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: Почечура Артемий Андреевич

Группа: М80-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата: 25.12.2021

Оценка: 5

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 3 – работа с потоками***

execve("./a.out", ["./a.out"], 0x7ffffe413170 /\* 19 vars \*/) = 0

brk(NULL) = 0x7fffc49a8000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffcbc5d6a0) = -1 EINVAL (Invalid argument)

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

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=32182, ...}) = 0

mmap(NULL, 32182, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fed3baa8000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", 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\220\201\0\0\0\0\0\0"..., 832) = 832

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0w\\\273\377\370\24Ef`xg\200\260\263\264\0"..., 68, 824) = 68

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=157224, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fed3bae0000

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0w\\\273\377\370\24Ef`xg\200\260\263\264\0"..., 68, 824) = 68

mmap(NULL, 140408, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fed3ba85000

mmap(0x7fed3ba8c000, 69632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x7000) = 0x7fed3ba8c000

mmap(0x7fed3ba9d000, 20480, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7fed3ba9d000

mmap(0x7fed3baa2000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c000) = 0x7fed3baa2000

mmap(0x7fed3baa4000, 13432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fed3baa4000

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\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 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\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) = 68

mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fed3b890000

mprotect(0x7fed3b8b5000, 1847296, PROT\_NONE) = 0

mmap(0x7fed3b8b5000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7fed3b8b5000

mmap(0x7fed3ba2d000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7fed3ba2d000

mmap(0x7fed3ba78000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7fed3ba78000

mmap(0x7fed3ba7e000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fed3ba7e000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fed3b880000

arch\_prctl(ARCH\_SET\_FS, 0x7fed3b880740) = 0

mprotect(0x7fed3ba78000, 12288, PROT\_READ) = 0

mprotect(0x7fed3baa2000, 4096, PROT\_READ) = 0

mprotect(0x7fed3bae9000, 4096, PROT\_READ) = 0

mprotect(0x7fed3badd000, 4096, PROT\_READ) = 0

munmap(0x7fed3baa8000, 32182) = 0

set\_tid\_address(0x7fed3b880a10) = 146

set\_robust\_list(0x7fed3b880a20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7fed3ba8cbf0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7fed3ba9a3c0}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7fed3ba8cc90, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7fed3ba9a3c0}, NULL, 8) = 0

rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

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

write(2, "\320\235\320\265\320\262\320\265\321\200\320\275\320\276\320\265 \320\272\320\276\320\273\320\270\321\207\320\265\321\201\321"..., 59Неверное количество аргументов

) = 59

exit\_group(1) = ?

+++ exited with 1 +++

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.