

SCR – SYSTEMY OPERACYJNE  
LISTA 4

---

**Zadania rozszerzone**

---

Prowadzący: Mgr inż. Arkadiusz Mielczarek

*Autor*  
Michał Maćkowiak

17 listopada 2020

# Contents

<b>1</b>	<b>Zadanie 5</b>	<b>1</b>
1.1	Polecenie: . . . . .	1
1.2	Rozwiązanie: . . . . .	1
<b>2</b>	<b>Zadanie 6</b>	<b>1</b>
2.1	Polecenie: . . . . .	1
2.2	Rozwiązanie: . . . . .	2
<b>3</b>	<b>Zadanie 7</b>	<b>5</b>
3.1	Polecenie: . . . . .	5
3.2	Rozwiązanie: . . . . .	5

# 1 Zadanie 5

## 1.1 Polecenie:

Zapoznaj się z dokumentacją polecenia *strace* na Linuxie. Opisz najważniejsze według siebie opcje.

## 1.2 Rozwiązanie:

*strace* jest Linuksowym odpowiednikiem solarisowego *truss*.

*strace* jest funkcją służącą do analizy interakcji programu z jądrem systemu Linux. Śledzi sygnały wysyłane i odbierane przez dany proces, mierzy czas poszczególnych wywołań oraz monitoruje wywołania systemowe. Uruchomienie programu za pomocą funkcji *strace* wyświetla log, w którym znajdziemy opisane krok po kroku działanie danego programu, co robi, z jakich plików korzysta, czy kończy pracę z błędem czy też bez. Wybrane opcje:

- *-o* – umożliwia zapis log'u do pliku,
- *-e trace = nazwa\_polecenia* – pozwala na śledzenie polecenia z wywołania np. *signal*, śledzi wszystkie wywołania systemowe związane z sygnałami (można wypisywać polecenia po przecinku),
- *-p numer\_PID* – umożliwia dołączenie do procesu o danym identyfikatorem PID i rozpoczyna śledzenie,
- *-t* – pokazuje godzinę wykonania poleceń,
- *-T* – pokazuje czas trwania wywołań systemowych.

# 2 Zadanie 6

## 2.1 Polecenie:

Wykorzystaj program *strace* na Linuxie do śledzenia wykonywania się programu:

- przeanalizuj wykonanie się programu wyświetlającego napis *Hello world* na ekranie,
- wykorzystaj program *strace* do znalezienia wszystkich plików konfiguracyjnych, jakie powłoka próbuje odczytać przy starcie,
- sprawdź czy plik edytowany w programie *pico* jest stale otwarty,
- odczytaj jakie file deskryptory posiada uruchomiona aplikacja wyświetlająca napis *Hello world* na ekranie,

## 2.2 Rozwiązanie:

(a)

```
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$ ./hello
Hello world
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$ strace ./hello
execve("./hello", [ "./hello" ], [ /* 23 vars */ ]) = 0
brk(NULL) = 0x5592dac20000
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) = 0x7fdc8626b000
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\3\0\0\0\0\0\0\0\3\0\0\0\1\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) = 0x7fdc8628d000
mmap(NULL, 3795296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdc85cc3000
mprotect(0x7fdc85e58000, 2097152, PROT_NONE) = 0
mmap(0x7fdc86058000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x195000) = 0x7fdc86058000
mmap(0x7fdc8605e000, 14688, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fdc8605e000
close(3) = 0
arch_prctl(ARCH_SET_FS, 0x7fdc8628e440) = 0
mprotect(0x7fdc86058000, 16384, PROT_READ) = 0
mprotect(0x5592d946c000, 4096, PROT_READ) = 0
mprotect(0x7fdc86286000, 4096, PROT_READ) = 0
munmap(0x7fdc8626b000, 103738) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 8), ...}) = 0
brk(NULL) = 0x5592dac20000
brk(0x5592dac41000) = 0x5592dac41000
write(1, "Hello world\n", 12Hello world
) = 12
exit_group(12) = ?
+++ exited with 12 +++
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$
```

Wyświetlenie napisu jest tak naprawdę realizowane jest w ostatnich liniijkach logu *strace*.

Przed wyświetleniem można zauważyć, że system najpierw np. żąda dostępu funkcją *access* albo np. za pomocą *mmap()* korzysta z memory mapingu.

Niby prosty program a wymaga wykonania wielu funkcji, dostępu wielu plików oraz wykonywania niskopoziomowych operacji na konkretnych adresach w pamięci komputera.

(b)

```
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$ strace -e trace=open,read bash
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
open("/lib/x86_64-linux-gnu/libtinfo.so.5", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260\315\0\0\0\0\0"... , 832) = 832
open("/lib/x86_64-linux-gnu/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\200\r\0\0\0\0\0"... , 832) = 832
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\3\0>\0\1\0\0\0\4\2\0\0\0\0"... , 832) = 832
open("/dev/tty", O_RDWR|O_NONBLOCK) = 3
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
open("/usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache", O_RDONLY) = 3
open("/etc/bash.bashrc", O_RDONLY) = 3
read(3, "# System-wide .bashrc file for i"... , 1863) = 1863
open("/home/mmackowi/.bashrc", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/home/mmackowi/.bash_history", O_RDONLY) = 3
read(3, "cat < FIFO\ncat < FIFO\ncat < FIFO"... , 7247) = 7247
open("/home/mmackowi/.bash_history", O_RDONLY) = 3
read(3, "cat < FIFO\ncat < FIFO\ncat < FIFO"... , 7247) = 7247
open("/lib/terminfo/x/xterm-256color", O_RDONLY) = 3
read(3, "\32\1%\0&\0\17\0\235\1\277\5xterm-256color|xterm"... , 4096) = 3430
read(3, "", 4096) = 0
open("/etc/inputrc", O_RDONLY) = 3
read(3, "# /etc/inputrc - global inputrc "... , 1748) = 1748
```

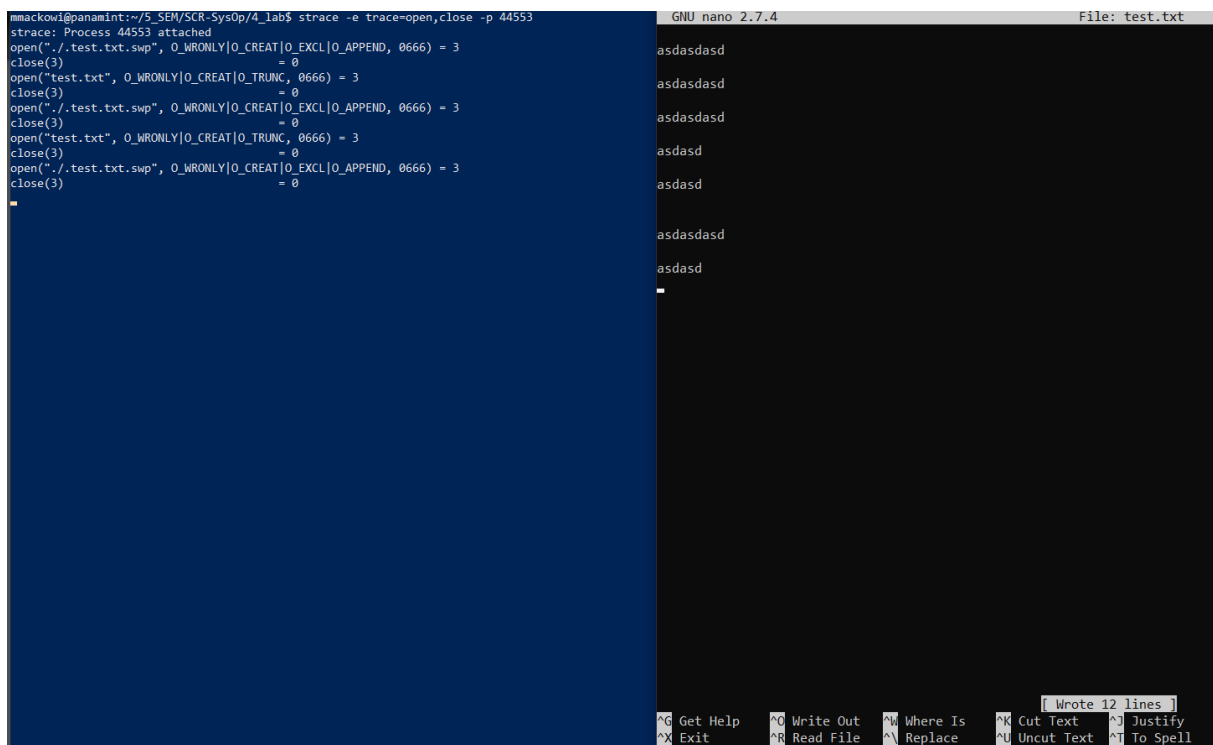
Za pomocą wywołania *strace -e trace=open,read bash*.

*read* pozwala na pokazanie z jakiego deskryptora pliku korzystamy.

*open* pozwala na pokazanie ścieżki pliku o danym deskrytorze.

% Nie jestem pewny czy nie powinno być *openat*, lecz po wywołaniu z tym poleceniem nie znajdowało żadnego. %

(c)



The screenshot shows a terminal window with two panes. The left pane displays the output of the `strace` command, tracking the `open` and `close` system calls for process 44553. The right pane shows the GNU nano 2.7.4 text editor editing the file `test.txt`. The editor contains 12 lines of the text `asdasdasd`. The terminal output shows the following sequence of operations:

```
mackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$ strace -e trace=open,close -p 44553
strace: Process 44553 attached
open("./test.txt.swp", O_WRONLY|O_CREAT|O_EXCL|O_APPEND, 0666) = 3
close(3) = 0
open("test.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
close(3) = 0
open("./test.txt.swp", O_WRONLY|O_CREAT|O_EXCL|O_APPEND, 0666) = 3
close(3) = 0
open("test.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
close(3) = 0
open("./test.txt.swp", O_WRONLY|O_CREAT|O_EXCL|O_APPEND, 0666) = 3
close(3) = 0
```

Śledzenie otwartego pliku w programie *pico* pokazało, że plik nie jest stale otwarty.

Otwiera się na moment edycji i zaraz potem zamyka.

Dla zapisu wciskając "ctrl + o" potem "enter". Najpierw się otwiera nasz plik, a potem zamyka.

(d)

```
mackowi@panamint:~/5_SEM/SCR-SysOp/4 lab$ strace -y ./hello
execve("./hello", ["/.hello"], [/* 23 vars */]) = 0
brk(NULL) = 0x558d143c6000
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</etc/ld.so.cache>
fstat(3</etc/ld.so.cache>, {st_mode=S_IFREG|0644, st_size=103738, ...}) = 0
mmap(NULL, 103738, PROT_READ, MAP_PRIVATE, 3</etc/ld.so.cache>, 0) = 0x7f956c37b000
close(3</etc/ld.so.cache>) = 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</lib/x86_64-linux-gnu/libc-2.24.so>
read(3</lib/x86_64-linux-gnu/libc-2.24.so>, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\0\1\0\0\0\4\2\0\0\0\0\0...", 832) = 832
fstat(3</lib/x86_64-linux-gnu/libc-2.24.so>, {st_mode=S_IFREG|0755, st_size=1689360, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f956c399000
mmap(NULL, 3795296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3</lib/x86_64-linux-gnu/libc-2.24.so>, 0) = 0x7f956bdd3000
mprotect(0x7f956bf68000, 2097152, PROT_NONE) = 0
mmap(0x7f956c168000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3</lib/x86_64-linux-gnu/libc-2.24.so>, 0x195000) = 0x7f956c168000
mmap(0x7f956c16e000, 14688, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f956c16e000
close(3</lib/x86_64-linux-gnu/libc-2.24.so>) = 0
arch_prctl(ARCH_SET_FS, 0x7f956c39a1c0) = 0
mprotect(0x7f956c168000, 16384, PROT_READ) = 0
mprotect(0x558d125e6000, 4096, PROT_READ) = 0
mprotect(0x7f956c396000, 4096, PROT_READ) = 0
munmap(0x7f956c37b000, 103738) = 0
fstat(1</dev/pts/9>, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 9), ...}) = 0
brk(NULL) = 0x558d143c6000
brk(0x558d143e7000) = 0x558d143e7000
write(1</dev/pts/9>, "Hello world\n", 12Hello world
) = 12
exit_group(12) = ?
+++ exited with 12 +++
```

Śledzenie file deskryptorów umożliwia flaga `-y`.

Jednym z deskryptorów jest 1 podana jako argument funkcji write().

### 3 Zadanie 7

### 3.1 Polecenie:

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?

### 3.2 Rozwiązanie:

By znaleźć sygnał, który zabił program należy śledzić *signal* w wywołaniu *strace*.

```
mrmackow@panamint:~/5_SEM/SCR-SysOp/4_lab$ strace -e trace=signal -b/bleedny.kod
Witajcie moi mili ...      @B@      @T 4 d      @zR Bx000000      @      @R Bx0000      @      @BF0000 ?@,*3*"      @ D
@      D | xe B000B000E0 @B@ @H00H8B0@r8A0@AB(B@ B000B0B @      @      @      @
@      @      @      @      @      @      @      @      @      @      @      @      @      @      @      @
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=0x555e7588c000} ---
+++ killed by SIGSEGV +++
Segmentation fault
mrmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$
```

Widać, że sygnałem który zabił program jest *SIGSEGV*, czyli inaczej popularny *Segmentation fault* - błąd naruszenia pamięci.

By wykorzystać *strace* do pomiaru czasu wykonania poszczególnych elementów programu korzystamy z flagi *-c*.

```
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$ strace -c ./bledny_kod
Witajcie moi mili ...
% time    seconds    usecs/call    calls    errors syscall
-----
0.00      0.000000      0            1         read
0.00      0.000000      0            2         write
0.00      0.000000      0            2         open
0.00      0.000000      0            2         close
0.00      0.000000      0            3         fstat
0.00      0.000000      0            5         mmap
0.00      0.000000      0            4         mprotect
0.00      0.000000      0            1         munmap
0.00      0.000000      0            3         brk
0.00      0.000000      0            3         3 access
0.00      0.000000      0            1         execve
0.00      0.000000      0            1         arch_prctl
-----
100.00    0.000000          28          3 total
Segmentation fault
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$
```

```
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$ strace -c ./hello
Hello world
% time    seconds    usecs/call    calls    errors syscall
-----
0.00      0.000000      0            1         read
0.00      0.000000      0            1         write
0.00      0.000000      0            2         open
0.00      0.000000      0            2         close
0.00      0.000000      0            3         fstat
0.00      0.000000      0            5         mmap
0.00      0.000000      0            4         mprotect
0.00      0.000000      0            1         munmap
0.00      0.000000      0            3         brk
0.00      0.000000      0            3         3 access
0.00      0.000000      0            1         execve
0.00      0.000000      0            1         arch_prctl
-----
100.00    0.000000          27          3 total
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$
```

Nie wiem czemu nie pokazuje czasu, ale pokazuje ilości wywołań systemowych.

Ewentualnie można użyć flagi *-r* co poda czas, ale w mniej ładny, w mojej opinii, sposób i bez ilości wywołań jak w *-c*.

Rysunek poniżej.



```

mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$ strace -r ./hello
0.000000 execve("./hello", ["/.hello"], [/* 23 vars */]) = 0
0.001114 brk(NULL) = 0x557859b16000
0.000125 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
0.000132 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
0.000091 open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
0.000105 fstat(3, {st_mode=S_IFREG|0644, st_size=103738, ...}) = 0
0.000095 mmap(NULL, 103738, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f3e4e5d3000
0.000088 close(3) = 0
0.000083 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
0.000091 open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
0.000085 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\0\1\0\0\0\4\2\0\0\0\0\0...", 832) = 832
0.000083 fstat(3, {st_mode=S_IFREG|0755, st_size=1689360, ...}) = 0
0.000076 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f3e4e5d1000
0.000090 mmap(NULL, 3795296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f3e4e02b000
0.000080 mprotect(0x7f3e4e1c0000, 2097152, PROT_NONE) = 0
0.000095 mmap(0x7f3e4e3c0000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x195000) = 0x7f3e4e3c0000
0.000101 mmap(0x7f3e4e3c6000, 14688, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f3e4e3c6000
0.000093 close(3) = 0
0.000103 arch_prctl(ARCH_SET_FS, 0x7f3e4e5d2440) = 0
0.000215 mprotect(0x7f3e4e3c0000, 16384, PROT_READ) = 0
0.000078 mprotect(0x5578587dd000, 4096, PROT_READ) = 0
0.000080 mprotect(0x7f3e4e5ee000, 4096, PROT_READ) = 0
0.000083 munmap(0x7f3e4e5d3000, 103738) = 0
0.000156 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 14), ...}) = 0
0.000149 brk(NULL) = 0x557859b16000
0.000068 brk(0x557859b37000) = 0x557859b37000
0.000082 write(1, "Hello world\n", 12)Hello world
) = 12
0.000113 exit_group(12) = ?
0.000192 +++ exited with 12 +++
mmackowi@panamint:~/5_SEM/SCR-SysOp/4_lab$

```