ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΡΟΥΣΙΑΣΗ / ΕΞΕΤΑΣΗ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Μεταπτυχιακός Φοιτητής
Τσιατσιάνας Ευάγγελος

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης
Επόπτης Μεταπτ. Εργασίας: Καθηγητής, Α. Σαββίδης

Τετάρτη, 08/04/2020, 16:00
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

"Αντίστροφη Εκσφαλμάτωση Κατά την Εκτέλεση στον LLDB"

ΠΕΡΙΛΗΨΗ

Η εκσφαλμάτωση είναι μία απαραίτητη διαδικασία, η οποία καταλαμβάνει μεγάλο μέρος της καθημερινής εργασίας των προγραμματιστών. Παρ' όλα αυτά, η συντριπτική πλειοψηφία των προγραμματιστών βασίζεται σήμερα, ως επί το πλείστον, σε εργαλεία που έχουν παραμείνει σχεδόν ίδια τα τελευταία 20 χρόνια, δηλαδή διαδραστικούς εκσφαλματωτές με δυνατότητα λειτουργίας αποκλειστικά προς την κανονική κατεύθυνση ροής της εκτέλεσης του προγράμματος. Η αιτία εντοπίζεται στο ότι περισσότερο προχωρημένες λύσεις, οι οποίες παρέχουν υποστήριξη για αντίστροφη εκσφαλμάτωση, δεν χρησιμοποιούνται, διότι είτε έχουν περιορισμένο πεδίο δράσης και δυνατοτήτων είτε είναι άγνωστες στην ευρύτερη προγραμματιστική κοινότητα.

Ως αποτέλεσμα, αποφασίσαμε να δημιουργήσουμε ένα πρότυπο σύστημα για αντίστροφη εκσφαλμάτωση κατά την εκτέλεση βασισμένο στον LLDB, έναν δημοφιλή, σύγχρονο και ευρέως χρησιμοποιούμενο διαδραστικό εκσφαλματωτή ανοιχτού κώδικα.

Στόχος της εργασίας αυτής ήταν τόσο η σχεδίαση και υλοποίηση του ανωτέρω προτύπου όσο και η χρήση της ως μία σύντομη εισαγωγή στους αντίστροφους εκσφαλματωτές. Αρχικά, περιγράφουμε τη γενικότερη διαδικασία εκσφαλμάτωσης και παρουσιάζουμε εργαλεία που προσφέρουν οι παραδοσιακοί εκσφαλματωτές στους προγραμματιστές. Στη συνέχεια, συστήνουμε στον αναγνώστη τις επιπλέον λειτουργίες που προσφέρονται από τους αντίστροφους εκσφαλματωτές και παρατηρούμε πώς οι τελευταίοι μεταβάλλουν και βελτιώνουν τη διαδικασία εκσφαλμάτωσης.

Επιπλέον, συζητούμε σχετικά με τις ποικίλες μεθόδους και τρόπους προσέγγισης που μπορούν να ακολουθηθούν για την υλοποίηση ενός αντίστροφου εκσφαλματωτή, ενώ παραθέτουμε και τις αδυναμίες καθενός τύπου εκσφαλματωτή. Ακολουθεί μία σύντομη επισκόπηση υπαρχόντων λύσεων, τόσο κατά την εκτέλεση όσο και εκ των υστέρων, εκσφαλμάτωσης, κατηγοριοποιημένων βάσει του πεδίου δράσης τους και της προσέγγισης που ακολουθούν.

Έπειτα, ο αναγνώστης συναντά μερικούς απαραίτητους τεχνικούς όρους, προτού προχωρήσει στην παρουσίαση της σχεδίασης και της μεθόδου υλοποίησης που επιλέχθηκε για το σύστημα που παρουσιάζουμε. Ξεκινούμε την παρουσίαση του συστήματος με την αρχιτεκτονική του LLDB σε επίπεδο κώδικα, εστιάζοντας στις δομές δεδομένων που αναπαριστούν το πρόγραμμα υπό εκσφαλμάτωση, καθώς και σε εκείνες που προστέθηκαν προκειμένου να εισάγουμε υποστήριξη για αντίστροφη εκσφαλμάτωση κατά την εκτέλεση.

Κατόπιν, εξηγούμε τη μέθοδο που ακολουθήθηκε με στόχο να προστεθεί η λειτουργία αντίστροφης εκσφαλμάτωσης κατά την εκτέλεση στον LLDB και αναφέρουμε τις εσωτερικές αλλαγές που απαιτήθηκαν, προκειμένου να φιλοξενηθεί η νέα λειτουργικότητα. Ακόμη, αναλύουμε πώς ο αντίστροφος εκσφαλματωτής λαμβάνει στγμιότυπα και καταγράφει την εκτέλεση του υπό εκσφαλμάτωση προγράμματος, καθώς και τη δομή των στιγμιοτύπων αυτών, τα οποία δύνανται να επαναφέρουν την εκτέλεση σε παρελθοντική κατάσταση δίχως ανάγκη τερματισμού του προγράμματος. Φροντίζουμε να εξηγήσουμε σε επίπεδο κώδικα πώς ο αντίστροφος εκσφαλματωτής χειρίζεται μεταβολές τιμής σε μεταβλητές, καταχωρητές και θέσεις μνήμης κατά τη λήψη και επαναφορά στιγμιοτύπων και συζητούμε όλα τα μέτρα που έχουν ληφθεί προκειμένου να ελαχιστοποιήσουμε την επιβάρυνση σε επίπεδο επιδόσεων και μνήμης, διατηρώντας παράλληλα την ακρίβεια και την ορθότητα της καταγεγραμμένης εκτέλεσης.

Τέλος, αναδεικνύουμε τη δύναμη του αντίστροφου εκσφαλματωτή μέσω ενός αριθμού σεναρίων και αξιολογούμε το συνολικό αποτέλεσμα της εργασίας, προτού παραθέσουμε τις λειτουργίες που λείπουν, αλλά έχουν ήδη προγραμματιστεί για μεταγενέστερη έκδοση του εκσφαλματωτή.

Tsiatsianas Evaggelos
M.Sc. Thesis

Computer Science Department
University of Crete
Master's Thesis Supervisor: Professor, A. Savidis

Wednesday 08/04/2020, 16:00 p.m
Computer Science Dept., University of Crete

“Live Reverse Debugging in LLDB”

ABSTRACT

Debugging is an essential process that occupies a great proportion of the daily work conducted by every developer. Nevertheless, the overwhelming majority of programmers nowadays mostly relies on almost the same tools as 20 years ago, i.e., forward-only interactive debuggers. The reason for this is that more advanced, reverse debugging solutions, are either very limited in their scope and capabilities or relatively unknown to the wider developer community and thus remain mostly unused.

As a result, we decided to create a proof of concept for live reverse debugging based on a well-known, modern and widely used open-source interactive debugger, LLDB.

The goal of this thesis was to design and implement the aforementioned proof of concept and also act as a brief introduction to reverse debuggers. We begin by describing the general debugging process and present the tools forward-only debuggers make available to the programmer. Then, we introduce the additional features provided by reverse

debuggers and how the latter affect and improve the debugging process. In addition, we discuss the multiple different methods and approaches that can be taken to implement a reverse debugger and also list the limitations of each debugger type. Following, is a brief overview of existing live and offline reverse debugging solutions, categorized based on their scope and solution approach.

Later on, the reader runs across a few essential terms, before moving on to the presentation of the design and implementation method selected for the resulting proof of concept. We begin the presentation of the work with the code-level architecture of LLDB, focusing on the data structures responsible for representing the program being debugged and the ones introduced, in order to add support for live reverse debugging.

Afterwards, we explain how the live reverse debugging functionality was added to LLDB, along with any internal modifications required, in order to accommodate the additional functionality. We also analyze the way the live reverse debugger records the execution of the program being debugged by capturing snapshots and the way those snapshots are structured and restored in order to move the execution backwards in time, without ever terminating the program. We make sure to explain at the code level how modifications to the program state, such as in variables, registers and memory are handled when capturing and restoring snapshots and discuss all measures taken in order to minimize the performance overhead and memory footprint of the solution, while also preserving the accuracy and correctness of the execution recording.

Ultimately, we demonstrate the power of the live reverse debugger via a number of case scenarios and evaluate the overall result of this work, before listing the missing, but already scheduled for a subsequent version, features of the live reverse debugger.