

Alat komandne linije *strace*

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Srđan Lazarević, 1096/2018
slazarevic1995@gmail.com

15. januar 2020

Sažetak

Da li ste se ikada zapitali kako korisnički programi interaguju sa operativnim sistemom, ili koji se tačno sistemski pozivi i signali izvrše u pozadini? Ukoliko je odgovor na prethodno pitanje pozitivan, ovaj rad će vam pružiti osnovne informacije i dati uvid u to šta se dešava ispod haube. U okviru rada biće predstavljen, kroz kôd i praktične primere, alat komandne linije *strace*, podržan na svim *Unix* i njemu sličnim operativnim sistemima, pomoću koga možemo dobiti odgovor na pitanja sa početka sekcije, odnosno pratiti izvršavanje programa i njegovu interakciju sa kernelom operativnog sistema.

Sadržaj

1	Uvod	2
2	Alat <i>strace</i> ispod haube	2
2.1	Sistemski poziv <i>ptrace</i>	2
2.2	Alat <i>strace</i> kao omotač	2
3	Formatiranje i filtriranje izlaza	3
4	Zdravo <i>strace</i>	4
5	Doprinos debugovanju	4
6	Zaključak	5
	Literatura	5

1 Uvod

Alat *strace*, karakterističan za *linux* i njemu slične operativne sisteme, predstavlja program komandne linije uz čiju pomoć možemo pratiti izvršavanje drugog procesa, odnosno njegovu interakciju sa kernelom. Ime alata *strace* potiče od termina sistem (eng. *system*) i tragač (eng. *tracer*), odnosno tragač sistemskih poziva i signala.

Alat se najviše koristi od strane sistem administratora i developera u oblasti programiranja uređaja sa ugrađenim računarom (eng. *embedded devices*), za potrebe debugovanja u slučajevima kada izvorni kôd nije dostupan ili nismo u mogućnosti da ponovo kompajliramo program [8].

```
$ strace [OPTIONS] args
```

Listing 1: Primer upotrebe alata *strace*

U najjednostavnijem slučaju upotrebe, alat *strace* će pokrenuti program koji mu je zadat kao argument. Svi sistemski pozivi od strane zadatog programa, kao i signali koje program prima će biti presretnuti i zabeleženi od strane alata i biti prikazani u redosledu izvršavanja [5].

2 Alat *strace* ispod haube

Da bi razumeli kako alat *strace* radi, moramo zapravo razumeti rad sistemskog poziva *ptrace*, na čijem interfejsu *strace* zapravo počiva.

2.1 Sistemski poziv *ptrace*

Sistemski poziv *ptrace*, iz zaglavlja *sys/ptrace.h*, pruža mogućnost da jedan proces posmatra i kontroliše izvršavanje drugog procesa, kao i da čita i menja memoriju i registre procesa koji kontroliše [6, 3].

Pored svih drugih pogodnosti koje ima, najvažnija pogodnost, u kontekstu alata *strace* (takođe i *ltrace* [2]), koju *ptrace* pruža je posmatranje procesa koji se izvršava.

2.2 Alat *strace* kao omotač

Operativni sistem se može grubo podeliti na dva načina rada:

- Način rada kernela (eng. *kernel mode*) - predstavlja interni način rada kernela operativnog sistema, u kojem su implementirani svi sistemski pozivi, koje kernel pruža kao interfejs
- Korisnički način rada (eng. *user mode*) - predstavlja način rada u kojem se izvršavaju sve korisničke aplikacije

Funkcije koje predstavljaju sistemske pozive su veoma slične regularnim korisničkim funkcijama, odnosno rade tako što im se proslede argumenti i imaju povratnu vrednost. Jedina razlika u tome je što sistemski pozivi pristupaju kernelu operativnog sistema, dok to nije slučaj sa regularnim korisničkim funkcijama.

Prebacivanje iz kernel u korisnički način rada, i obratno, je realizovano preko mehanizma zamki (end. *trap*), implementiranog na nivou operativnog sistema [7].

Alat *strace* kao omotač oko sistemskog poziviva *ptrace*, koristi upravo mogućnost da se detektuje svaki prelazak iz korisničkog u kernel način rada, i obratno, odnosno realizovanje mehanizma zamke.

Alat *strace* se pomoću *ptrace* nakači na proces koji mu je zadat kao argument, tada počinje praćenje izvršavanje procesa i čeka se trenutak poziva mehanizma zamke. Kada se mehanizam zamke pozove, mehanizam pruža informaciju koji sistemski poziv je izvršen, i to sa kojim argumentima je pozvan i koja je njegova povratna vrednost. Upravo tu informaciju za svaki sistemski poziv koji se desi u toku rada korisničkog programa *strace* pruža kao povratnu informaciju onome ko ga koristi.

Sličan je metod pomoću kojeg se detektuju sistemski signali [7].

Jedan poprilično jednostavan način implementacije alata *strace*, sa minimalnim setom funkcionalnosti, može se pronaći na github [stranici](#).

3 Formatiranje i filtriranje izlaza

Svaka linija koja predstavlja izlaz alata *strace* sastoji se od imena sistemskog poziva, praćenog argumentima sa kojima je pozvan i povratnom vrednošću. Jedna takva linija izgleda bi (u slučaju sistemskog poziva):

```
open("/dev/null", O_RDONLY) = 3
```

Listing 2: Format ispisa sistemskog poziva

U slučaju neuspešnog izvršavanja (ukoliko je povratna vrednost -1), dodatno *errno* simbol i string reprezentaciju simbole greške.

```
open("/foo/bar", O_RDONLY) = -1 ENOENT (No such file or directory)
```

Listing 3: Format ispisa u slučaju greške

Signali koje prima program su prikazani kao par simbola signala i *siginfo* strukture.

```
sigsuspend([] <unfinished ...>
--- SIGINT {si_signo=SIGINT, si_code=SI_USER, si_pid=...} ---
+++ killed by SIGINT +++
```

Listing 4: Format ispisa signala

Informacije o izvršavanju sistemskih poziva i signala, dobijene pomoću alata *strace*, možemo formatirati i filtrirati na različite načine, kao što su:

- **-a kolona** - poravnavanje povratnih vrednosti sistemskih poziva na određenu liniju (podrazumevano je 40).
- **-o fajl** - upisuje izlaz u zadati fajl umesto na *stderr*
- **-c** - prikazuje vreme, pozive i greške za svaki sistemski poziv i ispisuje izveštaj na kraju izvršavanja programa
- **-e trace=sisPoziv** - prati izvršavanje samo zadatih sistemskih poziva, na primer **-e trace=open,read,write**
- **-e status=status** - prati izvršavanje samo sistemskih poziva sa zadatim statusom. Podrazumevano *all*, mogući statusi: *successful*, *failed*, *unfinished*, *detached*, *unavailable*
- **-T** - prikazuje vreme potrebno za izvršavanje svakog sistemskog poziva

- `-d` - prikazuje debug informacije samog alata *strace*

Vredi napomenuti da izlazne informacije alata *strace* ispisuje na standardni izlaz greške (eng. *stderr*), dok se regularni ispis programa ispisuje na standardni izlaz (eng. *stdin*) (ukoliko nije navedeno drugačije).

Ostale informacije o filtriranju, formatiranju, kao i o načinima upotrebe alata *strace* možemo pronaći u okviru *man* strana [5].

4 Zdravo *strace*

Posmatrajmo ponašanje alata *strace* na primeru jednostavnog *Zdravo strace* programa:

```
$ cat zdravo_strace.cpp
#include <iostream>

int main(int argc, char const *argv[])
{
    std::cout << "Zdravo strace!" << std::endl;
    return 0;
}
$ g++ zdravo_strace.cpp -o zdravo_svete
```

Listing 5: *zdravo_strace.cpp*

Ukoliko bi naš jednostavni program *zdravo_strace* pokrenuli kroz *strace*, dobili bi sledeće informacije o izvršavanju sistemskih poziva:

```
$ strace ./zdravo_svete > log.txt
execve("./zdravo_svete", ["/zdravo_svete"], 0x7ffde57199d0 /* 59
    vars */) = 0
...
write(1, "Zdravo strace!\n", 14)          = 14
exit_group(0)                          = ?
+++ exited with 0 +++
```

Listing 6: *strace zdravo_strace*

Znajući da ispis programa ide na standardni izlaz (na osnovu poglavlja 3), možemo potražiti ispis u okviru *log.txt* datoteke.

```
$ cat log.txt
Zdravo strace!
```

Listing 7: Ispis programa

5 Doprinos debugovanju

Alat *strace* posebno dolazi do izražaja prilikom debugovanja problema na razvojnim pločama ili pri radu sa bibliotekama koje nije pisao sam developer, integrator ili sistem administrator. U nedostatku grafičkog okruženja, *strace* je često jedini alat, uz *gdb* [1] naravno, pomoću kojeg se može pratiti izvršavanje nekog procesa ili kompletnog sistema. Na primer, možemo se konektovati na razvojnu ploču pomoću *ssh* [4], i zatim zakačiti se na izvršavanje željenog procesa.

Alat *strace* nikako ne treba koristiti kao alat u produkcionom kodu, naime *strace* nakon svakog sistemskog poziva ili signala stopira izvršavanje

procesa, što može dovesti do velikog usporenja kako procesa tako i čitavog sistema, u slučaju da rad sistema zavisi od *strace*-ovanog procesa.

Na prostom primeru pozivanja *ls* sa i bez alat *strace* možemo uočiti značajan pad performansi u izvršavanju:

```
$ time ls > /dev/null
real    0m0,005s
user    0m0,005s
sys     0m0,000s
$ time strace ls > /dev/null
real    0m0,015s
user    0m0,010s
sys     0m0,006s
```

Listing 8: Pad performansi izvršavanja

6 Zaključak

U doba ekspanzije i pre svega otkrivanja novih oblasti razvoja softvera, pojavljuju se novi problemi, koje do sada nismo sretali. Alati komandne linije, koji se godinama razvijaju, postaju sve moćnije i moćnije sredstvo u prevenciji novonastalih problema koji potencijalno mogu imati fatalne posledice.

Alat *strace* i drugi alati komandne linije, pružaju mogućnost skroz novog pristupa problemima, iz skroz drugog ugla. Taj ugao može biti podjednako značajan sa strane industrijskog programiranja, kao i sa strane edukacije, gde alati postaju sredstvo koje pomaže u razumevanju kako nešto zaista radi ispod haube.

Literatura

- [1] gdb man pages. on-line at: <http://man7.org/linux/man-pages/man1/gdb.1.html>.
- [2] ltrace man page. on-line at: <http://man7.org/linux/man-pages/man1/ltrace.1.html>.
- [3] ptrace man page. on-line at: <http://man7.org/linux/man-pages/man2/ptrace.2.html>.
- [4] ssh man pages. on-line at: <http://man7.org/linux/man-pages/man1/ssh.1.html>.
- [5] strace man page. on-line at: <http://man7.org/linux/man-pages/man1/strace.1.html>.
- [6] Nikola Dimitrijevic. Sistemski poziv ptrace i njegova uloga u radu debagera. on-line at: http://www.verifikacijasoftware.matf.bg.ac.rs/vs/predavanja/03_dinamicka_analiza/02_NikolaDimitrijevic_Ptrace.pdf.
- [7] Red Hat. Understanding system calls on Linux with strace. on-line at: <https://opensource.com/article/19/10/strace>.
- [8] Strace community. Strace conferences proceedings. on-line at: <https://github.com/strace/strace-talks>.