Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Студент: Садаков А. А.

Группа: М8О–206Б–19

Вариант: 15

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

**Цель работы**

Приобретение практических навыков диагностики работы программного обеспечения на примере 2 лабораторной работы.

**Задание**

Провести диагностику работы 2 лабораторной работы при помощи strace, объяснить результат работы strace.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

Правило проверки: строка должна начинаться с заглавной буквы.

**Вывод strace**

execve("./a.out", ["./a.out"], 0x7fffc90950d0 /\* 28 vars \*/) = 0

brk(NULL) = 0x7ffff4c38000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffffd7b2af0) = -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=75514, ...}) = 0

mmap(NULL, 75514, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fcb8cfcd000

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\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2029224, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fcb8d010000

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\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68

mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fcb8cdd0000

mprotect(0x7fcb8cdf5000, 1847296, PROT\_NONE) = 0

mmap(0x7fcb8cdf5000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7fcb8cdf5000

mmap(0x7fcb8cf6d000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7fcb8cf6d000

mmap(0x7fcb8cfb8000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7fcb8cfb8000

mmap(0x7fcb8cfbe000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fcb8cfbe000

close(3) = 0

arch\_prctl(ARCH\_SET\_FS, 0x7fcb8d011380) = 0

mprotect(0x7fcb8cfb8000, 12288, PROT\_READ) = 0

mprotect(0x7fcb8d01b000, 4096, PROT\_READ) = 0

mprotect(0x7fcb8d00d000, 4096, PROT\_READ) = 0

munmap(0x7fcb8cfcd000, 75514) = 0

fstat(0, {st\_mode=S\_IFREG|0644, st\_size=44, ...}) = 0

brk(NULL) = 0x7ffff4c38000

brk(0x7ffff4c59000) = 0x7ffff4c59000

read(0, "ans\nAfaefaefe\naefef\naefe\nAAAA\naf"..., 4096) = 44

openat(AT\_FDCWD, "ans", O\_WRONLY|O\_CREAT|O\_APPEND, 0600) = 3

pipe([4, 5]) = 0

pipe([6, 7]) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fcb8d011650) = 2355

close(4) = 0

close(7) = 0

write(5, "\n\0\0\0", 4) = 4

write(5, "Afaefaefe\0", 10) = 10

read(6, "\1", 1) = 1

write(5, "\6\0\0\0", 4) = 4

write(5, "aefef\0", 6) = 6

read(6, "\0", 1) = 1

write(3, "ERROR\n", 6) = 6

write(5, "\5\0\0\0", 4) = 4

write(5, "aefe\0", 5) = 5

read(6, "\0", 1) = 1

write(3, "ERROR\n", 6) = 6

write(5, "\5\0\0\0", 4) = 4

write(5, "AAAA\0", 5) = 5

read(6, "\1", 1) = 1

write(5, "\4\0\0\0", 4) = 4

write(5, "afe\0", 4) = 4

read(6, "\0", 1) = 1

write(3, "ERROR\n", 6) = 6

write(5, "\10\0\0\0", 4) = 4

write(5, "sgrgrgr\0", 8) = 8

read(6, "\0", 1) = 1

write(3, "ERROR\n", 6) = 6

write(5, "\2\0\0\0", 4) = 4

write(5, "A\0", 2) = 2

read(6, "\1", 1) = 1

read(0, "", 4096) = 0

write(5, "\0\0\0\0", 4) = 4

wait4(-1, NULL, 0, NULL) = 2355

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=2355, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

close(3) = 0

close(5) = 0

close(6) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Описание работы**

execve("./a.out", ["./a.out"], 0x7fffc90950d0 /\* 28 vars \*/) = 0

Исполняет программу ./a.out с ключами ", ["./a.out"]. Возвращает 0 – успешное выполнение.

brk(NULL) = 0x555bdb126000

Устанавливает конец сегмента данных в значение NULL, возвращает указатель на начало новой области памяти = 0x555bdb126000.

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

Проверяет /etc/ld.so.preload на существование и на наличие прав на чтение (R\_OK), возвращает -1 – или не существует /etc/ld.so.preload или нет прав на чтение, errno устанавливается в ENOENT (компонент пути не существует или является "висячей" символической ссылкой).

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

Открывает /etc/ld.so.cache относительно дескриптора указанного каталога - AT\_FDCWD (относительно текущего рабочего каталога вызывающего процесса) с правами доступа - O\_RDONLY|O\_CLOEXEC (на чтение и устанавливает флаг close-on-exec на новом файловом дескрипторе). Возвращает новый файловый дескриптор 3.

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=88176, ...}) = 0

Заполняет структуру указанную вторым аргументом fstat информацией об файле с файловым дескриптором 3. Возвращает 0 – успешное выполнение.

mmap(NULL, 88176, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4ece6b2000

Создает отображение файла с файловым дескриптором 3 в память, начиная с адресса NULL (система сама выбирает), размер = 88176 байт, с правами защиты памяти на чтение PROT\_READ, задает тип отражаемого объекта MAP\_PRIVATE - создает неразделяемое отражение с механизмом copy-on-write, запись в эту область памяти не влияет на файл, не определено, являются или нет изменения в файле после вызова mmap видимыми в отраженном диапазоне. Возвращает указатель на начало отраженной памяти = 0x7f4ece6b2000.

close(3) = 0

Закрывает файл с файловым дескриптором 3. Возвращает 0 – успешное выполнение.

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000b\0\0\0\0\0\0"..., 832) = 832

Читает 832 байта данных из файла с файловым дескриптором 3 в буффер указанный вторым аргументом. Возвращает число успешно считанных байт = 832.

mprotect(0x7f4ece29c000, 2093056, PROT\_NONE) = 0

Контролирует доступ к области памяти начинающейся с адресса 0x7f4ece29c000 длины 2093056 байт, доступ к памяти запрещен - PROT\_NONE. Если программой производится запрещенный этой функцией доступ к памяти, то такая программа получает сигнал SIGSEGV. Возвращает 0 – успешное завершение.

arch\_prctl(ARCH\_SET\_FS, 0x7f4ece6ad740) = 0

Устанавливает специфичное для архитектуры состояние. Устанавливает 64 битную базу для регистра FS (ARCH\_SET\_FS) в значение 0x7f4ece6ad740. Возвращает 0 – успешное выполнение.

munmap(0x7f4ece6b2000, 88176) = 0

Снимает отражение из заданной области памяти, 0x7f4ece6b2000 указатель на начало памяти, длина = 88176 байт. Возвращает 0 – успешное выполнение.

write(5, "Afaefaefe\0", 10) = 10

Записывает 10 байта из буффера (второй аргумент) в файл с файловым дескриптором 5. Возвращает число успешно записанных байт = 10.

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f4ece6ada10) = 1614

Создает процесс-потомок с фалагами - CLONE\_CHILD\_CLEARTID (очищает id), CLONE\_CHILD\_SETTID (устанавливает id), SIGCHLD (сигнал о изменении статуса дочернего процесса), задает положение стека для процесса-потомка = NULL, задает указатель на id = 0x7f4ece6ada10. Возвращает pid процесса-потомка.

wait4(-1, NULL, 0, NULL) = 2355

Родительский процессе ждет завершение работы процесса с pid = 2355

**Вывод**

Strace позволяет просматривать различные системные вызовы, которые происходят при работе программы. Благодаря этому можно удобно анализировать работу программы и искать ошибки.