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: Андрюшин Лев Дмитриевич

Группа: М8О–212Б–22

Вариант: 2

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 №3:  
vingael@vingaelQWE:~/projects/OSI/OSI 3/src/build$ strace ./main 1.txt

execve("./main", ["./main", "1.txt"], 0x7fffcd6bd928 /\* 46 vars \*/) = 0

brk(NULL) = 0x55bf86a07000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffe37b14670) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe853332000

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=67439, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 67439, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fe853321000

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\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, 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, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe853000000

mprotect(0x7fe853028000, 2023424, PROT\_NONE) = 0

mmap(0x7fe853028000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fe853028000

mmap(0x7fe8531bd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fe8531bd000

mmap(0x7fe853216000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7fe853216000

mmap(0x7fe85321c000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe85321c000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe85331e000

arch\_prctl(ARCH\_SET\_FS, 0x7fe85331e740) = 0

set\_tid\_address(0x7fe85331ea10) = 21039

set\_robust\_list(0x7fe85331ea20, 24) = 0

rseq(0x7fe85331f0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fe853216000, 16384, PROT\_READ) = 0

mprotect(0x55bf86463000, 4096, PROT\_READ) = 0

mprotect(0x7fe85336c000, 8192, PROT\_READ) = 0

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

munmap(0x7fe853321000, 67439) = 0

openat(AT\_FDCWD, "1.txt", O\_RDONLY) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0664, st\_size=15, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 15, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fe85336b000

close(3) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fe85331ea10) = 21040

wait4(21040, Read value: 1.630000

Read value: 3.320000

Read value: 6.540000

Final sum: 11.490000

[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 21040

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=21040, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

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

getrandom("\xa1\xe6\x52\xae\x75\xa2\x4f\x5e", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55bf86a07000

brk(0x55bf86a28000) = 0x55bf86a28000

write(1, "Child process exited with status"..., 36Child process exited with status: 0

) = 36

munmap(0x7fe85336b000, 15) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**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.