Siluri kasutamisest

Kompileerimine

Kompileerimine koos silumisinfoga:

C	`gcc -o` *prog* `-g -Wall` *prog*`.c`
C++	`g++ -o` *prog* `-g -Wall` *prog*`.cpp`
Pascal	`fpc -g` *prog*`.pas`

Siluri lühikonspekt

Programmi laadimine silumiseks: `gdb` *prog*

Programmi laadimine *core* faili lahkamiseks: `gdb -c core` *prog*

Siluri sees tähtsamad käsud:

`quit`	väljub silurist
`run`	käivitab programmi, programm jookseb esimese katkestuspunkti või veani
`break` *xxx*	määrab katkestuspunkti, *xxx* võib olla funktsiooni nimi või reanumber
`delete break` *x*	kustutab katkestuspunkti number *x*
`start`	sama, mis `break main` + `run`
`cont`	jätkab programmi täitmist pärast katkestust
`step`	täidab programmi ühe rea, läheb alamprogrammide sisse
`next`	täidab programmi ühe rea, ei lähe alamprogrammide sisse
`print` *xxx*	väljastab avaldise *xxx* väärtuse
`disp` *xxx*	väljastab avaldise *xxx* väärtuse iga kord kui programm peatub
`undisp` *x*	loobub avaldise number *x* väljastamisest
`watch` *xxx*	programm katkestatakse, kui *xxx* väärtust muudetakse
`rwatch` *xxx*	programm katkestatakse, kui *xxx* väärtust vaadatakse
`bt`	väljastab kutsemagasini sisu, sisuliselt vastab küsimusele “kuidas me siia sattusime?”; eriti kasulik veakoha leidmiseks *core* faili lahkamisel
`help`	spikri teemade loetelu
`help` *xxx*	spikker käsu või teema *xxx* kohta

Põhjalikum manuaal: `http://www.gnu.org/software/gdb/documentation/`

Mälupiirangutest

Masina ülekoormamise vältimiseks on võimalik piirata mitmete ressursside kasutamist programmide poolt. Nende piirangute haldamiseks on kõige lihtsam kasutada `bash`i käsku `ulimit`.

`ulimit -a` näitab hetkel kehtivaid piiranguid. Sellest komplektist tasub esimeses järjekorras tähele panna `-c`, `-v`, `-s`.

Kui `-c` on 0, siis programmi kokkujooksmisel *core* faili ei tehta ja järelikult ei saa seda faili ka siluriga lahata. Soovitan oma lahenduste testimise ajaks panna `ulimit -c unlimited`.

Ülejäänud kaks on mälumahtude piirangud. Meie testimismasinas on pandud `ulimit -v 65536` ja `ulimit -s 8192` — see tähendab, et programmile lubatakse magasini (sinna lähevad kõigi funktsioonide parameetrid ja lokaalsed muutujad) kuni 8 MB ja kogu mälu koodile, andmetele ja magasinile kokku kuni 64 MB.

Lahkamisest: C ja C++

C ja C++ kasutajatel on üldiselt võimalik saada täpne veakoht automaatselt kätte alles siis, kui viga on juba nii suur, et opsüsteem asjasse sekkub.

Väike näide:

```
morph@Morphix:~$ ./x
Segmentation fault (core dumped)

morph@Morphix:~$ gdb -c core x
Core was generated by `./x'.
Program terminated with signal 11, Segmentation fault.
#0  0x08048364 in f () at x.c:3
3               *p = 0;
(gdb) bt
#0  0x08048364 in f () at x.c:3
#1  0x08048381 in main () at x.c:6
(gdb) list x.c:3
1       void f() {
2               int *p = 0;
3               *p = 0;
4       }
5       int main() {
6               f();
7               return 0;
8       }
(gdb)
```

Magasini väljatrükist on näha, et viga tekkis faili `x.c` real 3 funktsioonis `f()`, mida kutsuti välja faili `x.c` realt 6 funktsioonist `main()`, ja listingust on näha, et süüdi on tühiviida kasutamine.

Lahkamisest: Pascal

Pascali kasutajatel on võimalik tellida mitemesuguseid täpsemaid veakontrolle.

Näiteks ületäitumise ja massiivi indeksite piiridest väljumise korral vea saamiseks tuleks programmi algusesse panna kommentaar {`$R+`} (selle tähendus on *range check ON*). Sellise kontrolliga programm lõpetab veateatega kohe, kui vastav viga tekib.

Kuna programm lõpetab oma töö ise, mitte opsüsteemi veaga, siis *core* faili ei teki. Veakoha leidmiseks tuleks enne programmi siluri all käivitamist panna katkestuspunkt funktsioonile `SYSTEM_DO_EXIT` (see on protseduur, mida FreePascal ise kasutab, et programmist väljuda) ja siis vaadata programmi kutsemagasini.

Väike näide:

```
morph@Morphix:~$ ./x
Runtime error 201 at $080480D1
  $080480D1
  $080480E5

morph@Morphix:~$ gdb x
(gdb) break SYSTEM_DO_EXIT
Breakpoint 1 at 0x8053f0f
(gdb) run
Starting program: /home/morph/x
Breakpoint 1, 0x08053f0f in SYSTEM_DO_EXIT ()
(gdb) bt
#0  0x08053f0f in SYSTEM_DO_EXIT ()
#1  0x08053f45 in SYSTEM_HALT$BYTE ()
#2  0x08054013 in SYSTEM_HANDLEERRORADDRFRAME$LONGINT$POINTER$POINTER ()
#3  0x08054058 in SYSTEM_HANDLEERRORFRAME$LONGINT$POINTER ()
#4  0x08053aa0 in fpc_rangeerror ()
#5  0x080480d1 in P () at x.pas:6
#6  0x080480e5 in main () at x.pas:9
(gdb) list x.pas:6
1       {$R+}
2       procedure p;
3       var i : integer;
4       begin
5               i := 1000;
6               i := i * 1000;
7       end;
8       begin
9               p;
10      end.
(gdb)
```

Magasini väljatrükist on näha, et viga tekkis faili `x.pas` real 6 protseduuris `p`, mida kutsuti välja faili `x.pas` realt 9 põhiprogrammist, ja listingust on näha, et selle põhjustas ületäitumine korrutamisel.