Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

**УПРАВЛЕНИЕ ПРОЦЕССАМИ В ОС И ОБМЕН ДАННЫМИ МЕЖДУ НИМИ**

Студент: Корнев Максим Сергеевич

Группа: М8О–212Б–22

Вариант: 16

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Цель работы**

Приобретение практических навыков диагностики работы программного обеспечения.

**Постановка задачи**

При выполнении лабораторных работ по курсу ОС необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.

По итогам выполнения всех лабораторных работ отчет по данной ЛР должен содержать краткую сводку по исследованию написанных программ.

**Общие сведения о программе**

Strace - это утилита для отслеживания системных вызовов и сигналов, которые исполняет процесс. Она позволяет наблюдать взаимодействие процесса с операционной системой, что может быть полезным для отладки, анализа производительности и выявления проблем. Основные возможности strace включают в себя вывод информации о системных вызовах, сигналах, изменениях регистров, а также печать времени выполнения каждого системного вызова.

**Пример использования strace**

Действие утилиты продемонстрировано на примере лабораторной работы №1:  
radioactive@DESKTOP-RNP2IGB:/mnt/d/labs/os\_lab\_1/src$ strace ./main

execve("./main", ["./main"], 0x7fff2e90b850 /\* 26 vars \*/) = 0

brk(NULL) = 0x555712a07000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffd57cbd610) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fa0ada46000

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=21187, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 21187, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fa0ada40000

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\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, 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) = 0x7fa0ad818000

mmap(0x7fa0ad840000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fa0ad840000

mmap(0x7fa0ad9d5000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fa0ad9d5000

mmap(0x7fa0ada2d000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7fa0ada2d000

mmap(0x7fa0ada33000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fa0ada33000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fa0ad815000

arch\_prctl(ARCH\_SET\_FS, 0x7fa0ad815740) = 0

set\_tid\_address(0x7fa0ad815a10) = 981

set\_robust\_list(0x7fa0ad815a20, 24) = 0

rseq(0x7fa0ad8160e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fa0ada2d000, 16384, PROT\_READ) = 0

mprotect(0x555711960000, 4096, PROT\_READ) = 0

mprotect(0x7fa0ada80000, 8192, PROT\_READ) = 0

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

munmap(0x7fa0ada40000, 21187) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 55Введите имя файла для записи: ) = 55

read(0, test1.txt

"test1.txt\n", 256) = 10

openat(AT\_FDCWD, "test1.txt", O\_WRONLY|O\_CREAT, 0152260) = 3

pipe2([4, 5], 0) = 0

pipe2([6, 7], 0) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fa0ad815a10) = 982

close(4) = 0

close(7) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\201\321\202\321\200\320\276\320\272\321\203: \0", 30Введите строку: ) = 30

read(0, asdasd

"asdasd\n", 256) = 7

write(5, "asdasd\n", 7Ошибка: строка не заканчивается на ";" или "."

) = 7

read(6, "", 256) = 0

wait4(0, [{WIFEXITED(s) && WEXITSTATUS(s) == 255}], 0, NULL) = 982

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=982, si\_uid=1000, si\_status=255, si\_utime=0, si\_stime=0} ---

exit\_group(-1) = ?

+++ exited with 255 +++

**Вывод**

В ходе данной работы была изучена утилита strace, которая является удобным инструментом для отслеживания системных вызовов, используемых программой. Стоит заметить, что данная утилита выводит результат выполнения системных вызовов, что крайне полезно в процессе отладки программы. Данная утилита позволяет увидеть то, что происходит при запуске и выполнении программы с точки зрения операционной системы.