Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №8 по курсу**

**«Операционные системы»**

**ДИАГНОСТИКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Студент: Басова Татьяна Валентиновна

Группа: М8О–212Б–22

Вариант: 3

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

* Приобретение практических навыков диагностики работы программного обеспечения.

## Задание

При выполнении лабораторных работ по курсе ОС необходимо продемонстрировать системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.

По итогам выполнения всех лабораторных работа отчет по данной ЛР должен содержать краткую сводку по исследованию написанных программ.

**Общие сведения о программе**

Для анализа я буду использовать лабораторную работу номер 1. В ней мною были использованы следующие системные вызовы:

1. **wait** – приостанавливает выполнение вызвавшего процесса до тех пор, пока не прекратит выполнение один из его потомков.
2. **pipe** – создаёт однонаправленный канал данных, который можно использовать для взаимодействия между процессами.
3. **close** – закрывает файловый дескриптор, который после этого не ссылается ни на один и файл и может быть использован повторно.
4. **write -**  записывает до count байтов из буфера buf в файл, на который ссылается файловый описатель fd.
5. **read -** пытается записать count байтов файлового описателя fd в буфер, адрес которого начинается с buf.
6. **fork –** создаёт новый процесс (потомок), который является практически полной копией процесса-родителя, выполняющего этот вызов.
7. **exit -** "немедленно" завершает работу программы.
8. **dup2** - переназначает файловый дескриптор.
9. **exec** - позволяет запустить исполняемый файл изнутри программы.

**Пример работы strace (dtrace)**

execve("/usr/bin/ls", ["ls"], 0x7ffd9df2a680 /\* 59 vars \*/) = 0

brk(NULL) = 0x55f533595000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffee50d6dd0) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fec78d59000

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=66111, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 66111, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fec78d48000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libselinux.so.1", 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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=166280, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 177672, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fec78d1c000

mprotect(0x7fec78d22000, 139264, PROT\_NONE) = 0

mmap(0x7fec78d22000, 106496, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x6000) = 0x7fec78d22000

mmap(0x7fec78d3c000, 28672, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x20000) = 0x7fec78d3c000

mmap(0x7fec78d44000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000) = 0x7fec78d44000

mmap(0x7fec78d46000, 5640, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fec78d46000

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\0 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, 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, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fec78a00000

mmap(0x7fec78a28000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fec78a28000

mmap(0x7fec78bbd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fec78bbd000

mmap(0x7fec78c15000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7fec78c15000

mmap(0x7fec78c1b000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fec78c1b000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpcre2-8.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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=613064, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 615184, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fec78c85000

mmap(0x7fec78c87000, 438272, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7fec78c87000

mmap(0x7fec78cf2000, 163840, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x6d000) = 0x7fec78cf2000

mmap(0x7fec78d1a000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x94000) = 0x7fec78d1a000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fec78c82000

arch\_prctl(ARCH\_SET\_FS, 0x7fec78c82800) = 0

set\_tid\_address(0x7fec78c82ad0) = 6510

set\_robust\_list(0x7fec78c82ae0, 24) = 0

rseq(0x7fec78c831a0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fec78c15000, 16384, PROT\_READ) = 0

mprotect(0x7fec78d1a000, 4096, PROT\_READ) = 0

mprotect(0x7fec78d44000, 4096, PROT\_READ) = 0

mprotect(0x55f532a58000, 4096, PROT\_READ) = 0

mprotect(0x7fec78d93000, 8192, PROT\_READ) = 0

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

munmap(0x7fec78d48000, 66111) = 0

statfs("/sys/fs/selinux", 0x7ffee50d6e10) = -1 ENOENT (Нет такого файла или каталога)

statfs("/selinux", 0x7ffee50d6e10) = -1 ENOENT (Нет такого файла или каталога)

getrandom("\xde\x66\xe2\xff\xc9\x5d\x02\x5b", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55f533595000

brk(0x55f5335b6000) = 0x55f5335b6000

openat(AT\_FDCWD, "/proc/filesystems", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0444, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

read(3, "nodev\tsysfs\nnodev\ttmpfs\nnodev\tbd"..., 1024) = 391

read(3, "", 1024) = 0

close(3) = 0

access("/etc/selinux/config", F\_OK) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/locale/locale-archive", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=5721376, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 5721376, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fec78400000

close(3) = 0

ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0

ioctl(1, TIOCGWINSZ, {ws\_row=11, ws\_col=178, ws\_xpixel=0, ws\_ypixel=0}) = 0

openat(AT\_FDCWD, ".", O\_RDONLY|O\_NONBLOCK|O\_CLOEXEC|O\_DIRECTORY) = 3

newfstatat(3, "", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}, AT\_EMPTY\_PATH) = 0

getdents64(3, 0x55f53359b9f0 /\* 7 entries \*/, 32768) = 216

getdents64(3, 0x55f53359b9f0 /\* 0 entries \*/, 32768) = 0

close(3) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x4), ...}, AT\_EMPTY\_PATH) = 0

write(1, "calc.cpp CMakeLists.txt main.c"..., 46) = 46

close(1) = 0

close(2) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Найденные мной системные вызовы внутри strace**

1. **openat**(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) – работает так же как и open, но получает на вход файловый дескриптор, чтобы искать файлы по локальному пути относительно заданного файлового дескриптора.

2. Вызов **mmap** отражает length байтов, начиная со смещения offset файла (или другого объекта), определенного файловым описателем fd, в память, начиная с адреса start.

3. Системный вызов **munmap** удаляет все отражения из заданной области памяти, после чего все ссылки на данную область будут вызывать ошибку "неправильное обращение к памяти" (invalid memory reference).

4. **mprotect** контролирует доступ к области памяти. Если программой производится запрещенный этой функцией доступ к памяти, то такая программа получает сигнал SIGSEGV.

5. **access** проверяет, имеет ли процесс права на чтение или запись, или же просто проверяет, существует ли файл (или другой объект файловой системы), с именем pathname. Если pathname является символьной ссылкой, то проверяются права доступа к файлу, на который она ссылается.

6. **statfs** возвращает информацию о переданном пути к файлу или файловой системе.

7. **getrandom** используется для получения случайных данных.

**Вывод**

Процесс выполнения программы на С++ обычно включает следующие шаги:

1. Загрузка исполняемого файла в память.
2. Загрузка динамических библиотек (если они используются).
3. Инициализация статических переменных.
4. Выделение памяти для динамических переменных и структур данных.
5. Выполнение кода программы.
6. Освобождение выделенных ресурсов и завершение программы.

Примерно такую последовательность мы и видим, когда запускаем strace. Мы видим, что есть огромное множество системных вызовов, которые мы внутри своей программы не вызывали. Программа сама очищает память, занимает ее, использует файловые дескрипторы. Это и есть процессы загрузки и инициализации, которые происходят внутри системных библиотеку и динамических библиотек.

Но среди огромного количества системных вызовов есть и те, которые я действительно вызывала внутри кода. Благодаря strace я удостоверилась, что моя программа работает правильно и все действия в ней совершаются так, как я и задумывала.