Teaching Debugging Collaboratively—Midway Report

Sammy Furr

May 7, 2020

1 Introduction

Debugging is invaluable in writing and understanding code, yet it is rarely formally taught. We typically teach students programming structures, concepts, and languages, but we leave them to learn the tools they use to code by themselves. This approach often works well—the programmer's choice of editor is *very* personal, students figure out how to configure an individualized workflow. Perhaps because debuggers are tools, they often get lumped into the "teach yourself" category. Unlike editors or reference guides however, effectively using a debugger requires a set of high-level, platform agnostic, teachable skills. Teaching these skills is effective, and translates into better, faster, debugging and programming. [2] [3]

The use of a debugger is particularly important in a low-level programming classes, which is often the first time that students encounter concepts like assembly instructions and memory addresses. Debuggers such as GCC offer incredibly powerful tools to step through and inspect running code. Sadly, low level debugging tools are often less than intuitive, and provide little to no means for collaboration. Powerful tools such as RR exist that enable collaborative "record and replay" debugging [4], but they lack tools to visualize address spaces and control flow of programs. Research shows that visualization of these previously unexplored spaces aids students taking low-level or systems programming classes [5].

The necessity for accessible, collaborative tools for teaching all aspects of computer science, not the least debugging, has been exemplified by the current COVID-19 crisis. Many students don't have access to a high powered computer (or even a computer at all) to run tools that allow for collaboration and replay by capturing the entire state of a virtual machine [1]. Though the limitations of interacting with a debugger through a phone or low-powered laptop seem daunting, I believe they are actually constraints that can help better shape a visualized, collaborative approach.

For my senior project, I want to develop a front-end interface to RR that enables collaborative debugging and process visualization on a varitey of non-traditional platforms.

2 The Value of Teaching Debugging

3 Debugger: Low Level

4 Debugger: High Level

5 Next Steps

References

[1] Dolan-Gavitt, B., Hodosh, J., Hulin, P., Leek, T., and Whelan, R. Repeatable reverse engineering with panda. In *Proceedings of the 5th Program Protection and Reverse Engineering Workshop* (New York, NY, USA, 2015), PPREW-5, Association for Computing Machinery.

[2] Li, C., Chan, E., Denny, P., Luxton-Reilly, A., and Tempero, E. Towards a framework for teaching debugging. In *Proceedings of the Twenty-First Australasian Computing Education Conference* (New York, NY, USA, 2019), ACE 19, Association for Computing Machinery, p. 7986.

[3] Michaeli, T., and Romeike, R. Improving debugging skills in the classroom: The effects of teaching a systematic debugging process. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education* (New York, NY, USA, 2019), WiPSCE19, Association for Computing Machinery.

[4] O'Callahan, R., Jones, C., Froyd, N., Huey, K., Noll, A., and Partush, N. Engineering record and replay for deployability: Extended technical report. *CoRR abs/1705.05937* (2017).

[5] Walker, J., Wang, M., Carr, S., Mayo, J., and Shene, C.-K. A system for visualizing the process address space in the context of teaching secure coding in c. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2020), SIGCSE 20, Association for Computing Machinery, p. 10331039.