Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Студент: Старцев Иван Романович

Группа: М8О-201Б-21

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Описание работы strace
4. Демонстрация работы strace
5. Вывод

**Репозиторий**

https://github.com/IvanTvardovsky/OS-labs

**Постановка задачи**

Подробно рассказать о каждом системном вызове из утилиты strace на примере лабораторной работы №4.

**Описание работы strace**

execve — открывает файл на исполнение.

int execve(const char \*filename, char \*const argv[],  
char \*const envp[]);

execve() выполняет программу, задаваемую аргументом *filename*.

*argv* — это массив строковых параметров, передаваемых новой программе. По соглашению, в первой строке должно содержаться имя файла, относящееся к запускаемой программе.

*envp* — это массив строк в формате ключ=значение, которые передаются новой программе в качестве окружения (environment). Оба массива argv и envp завершаются указателем null.

brk — изменяет расположение маркера окончания неинициализированных данных, который определяет конец сегмента данных процесса.

int brk(void \*addr);

brk() устанавливает конец сегмента данных в значение, указанное в аргументе *addr*, если это значение является приемлемым, система имеет достаточно памяти и процесс не достиг максимально возможного размера своего сегмента данных

arch\_prtcl — устанавливает состояние процесса или потока, зависящее от архитектуры.

int arch\_prctl(int code, unsigned long addr);  
int arch\_prctl(int code, unsigned long \*addr);

Функция arch\_prctl() задаёт состояние процесса или нити, зависящие от архитектуры. В аргументе *code* выбирается подфункция и ей передаётся значение *addr*; параметр *addr* рассматривается либо как unsigned long при операциях «установки», либо как unsigned long \* при операциях «получения» значения.

access - для проверки существования файла

openat — открывает файл в определенной директории.

newfstatat — возвращает информацию о файле в буфер.

close — закрывает файловый дескриптор.

mmap — создает новое отображение памяти в адресном пространстве процесса.

ftruncate — устанавливает файлу необходимый размер.

munmap — удаляет отображение.

mprotect − контролирует доступ к области памяти.

set\_robust\_list - запрашивает ядро записать начало списка надёжных фьютексов, принадлежащего вызывающей нити

rt\_sigaction - получает и изменяет обработчик сигнала.

rt\_sigprocmask - используется для проверки или настройки сигнальной маски текущего процесса.

set\_tid\_address - устанавливает у вызывающей нити значение clear\_child\_tid равным tidptr (В ядре для каждой нити хранится два атрибута (адреса): set\_child\_tid и clear\_child\_tid. Их значение по умолчанию равно NULL)

futex - предоставляет программам метод для ожидания пока определённое условие не станет истинным

fstatat — требует права выполнения (поиска) на все каталоги, указанные в полном имени файла pathname. (опрашиваемый файл задаётся в виде файлового дескриптора fd.)

statfs - возвращает информацию о смонтированной файловой системе

clone - создаёт новый процесс подобно fork

clock\_nanosleep - позволяет вызывающей нити приостановить работу на некоторое время с наносекундной точностью

lseek - позволяет задавать смещение, которое будет находиться за существующим концом файла (но это не изменяет размер файла)

exit\_group - завершает исполнение всех потоков процесса.

**Демонстрация работы strace**

tvard@tvard-HVY-WXX9:~/os/OS-labs/lab4$ strace -f ./lab4

execve("./lab4", ["./lab4"], 0x7ffdd6445d68 /\* 72 vars \*/) = 0

brk(NULL) = 0x558c38da4000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffdabefae70) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc001cf8000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=59823, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 59823, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fc001ce9000

close(3) = 0

openat(AT\_FDCWD, "/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\0P\237\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

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2216304, ...}, 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, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fc001ac1000

mmap(0x7fc001ae9000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fc001ae9000

mmap(0x7fc001c7e000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fc001c7e000

mmap(0x7fc001cd6000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7fc001cd6000

mmap(0x7fc001cdc000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fc001cdc000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc001abe000

arch\_prctl(ARCH\_SET\_FS, 0x7fc001abe740) = 0

set\_tid\_address(0x7fc001abea10) = 6804

set\_robust\_list(0x7fc001abea20, 24) = 0

rseq(0x7fc001abf0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fc001cd6000, 16384, PROT\_READ) = 0

mprotect(0x558c383ef000, 4096, PROT\_READ) = 0

mprotect(0x7fc001d32000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7fc001ce9000, 59823) = 0

newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

getrandom("\x46\xc5\xc9\x64\x03\x90\x3c\x51", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x558c38da4000

brk(0x558c38dc5000) = 0x558c38dc5000

read(0, test.txt

"test.txt\n", 1024) = 9

read(0, output.txt

"output.txt\n", 1024) = 11

openat(AT\_FDCWD, "test.txt", O\_RDONLY) = 3

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 6839 attached

, child\_tidptr=0x7fc001abea10) = 6839

[pid 6804] wait4(6839, <unfinished ...>

[pid 6839] set\_robust\_list(0x7fc001abea20, 24) = 0

[pid 6839] dup2(3, 0) = 0

[pid 6839] execve("/home/tvard/os/OS-labs/lab4/child", ["/home/tvard/os/OS-labs/lab4/chil"..., "test.txt"], 0x7ffdabefb048 /\* 72 vars \*/) = 0

[pid 6839] brk(NULL) = 0x56263f5a6000

[pid 6839] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffeb4f54830) = -1 EINVAL (Invalid argument)

[pid 6839] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f45a51b8000

[pid 6839] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 6839] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 4

[pid 6839] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=59823, ...}, AT\_EMPTY\_PATH) = 0

[pid 6839] mmap(NULL, 59823, PROT\_READ, MAP\_PRIVATE, 4, 0) = 0x7f45a51a9000

[pid 6839] close(4) = 0

[pid 6839] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 4

[pid 6839] read(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

[pid 6839] pread64(4, "\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

[pid 6839] pread64(4, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

[pid 6839] pread64(4, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68, 896) = 68

[pid 6839] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

[pid 6839] pread64(4, "\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

[pid 6839] mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7f45a4f81000

[pid 6839] mmap(0x7f45a4fa9000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x28000) = 0x7f45a4fa9000

[pid 6839] mmap(0x7f45a513e000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1bd000) = 0x7f45a513e000

[pid 6839] mmap(0x7f45a5196000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x214000) = 0x7f45a5196000

[pid 6839] mmap(0x7f45a519c000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f45a519c000

[pid 6839] close(4) = 0

[pid 6839] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f45a4f7e000

[pid 6839] arch\_prctl(ARCH\_SET\_FS, 0x7f45a4f7e740) = 0

[pid 6839] set\_tid\_address(0x7f45a4f7ea10) = 6839

[pid 6839] set\_robust\_list(0x7f45a4f7ea20, 24) = 0

[pid 6839] rseq(0x7f45a4f7f0e0, 0x20, 0, 0x53053053) = 0

[pid 6839] mprotect(0x7f45a5196000, 16384, PROT\_READ) = 0

[pid 6839] mprotect(0x56263eb8f000, 4096, PROT\_READ) = 0

[pid 6839] mprotect(0x7f45a51f2000, 8192, PROT\_READ) = 0

[pid 6839] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 6839] munmap(0x7f45a51a9000, 59823) = 0

[pid 6839] getrandom("\x44\xc6\x9a\x70\xc6\x70\x16\x92", 8, GRND\_NONBLOCK) = 8

[pid 6839] brk(NULL) = 0x56263f5a6000

[pid 6839] brk(0x56263f5c7000) = 0x56263f5c7000

[pid 6839] openat(AT\_FDCWD, "test.txt", O\_RDONLY) = 4

[pid 6839] openat(AT\_FDCWD, "/dev/shm/shm", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0777) = 5

[pid 6839] ftruncate(5, 4096) = 0

[pid 6839] mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0x7f45a51f1000

[pid 6839] newfstatat(0, "", {st\_mode=S\_IFREG|0664, st\_size=75, ...}, AT\_EMPTY\_PATH) = 0

[pid 6839] read(0, "8.0 2.0 -4.0 -1.0\n0.0 3.2 2.09\n-"..., 4096) = 75

[pid 6839] close(5) = 0

[pid 6839] close(5) = -1 EBADF (Bad file descriptor)

[pid 6839] close(5) = -1 EBADF (Bad file descriptor)

[pid 6839] close(5) = -1 EBADF (Bad file descriptor)

[pid 6839] read(0, "", 4096) = 0

[pid 6839] close(5) = -1 EBADF (Bad file descriptor)

[pid 6839] newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

[pid 6839] write(1, "1 0 -0.1 9.75 1 \n", 171 0 -0.1 9.75 1

) = 17

[pid 6839] close(4) = 0

[pid 6839] exit\_group(0) = ?

[pid 6839] +++ exited with 0 +++

<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 6839

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=6839, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

openat(AT\_FDCWD, "output.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666) = 4

openat(AT\_FDCWD, "/dev/shm/shm", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0777) = 5

mmap(NULL, 4096, PROT\_READ, MAP\_SHARED, 5, 0) = 0x7fc001d31000

newfstatat(4, "", {st\_mode=S\_IFREG|0664, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

munmap(0x7fc001d31000, 4096) = 0

unlink("/dev/shm/shm") = 0

close(5) = 0

write(4, "1 0 -0.1 9.75 1 ", 16) = 16

close(4) = 0

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Проделав лабораторную работу, я приобрёл навыки, необходимые для работы с strace, а также изучил системные вызовы.