Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №8 по курсу**

**«Операционные системы»**

Студент: Чапалда Мария Олеговна

Группа: М8О-201Б-22

Вариант: -

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Содержание**

[**Постановка задачи** 3](#_Toc130665298)

[**Цель работы** 3](#_Toc130665299)

[**Задание** 3](#_Toc130665300)

[**Системные вызовы** 4](#_Toc130665301)

[**Демонстрация работы** 6](#_Toc130665302)-7

[**Выводы**](#_Toc130665303) 7

# **Постановка задачи**

## **Цель работы**

Приобретение практических навыков диагностики работы программного обеспечения.

## **Задание**

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

# **Системные вызовы**

1) **int execve(const char \****filename***, char \*const***argv* **[], char \*const***envp***[]);** 

**execve()** выполняет программу, заданную параметром *filename*. Программа должна быть или двоичным исполняемым файлом, или скриптом, начинающимся со строки вида "**#!***интерпретатор*[аргументы]". В последнем случае интерпретатор -- это правильный путь к исполняемому файлу, который не является скриптом; этот файл будет выполнен как **интерпретатор** [arg] *filename*. *argv* -- это массив строк, аргументов новой программы. *envp* -- это массив строк в формате **key=value**, которые передаются новой программе в качестве окружения (environment). Как *argv*, так и *envp* завершаются нулевым указателем. К массиву аргументов и к окружению можно обратиться из функции **main**(), которая объявлена как **int main(int argc, char \*argv[], char \*envp[])**.

2) **int access(const char \****pathname***, int***mode***);**

**access** проверяет, имеет ли вызвавший процесс права доступа к файлу *pathname*. Если *pathname* является символьной ссылкой, то проверяются права доступа к файлу, на который она ссылается.

Аргумент *mode* --- это маска выполняемых проверок доступа; может быть равна значению **F\_OK**, или состоять из одного или несколько побитово сложенных **R\_OK**, **W\_OK** и **X\_OK**. **F\_OK** проверяет существование файла. **R\_OK**, **W\_OK** и **X\_OK** запрашивают проверку, соответственно, существования файла и возможности его чтения, записи или выполнения.

3) **ssize\_t read(int***fd***, void \****buf***, size\_t***count***);**

Вызов **read**() пытается прочитать *count* байт из файлового дескриптора *fd* в буфер, начинающийся по адресу *buf*.

Для файлов, поддерживающих смещения, операция чтения начинается с текущего файлового смещения, и файловое смещение увеличивается на количество прочитанных байт. Если текущее файловое смещение находится за концом файла, то ничего не читается и **read**() возвращает ноль.

Если значение *count* равно 0, то **read**() *может* обнаружить ошибки, описанные далее. При отсутствии ошибок, или если **read**() не выполняет проверки, то **read**() с *count* равным 0 возвращает 0 и ничего не меняет.

Если значение *count* больше **SSIZE\_MAX**, то результат не определён.

При успешном выполнении возвращается количество прочитанных байт (ноль означает конец файла), а позиция в файле увеличивается на это значение.

4) **void \*mmap(void \***addr, **size\_t** length, **int** prot, **int** flags, **int** fd, **off\_t** offset);

**mmap()** создает новое отображение в виртуальном адресном пространстве вызывающего процесса. Начальный адрес для нового сопоставления указан в addr. Аргумент length задает длину сопоставления (которая должна быть больше 0). Если addr равен NULL, то ядро выбирает адрес (выровненный по странице), по которому будет создано сопоставление; это наиболее переносимый метод создания нового сопоставления. Если addr не равен NULL, то ядро воспринимает это как подсказку о том, где разместить отображение; в Linux ядро выберет ближайшую границу страницы (но всегда выше или равна значению, указанному /proc/sys/vm/mmap\_min\_addr) и попытайтесь создать там сопоставление. Если там уже существует другое сопоставление, ядро выбирает новый адрес, который может зависеть от подсказки, а может и не зависеть. Адрес нового сопоставления возвращается в результате вызова.

# 

# **Демонстрация работы**

execve("./parent", ["./parent"], 0x7ffc0ecc7770 /\* 71 vars \*/) = 0 brk(NULL) = 0x562706e4d000 arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffc7f0c8890) = -1 EINVAL (Недопустимый аргумент) 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 fstat(3, {st\_mode=S\_IFREG|0644, st\_size=61656, ...}) = 0 mmap(NULL, 61656, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f5d650f9000 close(3) = 0 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.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 \341\t\0\0\0\0\0"..., 832) = 832 fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1956992, ...}) = 0 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5d650f7000 mmap(NULL, 1972224, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5d64f15000 mprotect(0x7f5d64fab000, 1290240, PROT\_NONE) = 0 mmap(0x7f5d64fab000, 987136, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x96000) = 0x7f5d64fab000 mmap(0x7f5d6509c000, 299008, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x187000) = 0x7f5d6509c000 mmap(0x7f5d650e6000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d0000) = 0x7f5d650e6000 mmap(0x7f5d650f4000, 10240, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5d650f4000 close(3) = 0 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_[s.so](https://s.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\3405\0\0\0\0\0\0"..., 832) = 832 fstat(3, {st\_mode=S\_IFREG|0644, st\_size=104984, ...}) = 0 mmap(NULL, 107592, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5d64efa000 mmap(0x7f5d64efd000, 73728, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f5d64efd000 mmap(0x7f5d64f0f000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x15000) = 0x7f5d64f0f000 mmap(0x7f5d64f13000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f5d64f13000 close(3) = 0 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/[libc.so](https://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\300A\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\346x\376T\245\322\302\t/\216G\353\v3\20^"..., 68, 880) = 68 fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2029592, ...}) = 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\346x\376T\245\322\302\t/\216G\353\v3\20^"..., 68, 880) = 68 mmap(NULL, 2037344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5d64d08000 mmap(0x7f5d64d2a000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x22000) = 0x7f5d64d2a000 mmap(0x7f5d64ea2000, 319488, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19a000) = 0x7f5d64ea2000 mmap(0x7f5d64ef0000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f5d64ef0000 mmap(0x7f5d64ef6000, 13920, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5d64ef6000 close(3) = 0 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/[libm.so](https://libm.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\300\323\0\0\0\0\0\0"..., 832) = 832 fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1369384, ...}) = 0 mmap(NULL, 1368336, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5d64bb9000 mmap(0x7f5d64bc6000, 684032, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xd000) = 0x7f5d64bc6000 mmap(0x7f5d64c6d000, 626688, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xb4000) = 0x7f5d64c6d000 mmap(0x7f5d64d06000, 8192, PROT\_READ

execve("./main", ["./main", "3"], 0x7ffc80c88818 /\* 30 vars \*/) = 0

brk(NULL) = 0x564727a2c000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffc36876410) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe20f83f000

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=16483, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 16483, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fe20f83a000

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) = 0x7fe20f612000

mmap(0x7fe20f63a000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fe20f63a000

mmap(0x7fe20f7cf000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fe20f7cf000

mmap(0x7fe20f827000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7fe20f827000

mmap(0x7fe20f82d000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe20f82d000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe20f60f000

arch\_prctl(ARCH\_SET\_FS, 0x7fe20f60f740) = 0

set\_tid\_address(0x7fe20f60fa10) = 4775

set\_robust\_list(0x7fe20f60fa20, 24) = 0

rseq(0x7fe20f6100e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fe20f827000, 16384, PROT\_READ) = 0

mprotect(0x564727246000, 4096, PROT\_READ) = 0

mprotect(0x7fe20f879000, 8192, PROT\_READ) = 0

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

munmap(0x7fe20f83a000, 16483) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}, AT\_EMPTY\_PATH) = 0

getrandom("\x1d\x3f\xce\xfa\x8b\x9a\x55\xcf", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x564727a2c000

brk(0x564727a4d000) = 0x564727a4d000

newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}, AT\_EMPTY\_PATH) = 0

write(1, "length of arrays N: ", 20) = 20

read(0, "2\n", 1024) = 2

write(1, "number of arrays K: ", 20) = 20

read(0, "3\n", 1024) = 2

read(0, "1 2 3\n", 1024) = 6

read(0, "4 5 6\n", 1024) = 6

write(1, "horizontal\n", 11) = 11

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7fe20f6a3870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7fe20f654520}, NULL, 8) = 0

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fe20ee0e000

mprotect(0x7fe20ee0f000, 8388608, PROT\_READ|PROT\_WRITE) = 0

rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fe20f60e910, parent\_tid=0x7fe20f60e910, exit\_signal=0, stack=0x7fe20ee0e000, stack\_size=0x7fff00, tls=0x7fe20f60e640} => {parent\_tid=[0]}, 88) = 4848

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fe20e60d000

mprotect(0x7fe20e60e000, 8388608, PROT\_READ|PROT\_WRITE) = 0

rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_

SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fe20ee0d910, parent\_tid=0x7fe20ee0d910, exit\_signal=0, stack=0x7fe20e60d000, stack\_size=0x7fff00, tls=0x7fe20ee0d640} => {parent\_tid=[0]}, 88) = 4849

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fe20de0c000

mprotect(0x7fe20de0d000, 8388608, PROT\_READ|PROT\_WRITE) = 0

rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fe20e60c910, parent\_tid=0x7fe20e60c910, exit\_signal=0, stack=0x7fe20de0c000, stack\_size=0x7fff00, tls=0x7fe20e60c640} => {parent\_tid=[0]}, 88) = 4850

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

write(1, "9 12 \n", 6) = 6

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

# **Выводы**

Strace - это инструмент для Linux, который отслеживает системные вызовы, представляющие собой интерфейс между процессом и операционной системой. При помощи Strace можно выяснить, какие действия выполняет процесс в текущий момент. Этот инструмент может быть очень полезен при отладке программ.