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: Фролов М.А.

Группа: М80-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата: 11.12.2021

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 –Управление потоками в ОС**

execve("./a.out", ["./a.out", "11"], 0x7ffce7b69b70 /\* 60 vars \*/) = 0

brk(NULL) = 0x55bc39a84000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffcea239370) = -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=59885, ...}) = 0

mmap(NULL, 59885, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4bb31d6000

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\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 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) = 0x7f4bb31d4000

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68

mmap(NULL, 140408, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4bb31b1000

mmap(0x7f4bb31b8000, 69632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x7000) = 0x7f4bb31b8000

mmap(0x7f4bb31c9000, 20480, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f4bb31c9000

mmap(0x7f4bb31ce000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c000) = 0x7f4bb31ce000

mmap(0x7f4bb31d0000, 13432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4bb31d0000

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) = 0x7f4bb2fbf000

mprotect(0x7f4bb2fe4000, 1847296, PROT\_NONE) = 0

mmap(0x7f4bb2fe4000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7f4bb2fe4000

mmap(0x7f4bb315c000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7f4bb315c000

mmap(0x7f4bb31a7000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f4bb31a7000

mmap(0x7f4bb31ad000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4bb31ad000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4bb2fbc000

arch\_prctl(ARCH\_SET\_FS, 0x7f4bb2fbc740) = 0

mprotect(0x7f4bb31a7000, 12288, PROT\_READ) = 0

mprotect(0x7f4bb31ce000, 4096, PROT\_READ) = 0

mprotect(0x55bc37b5f000, 4096, PROT\_READ) = 0

mprotect(0x7f4bb3212000, 4096, PROT\_READ) = 0

munmap(0x7f4bb31d6000, 59885) = 0

set\_tid\_address(0x7f4bb2fbca10) = 6167

set\_robust\_list(0x7f4bb2fbca20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f4bb31b8bf0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f4bb31c63c0}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f4bb31b8c90, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f4bb31c63c0}, NULL, 8) = 0

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

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

read(0, "f", 1) = 1

brk(NULL) = 0x55bc39a84000

brk(0x55bc39aa5000) = 0x55bc39aa5000

read(0, "i", 1) = 1

read(0, "l", 1) = 1

read(0, "e", 1) = 1

read(0, ".", 1) = 1

read(0, "t", 1) = 1

read(0, "x", 1) = 1

read(0, "t", 1) = 1

read(0, "\n", 1) = 1

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

mmap(NULL,

8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f4bb27bb000

mprotect(0x7f4bb27bc000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f4bb2fbafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[6189], tls=0x7f4bb2fbb700, child\_tidptr=0x7f4bb2fbb9d0) = 6189

futex(0x7f4bb2fbb9d0, FUTEX\_WAIT, 6189, NULL) = 0

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

write(1, "1e78\n", 5) = 5

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.