Runtime error checking for novice C programmers

Matthew Heinsen Egan, Chris McDonald
Computer Science and Software Engineering
University of Western Australia
Crawley, WA 6009, Australia
m.heinsen.egan@graduate.uwa.edu.au, chris.mcdonald@uwa.edu.au

ABSTRACT

We present SeeC, a novice-focused tool for the C programming language that records the execution of student programs, detects runtime errors, and allows students to review their program's execution in a graphical environment.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging; K.3.2 [**Computers and Education**]: Computer and Information Science Education

General Terms

Human Factors, Languages

Keywords

Novice programmers, debuggers

1. INTRODUCTION

Newcomers to the C programming language often experience great difficulties with runtime errors, exacerbated by the language's limited error checking and its concept of undefined behavior. Typical errors include array-bounds violations, use of uninitialized memory, dereferencing invalid pointers, passing invalid pointers to standard library functions, and passing non-terminated character arrays to string handling functions.

2. SEEC

SeeC is a novice-focused tool which can detect all of these errors and describe them with regard to the user's source code. Furthermore, SeeC records the program's execution and provides a simple graphical interface for the user to move backwards and forwards through their program's execution history. This allows users to reason backwards from a runtime error in order to determine the true cause of the error, or to simply inspect the behavior of their program.

ITiCSE 2013, University of Kent, Canterbury, England

SeeC uses LLVM, introduced by Lattner and Adve [1], to perform compile-time instrumentation on student programs: inserting code for execution tracing and error detection, and redirecting function calls from the C standard library to error-checking wrapper functions. These functions check input values against information from the execution tracing system, allowing them to detect invalid usage such as passing a non-terminated character array to a function that expects a C string. This allows SeeC to produce errors that describe the program's misuse of the standard library, rather than the errors that *result* from that misuse.

3. COMPARISON WITH MEMCHECK

C's long and fruitful life has seen the creation of numerous debuggers and error detectors, such as the Valgrind tool Memcheck, which detects various runtime memory errors [2]. Valgrind's dynamic instrumentation can be used with precompiled binaries, but compile-time instrumentation allows access to more semantic information, providing the opportunity for increased error detection with precise reporting. SeeC can show the exact expression that is responsible for an error, rather than only showing the containing line.

We compared the error detection of SeeC and Memcheck by testing the correctness of 170 student project solutions collected during the 2nd-semester 2012 presentation of our University's first year course on the C programming language and Operating Systems. SeeC detected errors in forty-three student programs for which Memcheck detected no errors (seven of these were "benign" uses of uninitialized memory, allowed by Memcheck). For all programs in which Memcheck detected errors, SeeC also detected errors.

4. ACKNOWLEDGEMENTS

This research is partially supported by an Australian Postgraduate Award.

5. REFERENCES

[1] C. Lattner and V. Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, Palo Alto, California, Mar 2004.

[2] J. Seward and N. Nethercote. Using valgrind to detect undefined value errors with bit-precision. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 2–2, Berkeley, CA, USA, 2005. USENIX Association.