COM S 413/513 Project 1: Understanding bugs

Learning Objectives

1. Get further familiar with the terminologies related to bugs
2. Understand the challenge of bug understanding and the need of automatic tool support
3. Improve problem solving skills on code inspection and bug diagnosis
4. Work with real-world software and bugs

Description

In this homework, you are going to exercise your code inspection skills to analyze and understand bugs. As a first step, you will download the dbg benchmark located here: https://dbgbench.github.io/ This benchmark documented a set of real-world bugs from unix utilities *find* and *grep*. Here are some guidance on how you can explore this website effectively:

1. You can start under “use and reproduce”, and follow “Step-by-Step instructions”. The website also lists a set of tutorial materials (Click “tutorial materials” under “setup and infrastructure”) that can help you further understand the two programs.
2. Click “Data for find” and “Data for grep” on the website, you will find detailed explanations of the bugs, including the root causes, symptoms and the tests that can trigger the bugs. There are also a set of patches for each bug developed by different participants of the research as well as the patch from the developer’s original fix. You can find further information about the benchmark following the paper “Where is the bug and how is it fixed? an experiment with practitioners” https://dl.acm.org/citation.cfm?id=3106255

In this assignment, you are going to study and reproduce one of these bugs of your selection. Here are the list of steps to follow:

1. (7 pt) Study “simplified bug report” listed in the bug description, and create a data entry consisting of “program, bugid, bug type, bug location, fault location, fault condition, patch location, test input that triggers the bug” and save it in a file called bug.txt. Note that the terminologies of bug and fault in the dbg benchmark may be different from the ones taught in our class. Please use the terminologies taught in our class.
    a. Fault location and fault condition (assertion): 4 pt

   b. The rest: 3 pt (0.5 pt each item)

2. (8 pt) Create a slice for the fault by deleting irrelevant statements in the original code or by selecting relevant statements from the original code. The slice should be executable and as small as possible (we are going to create a leadboard for the class to compete the size of the slice). For this question, you will need to submit two sets of .c/.cpp files, the original buggy files and the files that contain the slice. You will also report the sizes, in terms of the lines of code for the original program and for your slice. Please use the tool I upload to the course website under the directory of *tools.* You can list this number in readme.txt
   a. executable (2 pt)
   b. reproduce the same bug (3 pt)
   c. small (2 pt)
   d. available of two sets of c/cpp files and size of the slice (1 pt)

3. (5 pt) Answer the following questions in readme.txt
   a. Can the input that triggers the original bug also crash the slice?
   b. What are your thoughts/findings/experiences of analyzing this bug?

If you are interested to evaluate your ability of bug diagnosis, you can also fill the questionnaire located https://dbgbench.github.io/questionnaire.pdf Then you can compare your answers with the survey results the researchers had generated over a set of experienced participates in the paper.

Deliverables (20 pt)

Please zip the following files and submit the zipped file to canvas under the “project 1: understanding bugs” column.

1. bug.txt for Q1
2. two sets of .c/.cpp files for Q2
3. readme.txt for Q2 and Q3

The homework is due *Feb 7, Fri 11:59pm*. You can continuously submit your homework without a penalty after the deadline. But once the TA started grading the homework, we will no longer accept late homework or modifications. This project can be challenging if it is your first time for handling real-world bugs. So start early!