Razvoj programa za x86_64 arhitekturu
(uz korišćenje dibagera gdb)

Sadržaj

- Uspostavljanje radnog okruženja
- Prevođenje C programa
- Izvršavanje programa
- Izvršavanje pod kontrolom dibagera
- Programiranje na asembleru
- Debagovanje asemblerskog programa

Uvod

- Hello World na višem programskom jeziku (C):

hello.c

```
   #include <stdio.h>

int main () {
   int i;
   for (i=1; i<=3; i++) {
       printf("Zdravo svete %d. put\n", i);
   }
   return 0;
}
```

Uspostavljanje radnog okruženja

- Radi se na Ubuntu 18.04 LTS operativnom sistemu koji se može instalirati i iz Windowsa kao Oracle VirtualBox kao virtuelna mašina:
- Otvoriti terminal i zadati sledeće komande:
  - Instalirati Linaro gnu alate:
    $ sudo apt install build-essential
  - Instalirati odgovarajući dibager:
    $ sudo apt install gdb

Prevođenje C programa

- Napraviti fajl hello.c sa izvornim programom:

  $ nano hello.c

  (izlazak uz snimanje izmena sa ctrl-x)

- Prevođenje programa (otkucati u terminalu):

  $ gcc hello.c –o hello

  (ulazni fajl je hello.c a izlazni fajl sa mašinskim kodom se zove hello, tako je zadato opcijom –o)

- Provera da li smo stvarno dobili izvršni fajl (fajl komanda ispisuje tip fajla):

  $ file hello

  hello: ELF 64-bit LSB shared object, x86-64, ...

  (ELF je izvršni tip fajla na Linuxu, slično .exe fajlu na windowsu)

Izvršavanje programa

- Pokretanje programa bez dibagera
  (ne može samo hello jer bi Linux mislio da želimo ugrađenu komandu i ne bi tražio u tekućem folderu):

```
$ ./hello
Zdravo svete 1. put
Zdravo svete 2. put
Zdravo svete 3. put
```

Izvršavanje pod kontrolom dibagera

- Debugger je alat koji služi za pomoć pri otklanjanju grešaka u programu
- Debugger omogućava izvršavanje programa korak po korak, prekid izvršavanja programa u unapred definisanoj tački (breakpoint), pregledanje vrednosti lokalnih promenljivih, itd.
- Prevođenje programa sa ugrađivanjem debuggerskih informacija u izvršni program (-g):

```
$ gcc –g hello.c –o hello
```

Izvršavanje pod kontrolom dibagera

- Otvoriti dva prozora terminala – u prvom će se prikazivati ulaz/izlaz programa, u drugom zadajemo komande dibageru
- U prvom prozoru proveravamo id prozora:

  $ tty

  /dev/pts/1

- Izvršavanje programa (iz foldera gde je hello fajl)

  $ gdb hello –tty /dev/pts/1

GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git

Copyright (C) 2018 Free Software Foundation, Inc.

. . . . .

Reading symbols from hello...done.

(gdb)

- Dobija se komandni prompt dibagera gde se mogu zadavati komande

Izvršavanje pod kontrolom dibagera

- Zadajemo breakpoint na početak main funkcije i komandu run da se program počne izvršavati

  (gdb) **break main**

  Breakpoint 1 at 0x652: file hello.c, line 5.

  (gdb) **run**

  Starting program: /home/db/temp/hello

  Breakpoint 1, main () at hello.c:5

  5 for (i=1; i<=3; i++) {

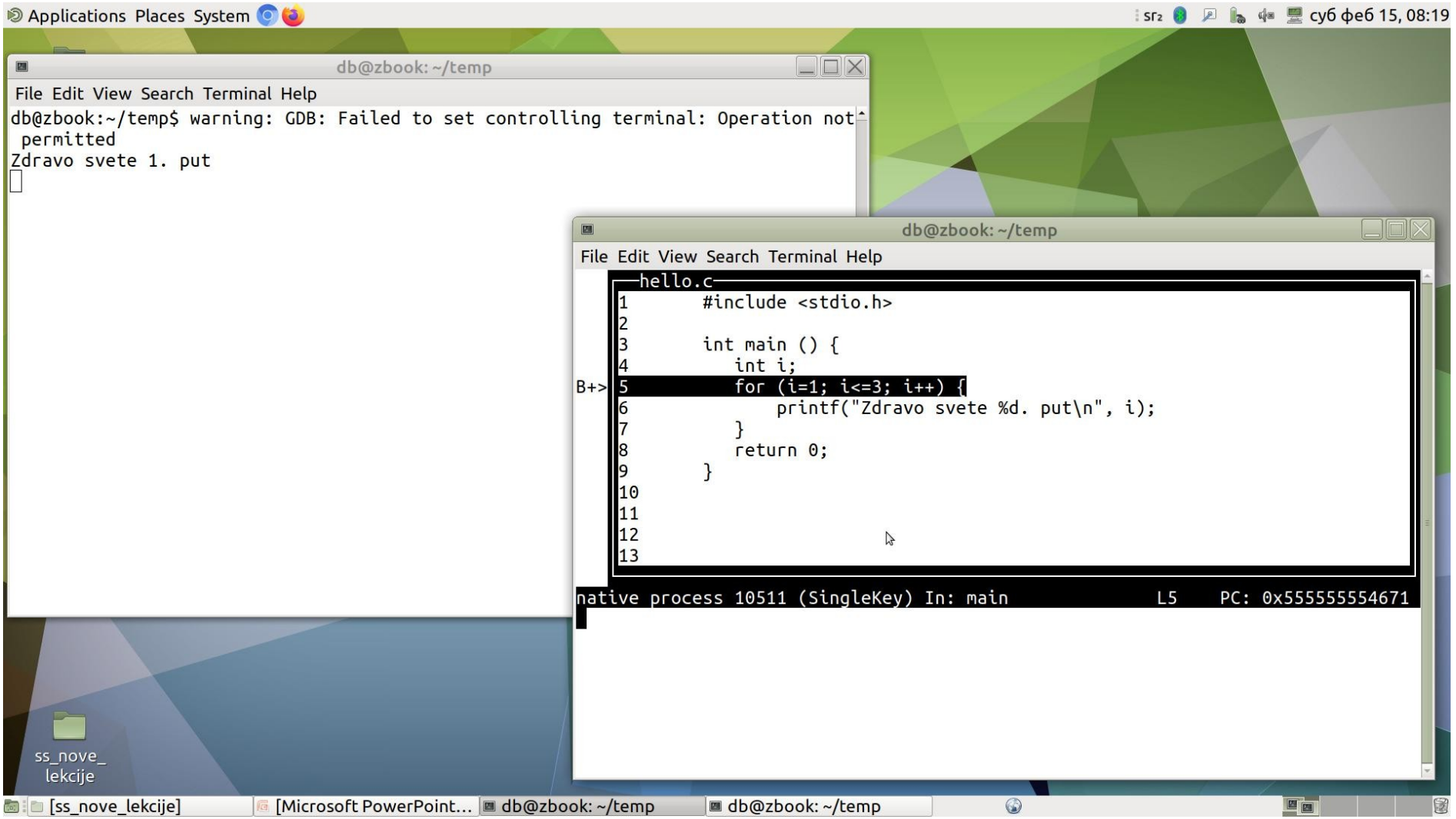
  (gdb)

- Izvršavanje je pauzirano na liniji 5, što je prva izvršna linija funkcije main()

Izvršavanje pod kontrolom dibagera

- Najzodnije je koristiti tui režim gdba, koji možemo aktivirati sa ctrl-x 1
- Dodatno, da ne bismo kucali komande u (gdb) promptu, možemo aktivirati single key režim sa ctrl-x s
- Sada možemo upravljati gdb-om pojedinim tasterima sa tastature

Izvršavanje pod kontrolom dibagera (tui režim, single key)



Gdb tui single key komande

- s – **step** izvršavanje programa za jednu liniju uz ulaženje u pozvane funkcije
- n – **next** slično kao step, ali se ne ulazi u pozvane funkcije, nego se poziv izvrši kompletno u jednom next koraku
- v – **info locals**, prikaz vrednosti promenljivih
- f – **finish**, izvršavanje do izlaska iz tekuće funkcije
- c – **continue**, izvršavanje “punom brzinom” do sledeće prekidne tačke ili kraja programa
- q – izlaz iz single key režima, dobija se (gdb) prompt

Neke korisne (gdb) komande

- Promena vrednosti promenljive i

  (gdb) set var i = 4

- Ispis vrednosti promenljive

  (gdb) print i

  $1 = 4

  - ispisivanje na kom mestu je prekinuto izvršavanje programa (w u single key režimu)

  (gdb) **where**

Kraj izvršavanja programa

- Posle nekog continue, gdb će ispisati:

  [Inferior 1 (process 9535) exited normally]
- Izlaz iz gdb: **quit**
- U drugom terminalu vidi se izlaz programa:

  Zdravo svete 1. put

  Zdravo svete 2. put

  Zdravo svete 3. put

Programiranje na asembleru

```
 .intel_syntax noprefix
    .text # direktno poziva sistemske servise linux kernela
    .globl _start
_start:
     # syscall write
     mov rax, 1 # redni broj poziva
     mov rdi, 1 # stdout
     mov rsi, offset HelloWorldString # poruka
     mov rdx, 12 # duzina poruke
     syscall     # obavlja poziv
     # syscall exit
     mov rax, 60 # redni broj poziva
     mov rdx, 0 # izlazni status programa
     syscall # obavlja poziv
     .section .rodata
HelloWorldString: .string "Hello World\n"
```

asmhello.s

Programiranje na asembleru

- Opis linux sistemskih servisa može se dobiti sa:

  https://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/
- Konvencija pozivanja iz x86_64 asemblera:
  - Redni broj poziva ide u rax (brojevi se mogu videti u /usr/arm-linux-gnueabi/include/asm/unistd.h)
  - Ostali argumenti idu u rdi,rsi, rdx, r10, r8 i r9.
  - Kernel se instrukcijom softverskog prekida syscall
  - Rezultat sistemskog poziva je u rax (0-sve u redu, <0 greška)
  - Sist. Pozivi ne čuvaju rcx ni r11.

Programiranje na asembleru

- Prevođenje:

  $ arm-linux-gnueabi-gcc -nostdlib -no-pie -g asmhello.s -o asmhello

- Dakle, samo se doda uputstvo kompajleru da ne linkuje C standardne biblioteke sa programom i ne pravi poziciono nezavisni kod
- Izvršavanje i debagovanje ide isto kao što je opisano za C program (samo treba **break _start**)

Debagovanje asemblerskog programa

- Korisno je u tui režimu otvoriti dodatan prozor koji ispisuje vrednosti procesorskih registara

  (gdb) **layout regs**

  otvara novi prozor (osim izvornog programa) u kome prikazuje vrednosti procesorskih registara
- Tekući prozor menja se sa **ctrl-x o**
- **winheight regs 20** zadaje visinu regs prozora (slično tome i za src prozor)
- Promena vrednosti registra (koristiti $ime registra):

  (gdb) set $rax = 5

Debagovanje asemblerskog programa