МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика"

Кафедра 806 "Вычислительная математика и программирование"

Лабораторная работа №8

По курсу «Операционные системы»

Студент: Татаркин И. В.

Группа: М8О-208Б-23

Преподаватель: Живалев Е. А.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Тема:** Диагностика программного обеспечения

**Цель работы:** Приобретение практических навыков диагностики работы программного обеспечения.

**Задачи:**

1. Провести диагностику одной из программ, написанных в ходе работ 1-7, с помощью утилиты *strace.*
2. Рассмотреть использовавшиеся системные вызовы.

**Ход работы:**

strace — это утилита для диагностики и отладки программ в операционных системах семейства Unix/Linux. Она позволяет отслеживать системные вызовы и сигналы, которые приложение выполняет или получает во время своей работы.

Трассировка использования *strace* для лабораторной работы №3*:*

execve("./lab3", ["./lab3"], 0x7ffe3fd709c0 /\* 53 vars \*/) = 0

brk(NULL) = 0x6163ef9e5000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffa00a8e80) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x758b60512000

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=63219, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 63219, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x758b60502000

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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 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) = 0x758b60200000

mprotect(0x758b60228000, 2023424, PROT\_NONE) = 0

mmap(0x758b60228000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x758b60228000

mmap(0x758b603bd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x758b603bd000

mmap(0x758b60416000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x758b60416000

mmap(0x758b6041c000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x758b6041c000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x758b604ff000

arch\_prctl(ARCH\_SET\_FS, 0x758b604ff740) = 0

set\_tid\_address(0x758b604ffa10) = 159637

set\_robust\_list(0x758b604ffa20, 24) = 0

rseq(0x758b605000e0, 0x20, 0, 0x53053053) = 0

mprotect(0x758b60416000, 16384, PROT\_READ) = 0

mprotect(0x6163ee0fa000, 4096, PROT\_READ) = 0

mprotect(0x758b6054c000, 8192, PROT\_READ) = 0

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

munmap(0x758b60502000, 63219) = 0

openat(AT\_FDCWD, "/dev/shm/shared\_mem", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 3

ftruncate(3, 4096) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x758b6054b000

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

getrandom("\x97\x08\x99\xde\x58\x35\x4b\x58", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x6163ef9e5000

brk(0x6163efa06000) = 0x6163efa06000

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

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 93Введите имя файла (или введите 'exit' для завершения): ) = 93

read(0, wer.txt

"wer.txt\n", 1024) = 8

write(1, "\320\242\320\265\320\277\320\265\321\200\321\214 \320\262\320\262\320\265\320\264\320\270\321\202\320\265 \320\275\320\265"..., 174Теперь введите несколько групп чисел. Введите 'done', чтобы завершить ввод и выполнить обработку.

) = 174

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\263\321\200\321\203\320\277\320\277\321\203 \321\207\320\270"..., 106Введите группу чисел через пробел (или 'done' для обработки): ) = 106

read(0, 3 4 5 6

"3 4 5 6\n", 1024) = 8

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\263\321\200\321\203\320\277\320\277\321\203 \321\207\320\270"..., 106Введите группу чисел через пробел (или 'done' для обработки): ) = 106

read(0, 2 0 -4 1

"2 0 -4 1\n", 1024) = 9

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\263\321\200\321\203\320\277\320\277\321\203 \321\207\320\270"..., 106Введите группу чисел через пробел (или 'done' для обработки): ) = 106

read(0, done

"done\n", 1024) = 5

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x758b604ffa10) = 159873

wait4(-1, Child process started.

Содержимое shared\_mem:

wer.txt

3 4 5 6

2 0 -4 1

Обработка файла: wer.txt

Входные данные для calculate\_sum: 3 4 5 6

Результат calculate\_sum: 18

Входные данные для calculate\_sum: 2 0 -4 1

Результат calculate\_sum: -1

Дочерний процесс завершен, результаты записаны в wer.txt.

NULL, 0, NULL) = 159873

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=159873, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271 \320\277\321\200\320\276\321"..., 161Родительский процесс завершён обработку для файла 'wer.txt'. Вы можете ввести новые данные.

) = 161

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 93Введите имя файла (или введите 'exit' для завершения): ) = 93

read(0, exit

"exit\n", 1024) = 5

write(1, "\320\222\321\213\321\205\320\276\320\264 \320\270\320\267 \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274"..., 36Выход из программы.

) = 36

munmap(0x758b6054b000, 4096) = 0

unlink("/dev/shm/shared\_mem") = 0

exit\_group(0) = ?

+++ exited with 0 +++

Использовавшиеся системные вызовы:

1. **execve** – Выполняет запуск нового процесса, заменяя текущий процесс указанной программой.
2. **read** – Считывает данные из файла или ввода.
3. **write** – Записывает данные в файл или вывод.
4. **mmap** – Отображает файлы или устройства в память, а также выделяет анонимную память.
5. **munmap** – Освобождает ранее отображённую память.
6. **openat** – Открывает файл или каталог, относительно заданного файлового дескриптора.
7. **fstat** – Получает информацию о файле по его дескриптору.
8. **close** – Закрывает файловый дескриптор.
9. **clone** – Создаёт новый процесс или поток.
10. **access** – Проверяет доступность файла по заданным правам (чтение, запись, выполнение).
11. **prlimit64** – Получает или изменяет текущие ограничения ресурсов (например, размер стека).
12. **mprotect** – Изменяет права доступа к памяти (например, разрешение на чтение/запись/выполнение).
13. **getrandom** – Генерирует случайные числа из ядра.
14. **set\_tid\_address** – Устанавливает адрес для хранения идентификатора потока.
15. **set\_robust\_list** – Устанавливает указатель на список «робастных мьютексов» (защищённых от прерываний).
16. **arch\_prctl** – Настраивает параметры архитектуры, например, указывает адрес указателя сегмента FS для текущего потока.
17. **clock\_nanosleep** – Приостанавливает выполнение потока на указанное время.
18. **futex** – Оперирует синхронизацией между потоками.
19. **unlink** – Удаляет файл или символическую ссылку.

**Вывод:** В ходе работы была проведена диагностика ранее созданной программы с использованием инструмента strace. Анализ системных вызовов позволил выявить их ключевую роль в обеспечении взаимодействия приложения с ресурсами операционной системы. Были подробно изучены используемые механизмы работы с памятью, синхронизации процессов, ввода-вывода и передачи данных между компонентами. Полученные результаты подтвердили, что работа программы соответствует теоретическим ожиданиям и корректно реализует взаимодействие с основными механизмами операционной системы.