this does is make the code easier to understand and helps prevent bugs like the one just described. This practice is especially helpful when code must be moved about, which happens often. In addition, if coders want to go one step further in making their code understandable, a brief comment can be added after the tag, like this

```
}//for all transactions over a thousand dollars
```

This would also eliminate the usefulness of putting the opening brace on a line by itself where it would be separated, from a syntactic viewpoint, from the construct it is punctuating, while creating an almost blank line that could better serve to separate logically distinct parts of a program.

I thus propose adoption of this practice by all software engineers and coders forthwith, as well as taught to all beginners from the get-go.

A. Frank Ackerman, Butte, MT

Surprisingly Deep Roots of Word Processor Interface Design

The Research Highlight "Soylent: A Word Processor with a Crowd Inside" by Michael Bernstein et al. (Aug. 2015) reminded me how long software developers have been pursuing such basic concepts as reducing redundancy and improving readability in computer-generated text. Soylent recruits volunteer humans via the Web, through a novel form of crowdsourcing, to accomplish what has long been a goal for natural language processing—improving readability and reducing redundancy in computer-produced text. Early work on auto-