Debagovanje u LLDB-u

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Momir Adžemovic, Miloš Miković, Marko Spasić,
Mladen Dobrašinović

momir.adzemovic@gmail.com, spaskeasm@gmail.com,
milos.mikovicpos@gmail.com, dobrasinovic.mladen@gmail.com

1. april 2020.

Sažetak

Ovaj rad predstavlja grupni projekat u okviru kursa Metologija stručnog i naučnog rada. Ovo je dobra prilika da podelimo sa kolegama naša znanja koja smo stekli ovim istraživanjem koje ima velike primene u praksi. Rad većinom pokriva interesatne informacije o debageru LLDB kao jednom od produžetaka LLVM-a, način korišćenja LLDB i poređenje sa ostalim debagerima.

Sadržaj

1 Uvod

Kada budete predavali seminarski rad, imenujete datoteke tako da sadrže redni broj teme, temu seminarskog rada, kao i prezimena članova grupe. Precizna uputstva na temu imenovnja će biti data na formi za predaju seminarskog rada. Predaja seminarskih radova biće isključivo preko veb forme, a NE slanjem mejla. Link na formu će biti dat u okviru obaveštenja na strani kursa. Vodite računa da prilikom predavanja seminarskog rada predate samo one fajlove koji su neophodni za ponovno generisanje pdf datoteke. To znači da pomoćne fajlove, kao što su .log, .out, .blg, .toc, .aux i slično, **ne treba predavati**.

2 Šta je debagovanje

Debagovanje je proces identifikacije pravog problema i njegovo ispravljanje. "Debagovanje je duplo teže od kodiranja, ako napisete kod na najlukaviji (odnosno najkomplikovaniji) način, po definiciji niste dovoljno pametni da ga debagujete.(Brian w. Keringhan) [1]
Koraci pri debagovanju [10]:

1. uočavanje da postoji greška
2. razumevanje greške
3. lociranje greške
4. ispravljanje greške

Često je najteži deo ispravno razumevanje i rano otkrivanje greške, kada se greška locira, ispravljanje najčešće nije veliki problem.

2.1 Bagovi uopšteno

Postojanje grešaka(bagova) se često neopravdano poistovećuje sa propustima u programiranju. U širem kontekstu bag, greška, defekt ili propust se odnosi na bilo koju vrstu problema u bilo kojoj fazi procesa razvoja kao što su greške u projektovanju, planiranju, arhitekturi, dizajnu... Zato se često termini propust i greška koriste u širem kontekstu razvoja a termin bag u užem i vezan je za propuste u programiranju.

Jedna od najčešćih klasifikacija bagova prema načinu ispoljavanja obuhvata:

1. **Nekonzistentnosti u korisničkom interfejsu**: često je slučaj da se komanda ctr+f koristi za pretraživanje dokumenta, Outlook koristi tu komandu za prosleđivanje poruke
2. **Neispunjena očekivanja**: dobijanje neočekivanog(pogrešnog) rezultata
3. **Slabe performanse**: stalno ili povremeno čekanje rezultata zbog lošeg odziva sistema, takvi programi su često ne upotrebljivi
4. **Padovi sistema i oštećenja podataka**: predstavljaju najopasniji vid bagova, mogu trajno oštetiti sistem i podatke

Bagovi su jako neugodni i treba ih sistematski otklanjati čestim refaktorisanjem i planskim gradjenjem koda.Jedne od okolnosti koje pogoduju nastajanju bagova su nedovoljna stručnost razvojnog tima i povećan stres na poslu, a informisanost, sistematičnost i redovnost ih suzbijaju. [1]

2.2 Metode debagovanja

1. Neformalno debagovanje Neformalno debagovanje predstavlja jednostavan i površan pristup i čine ga dva koraka.
   (a) Pokušati sa nekom jednostavnom popravkom.
   (b) Ponavljati korak (a) dok se problem ne reši

   Ovaj metod se ne preporučuje u praksi i često može da proizvede nove probleme, pogotovo ako vršimo puno sitnih popravki za koje nismo sigurni da će rešiti problem.Ponekad, ako su u pitanju sitne greške, ovaj metod se može oprezno koristiti jer dovodi do brzog rešenja problema.[1]
2. Empirijski naučni metod Ovaj postupak je sličan uobičajenom istaživačkom metodu u prirodnim naukama. Čine ga sledeći koraci:
   (a) Posmatrati uočen problem
   (b) Postaviti hipoteze o uzroku problema
   (c) Na osnovu hipoteza napraviti predviđanje
   (d) ponašanja
   (e) Eksperimentalno proveriti ispravnost predviđanja Ponavljati prethodne korake uz popravljanje ili menjanje hipoteze, sve dok se ne potvrdi ispravnost hipoteze ili ne ponestanu mogućnosti za njeno dalje unapređivanje.

   Uopšteno gledano ovo je najbolji pristup debagovanju.Često je jako zahtevan i oduzima dosta vremena ali je sa druge strane je temeljan i koncizan.[1]
3. Heurističko debagovanje Ova vrsta debagovanja podrazumeva postojanje heuristike(skup pravila) koja olakšava brže i efikasnije pronalaženje grešaka.Često se za određene skupove problema prave različite heuristike, koje se testiraju u praksi i kasnije koriste kao pravila pri otklanjanju određenih vrsta bagova.Ovakve heuristike odlikuje izbegavanje pravljenja previda pri posmatranju, sužavanje skupa kandidata za iskazivanje hipoteza, usmeravanje posmatranja prema uzroku problema i dr.Heuristike nisu optimalna rešenja niti egzaktna pravila koja vode rešenju problema, ali često su jako efikasne i daju "dovoljno dobra rešenja".[1]

2.3 Osnovna pravila debagovanja

Neka od osnovnih pravila debagovanja su[1]:

1. Razumeti sistem
2. Navesti sistem na grešku
3. Najpre posmatrati pa zatim razmišljati
4. Podeli pa vladaj
5. Praviti samo jednu po jednu izmenu
6. Proveriti naizgled trivijalne stvari
7. Praviti i čuvati tragove izvršavanja
8. Ako nismo popravili bag, onda on nije popravljen
9. Zatražiti tuđe mišljenje Ovde mogu da objanim pravila

2.4 Tehnike za prevenciju bagova

Tehnike za prevenciju bagova mogu biti unutrašnje i spoljašnje.Unutrašnje predstavljaju sve ono što se ugrađuje u programski kod samo radi pomoći u prevenciji i otklanjanju grešaka.Neke od njih su pravljenje pretpostavki(asserts), komentarisanje značajnih odluka i mesta u kodu, testiranje jedinica koda... Spoljašnje tehnike i alati se koriste pri razvoju i ne ugrađuju se nužno u programski kod, ali se koriste u čitavom razvojnom ciklusu.Neki od spoljašnjih alata su debager, alati za praćenje verzija programskog koda, alati za podršku i praćenje komunikacije, alati za automatizovanje pravljenja dokumentacije[1].

2.5 Debager

Debager je računarski program koji se koristi za uklanjanje grešaka, testiranje rada i proveru ispravnosti drugih programa.Debugeri daju napredne funkcije kao što su pokretanje programa korak po korak (single-stepping), praćenje vrednosti promenjivih kao i stek okvira, praćenje na nivou instrukcija i stanja procesora, zaustavljanje ili pauziranje izvršavanja programa na takozvanim tačkama prekida (breakpoint), a neki čak i omogućavaju menjanje programa tokom izvršavanja.

Većina popularnih debugera daje samo jednostavno komandno-linijsko okruženje(command-line interface - CLI), često iz razloga da maksimizuju portabilnost i minimizuju trošenje sistemskih resursa računara. Ipak, popravljanje grešaka u programu preko grafičkog korisničkog okruženja (GUI) debugera se često smatra jednostavnijim, produktivnijim i ugodnijim za rad.Neki debugeri pružaju i mogućnosti obrnutog debagovanja(debagovanje unazad) koje omogućava da se vratimo na prethodno stanje programa(Step Beckward).Jedan od debugera koji pruža ovu mogućnost je IntelliTrace koji se koristi u Microsoft-ovom razvojnom okruzenju VisualStudio.Debagovanje unazad je jako korisno i sve se više koriste debugeri koji omogućavaju ovo svojstvo.Mana debagovanja unazad je usporavanje čitavog procesa debagovnja pa čak i do dva puta.Debugeri mogu biti zavisni od programskog jezika, ako se mogu koristiti za debagovanje jednog konkretnog jezika ili mogu biti višejezični i koristiti se za debagovanje više programskih jezika.Neki debugeri uključuju i zaštitu memorije kako bi izbegli prekoračenje bafera, ili onemogućili korisnika da pristupa memoriji za koju nema dozvolu i slično.[12]

2.6 Lista debagera

Najčešće korišćeni debugeri[6][3]:

1. GDB(gnu debuger)
2. Firefox javascript
3. Lldb
4. IntelliTrace
5. Windbg
6. Valgrind
7. Watcom

Mejnfrejm debugeri[6][3]:

1. XPEDITER
2. z/XDC

3. IBM Oliver
4. CA/EZTEST

3 Gde se on koristi i za koje jezike?

LLVM debager se koristi za debagovanje programa pisanih u programskim jezicima C, Objective-C, i C++. Postoji i verzija za debagovanje programa napisanih u Swift programskom jeziku, tu verziju održava Swift zajednica. Dostupan je na FreeBSD, Linux, macOS, NetBSD, i od 2015 na Windows platformi. Komplentnost skupa funkcionalnosti varira od platforme do platforme.

- FreeBSD - zaostaje za Linux-om, ali brzo napreduje.
- Linux - Približava se kompletnosti funkcionalnosti za debagovanje x86-64, i386, ARM, AArch64, IBM POWER (ppc64), IBM Z (s390x=, i MIPS64 programa.
- macOS - LLDB je sistemski debager na macOS, iOS, tvOS, i watchOS za x86, i386, ARM, i AArch64 debagovanje. Na ovoj platformi ima najbogatiji skup funkcionalnosti koje implementira.
- Windows - I dalje u razvojnoj fazi, ali već koristan za i386 programe.

Skup funkcionalnosti se iz godine u godinu unapređuje i teži se da bude kompletan na svim platformama. Najbolja i najpotpunija podrška je trenutno na Linux i macOS platformama.

4 Koja razvojna okruzenja podrzavaju upotrebu ovog debagera i kako?

LLDB se može koristiti u Visual Studio Code, Eclipse, CLion, Xcode 5 gde je i podrazumevani debager. Instalacija u Visual Studio Code (VSC) se svodi na instaliranje dodataka sa VSC repozitorijuma. Komande LLDB se zadaju preko VCS grafičkog korisničkog interfejsa. Podržava:

- Debugging on Linux (x64 or ARM), macOS and Windows*,
- Conditional breakpoints, function breakpoints, data breakpoints, logpoints,
- Launch debuggee in integrated or external terminal,
- Disassembly view with instruction-level stepping,
- Loaded modules view,
- Python scripting,
- HTML rendering for advanced visualizations,
- Rust language support with built-in visualizars for vectors, strings and other standard types,
- Global and workspace defaults for launch configurations,
- Remote debugging,
- Reverse debugging (experimental, requires compatible backend).

U Eclipse razvojnom okruženju se korišćenje omogućava instaliranjem Eclipse dodataka koji integrše postojeći lldb na sistemu sa eclipse razvojnim okruženjem. Radi na svim platformama koje podržava lldb i eclipse. Za razliku od VCS ima nekoliko ograničenja:

Feature	Free BSD	Linux	macOS	NetBSD	Windows
Backtracing	YES	YES	YES	YES	YES
Breakpoints	YES	YES	YES	YES	YES
C++11:	YES	YES	YES	YES	Unknown
Commandline tool	YES	YES	YES	YES	YES
Core file debugging	YES (ELF)	YES (ELF)	YES (MachO)	YES (ELF)	YES (Minidump)
Remote debugging	NO	YES (lldb-server)	YES (debugserver)	YES (lldb-server)	NO
Disassembly	YES	YES	YES	YES	YES
Expression evaluation	Unknown	YES (known issues)	YES	YES (known issues)	YES (known issues)
JIT debugging	Unknown	Symbolic debugging only	Untested	Work In Progress	NO
Objective-C 2.0:	Unknown	N/A	YES	Unknown	N/A

Slika 1: feature table

- Debagovanje na daljinu nije moguće
- Core dump debagovanje nije moguće
- Watch points ne radi
- Ne može se izmeniti vrednost promenljivih tokom debagovanja
- Ne može se menjati sadržaj memorije
- Skoči na liniju, pomeri se na liniju nije implementirano Modules view se ne popuni

LLLDB dolazi u paketu zajedno sa CLion razvojnim okruženjem na Linux i macOS platformama. Postoji i eksperemintalna verzija LLDB ba-

ziranog debagera sa MSVC razvojne alate na Windows platformi.

Da bi omogućili korišćenje lldb potrebno je u podešavanjima za dati projekat odabrati postojeći lldb debager. Ne postavlja nikakva ograničenja na LLDB debager kao što je to slučaj kod Eclipse razvojnog okruženja.

5 Upoznavanje sa LLDB-om

LLDB podržava standardne funkcije debagovanja preko komandne linije i može se koristiti kao debager u interaktivnom razvojnom okruženju. Konkretno, sa debagerom pokrenutim nad programom prevedenim sa debug opcijama omogućava se [7]:

- Aktiviranje procesa programa sa određenim argumentima komandne linije (eng. command line arguments).
- Korišćenje breakpoint-a (određenog reda ili funkcije u izvornom kodu pri kojima debager zaustavlja izvršavanje programa kada se stigne do odgovarajućeg dela izvršnog koda).
- Korišćenje watchpoint-a (određene promenljive, takve da debager zaustavlja proces ili nit kada se njeno stanje promeni).
- Korišćenje dodatnih uslova nad breakpoint-ovima i watchpoint-ovima.
  - Nastavljanje ili pokretanje programa.
- Pokretanje procesa red po red (sa „ulaženjem" u funkciju ili bez).
- Istraživanje promenljivih ili memorije procesa.
- Izvršavanje proizvoljnog izraza nad stanjem procesa (npr. menjanje neke promenljive na steku).
- Istraživanje steka okvira poziva.
- Izvršavanje drugih naprednih i raznih funkcija.

Ono što ističe LLDB je omogućavanje korišćenja eksternih skripti za debagovanje preko javnog API-a za Python, izršavanje proizvoljnog Python koda unutar debagera [13] (preko ugnježdenog interpretatora (eng. embedded interpreter) i omogućavanje REPL (Read-Evaluate-Print-Loop) funkcija za programske jezike zajedno sa mogućnostima debagovanja [4].

5.1 LLDB interfejs komandne linije

LLDB interfejs komandne linije (eng. command line interface) se aktivira pozivom `lldb` u ljusci (eng. shell) sa programom koji želimo debagovati kao argumentom. Program komandne linije `lldb` se odlikuje strukturisanom sintaksom osnovnih komandi koja je sledećeg oblika [14]:

Primer 5.1

```
<imenica> <glagol> [-opcije [vrednost-opcije]] [argument [argument...]]
```

U ovakvom obliku, imenica se zove i komanda, a glagol potkomanda. Postoje i skraćenice (eng. alias) za komande koje mogu odstupati od ovog oblika. Upravo zato što je ovaj format komandi jako strukturisan mogu nam biti pogodni skraćeni oblici komandi koji mogu biti sličniji onome što je poznato korisnicima drugih debagera [5]. U tabeli 1 su date neke od osnovnih komandi kao primer korišćenja interfejsa i reprezentativni prikaz širokog skupa mogućnosti LLDB-a koji nije naveden u potpunosti u ovom radu. Posebno se ističu komande `help` i `apropos`, koje mogu biti korisne početnicima u korišćenju ovog alata.

Tabela 1: Upotreba interfejsa komandne linije LLDB-a [7][14].

`process launch -- <argumenti>`	Pokreće izabrani program sa datim argumentima.
`thread step-in`	U trenutnoj niti nastavlja izvršavanje programa sledeće instrukcije izvornog koda, ulazeći u pozive funkcija.
`thread step-inst-over`	U trenutnoj niti nastavlja izvršavanje programa sledeće instrukcije izvršnog koda, ne ulazeći u pozive funkcija.
`breakpoint set --file 1.c --line 42`	Postavlja breakpoint na red 42 u izvornom kodu programa 1.c.
`breakpoint list`	Ispisuje postojeće breakpoint-ove debagera.
`breakpoint disable 1`	Deaktivira breakpoint 1.
`apropos <ključna_reč>`	Traži u pomoći za upotrebu komandi (eng. command help) datu ključnu reč.
`help`	Štampa pomoć za komande. (help se može koristiti i za nalaženje pomoći za upotrebu potkomandi određene komande [5]).

6 Poredjenje sa drugim popularnim debagerima

Potrebno je naglasiti da pri poređenju različitih debagera ne možemo objektivno odrediti koji je debager najbolji, jer to dosta zavisi koji se operativni sistem koristi, a i samih preferenci korisnika. Visual Studio Code, jedan od popularnijih editora, koristi LLDB, GDB i VSD za C++ u zavisnosti od od operativnog sistema na kojem je instaliran [**?**]:

- **Linux**: GDB
- **macOS**: LLDB or GDB
- **Windows**: the Visual Studio Windows Debugger or GDB (using Cygwin or MinGW)

6.1 Poređenje: GDB i LLDB

Debuger GDB predstavlja standard za GNU sisteme (ne striktno samo za GNU) [2]. Ako se proverava kvalitet debagera LLDB, onda u potpunosti ima smisla upoređivati ga prvo sa GDB debagerom kao jednim od najpopularnijih debagera. Debager LLDB u debagovanju velikih programa pokazuje bolje performanse od GDB debagera i ima dobar korisnički interfejs [9]. Način korišćenje ova dva debagera je veoma sličan i skup komandi se većinom poklapa. Postoji zvaničan rečnik koji prevodi komande iz GDB u LLDB [7]. Novije verzije GDB podržavaju MacOS, ali u proteklih par godina se pretežno koristio LLDB kao glavni debager za MacOS.

	LLDB	GDB	Visual Studio Debugger
Podrška za programske jezike	C, C++, Objective C	C, C++, Objective C, Java, Fortran etc.	C#, C++, Visual Basic, JavaScript etc.
Implementacija	C++	C	C++/C#
Podrška za operativne sisteme	Unix, Windows, MacOS	Unix, Windows, MacOS	Windows
Razvijen od strane	GNU Project	LLVM developer group	Microsoft
UI	TUI	TUI	GUI

Slika 2: LLDB, GDB, Visual Studio Debugger [2][8][11]

6.2 Visual Studio Debugger i LLDB

VIsual studio debugger je takođe jedan od poznatijih debagera koji možemo da upoređujemo sa LLDB-om. Prednost VSD u odnosu na LLDB je u tome što VSD nudi grafički point-and-click korisnički interfejs, a prednost LLDB je u broju operativnih sistema koji za koju ima podršku [11].

7 Zaključak

Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.

Literatura

[1] Vladimir Filipovic. Debagovanje, 2016. http://poincare.matf.bg.ac.rs/~vladaf/Courses/Matf%20RS2/Prezentacije/CC%2023%20-%20Debagiranje%20-%20Goran%20Vinterhalter.pdf.

[2] Free Software Foundation. GDB: The GNU Project Debugger, 2020. on-line at: https://www.gnu.org/software/gdb/.

[3] Rubaiat Hossain. Best Linux Debuggers for Modern Software Engineers, 2020. https://www.ubuntupit.com/best-linux-debuggers-for-modern-software-engineers/.

[4] Apple Inc. REPL and Debugger, 2020. on-line at: https://swift.org/lldb/#why-combine-the-repl-and-debugger.

[5] Apple Inc. Understanding LLDB Command Syntax, 2020. on-line at: https://developer.apple.com/library/archive/documentation/General/Conceptual/lldb-guide/chapters/C2-Understanding-LLDB-Command-Syntax.html.

[6] Linux Links. Best Free Linux Debuggers, 2020. https://www.linuxlinks.com/debuggers/.

[7] LLVM. GDB to LLDB command map, 2020. on-line at: https://lldb.llvm.org/use/map.html.

[8] LLVM. The LLDB Debugger, 2020. on-line at: https://lldb.llvm.org.

[9] LLVM. The LLDB Debugger, 2020. on-line at: http://blog.llvm.org/2010/06/new-lldb-debugger.html.

[10] Sasa Malkov. Debagovanje, 2019. http://poincare.matf.bg.ac.rs/~smalkov/files/rs.r290.2019/public/Predavanja/Razvoj%20softvera.08.2019%20-%20Debagovanje.p4.pdf.

[11] Microsoft. Visual Studio Debugger Suport, 2019. https://docs.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs-2019.

[12] Margaret Rouse. Debagovanje, 2020. https://searchsoftwarequality.techtarget.com/definition/debugging.

[13] The LLDB Team. Python Scripting, 2020. on-line at: https://lldb.llvm.org/use/python.html.

[14] The LLDB Team. Tutorial, 2020. on-line at: https://lldb.llvm.org/use/tutorial.html.