Dodatna tema

Ispitivanje programa i korekcija grešaka

Dibager (Debugger)

- Program pomoću kog je moguće pratiti izvršavanje ciljnog programa
- Veća kontrola izvršavanja programa
    - Izvršavanje programa "korak po korak"
    - Pregled stanja promenljivih u svakom trenutku

GNU Project Debugger (GDB)

Njegove mogućnosti:

- Pokretanje programa, uz prosleđivanje vrednosti koje utiču na njegovo izvršavanje
- Moguće zaustavljanje programa na osnovu navedenih uslova
- Ispitivanje uzroka prevremenog zaustavljanja programa
- Promene vrednosti "na živo", odnosno, uticanje na vrednosti promenljivih u okviru programa tokom izvršavanja

Potrebno je

1. Proveriti da li je GDB instaliran
    - Komanda: `which gdb`
    - Ukoliko se pojavi putanja poput `/usr/bin/gdb`, GDB je instaliran
    - Ukoliko nema ispisa, instalirati GDB
    - Komanda za Ubuntu sisteme (od verzije 16.04 dovoljno samo `apt`, umesto `apt-get`):

```
sudo apt-get -y install gdb
```

2. Kompajlirati program sa debug informacijama
    - Dodati flag `-g` prilikom kompajliranja

```
gcc -g -o program program.c
```

Primer 1

```c
#include <stdio.h>

int find_min(int *, int);

int main() {
    int a[] = {1, 3, -1, 5, 2};
    int b[] = {1, 3, -1, 5, 2, -2, -3};
    int c[] = {1, 3, 7, 5, 2};

    printf("Minimum niza a je: %d\n", find_min(a, 5));
    printf("Minimum niza b je: %d\n", find_min(b, 5));
    printf("Minimum niza c je: %d\n", find_min(c, 5));

    return 0;
}

```

```
17 int find_min(int *a, int n) {
18     int min = 0;
19 
20     int i;
21     for(i = 0;i <= n;i++) {
22         if(a[i] < min) {
23             min = a[i];
24         }
25     }
26 
27     return min;
28 }
29 
```

Ispis:

```
Minimum niza a je: -1
Minimum niza b je: -2
Minimum niza c je: 0
```

Prepoznavanje problema

Niz	Očekivana vrednost	Dobijena vrednost	Tačnost
[1, 3, -1, 5, 2]	-1	-1	✓
[1, 3, -1, 5, 2, -2, -3]	-3	-2	✗
[1, 3, 7, 5, 2]	1	0	✗

Kako odrediti uzroke nekorektnog ponašanja programa?

Kompajliranje i pokretanje u dibageru

1. Kompajliranje programa sa debug informacijama

```
gcc -g -o buggy-program buggy-program.c
```

2. Pokretanje dibagera za određeni program

```
gdb buggy-program
```

3. Pokretanje programa u okviru dibagera

```
(gdb) run
```

4. Izlazak iz dibagera

```
(gdb) quit
```

Napomena: Kod 3. i 4. `(gdb)` je oznaka koju ispisuje dibager kako bi dao korisniku do znanja da koristi njegovu konzolu

Kontrola izvršavanja programa

- Da se program ne bi samo izvršio, potrebno je postaviti barem jedan breakpoint

```
(gdb) break 10
```

- U slučaju uspešno postavljenog breakpoint-a, ispisuje se poruka

```
Breakpoint 1 at 0x400624: file buggy-program.c, line 10.
```

- Nakon postavljanja breakpoint-a, ponovo pokrenuti program
- Program bi trebalo da se zaustavi u funkciji main, na 10. liniji

```
(gdb) run
Starting program: /home/student/buggy-program
Breakpoint 1, main () at buggy-program.c:10
10          printf("Minimum niza a je: %d\n", find_min(a, 5));
```

Kontrola izvršavanja programa

- Pomeranje na sledeću naredbu, komanda `next`

```
(gdb) next
Minimum niza a je: -1
11          printf("Minimum niza b je: %d\n", find_min(b, 5));
```

- Pomeranje na sledeću naredbu, komanda `step`

```
(gdb) step
find_min (a=0x7fffffffdad0, n=5) at buggy-program.c:18
18          int min = 0;
```

- Razlika između `step` i `next`
  - Ukoliko je na trenutnoj liniji poziv funkcije, `step` ulazi u funkciju
  - `next` ne ulazi u funkciju, već nastavlja na sledeću liniju
  - Napomena: izbor komandi ne utiče na izvršavanje programa

Kontrola izvršavanja programa

- Postavljanje novog breakpoint-a, na liniju 22

```
(gdb) break 22
```

- Komanda `continue` nastavlja izvršavanje programa do sledećeg breakpoint-a

```
(gdb) continue
Continuing.

Breakpoint 2, find_min (a=0x7fffffffd9d0, n=5) at buggy-program.c:22
22              if(a[i] < min) {
```

- Uraditi `continue` 5 puta

Ispisivanje vrednosti promenljivih

- Ispis vrednosti pomoću komande `print`

```
(gdb) print i
$15 = 5
(gdb) print a[i]
$16 = -2
```

- Trik za ispisivanje celokupnog niza

```
(gdb) print * a@7
$18 = {1, 3, -1, 5, 2, -2, -3}
```

- Moguće je ispisivati polja, kao i celokupne strukture
- Moguće je koristiti dereferenciranje prilikom ispisa vrednosti (`*` i `->`)
- Pitanje: Da li moguće zaključiti zašto funkcija `find_min` ne radi dobro za niz `b`?

Upravljanje breakpoint-ima

- Disable svih breakpoint-ova i enable breakpoint-a 1

```
(gdb) disable breakpoints
(gdb) enable 1
```

- Brisanje

```
(gdb) delete 1
```

- Pregled postavljenih breakpoint-a

```
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
2       breakpoint     keep n   0x00000000004006c0 in find_min at buggy-program.c:22
        breakpoint already hit 12 times
```

Ostale korisne komande

- Postavljanje breakpoint-a na funkciju

```
(gdb) break find_min
```

- Breakpoint sa uslovom

```
(gdb) break 22 if i==5
```

- Praćenje promena stanja promenljive (čitanje i pisanje)
  - Napomena: Promeljiva mora biti u opsegu u trenutku praćenja

```
(gdb) watch a
```

Ostale korisne komande

- Ispisivanje stanja steka funkcija u trenutnoj liniji izvršavanja

```
(gdb) backtrace
#0  find_min (a=0x7fffffffda10, n=5) at buggy-program.c:22
#1  0x0000000000400657 in main () at buggy-program.c:11
```

- Odabir stek frejma (funkcije, radi pregleda promenljivih)

```
(gdb) frame 1
#1  0x0000000000400657 in main () at buggy-program.c:11
11          printf("Minimum niza b je: %d\n", find_min(b, 5));
```

- Ispisivanje tipa promenljive

```
(gdb) whatis i
type = int
```

Ostale korisne komande

- Izlazak iz trenutne funkcije

```
(gdb) finish
```

- Postavljanje nove vrednosti promenljive

```
(gdb) set variable i = 2
```

- Prikaz liste svih komandi i njihovih kratkih objašnjenja

```
(gdb) help all
```

- Pojašnjenje pojedinačne komande

```
(gdb) help break
```

Zadatak 1

Pomoću komandi GDB alata pronaći razlog zašto se ne ispisuje odgovarajući minimum niza c u primeru 1. Ispraviti sve pronađene greške u programu.