Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

**Приобретение практических навыков диагностики работы программного обеспечения**

Студент: Гречников А.В.

Группа: М80-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата: 18.12.2021

Оценка: 4

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

1. **Постановка задачи**

При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются.

Используемые утилиты: strace.

1. **Описание используемых утилит**

Strace — это утилита Linux, отслеживающая системные вызовы, которые представляют собой механизм трансляции, обеспечивающий интерфейс между процессом и операционной системой. Использование данной утилиты позволяет понять, что процесс пытается сделать в данное время. Strace может быть полезен при отладке программ.

Для удобства работы с протоколом утилиты можно использовать следующие ключи:

* -o file – Перенаправить протокол утилиты в файл file
* -e trace=filters – Указать выражения, по которым будут фильтроваться системные вызовы. Например -e trace=write,%process задаёт фильтрацию по системным вызовам write и по группе системных вызовов, связанных с межпроцессорным взаимодействием.
* -f – Отслеживать системные вызовы в дочерних процессах
* -y – Заменить в протоколе все файловые дескрипторы на имена соответствующих им файлов (где возможно).
* -p file – Отслеживать только обращения к файлу file
* -k – Отображать стек вызовов

1. **Пример использования утилит**

**Лабораторная работа 5 – Динамические библиотеки**

execve("./prog2", ["./prog2"], 0x7ffd2d8f4bc0 /\* 50 vars \*/) = 0

brk(NULL) = 0x564c06251000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffc4f769e30) = -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

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=74985, ...}) = 0

mmap(NULL, 74985, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7ff5d8d5a000

close(3) = 0

openat(AT\_FDCWD, "/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\0\0\3\0>\0\1\0\0\0 \22\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=18816, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff5d8da5000

mmap(NULL, 20752, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff5d8d52000

mmap(0x7ff5d8d53000, 8192, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000) = 0x7ff5d8d53000

mmap(0x7ff5d8d55000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7ff5d8d55000

mmap(0x7ff5d8d56000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7ff5d8d56000

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\0\360q\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\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2029224, ...}) = 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

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff5d8b5a000

mprotect(0x7ff5d8b7f000, 1847296, PROT\_NONE) = 0

mmap(0x7ff5d8b7f000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7ff5d8b7f000

mmap(0x7ff5d8cf7000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7ff5d8cf7000

mmap(0x7ff5d8d42000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7ff5d8d42000

mmap(0x7ff5d8d48000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7ff5d8d48000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff5d8da2000

arch\_prctl(ARCH\_SET\_FS, 0x7ff5d8da2740) = 0

mprotect(0x7ff5d8d42000, 12288, PROT\_READ) = 0

mprotect(0x7ff5d8d56000, 4096, PROT\_READ) = 0

mprotect(0x564c05450000, 4096, PROT\_READ) = 0

mprotect(0x7ff5d8d9f000, 4096, PROT\_READ) = 0

munmap(0x7ff5d8d5a000, 74985) = 0

brk(NULL) = 0x564c06251000

brk(0x564c06272000) = 0x564c06272000

openat(AT\_FDCWD, "./lib1.so", 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@\20\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0775, st\_size=15672, ...}) = 0

getcwd("/home/artem", 128) = 12

mmap(NULL, 16424, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff5d8d6a000

mmap(0x7ff5d8d6b000, 4096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000) = 0x7ff5d8d6b000

mmap(0x7ff5d8d6c000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7ff5d8d6c000

mmap(0x7ff5d8d6d000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7ff5d8d6d000

close(3) = 0

mprotect(0x7ff5d8d6d000, 4096, PROT\_READ) = 0

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

read(0, 1 15 30

"1 15 30\n", 1024) = 8

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

write(1, "GCF(15, 30) = 15\n", 17GCF(15, 30) = 15

) = 17

read(0, 2 4 5

"2 4 5\n", 1024) = 6

write(1, "Square = 20.000000\n", 19Square = 20.000000

) = 19

read(0, 0

"0\n", 1024) = 2

munmap(0x7ff5d8d6a000, 16424) = 0

openat(AT\_FDCWD, "./lib2.so", 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@\20\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0775, st\_size=15768, ...}) = 0

getcwd("/home/artem", 128) = 12

mmap(NULL, 16424, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff5d8d6a000

mmap(0x7ff5d8d6b000, 4096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000) = 0x7ff5d8d6b000

mmap(0x7ff5d8d6c000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7ff5d8d6c000

mmap(0x7ff5d8d6d000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7ff5d8d6d000

close(3) = 0

mprotect(0x7ff5d8d6d000, 4096, PROT\_READ) = 0

read(0, 2 4 5

"2 4 5\n", 1024) = 6

write(1, "Square = 10.000000\n", 19Square = 10.000000

) = 19

read(0, "", 1024) = 0

munmap(0x7ff5d8d6a000, 16424) = 0

exit\_group(0) = ?

+++ exited with 0 +++

1. **Выводы**

Утилиты strace и ltrace – простые и надежные инструменты для отслеживания системных вызовов. Они полезны при отладке и тестировании программ. Несмотря на то, что на первый взгляд протоколы этих утилит кажутся чересчур объёмными и непонятными, при помощи различных ключей можно отфильтровать их «на свой вкус».

Утилита strace позволяет отслеживать большинство системных вызовов. Но если требуется пронаблюдать за более узкими аспектами работы программ, то на помощь придёт ltrace (для отслеживания вызовов динамических библиотек), ftrace (для отслеживания работы операционной системы), perf, SystemTap и многие другие.

Данные утилиты пригодились мне для изучения принципов работы системных вызовов и поиска ошибок в моих лабораторных работах.