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

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada. Obratite pažnju da je pored ove .pdf datoteke, u prilogu i odgovarajuća .tex datoteka, kao i .bib datoteka korišćena za generisanje literature. Na prvoj strani seminarskog rada su naslov, apstrakt i sadržaj, i to sve mora da stane na prvu stranu! Kako bi Vaš seminarski zadovoljio standarde i očekivanja, koristite uputstva i materijale sa predavanja na temu pisanja seminarskih radova. Ovo je samo šablon koji se odnosi na fizički izgled seminarskog rada (šablon koji *morate* da koristite!) kao i par tehničkih pomoćnih uputstava. Pročitajte tekst pažljivo jer on sadrži i važne informacije vezane za zahteve obima i karakteristika seminarskog rada.

Sadržaj

1 Uvod **3**

2 Osnovna uputstva **3**

3 Šta je debagovanje **3**
3.1 Bagovi uopšteno 3
3.2 Metode debagovanja 4
3.3 Osnovna pravila debagovanja 5
3.4 Tehnike za prevenciju bagova 5
3.5 Debager 5
3.6 Lista debagera 6

4 Upoznavanje sa LLDB-om **6**
4.1 LLDB interfejs komandne linije 7

5 Engleski termini i citiranje **7**

6 Slike i tabele **8**

7 Kôd i paket listings **9**

8 Poredjenje sa drugim popularnim debagerima **9**
8.1 Poređenje: GDB i LLDB 10
8.2 Visual Studio Debugger i LLDB 10

9 Zaključak **10**

Literatura **10**

A Dodatak **11**

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 Osnovna uputstva

Vaš seminarski rad mora da sadrži najmanje jednu **sliku**, najmanje jednu **tabelu** i najmanje **sedam referenci** u spisku literature. Najmanje jedna slika treba da bude originalna i da predstavlja neke podatke koje ste Vi osmislili da treba da prezentujete u svom radu. Isto važi i za najmanje jednu tabelu. Od referenci, neophodno je imati bar jednu **knjigu**, bar jedan **naučni članak** iz odgovarajućeg časopisa i bar jednu adekvatnu **veb adresu**.

Dužina seminarskog rada treba da bude od 10 do 12 strana. Svako prekoračenje ili potkoračenje biće kažnjeno sa odgovarajućim brojem poena. Eventualno, nakon strane 12, može se javiti samo tekst poglavlja **Dodatak** koji sadrži nekakav dodatni kôd, ali je svakako potrebno da rad može da se pročita i razume i bez čitanja tog dodatka.

Ко жели, може да пише рад ћирилицом. У том случају, неопходно је да су инсталирани одговарајући пакети: texlive-fonts-extra, texlive-latex-extra, texlive-lang-cyrillic, texlive-lang-other.

Nemojte koristiti stari način pisanja slova, tj ovo:

```
\v{s} i \v{c} i \'c ...
```

Koristite direknto naša slova:

```
š i č i ć ...
```

3 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) Koraci pri debagovanju: uočavanje da postoji greška razumevanje greške lociranje greške ispravljanje greške Često je najteži deo ispravno razumevanje i rano otkrivanje greške, kada se greška locira, ispravljanje najčešće nije veliki problem.

3.1 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.

3.2 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.
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.
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".

3.3 Osnovna pravila debagovanja

Neka od osnovnih pravila debagovanja su:

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

3.4 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.

3.5 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.

3.6 Lista debagera

Najčešće korišćeni debugeri:

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

Mejnfrejm debugeri:

1. XPEDITER
2. z/XDC
3. IBM Oliver
4. CA/EZTEST

4 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 [4]:

- 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 [8] (preko ugnježdenog interpretatora (eng. embedded interpreter) i omogućavanje REPL (Read-Evaluate-Print-Loop) funkcija za programske jezike zajedno sa mogućnostima debagovanja [2].

4.1 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 [9]:

Primer 4.1

```
<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 [3]. 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 [4][9].

`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 [3]).

5 Engleski termini i citiranje

Na svakom mestu u tekstu naglasiti odakle tačno potiču informacije. Uz sve novouvedene termine u zagradi naglasiti od koje engleske reči termin potiče.

Naredni primeri ilustruju način uvođenja enlegskih termina kao i citiranje.

Primer 5.1 *Problem zaustavljanja (eng.* halting problem*) je neodlučiv.*

Primer 5.2 *Za prevođenje programa napisanih u programskom jeziku C može se koristiti GCC kompajler.*

Primer 5.3 *Da bi se ispitivala ispravost softvera, najpre je potrebno precizno definisati njegovo ponašanje.*

Reference koje se koriste u ovom tekstu zadate su u datoteci *seminarski.bib*. Prevođenje u pdf format u Linux okruženju može se uraditi na sledeći način:

```
pdflatex TemaImePrezime.tex
bibtex TemaImePrezime.aux
pdflatex TemaImePrezime.tex
pdflatex TemaImePrezime.tex
```

Prvo latexovanje je neophodno da bi se generisao *.aux* fajl. *bibtex* proizvodi odgovarajući *.bbl* fajl koji se koristi za generisanje literature. Potrebna su dva prolaza (dva puta pdflatex) da bi se reference ubacile u tekst (tj da ne bi ostali znakovi pitanja umesto referenci). Dodavanjem novih referenci potrebno je ponoviti ceo postupak.

Broj naslova i podnaslova je proizvoljan. Neophodni su samo Uvod i Zaključak. Na poglavlja unutar teksta referisati se po potrebi.

Primer 5.4 *U odeljku* **??** *precizirani su osnovni pojmovi, dok su zaključci dati u odeljku 9.*

Još jednom da napomenem da nema razloga da pišete:

```
\v{s} i \v{c} i \'c ...
```

Možete koristiti srpska slova

```
š i č i ć ...
```

6 Slike i tabele

Slike i tabele treba da budu u svom okruženju, sa odgovarajućim naslovima, obeležene labelom da koje omogućava referenciranje.

Primer 6.1 *Ovako se ubacuje slika. Obratiti pažnju da je dodato i*

```
\usepackage{graphicx}
```

Na svaku sliku neophodno je referisati se negde u tekstu. Na primer, na slici **??** *prikazane su pande.*

Primer 6.2 *I tabele treba da budu u svom okruženju, i na njih je neophodno referisati se u tekstu. Na primer, u tabeli 2 su prikazana različita poravnanja u tabelama.*

Tabela 2: Razlčita poravnanja u okviru iste tabele ne treba koristiti jer su nepregledna.

centralno poravnanje	levo poravnanje	desno poravnanje
a	b	c
d	e	f

7 Kôd i paket listings

Za ubacivanje koda koristite paket **listings**: https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings

Primer 7.1 *Primer ubacivanja koda za programski jezik Python dat je kroz listing 1. Za neki drugi programski jezik, treba podesiti odgvarajući programski jezik u okviru defnisanja stila.*

```
# This program adds up integers in the command line
import sys
try:
    total = sum(int(arg) for arg in sys.argv[1:])
    print 'sum =', total
except ValueError:
    print 'Please supply integer arguments'
```

Listing 1: Primer ubacivanja koda u tekst

8 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)

	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 1: LLDB, GDB, Visual Studio Debugger [1][5][7]

8.1 Poređenje: GDB i LLDB

Debuger GDB predstavlja standard za GNU sisteme (ne striktno samo za GNU) [1]. 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 [6]. Način korišćenje ova dva debagera je veoma sličan i skup komandi se većinom poklapa. Postoji zvaničan rečnik koji prevodi komande iz GDB u LLDB [4]. Novije verzije GDB podržavaju MacOS, ali u proteklih par godina se pretežno koristio LLDB kao glavni debager za MacOS.

8.2 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 [7].

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

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

[3] 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.

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

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

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

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

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

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

A Dodatak

Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe.