

SCR - systemy operacyjne

# Sprawozdanie z listy IV

Tworzenie i operacje na procesach



Politechnika  
Wrocławska

17-11-2020

Paweł Niedziółka

# Spis treści

<b>1</b>	<b>Polecenie strace</b>	<b>1</b>
<b>2</b>	<b>Wykorzystanie programu strace do śledzenia wykonywania się programu</b>	<b>5</b>
2.1	Przeanalizuj wykonanie się programu wyświetlającego napis Hello world na ekranie . . . . .	5
2.2	Wykorzystaj program strace do znalezienia wszystkich plików konfiguracyjnych, jakie powłoka próbuje odczytać przy starcie . . . . .	6
2.3	Sprawdź czy plik edytowany w programie pico jest stale otwarty . . . . .	7
2.4	Odczytaj jakie file deskryptory posiada uruchomiona aplikacja wyświetlająca napis Hello world na ekranie . . . . .	8
<b>3</b>	<b>Szukanie błędu w programie przy użyciu strace</b>	<b>9</b>

# 1. Polecenie strace

Zapoznano się z instrukcją polecenia strace (polecenie truss na Solarisie). I wybrano następujące funkcje programu:

- Podłączanie się do istniejącego już procesu  
Polecenie strace można wywołać na dwa sposoby - pierwszy to wywołanie polecenia i programu jednocześnie. Natomiast istnieje możliwość podglądu wywołań poleceń systemowych przez proces który już istnieje. Wywołuje się go w następujący sposób:

sudo strace -p PID

```
write(2, "\n", 1) = 1
lwp_sigmask(SIG_SETMASK, 0x00000002, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
ioctl(0, TCSETSW, 0x000E2EDC) = 0
lwp_sigmask(SIG_SETMASK, 0x00000000, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
sigaction(SIGINT, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGTERM, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGQUIT, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGALRM, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGTSTP, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGTTOU, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGTTIN, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGWINCH, 0xFFBFF028, 0xFFBFF0C8) = 0
sigaction(SIGINT, 0xFFBFF068, 0xFFBFF108) = 0
sigaction(SIGINT, 0xFFBFFB98, 0xFFBFFC38) = 0
time() = 1605636175
lwp_sigmask(SIG_SETMASK, 0x06820000, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
ioctl(255, TIOCGSID, 0xFFBFF064) = 0
getsid(0) = 11999
ioctl(255, TIOCSGRP, 0xFFBFF110) = 0
lwp_sigmask(SIG_SETMASK, 0x00000000, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
sigaction(SIGINT, 0xFFBFF068, 0xFFBFF108) = 0
lwp_sigmask(SIG_SETMASK, 0x00000002, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
ioctl(0, TIOCGWINSZ, 0xFFBFEF80) = 0
ioctl(0, TIOCSWINSZ, 0xFFBFEF80) = 0
Received signal #20, SIGWINCH [caught]
lwp_sigmask(SIG_SETMASK, 0x00080002, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
setcontext(0xFFBFE948)
ioctl(0, TCGETS, 0xFFBFF0C4) = 0
ioctl(0, TCSETSW, 0xFFBFF0C4) = 0
lwp_sigmask(SIG_SETMASK, 0x00000000, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
lwp_sigmask(SIG_SETMASK, 0x06802006, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
sigaction(SIGINT, 0xFFBFEF08, 0xFFBFEFA8) = 0
sigaction(SIGTERM, 0xFFBFEF08, 0xFFBFEFA8) = 0
sigaction(SIGTERM, 0xFFBFEFA8, 0xFFBFF048) = 0
sigaction(SIGQUIT, 0xFFBFEF08, 0xFFBFEFA8) = 0
sigaction(SIGQUIT, 0xFFBFEFA8, 0xFFBFF048) = 0
sigaction(SIGALRM, 0xFFBFEF88, 0xFFBFF028) = 0
sigaction(SIGTSTP, 0xFFBFEF08, 0xFFBFEFA8) = 0
sigaction(SIGTSTP, 0xFFBFEFA8, 0xFFBFF048) = 0
sigaction(SIGTTOU, 0xFFBFEF08, 0xFFBFEFA8) = 0
sigaction(SIGTTOU, 0xFFBFEFA8, 0xFFBFF048) = 0
sigaction(SIGTTIN, 0xFFBFEF08, 0xFFBFEFA8) = 0
sigaction(SIGTTIN, 0xFFBFEFA8, 0xFFBFF048) = 0
lwp_sigmask(SIG_SETMASK, 0x00000000, 0x00000000) = 0xFFBFEFF [0x0000FFFF]
sigaction(SIGWINCH, 0xFFBFEF08, 0xFFBFEFA8) = 0
diablo-bash-3.2$ write(2, " d i a b l o - b a s h -...", 17) = 17
read(0, 0xFFBFEF4F, 1) (sleeping...)
```

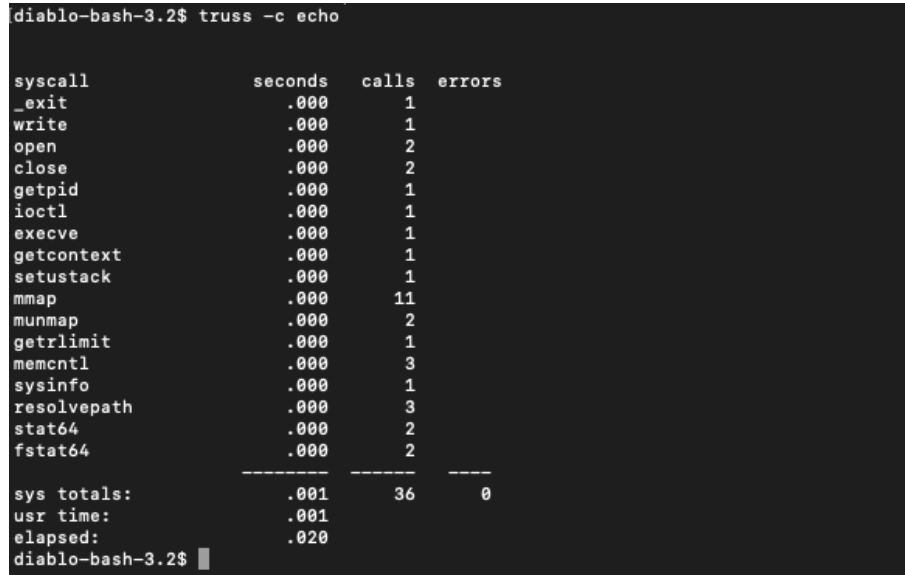
Rysunek 1.1: Podłączanie się do procesu

Aby polecenie działało poprawnie należy je wywołać jako superuser. Ta opcja według mnie jest bardzo ważna, ponieważ, gdyby nie ona, to nie moglibyśmy zobaczyć wywołań poleceń systemowych wszystkich procesów (nie mówię tu o tych w bashu, tylko np. działających jako system operacyjny).

- Zliczanie czasu wykonania i błędów

Drugą opcją, która według mnie jest ważna jest polecenie:

strace -c POLECENIE



syscall	seconds	calls	errors
_exit	.000	1	
write	.000	1	
open	.000	2	
close	.000	2	
getpid	.000	1	
ioctl	.000	1	
execve	.000	1	
getcontext	.000	1	
setustack	.000	1	
mmap	.000	11	
munmap	.000	2	
getrlimit	.000	1	
memcntl	.000	3	
sysinfo	.000	1	
resolvepath	.000	3	
stat64	.000	2	
fstat64	.000	2	
-----			
sys totals:	.001	36	0
usr time:	.001		
elapsed:	.020		

Rysunek 1.2: Wygenerowany raport

Generuje to listę poleceń systemowych, która jest posortowana malejąco pod względem czasu w nich spędzonego. Można też tam zobaczyć błędy i liczbę wystąpień wywołań. Może to być przydatna opcja, kiedy pojawi się problem z wydajnością - wówczas możemy ocenić co zżera tak dużo zasobów.

- Zapisanie logów

Przydatne też może być zapisanie logów na dysku - czasami potrafią być one bardzo długie i nieczytelne w terminalu, natomiast w pliku bardzo łatwo można szukać potrzebnych informacji:

strace -o PLIK POLECENIE

```

Last login: Tue Nov 17 19:05:31 on ttys001
pawel@MacBook-Air-Pawe ~ % ssh diablo.ict.pwr.wroc.pl -l pniedzio
[Password:
Last login: Tue Nov 17 19:05:37 2020 from gl131-32.master.pl

-----
Oracle Corporation      SunOS 5.10      Generic Patch      SUNW,Sun-Fire-880
*****
*
*                      D I A B L O
*                      Solaris
*
*      przeglądarki www:  w3m,lynx,firefox
*      klienci pocztowe:  mail,alpine,thunderbird
*      edytory tekstu:    vi,vim,nano,emacs,gedit
*      programy inżynierskie:  scip
*      narzędzia programistyczne: cc,gcc,git,svn
*      narzędzie do tworzenia dokumentów tekstowych: latex
*      pakiety biurowe:    soffice
*      czytniki pdf:      evince,acroread,xpdf
*      programy graficzne: gimp,librecad
*
*      W przypadku problemów prosimy pisać na adres:
*      admin@kcir.pwr.edu.pl
*      Aktualności: http://diablo.kcir.pwr.edu.pl
*****
diablo-bash-3.2$ truss -o echo.txt echo

diablo-bash-3.2$

```

Rysunek 1.3: Zapisanie logów

```

GNU nano 2.2.6      File: echo.txt
[execve("/usr/bin/echo", 0xFFBFFD2C, 0xFFBFFD34)  argc = 1
sysinfo(SI_MACHINE, "sun4u", 257)              = 6
mmap(0x00000000, 32, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON, -1, 0) = 0xFF3E0000
mmap(0x00000000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0) = 0xFF390000
mmap(0x00000000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0) = 0xFF380000
mmap(0xFF3A0000, 17984, MC_ADVICE, MADV_WILLNEED, 0, 0) = 0
mmap(0x00000000, 8192, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON, -1, 0) = 0xFF370000
mmap(0x00010000, 1684, MC_ADVICE, MADV_WILLNEED, 0, 0) = 0
resolvepath("/usr/lib/ld.so.1", "/lib/ld.so.1", 1023) = 12
resolvepath("/usr/bin/echo", "/usr/bin/echo", 1023) = 13
stat64("/usr/bin/echo", 0xFFBFF7F0)             = 0
open("/usr/lib/ld/config", O_RDONLY)              = 3
fstat64(3, 0xFFBFF360)                           = 0
mmap(0x00000000, 148, PROT_READ, MAP_SHARED, 3, 0) = 0xFF360000
close(3)                                           = 0
stat64("/lib/libc.so.1", 0xFFBFF5F0)             = 0
resolvepath("/lib/libc.so.1", "/lib/libc.so.1", 1023) = 14
open("/lib/libc.so.1", O_RDONLY)                  = 3
mmap(0x00010000, 32768, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_ALIGN, 3, 0) = 0xFF350000
mmap(0x00010000, 126884, PROT_NONE, MAP_PRIVATE|MAP_NORESERVE|MAP_ANON|MAP_ALIGN, -1, 0) = 0xFF200000
mmap(0xFF200000, 1247157, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_TEXT, 3, 0) = 0xFF200000
mmap(0xFF342000, 35965, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_INITDATA, 3, 1253376) = 0xFF342000
mmap(0xFF34C000, 1616, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_ANON, -1, 0) = 0xFF34C000
munmap(0xFF333000, 65536)                        = 0
munmap(0xFF350000, 32768)                        = 0
close(3)                                           = 0
mmap(0xFF200000, 146148, MC_ADVICE, MADV_WILLNEED, 0, 0) = 0
mmap(0x00010000, 24576, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON|MAP_ALIGN, -1, 0) = 0xFF350000
getcontext(0xFFBFF640)                           = 0
getrlimit(RLIMIT_STACK, 0xFFBFF640)              = 0
getpid()                                           = 12702 [12701]
setustack(0xFF352A88)

[ Read 36 lines
^G Get Help      ^C WriteOut     ^R Read File    ^N Prev Page    ^X Cut Text     ^G Cur Pos
^X Exit          ^_ Justify      ^W Where Is    ^J Next Page   ^U UnCut Text  ^H To Spell

```

Rysunek 1.4: Odczytanie logów

- Wyświetlanie logów polecenia systemowego

Według mnie ostatnią najważniejszą opcją jest możliwość wyświetlenia logów dla dowolnego polecenia systemowego, ale za pomocą tego wywołania można także sprawdzić kilka poleceń systemowych:

```
strace -e trace=POLECENIESYS POLECENIE
strace -e trace=POLECENIESYS, POLECENIESYS POLECENIE
```

Próbowałem to wywołanie zrobić za pomocą truss (na diablo), ale niestety nie zadziało.

```
[diablo-bash-3.2$ truss -e trace=open,close echo
truss: cannot find program: trace=open,close
diablo-bash-3.2$
```

Rysunek 1.5: truss -e (diablo)

Jednak po przełączeniu się na panaminta polecenie wykonało się poprawnie:

```
pniedzio@panamint:~$ strace -e trace=open,close echo
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
close(3)                                = 0
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
close(3)                                = 0
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
close(3)                                = 0

close(1)                                = 0
close(2)                                = 0
+++ exited with 0 +++
pniedzio@panamint:~$
```

Rysunek 1.6: strace -e (panamint)

## 2. Wykorzystanie programu strace do śledzenia wykonywania się programu

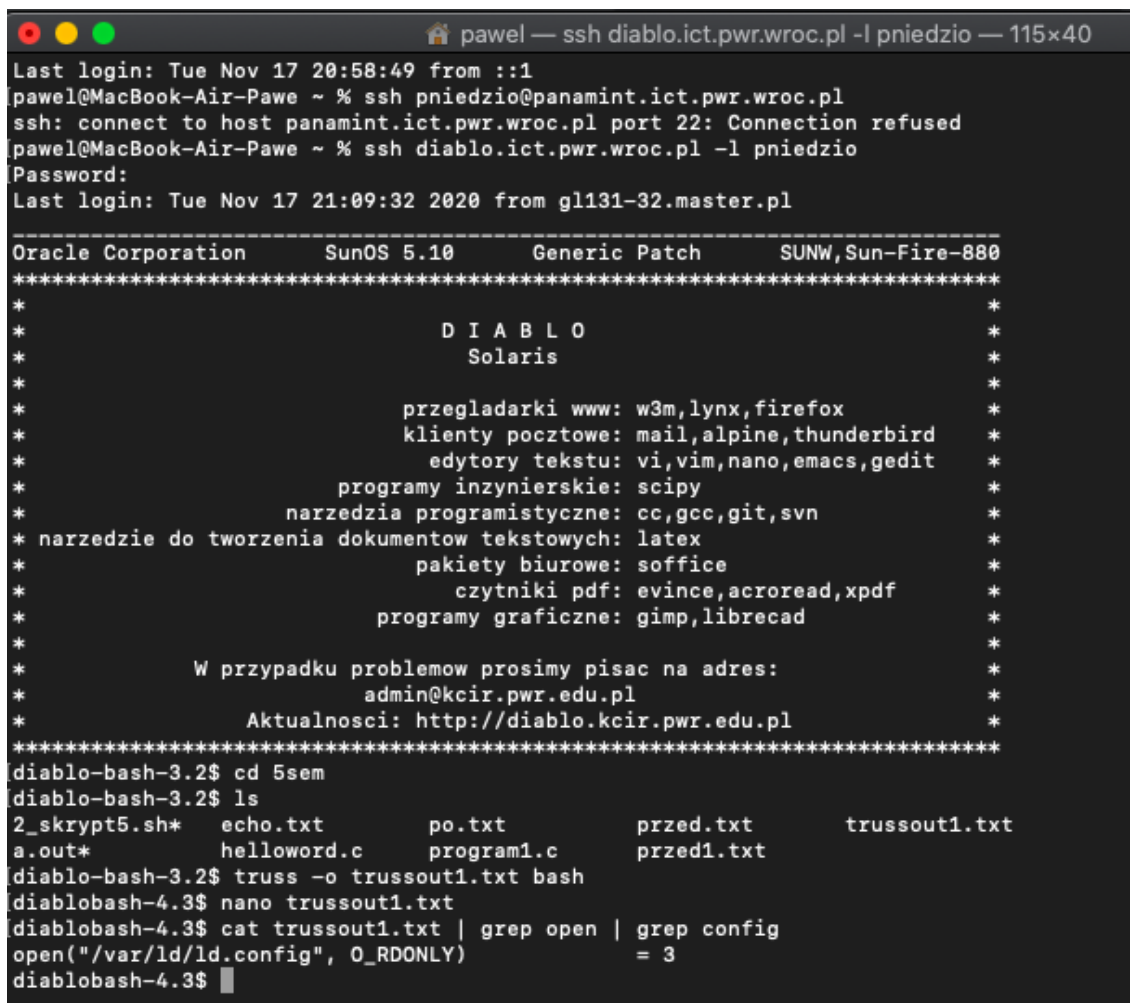
### 2.1 Przeanalizuj wykonanie się programu wyświetlającego napis Hello world na ekranie

```
pniedzio@panamint:~/5sem$ strace ./a.out
execve("./a.out", ["/a.out"], [/* 23 vars */]) = 0
brk(NULL)                               = 0x55dabfa18000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=103738, ...}) = 0
mmap(NULL, 103738, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0af76a3000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\4\2\0\0\0\0...", 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1689360, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0af76a1000
mmap(NULL, 3795296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0af70fb000
mprotect(0x7f0af7290000, 2097152, PROT_NONE) = 0
mmap(0x7f0af7490000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x195000) = 0x7f0af7490000
mmap(0x7f0af7496000, 14688, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f0af7496000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f0af76a2440) = 0
mprotect(0x7f0af7490000, 16384, PROT_READ) = 0
mprotect(0x55dabdfa2000, 4096, PROT_READ) = 0
mprotect(0x7f0af76be000, 4096, PROT_READ) = 0
munmap(0x7f0af76a3000, 103738)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 2), ...}) = 0
brk(NULL)                               = 0x55dabfa18000
brk(0x55dabfa39000)                     = 0x55dabfa39000
write(1, "Hello word", 10Hello word)     = 10
exit_group(0)                           = ?
+++ exited with 0 +++
pniedzio@panamint:~/5sem$
```

Rysunek 2.1: Strace programu wypisującego hello word

Jak widać na samym początku program się wykonuje, zmienia się rozmiar segmentu danych i jest sprawdzany dostęp do pliku. Następnie otrzymywany jest status pliku i mapowany jest program do pamięci. Procesy się powtarzają, zmienia się stan wątku, a następnie ustawia się ochronę obszaru pamięci. Można też zobaczyć, że program otwiera biblioteki. Napis jest wyświetlany dopiero po wszystkich tych niskopoziomych komendach.

## 2.2 Wykorzystaj program strace do znalezienia wszystkich plików konfiguracyjnych, jakie powłoka próbuje odczytać przy starcie



```
pawel — ssh diablo.ict.pwr.wroc.pl -l pniedzio — 115x40
Last login: Tue Nov 17 20:58:49 from ::1
pawel@MacBook-Air-Pawe ~ % ssh pniedzio@panamint.ict.pwr.wroc.pl
ssh: connect to host panamint.ict.pwr.wroc.pl port 22: Connection refused
pawel@MacBook-Air-Pawe ~ % ssh diablo.ict.pwr.wroc.pl -l pniedzio
Password:
Last login: Tue Nov 17 21:09:32 2020 from gl131-32.master.pl

-----
Oracle Corporation      SunOS 5.10      Generic Patch      SUNW,Sun-Fire-880
*****
*
*                      D I A B L O                      *
*                      Solaris                          *
*
*                      przegladarki www: w3m,lynx,firefox *
*                      klienty pocztowe: mail,alpine,thunderbird *
*                      edytory tekstu: vi,vim,nano,emacs,gedit *
*                      programy inzynierskie: scipy          *
*                      narzedzia programistyczne: cc,gcc,git,svn *
* narzedzie do tworzenia dokumentow tekstowych: latex      *
*                      pakiety biurowe: soffice              *
*                      czytniki pdf: evince,acroread,xpdf   *
*                      programy graficzne: gimp,librecad    *
*
*                      W przypadku problemow prosimy pisac na adres: *
*                      admin@kcir.pwr.edu.pl                 *
*                      Aktualnosci: http://diablo.kcir.pwr.edu.pl *
*****
diablo-bash-3.2$ cd 5sem
diablo-bash-3.2$ ls
2_skrypt5.sh*  echo.txt      po.txt        przed.txt     trussout1.txt
a.out*        helloworld.c  program1.c    przed1.txt
diablo-bash-3.2$ truss -o trussout1.txt bash
diablo-bash-4.3$ nano trussout1.txt
diablo-bash-4.3$ cat trussout1.txt | grep open | grep config
open("/var/ld/ld.config", O_RDONLY)
= 3
diablo-bash-4.3$
```

Rysunek 2.2: Plik przed1.txt (2)

Aby wykonać ten podpunkt zastosowałem polecenie truss (miałem problemy z zalogowaniem się na panaminta), przekierowałem trussa do pliku trussout1.txt, następnie wyszukałem w pliku linie które zawierały wyraz "open" oraz "config". Tak otrzymałem widoczny na ekranie plik konfiguracyjny.



## 2.3 Sprawdź czy plik edytowany w programie pico jest stale otwarty

```
pniedzio@panamint:~/5sem$ strace -o przed1.txt -e trace=open,close pico helloworld.c
pniedzio@panamint:~/5sem$ nano przed1.txt
pniedzio@panamint:~/5sem$
```

Rysunek 2.3: Otworzenie strace i pliku w pico

```
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
open("/lib/x86_64-linux-gnu/libz.so.1", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
open("/lib/x86_64-linux-gnu/libncursesw.so.5", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
open("/lib/x86_64-linux-gnu/libtinfo.so.5", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
open("/lib/x86_64-linux-gnu/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
open("/etc/nanorc", O_RDONLY) = 3
open("/usr/share/nano", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 4
open("/usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache", O_RDONLY) = 5
close(5) = 0
close(4) = 0
open("/usr/share/nano/nanorc.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/autoconf.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/perl.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/asm.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/elisp.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/go.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/tcl.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/man.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/fortran.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/c.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/spec.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/postgresql.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/po.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/xml.nanorc", O_RDONLY) = 4
close(4) = 0
open("/usr/share/nano/luau.nanorc", O_RDONLY) = 4
```

Rysunek 2.4: Plik przed1.txt (1)

```
GNU nano 2.7.4                                     File: przed1
close(4)                                             = 0
open("/usr/share/nano/html.nanorc", O_RDONLY) = 4
close(4)                                             = 0
open("/usr/share/nano/gentoo.nanorc", O_RDONLY) = 4
close(4)                                             = 0
close(3)                                             = 0
open("/home/pniedzio/.nanorc", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/home/pniedzio/.nano/search_history", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/lib/terminfo/x/xterm-256color", O_RDONLY) = 3
close(3)                                             = 0
close(4)                                             = 0
close(3)                                             = 0
open("./.helloworld.c.swp", O_WRONLY|O_CREAT|O_EXCL|O_APPEND, 0666) = 3
close(3)                                             = 0
open("/home/pniedzio/5sem/helloworld.c", O_RDONLY) = 3
open("/usr/share/locale/locale.alias", O_RDONLY|O_CLOEXEC) = 4
close(4)                                             = 0
open("/usr/share/locale/en_US/LC_MESSAGES/nano.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/share/locale/en/LC_MESSAGES/nano.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
close(3)                                             = 0
close(3)                                             = -1 EBADF (Bad file descriptor)
open("/home/pniedzio/5sem/helloworld.c", O_WRONLY|O_CREAT|O_APPEND, 0666) = 3
close(3)                                             = 0
close(3)                                             = -1 EBADF (Bad file descriptor)
+++ exited with 0 +++
```

Rysunek 2.5: Plik przed1.txt (2)

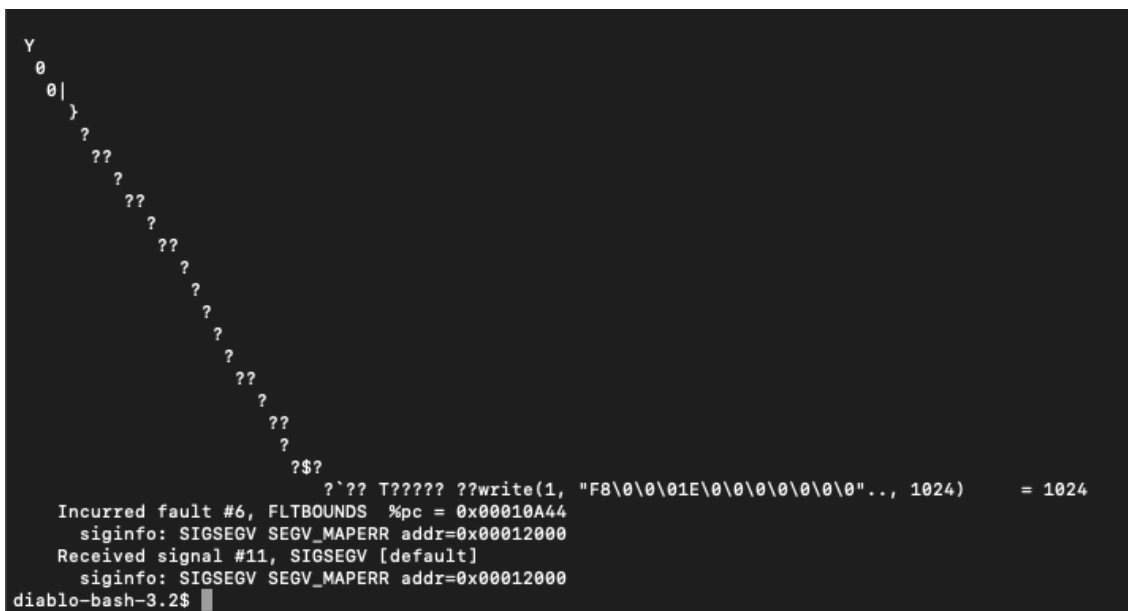
Jak widać każde otwarcie pliku kończy się jego zamknięciem `close`. Plik jest otwarty tylko wtedy wykonywane są w nim kluczowe operacje. Jeżeli dochodzi do edycji i zapisu pliku to następuje otwarcie pliku i zapis - plik **nie jest** stale otwarty.

## 2.4 Odczytaj jakie file deskryptory posiada uruchomiona aplikacja wyświetlająca napis Hello world na ekranie

brak —

### 3. Szukanie błędu w programie przy użyciu strace

Wykorzystaj program strace do znalezienia błędu w programie program.c. Jaki sygnał zabił program? Jak można wykorzystać strace do pomiaru czasu wykonania poszczególnych elementów programu?



Rysunek 3.1: Sygnał

Program zabił sygnał Segmentation Fault - błąd ochrony pamięci - musi być to złe alokowanie pamięci i złe wykorzystanie wskaźnika.