

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу
«Операционные системы» III
Семестр**

Студент:	Дегтярев Д.А.
Группа:	М80-207Б-21
Преподаватель:	Миронов Е.С
Оценка:	
Дата:	

Москва 2023

1. Постановка задачи

Приобретение практических навыков диагностики работы программного обеспечения.

Используемое средство диагностики — утилита **strace**.

2. Демонстрация полного функционала утилиты strace на примере ЛР №2

Strace - это утилита для диагностики, отладки и поиска неисправностей программ для Linux. Она используется для мониторинга взаимодействиями между процессами и ядром Linux, которые включают в себя системные вызовы, доставку сигналов и изменение состояния процесса. Работа strace становится возможной благодаря функции ядра, известной как ptrace.

Отображение всех вызовов:

```
[denis@denis-sat850 build]$ strace ./main
```

```
execve("./main", [ "./main" ], 0x7fffbad9d810 /* 81 vars */) = 0
```

```
brk(NULL) = 0x56492008a000
```

```
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdfabcedf0) = -1 EINVAL (Недопустимый аргумент)
```

```
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)
```

```
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
```

```
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=145939, ...}, AT_EMPTY_PATH) = 0
```

```
mmap(NULL, 145939, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd0d1bbe000
```

```
close(3) = 0
```

```
openat(AT_FDCWD, "/usr/lib/libstdc++.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\0\0\0\0\0\0"..., 832) = 832
```

```
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=19198496, ...}, AT_EMPTY_PATH) = 0
```

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd0d1bbc000
```

```
mmap(NULL, 2320384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd0d1985000
```

```

mmap(0x7fd0d1a1e000, 1138688, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x99000) = 0x7fd0d1a1e000

mmap(0x7fd0d1b34000, 487424, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1af000) = 0x7fd0d1b34000

mmap(0x7fd0d1bab000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x225000) = 0x7fd0d1bab000

mmap(0x7fd0d1bb9000, 10240, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd0d1bb9000

close(3) = 0

openat(AT_FDCWD, "/usr/lib/libm.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\0\0\0\0\0\0"...,
832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=944600, ...}, AT_EMPTY_PATH) =
0

mmap(NULL, 946368, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd0d189d000

mmap(0x7fd0d18ab000, 499712, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7fd0d18ab000

mmap(0x7fd0d1925000, 385024, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x88000) = 0x7fd0d1925000

mmap(0x7fd0d1983000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe5000) = 0x7fd0d1983000

close(3) = 0

openat(AT_FDCWD, "/usr/lib/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"...,
832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=571848, ...}, AT_EMPTY_PATH) =
0

mmap(NULL, 127304, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd0d187d000

mmap(0x7fd0d1880000, 94208, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7fd0d1880000

mmap(0x7fd0d1897000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1a000) = 0x7fd0d1897000

mmap(0x7fd0d189b000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x7fd0d189b000

close(3) = 0

```

```

openat(AT_FDCWD, "/usr/lib/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\0P4\2\0\0\0\0\0"... , 832)
= 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"... ,
784, 64) = 784

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1953472, ...}, AT_EMPTY_PATH) =
0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"... ,
784, 64) = 784

mmap(NULL, 1994384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd0d1696000

mmap(0x7fd0d16b8000, 1421312, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7fd0d16b8000

mmap(0x7fd0d1813000, 356352, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x17d000) = 0x7fd0d1813000

mmap(0x7fd0d186a000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d4000) = 0x7fd0d186a000

mmap(0x7fd0d1870000, 52880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd0d1870000

close(3) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fd0d1694000

arch_prctl(ARCH_SET_FS, 0x7fd0d1695200) = 0

set_tid_address(0x7fd0d16954d0) = 5190

set_robust_list(0x7fd0d16954e0, 24) = 0

rseq(0x7fd0d1695b20, 0x20, 0, 0x53053053) = 0

mprotect(0x7fd0d186a000, 16384, PROT_READ) = 0

mprotect(0x7fd0d189b000, 4096, PROT_READ) = 0

mprotect(0x7fd0d1983000, 4096, PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fd0d1692000

mprotect(0x7fd0d1bab000, 53248, PROT_READ) = 0

mprotect(0x56491fb0f000, 4096, PROT_READ) = 0

mprotect(0x7fd0d1c13000, 8192, PROT_READ) = 0

```

```

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0

munmap(0x7fd0d1bbe000, 145939) = 0

getrandom("\x7b\x5d\xb4\x4a\x37\xf1\xce\xef", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x56492008a000

brk(0x5649200ab000) = 0x5649200ab000

futex(0x7fd0d1bb96bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0

openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=3053536, ...}, AT_EMPTY_PATH) =
0

mmap(NULL, 3053536, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd0d13a8000

close(3) = 0

openat(AT_FDCWD, "/usr/share/locale/locale.alias", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2998, ...}, AT_EMPTY_PATH) = 0

read(3, "# Locale name alias data base.\n#"... , 4096) = 2998

read(3, "", 4096) = 0

close(3) = 0

openat(AT_FDCWD, "/usr/lib/locale/ru_RU.ISO-8859-5/LC_IDENTIFICATION",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/usr/lib/locale/ru_RU.iso88595/LC_IDENTIFICATION",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/usr/lib/locale/ru_RU/LC_IDENTIFICATION", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/usr/lib/locale/ru.ISO-8859-5/LC_IDENTIFICATION",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/usr/lib/locale/ru.iso88595/LC_IDENTIFICATION",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/usr/lib/locale/ru/LC_IDENTIFICATION", O_RDONLY|O_CLOEXEC) = -
1 ENOENT (Нет такого файла или каталога)

pipe2([3, 4], 0) = 0

pipe2([5, 6], 0) = 0

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fd0d16954d0) = 5191

```

```

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fd0d16954d0) = 5192

close(3)                                = 0

close(5)                                = 0

openat(AT_FDCWD, "./file1.txt", O_RDONLY) = -1 ENOENT (Нет такого файла или
каталога)

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5191, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5192, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---

openat(AT_FDCWD, "./file2.txt", O_RDONLY) = -1 ENOENT (Нет такого файла или
каталога)

close(4)                                = 0

close(6)                                = 0

exit_group(0)                            = ?

+++ exited with 0 +++

```

Очень «многословно» и не очень познавательно. Слишком много не интересующих нас вызовов. Исправим это.

Отображение определенных вызовов:

Опция -e служит для отображения лишь определенных вызовов.

Например — отобразить только вызовы close():

```

[leo@pc src]$ strace -e close ./main out1 out2
<test01.txt close(3)                = 0 close(3)
= 0
Name of the 1st child-output file: out1
Name of the 2nd child-output file: out2
close(3)                            = 0
close(5)                            = 0 Enter
strings to process:
close(4)                            = 0 close(6)
= 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=13590, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=13589, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} --Child 1 exited, returned 0
Child 2 exited, returned 0

```

+++ exited with 0 +++

Можно и **несколько типов вызовов сразу**: -e trace= и через запятую — список вызовов.

```
[denis@denis-sat850 build]$ strace -e trace=mmap,close ./main
mmap(NULL, 145939, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fee9e292000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7fee9e290000
mmap(NULL, 2320384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fee9e059000
mmap(0x7fee9e0f2000, 1138688, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x99000) = 0x7fee9e0f2000
mmap(0x7fee9e208000, 487424, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1af000) = 0x7fee9e208000
mmap(0x7fee9e27f000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x225000) = 0x7fee9e27f000
mmap(0x7fee9e28d000, 10240, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fee9e28d000
close(3)                                = 0
mmap(NULL, 946368, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fee9df71000
mmap(0x7fee9df7f000, 499712, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7fee9df7f000
mmap(0x7fee9dff9000, 385024, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x88000) = 0x7fee9dff9000
mmap(0x7fee9e057000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe5000) = 0x7fee9e057000
close(3)                                = 0
mmap(NULL, 127304, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fee9df51000
mmap(0x7fee9df54000, 94208, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7fee9df54000
mmap(0x7fee9df6b000, 16384, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) = 0x7fee9df6b000
mmap(0x7fee9df6f000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x7fee9df6f000
close(3)                                = 0
mmap(NULL, 1994384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fee9dd6a000
mmap(0x7fee9dd8c000, 1421312, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7fee9dd8c000
mmap(0x7fee9dee7000, 356352, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17d000) = 0x7fee9dee7000
```

```

mmap(0x7fee9df3e000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d4000) = 0x7fee9df3e000
mmap(0x7fee9df44000, 52880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fee9df44000
close(3) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7fee9dd68000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7fee9dd66000
mmap(NULL, 3053536, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fee9da7c000
close(3) = 0
close(3) = 0
close(3) = 0
close(5) = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5255, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
close(4) = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5256, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
close(6) = 0
+++ exited with 0 +++

```

Но на вызовы дочерних процессов взглянуть без определенного ключа не удастся.

Отслеживание дочерних процессов:

Отслеживать дерево процессов целиком помогает флаг -f, с которым strace отслеживает системные вызовы в процессах-потомках. К каждой строке вывода при этом добавляется pid процесса, делающего системный вызов:

```

[leo@pc src]$ strace -f -e trace=write,read ./main out1 out2 <test01.txt read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\202\2\0\0\0\0"..., 832) = 832
write(1, "Name of the 1st child-output fil"..., 40Name of the 1st child-output file: out1
) = 40
write(1, "Name of the 2nd child-output fil"..., 40Name of the 2nd child-output file: out2
) = 40 strace: Process 24231
attached strace: Process
24232 attached
[pid 24230] write(1, "Enter strings to process: \n", 27Enter strings to process:
) = 27
[pid 24230] read(0, <unfinished ...>
[pid 24231] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\202\2\0\0\0\0"...,
832) =
832

```



```

[pid 24232] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\202\2\0\0\0\0\0"...,
832) = 832
[pid 24230] <... read resumed>"\342\200\234The unexamined life is not wo"..., 4096) = 286
[pid 24230] write(6, "\342\200\234The unexamined life is not wo"..., 59) = 59
[pid 24230] write(6, "https://www.google.com/search?q="..., 130) = 130
[pid 24230] write(4, "123\n", 4) = 4
[pid 24230] write(4, "...biba\n", 8) = 8
[pid 24230] write(4, "boba...\n", 8) = 8
[pid 24230] write(6, "\342\200\234If God did not exist, it woul"..., 77) = 77
[pid 24230] read(0, "", 4096) = 0
[pid 24231] read(0, "123\n...biba\nboba...\n", 4096) = 20
[pid 24231] read(0, "", 4096) = 0
[pid 24231] write(1, "1st Child 24231: Started!\nReceiv"..., 182 <unfinished ...>
[pid 24232] read(0, "\342\200\234The unexamined life is not wo"..., 4096) = 266
[pid 24231] <... write resumed> = 182
[pid 24232] read(0, "", 4096) = 0
[pid 24232] write(1, "2nd Child 24232: Started!\nReceiv"..., 608) = 608
[pid 24232] +++ exited with 0 +++
[pid 24230] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=24232,
si_uid=1000, si_status=0, si_utime=0, si_stime=0} --[pid 24231] +++ exited with 0 +++
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=24231, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---
write(1, "Child 1 exited, returned 0\n", 28Child 1 exited, returned 0
) = 28
write(1, "Child 2 exited, returned 0\n", 28Child 2 exited, returned 0
) = 28
+++ exited with 0 +++

```

В данном случае будет полезной фильтрация по группам вызовов:

```

[denis@denis-sat850 build]$ strace -f -e trace=%process ./main

execve("./main", [ "./main" ], 0x7ffc22f643e8 /* 81 vars */) = 0

clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 5373
attached

, child_tidptr=0x7f0999d8b4d0) = 5373

[pid 5372] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 5374
attached

, child_tidptr=0x7f0999d8b4d0) = 5374

[pid 5373] execve("./child2", ["child2"], 0x7ffd0f0b8ae8 /* 81 vars */
<unfinished ...>

```

```
[pid 5374] execve("./child1", ["child1"], 0x7ffd0f0b8ae8 /* 81 vars */
<unfinished ...>

[pid 5373] <... execve resumed>          = -1 ENOENT (Нет такого файла или
каталога)

[pid 5374] <... execve resumed>          = -1 ENOENT (Нет такого файла или
каталога)

[pid 5372] exit_group(0)                  = ?

[pid 5374] exit_group(0 <unfinished ...>

[pid 5373] exit_group(0 <unfinished ...>

[pid 5374] <... exit_group resumed>      = ?

[pid 5373] <... exit_group resumed>      = ?

[pid 5372] +++ exited with 0 +++

[pid 5374] +++ exited with 0 +++

+++ exited with 0 +++
```

Пути к файлам вместо дескрипторов:

Ключ -у позволяет взглянуть на название файлов, с которыми работает процесс, вместо файловых дескрипторов:

```
[denis@denis-sat850
build]$ strace -y -e
read ./main

read(3</usr/lib/libstdc+
+.so.6.0.30>,
"\177ELF\2\1\1\3\0\0\0\0
\0\0\0\0\3\0>\0\1\0\0\0\
0\0\0\0\0\0\0\0"... ,
832) = 832

read(3</usr/lib/libm.so.
6>,
"\177ELF\2\1\1\3\0\0\0\0
\0\0\0\0\3\0>\0\1\0\0\0\
0\0\0\0\0\0\0\0"... ,
832) = 832

read(3</usr/lib/libgcc_s
.so.1>,
"\177ELF\2\1\1\0\0\0\0\0
\0\0\0\0\3\0>\0\1\0\0\0\
0\0\0\0\0\0\0\0"... ,
832) = 832
```

```
read(3</usr/lib/libc.so.
6>,
"\177ELF\2\1\1\3\0\0\0\0
\0\0\0\0\3\0>\0\1\0\0\0P
4\2\0\0\0\0\0"... , 832)
= 832
```

```
read(3</usr/share/locale
/locale.alias>, "#
Locale name alias data
base.\n#"..., 4096) =
2998
```

```
read(3</usr/share/locale
/locale.alias>, "",
4096) = 0
```

```
--- SIGCHLD
{si_signo=SIGCHLD,
si_code=CLD_EXITED,
si_pid=5440,
si_uid=1000,
si_status=0, si_utime=0,
si_stime=0} ---
```

```
+++ exited with 0 +++
```

Отображение времени выполнения вызова:

Посмотреть время выполнения того или иного вызова позволяет ключ **-t**:

```
[leo@pc src]$ strace -y -t -e read ./main out1 out2 <test01.txt
```

```
02:47:09 read(3</usr/lib/libstdc++.so.6.0.30>,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
```

```
02:47:09 read(3</usr/lib/libm.so.6>,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
```

```
02:47:09 read(3</usr/lib/libgcc_s.so.1>,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
```

```
02:47:09 read(3</usr/lib/libc.so.6>,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P4\2\0\0\0\0\0"... , 832) = 832
```

```
02:47:09 read(3</usr/share/locale/locale.alias>, "# Locale name alias data
base.\n#"..., 4096) = 2998
```

```
02:47:09 read(3</usr/share/locale/locale.alias>, "", 4096) = 0
```

```
02:47:09 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5492,
si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
```

```
02:47:09 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5493,
si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
```

```
02:47:09 +++ exited with 0 +++
```

Печать “относительной” времени вызова системных процессов:

```
[denis@denis-sat850 build]$ strace -r -e close ./main
```

```
0.000000 close(3) = 0
0.000558 close(3) = 0
0.000374 close(3) = 0
0.000307 close(3) = 0
0.000403 close(3) = 0
0.002399 close(3) = 0
0.000210 close(3) = 0
0.000651 close(3) = 0
0.000051 close(5) = 0
0.000373 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5536,
si_uid=1000, si_status=0, si_etime=0, si_stime=0} ---
0.000088 close(4) = 0
0.000051 close(6) = 0
0.000483 +++ exited with 0 +++
```

Статистика системных вызовов:

С помощью опции -c — можно получить наглядную статистику выполнения программы:

```
[denis@denis-sat850 build]$ strace -c ./main
```

% time	seconds	usecs/call	calls	errors	syscall
37,92	0,000182	91	2		clone
16,25	0,000078	5	15	8	openat
11,88	0,000057	2	23		mmap
8,75	0,000042	7	6		mprotect

6,25	0,000030	2	11	close
3,33	0,000016	8	2	pipe2
3,12	0,000015	15	1	munmap
2,71	0,000013	1	7	newfstatat
2,50	0,000012	2	6	read
2,08	0,000010	5	2	pread64
1,25	0,000006	2	3	brk
0,83	0,000004	4	1	getrandom
0,62	0,000003	1	2	1 arch_prctl
0,62	0,000003	3	1	futex
0,62	0,000003	3	1	prlimit64
0,42	0,000002	2	1	set_tid_address
0,42	0,000002	2	1	set_robust_list
0,42	0,000002	2	1	rseq
0,00	0,000000	0	1	1 access
0,00	0,000000	0	1	execve

100,00	0,000480	5	88	10 total

Вывод

По мере выполнения данной лабораторной работы я освоил диагностику работы ПО с помощью утилиты strace — простого и надёжного инструмента. Strace позволяет отслеживать выполнение системных вызовов и сигналов к ядру системы. Ее функционал прост, но широк; работать с ней комфортно и, главное, полезно.

Я научился просматривать как все вызовы сразу, так и несколько отдельных; получилось отследить работу и дочерних процессов. Научился отображать названия, пути к файлам, с которыми работает программа, вместо дескрипторов. Получилось взглянуть на удобную статистику системных вызовов и наблюдать время выполнения (обычное и относительное) тех или иных вызовов. Смог подключиться к уже запущенному процессу.

Все это — лишь малая часть функционала утилиты `strace`. С ее помощью можно делать еще много интересных вещей: намеренно ломать программу (инъекция ошибок) для выявления уязвимостей, наблюдать стек процесса в момент системного вызова и другое.

Но больше всего мне понравилась возможность отследить время тех или иных вызовов, т.к. это в будущем позволит определить, в каком участке кода программа могла зациклиться; или понять, в каком месте стоит произвести оптимизацию. Также интересной и полезной я нахожу возможность отобразить статистику системных вызовов в удобной форме, в виде таблицы.