Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Группа: М8О-215Б-23

Студент: Лизунов К.Р.

Преподаватель: Миронов Е.С.

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

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

Москва, 2024

**Постановка задачи**

**Вариант 15.**

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы инвертируют строки.

Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* **pid\_t fork(void)** — создаёт дочерний процесс. В программе используется для создания двух дочерних процессов, каждый из которых будет выполнять свою часть задачи.
* **int execl(const char \*path, const char \*arg, ...)** — загружает и исполняет новый образ программы. Дочерние процессы запускают программу child, передавая имя объекта разделяемой памяти в качестве аргумента.
* **int open(const char \*pathname, int flags, mode\_t mode)** — открытие/создание файла. В родительском процессе открываются два файла для записи результатов обработки данных дочерними процессами.
* **close(int fd)** — закрывает файл. Используется для закрытия всех открытых файлов в родительском и дочерних процессах после их использования.
* **int shm\_open(const char \*name, int oflag, mode\_t mode)** — создаёт или открывает разделяемую память. Родительский процесс создаёт два объекта разделяемой памяти для обмена данными с каждым из дочерних процессов.
* **int shm\_unlink(const char \*name)** — удаляет разделяемую память по имени. Используется для очистки системных ресурсов после завершения работы всех процессов.
* **int ftruncate(int fd, off\_t length)** — изменяет размер открытого файла. Применяется для установки размера разделяемой памяти, достаточного для размещения структуры данных.
* **void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset)** — сопоставляет область памяти с файлом. Используется для отображения разделяемой памяти в адресное пространство процессов, чтобы они могли взаимодействовать с ней.
* **int munmap(void \*addr, size\_t length)** — отменяет сопоставление области памяти. Применяется для отмены отображения разделяемой памяти после завершения работы.
* **int sem\_post(sem\_t \*sem)** — сигнализирует (разблокирует) семафор. Родительский процесс использует sem\_post для подачи сигнала дочернему процессу, что данные готовы для обработки.
* **int sem\_wait(sem\_t \*sem)** — ожидает (блокируется) на семафоре. Дочерний процесс использует sem\_wait, чтобы дождаться сигнала от родителя перед началом обработки данных.

**Алгоритм работы программы**

Программа создает два дочерних процесса, используя fork(). Родительский процесс устанавливает общую память (shared memory) с помощью shm\_open и отображает её в адресное пространство с помощью mmap. Для синхронизации между процессами используются семафоры. Родительский процесс считывает строки ввода, распределяет их между дочерними процессами в зависимости от длины строки, а дочерние процессы обрабатывают данные (реверсируют строки) и записывают результат в соответствующие файлы.

**Код программы**

**main.cpp**

#include <iostream>

#include <string>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/mman.h>

#include <semaphore.h>

#include <sys/wait.h>

#include <cstring>

#include <cstdlib>

//обмен данными через отображаемый файл

struct shared\_data {

    sem\_t sem\_parent;

    sem\_t sem\_child;

    char buffer[1024];

    int terminate;

};

int main() {

    std::cout << "name for child process 1: ";

    std::string file1\_name;

    std::getline(std::cin, file1\_name);

    std::cout << "name for child process 2: ";

    std::string file2\_name;

    std::getline(std::cin, file2\_name);

    int file1 = open(file1\_name.c\_str(), O\_WRONLY | O\_CREAT | O\_TRUNC, 0644);

    int file2 = open(file2\_name.c\_str(), O\_WRONLY | O\_CREAT | O\_TRUNC, 0644);

    if (file1 < 0 || file2 < 0) {

        perror("Can't open file");

        exit(1);

    }

    const char \*shm\_name1 = "/shm\_child1";

    const char \*shm\_name2 = "/shm\_child2";

    int shm\_fd1 = shm\_open(shm\_name1, O\_CREAT | O\_RDWR, 0666);

    int shm\_fd2 = shm\_open(shm\_name2, O\_CREAT | O\_RDWR, 0666);

    if (shm\_fd1 == -1 || shm\_fd2 == -1) {

        perror("Can't create shared memory object");

        exit(1);

    }

    ftruncate(shm\_fd1, sizeof(shared\_data));

    ftruncate(shm\_fd2, sizeof(shared\_data));

    shared\_data \*shm\_ptr1 = (shared\_data \*) mmap(NULL, sizeof(shared\_data), PROT\_READ | PROT\_WRITE, MAP\_SHARED, shm\_fd1, 0);

    shared\_data \*shm\_ptr2 = (shared\_data \*) mmap(NULL, sizeof(shared\_data), PROT\_READ | PROT\_WRITE, MAP\_SHARED, shm\_fd2, 0);

    if (shm\_ptr1 == MAP\_FAILED || shm\_ptr2 == MAP\_FAILED) {

        perror("Can't mmap shared memory");

        exit(1);

    }

    sem\_init(&shm\_ptr1->sem\_parent, 1, 0);

    sem\_init(&shm\_ptr1->sem\_child, 1, 0);

    sem\_init(&shm\_ptr2->sem\_parent, 1, 0);

    sem\_init(&shm\_ptr2->sem\_child, 1, 0);

    shm\_ptr1->terminate = 0;

    shm\_ptr2->terminate = 0;

    pid\_t pid1 = fork();

    if (pid1 < 0) {

        perror("Can't fork");

        exit(1);

    }

    if (pid1 == 0) {

        munmap(shm\_ptr2, sizeof(shared\_data));

        close(shm\_fd2);

        if (dup2(file1, STDOUT\_FILENO) < 0) {

            perror("Can't redirect stdout for child process 1");

            exit(1);

        }

        close(file1);

        close(file2);

        execl("./child", "./child", shm\_name1, NULL);

        perror("Can't execute child process 1");

        exit(1);

    }

    pid\_t pid2 = fork();

    if (pid2 < 0) {

        perror("Can't fork");

        exit(1);

    }

    if (pid2 == 0) {

        munmap(shm\_ptr1, sizeof(shared\_data));

        close(shm\_fd1);

        if (dup2(file2, STDOUT\_FILENO) < 0) {

            perror("Can't redirect stdout for child process 2");

            exit(1);

        }

        close(file2);

        close(file1);

        execl("./child", "./child", shm\_name2, NULL);

        perror("Can't execute child process 2");

        exit(1);

    }

    close(file1);

    close(file2);

    while (true) {

        std::string s;

        std::getline(std::cin, s);

        if (s.empty()) {

            shm\_ptr1->terminate = 1;

            shm\_ptr2->terminate = 1;

            sem\_post(&shm\_ptr1->sem\_parent);

            sem\_post(&shm\_ptr2->sem\_parent);

            break;

        }

        if (s.size() > 10) {

            strcpy(shm\_ptr2->buffer, s.c\_str());

            sem\_post(&shm\_ptr2->sem\_parent);

            sem\_wait(&shm\_ptr2->sem\_child);

        } else {

            strcpy(shm\_ptr1->buffer, s.c\_str());

            sem\_post(&shm\_ptr1->sem\_parent);

            sem\_wait(&shm\_ptr1->sem\_child);

        }

    }

    waitpid(pid1, NULL, 0);

    waitpid(pid2, NULL, 0);

    sem\_destroy(&shm\_ptr1->sem\_parent);

    sem\_destroy(&shm\_ptr1->sem\_child);

    sem\_destroy(&shm\_ptr2->sem\_parent);

    sem\_destroy(&shm\_ptr2->sem\_child);

    munmap(shm\_ptr1, sizeof(shared\_data));

    munmap(shm\_ptr2, sizeof(shared\_data));

    close(shm\_fd1);

    close(shm\_fd2);

    shm\_unlink(shm\_name1);

    shm\_unlink(shm\_name2);

    return 0;

}

**Child.cpp**

#include <iostream>

#include <string>

#include <algorithm>

#include <fcntl.h>

#include <unistd.h>

#include <sys/mman.h>

#include <semaphore.h>

#include <sys/stat.h>

#include <cstring>

#include <cstdlib>

struct shared\_data {

    sem\_t sem\_parent;

    sem\_t sem\_child;

    char buffer[1024];

    int terminate;

};

int main(int argc, char \*argv[]) {

    if (argc != 2) {

        std::cerr << "Usage: ./child <shm\_name>" << std::endl;

        return 1;

    }

    const char \*shm\_name = argv[1];

    int shm\_fd = shm\_open(shm\_name, O\_RDWR, 0666);

    if (shm\_fd == -1) {

        perror("Can't open shared memory object");

        exit(1);

    }

    shared\_data \*shm\_ptr = (shared\_data \*) mmap(NULL, sizeof(shared\_data), PROT\_READ | PROT\_WRITE, MAP\_SHARED, shm\_fd, 0);

    if (shm\_ptr == MAP\_FAILED) {

        perror("Can't mmap shared memory");

        exit(1);

    }

    while (true) {

        sem\_wait(&shm\_ptr->sem\_parent);

        if (shm\_ptr->terminate) {

            sem\_post(&shm\_ptr->sem\_child);

            break;

        }

        std::string str(shm\_ptr->buffer);

        std::reverse(str.begin(), str.end());

        std::cout << str << std::endl;

        std::cout.flush();

        sem\_post(&shm\_ptr->sem\_child);

    }

    munmap(shm\_ptr, sizeof(shared\_data));

    close(shm\_fd);

    return 0;

}

**Протокол работы программы**

**Тестирование:**

user@MacBook-Air-User build % ./main

name for child process 1: one

name for child process 2: two

123456789

12345678900

**Первое число запишется в one, второе в two**  
  
  
**strace**

execve("./main", ["./main"], 0x7ffda58b1c88 /\* 26 vars \*/) = 0

brk(NULL) = 0xb564000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fdc4757e000

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=25258, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fdc47577000

close(3) = 0

openat(AT\_FDCWD, "/usr/local/lib64/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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fdc47309000

mmap(0x7fdc473ae000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000) = 0x7fdc473ae000

mmap(0x7fdc474d7000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7fdc474d7000

mmap(0x7fdc47565000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000) = 0x7fdc47565000

mmap(0x7fdc47573000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fdc47573000

close(3) = 0

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fdc4722a000

mmap(0x7fdc4723a000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000) = 0x7fdc4723a000

mmap(0x7fdc472ad000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000) = 0x7fdc472ad000

mmap(0x7fdc47307000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000) = 0x7fdc47307000

close(3) = 0

openat(AT\_FDCWD, "/usr/local/lib64/libgcc\_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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fdc471fd000

mmap(0x7fdc47201000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7fdc47201000

mmap(0x7fdc47224000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000) = 0x7fdc47224000

mmap(0x7fdc47228000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000) = 0x7fdc47228000

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\20t\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

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, 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, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fdc4701c000

mmap(0x7fdc47042000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000) = 0x7fdc47042000

mmap(0x7fdc47197000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000) = 0x7fdc47197000

mmap(0x7fdc471ea000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7fdc471ea000

mmap(0x7fdc471f0000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fdc471f0000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fdc4701a000

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fdc47017000

arch\_prctl(ARCH\_SET\_FS, 0x7fdc47017740) = 0

set\_tid\_address(0x7fdc47017a10) =

set\_robust\_list(0x7fdc47017a20, 24) = 0

rseq(0x7fdc47018060, 0x20, 0, 0x53053053) = 0

mprotect(0x7fdc471ea000, 16384, PROT\_READ) = 0

mprotect(0x7fdc47228000, 4096, PROT\_READ) = 0

mprotect(0x7fdc47307000, 4096, PROT\_READ) = 0

mprotect(0x7fdc47565000, 45056, PROT\_READ) = 0

mprotect(0x404000, 4096, PROT\_READ) = 0

mprotect(0x7fdc475b0000, 8192, PROT\_READ) = 0

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

munmap(0x7fdc47577000, 25258) = 0

futex(0x7fdc4757373c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

getrandom("\xfb\x4d\xfb\x03\x28\xc8\xfb\x2e", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0xb564000

brk(0xb585000) = 0xb585000

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

write(1, "Enter file's name for child proc"..., 39) = 39

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

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

write(1, "Enter file's name for child proc"..., 39) = 39

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

openat(AT\_FDCWD, "o", O\_WRONLY|O\_CREAT|O\_TRUNC, 0644) = 3

openat(AT\_FDCWD, "oo", O\_WRONLY|O\_CREAT|O\_TRUNC, 0644) = 4

openat(AT\_FDCWD, "/dev/shm/shm\_child1", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 5

openat(AT\_FDCWD, "/dev/shm/shm\_child2", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 6

ftruncate(5, 1096) = 0

ftruncate(6, 1096) = 0

mmap(NULL, 1096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0x7fdc4757d000

mmap(NULL, 1096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 6, 0) = 0x7fdc4757c000

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fdc47017a10) =

set\_robust\_list(0x7fdc47017a20, 24 <unfinished ...>

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD <unfinished ...>

<... set\_robust\_list resumed>) = 0

<... clone resumed>, child\_tidptr=0x7fdc47017a10) =

set\_robust\_list(0x7fdc47017a20, 24 <unfinished ...>

munmap(0x7fdc4757c000, 1096 <unfinished ...>

close(3 <unfinished ...>

<... set\_robust\_list resumed>) = 0

<... close resumed>) = 0

<... munmap resumed>) = 0

close(4 <unfinished ...>

munmap(0x7fdc4757d000, 1096 <unfinished ...>

<... close resumed>) = 0

close(6 <unfinished ...>

read(0, <unfinished ...>

<... munmap resumed>) = 0

<... close resumed>) = 0

close(5 <unfinished ...>

dup2(3, 1 <unfinished ...>

<... close resumed>) = 0

<... dup2 resumed>) = 1

dup2(4, 1 <unfinished ...>

close(3 <unfinished ...>

<... dup2 resumed>) = 1

<... close resumed>) = 0

close(4 <unfinished ...>

close(4 <unfinished ...>

<... close resumed>) = 0

close(3 <unfinished ...>

<... close resumed>) = 0

<... close resumed>) = 0

execve("./child", ["./child", "/shm\_child1"], 0x7ffe26e4dce8 /\* 26 vars \*/ <unfinished ...>

execve("./child", ["./child", "/shm\_child2"], 0x7ffe26e4dce8 /\* 26 vars \*/ <unfinished ...>

<... execve resumed>) = 0

<... execve resumed>) = 0

brk(NULL <unfinished ...>

brk(NULL <unfinished ...>

<... brk resumed>) = 0x20503000

<... brk resumed>) = 0x3531000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b05448000

<... mmap resumed>) = 0x7fe8e7e52000

access("/etc/ld.so.preload", R\_OK <unfinished ...>

access("/etc/ld.so.preload", R\_OK <unfinished ...>

<... access resumed>) = -1 ENOENT (No such file or directory)

<... access resumed>) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

<... openat resumed>) = 3

<... openat resumed>) = 3

newfstatat(3, "", <unfinished ...>

newfstatat(3, "", <unfinished ...>

<... newfstatat resumed>{st\_mode=S\_IFREG|0644, st\_size=25258, ...}, AT\_EMPTY\_PATH) = 0

<... newfstatat resumed>{st\_mode=S\_IFREG|0644, st\_size=25258, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 3, 0 <unfinished ...>

mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 3, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b05441000

<... mmap resumed>) = 0x7fe8e7e4b000

close(3 <unfinished ...>

close(3 <unfinished ...>

<... close resumed>) = 0

<... close resumed>) = 0

openat(AT\_FDCWD, "/usr/local/lib64/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

openat(AT\_FDCWD, "/usr/local/lib64/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

<... openat resumed>) = 3

<... openat resumed>) = 3

read(3, <unfinished ...>

read(3, <unfinished ...>

<... read resumed>"\177ELF\2\1\1\3\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

<... read resumed>"\177ELF\2\1\1\3\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, "", <unfinished ...>

newfstatat(3, "", <unfinished ...>

<... newfstatat resumed>{st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

<... newfstatat resumed>{st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b051d3000

<... mmap resumed>) = 0x7fe8e7bdd000

mmap(0x7f2b05278000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000 <unfinished ...>

mmap(0x7fe8e7c82000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000 <unfinished ...>

<... mmap resumed>) = 0x7f2b05278000

<... mmap resumed>) = 0x7fe8e7c82000

mmap(0x7f2b053a1000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000 <unfinished ...>

mmap(0x7fe8e7dab000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000 <unfinished ...>

<... mmap resumed>) = 0x7f2b053a1000

<... mmap resumed>) = 0x7fe8e7dab000

mmap(0x7f2b0542f000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000 <unfinished ...>

mmap(0x7fe8e7e39000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000 <unfinished ...>

<... mmap resumed>) = 0x7f2b0542f000

<... mmap resumed>) = 0x7fe8e7e39000

mmap(0x7f2b0543d000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

mmap(0x7fe8e7e47000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b0543d000

<... mmap resumed>) = 0x7fe8e7e47000

close(3 <unfinished ...>

close(3 <unfinished ...>

<... close resumed>) = 0

<... close resumed>) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

<... openat resumed>) = 3

<... openat resumed>) = 3

read(3, <unfinished ...>

read(3, <unfinished ...>

<... read resumed>"\177ELF\2\1\1\3\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

<... read resumed>"\177ELF\2\1\1\3\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, "", <unfinished ...>

newfstatat(3, "", <unfinished ...>

<... newfstatat resumed>{st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

<... newfstatat resumed>{st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b050f4000

<... mmap resumed>) = 0x7fe8e7afe000

mmap(0x7f2b05104000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000 <unfinished ...>

mmap(0x7fe8e7b0e000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000 <unfinished ...>

<... mmap resumed>) = 0x7f2b05104000

<... mmap resumed>) = 0x7fe8e7b0e000

mmap(0x7f2b05177000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000 <unfinished ...>

mmap(0x7fe8e7b81000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000 <unfinished ...>

<... mmap resumed>) = 0x7f2b05177000

<... mmap resumed>) = 0x7fe8e7b81000

mmap(0x7f2b051d1000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000 <unfinished ...>

mmap(0x7fe8e7bdb000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000 <unfinished ...>

<... mmap resumed>) = 0x7f2b051d1000

<... mmap resumed>) = 0x7fe8e7bdb000

close(3 <unfinished ...>

close(3 <unfinished ...>

<... close resumed>) = 0

<... close resumed>) = 0

openat(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC <unfinished ...>

openat(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC <unfinished ...>

<... openat resumed>) = 3

<... openat resumed>) = 3

read(3, <unfinished ...>

read(3, <unfinished ...>

<... read resumed>"\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

<... read resumed>"\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, "", <unfinished ...>

newfstatat(3, "", <unfinished ...>

<... newfstatat resumed>{st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

<... newfstatat resumed>{st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b050c7000

<... mmap resumed>) = 0x7fe8e7ad1000

mmap(0x7f2b050cb000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000 <unfinished ...>

mmap(0x7fe8e7ad5000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000 <unfinished ...>

<... mmap resumed>) = 0x7f2b050cb000

<... mmap resumed>) = 0x7fe8e7ad5000

mmap(0x7f2b050ee000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000 <unfinished ...>

mmap(0x7fe8e7af8000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000 <unfinished ...>

<... mmap resumed>) = 0x7f2b050ee000

<... mmap resumed>) = 0x7fe8e7af8000

mmap(0x7f2b050f2000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000 <unfinished ...>

mmap(0x7fe8e7afc000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000 <unfinished ...>

<... mmap resumed>) = 0x7f2b050f2000

<... mmap resumed>) = 0x7fe8e7afc000

close(3 <unfinished ...>

close(3 <unfinished ...>

<... close resumed>) = 0

<... close resumed>) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

<... openat resumed>) = 3

<... openat resumed>) = 3

read(3, <unfinished ...>

read(3, <unfinished ...>

<... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) = 832

<... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) = 832

pread64(3, <unfinished ...>

pread64(3, <unfinished ...>

<... pread64 resumed>"\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 resumed>"\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

newfstatat(3, "", <unfinished ...>

newfstatat(3, "", <unfinished ...>

<... newfstatat resumed>{st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, AT\_EMPTY\_PATH) = 0

<... newfstatat resumed>{st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, AT\_EMPTY\_PATH) = 0

pread64(3, <unfinished ...>

pread64(3, <unfinished ...>

<... pread64 resumed>"\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 resumed>"\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, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

mmap(NULL, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b04ee6000

<... mmap resumed>) = 0x7fe8e78f0000

mmap(0x7f2b04f0c000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000 <unfinished ...>

mmap(0x7fe8e7916000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000 <unfinished ...>

<... mmap resumed>) = 0x7f2b04f0c000

<... mmap resumed>) = 0x7fe8e7916000

mmap(0x7f2b05061000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000 <unfinished ...>

mmap(0x7fe8e7a6b000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000 <unfinished ...>

<... mmap resumed>) = 0x7f2b05061000

<... mmap resumed>) = 0x7fe8e7a6b000

mmap(0x7f2b050b4000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000 <unfinished ...>

mmap(0x7fe8e7abe000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000 <unfinished ...>

<... mmap resumed>) = 0x7f2b050b4000

<... mmap resumed>) = 0x7fe8e7abe000

mmap(0x7f2b050ba000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

mmap(0x7fe8e7ac4000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b050ba000

<... mmap resumed>) = 0x7fe8e7ac4000

close(3 <unfinished ...>

close(3 <unfinished ...>

<... close resumed>) = 0

<... close resumed>) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b04ee4000

<... mmap resumed>) = 0x7fe8e78ee000

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

<... mmap resumed>) = 0x7f2b04ee1000

<... mmap resumed>) = 0x7fe8e78eb000

arch\_prctl(ARCH\_SET\_FS, 0x7f2b04ee1740 <unfinished ...>

arch\_prctl(ARCH\_SET\_FS, 0x7fe8e78eb740 <unfinished ...>

<... arch\_prctl resumed>) = 0

<... arch\_prctl resumed>) = 0

set\_tid\_address(0x7f2b04ee1a10 <unfinished ...>

set\_tid\_address(0x7fe8e78eba10 <unfinished ...>

<... set\_tid\_address resumed>) =

<... set\_tid\_address resumed>) =

set\_robust\_list(0x7f2b04ee1a20, 24 <unfinished ...>

set\_robust\_list(0x7fe8e78eba20, 24 <unfinished ...>

<... set\_robust\_list resumed>) = 0

<... set\_robust\_list resumed>) = 0

rseq(0x7f2b04ee2060, 0x20, 0, 0x53053053 <unfinished ...>

rseq(0x7fe8e78ec060, 0x20, 0, 0x53053053 <unfinished ...>

<... rseq resumed>) = 0

<... rseq resumed>) = 0

mprotect(0x7f2b050b4000, 16384, PROT\_READ <unfinished ...>

mprotect(0x7fe8e7abe000, 16384, PROT\_READ <unfinished ...>

<... mprotect resumed>) = 0

<... mprotect resumed>) = 0

mprotect(0x7f2b050f2000, 4096, PROT\_READ <unfinished ...>

mprotect(0x7fe8e7afc000, 4096, PROT\_READ <unfinished ...>

<... mprotect resumed>) = 0

<... mprotect resumed>) = 0

mprotect(0x7f2b051d1000, 4096, PROT\_READ <unfinished ...>

mprotect(0x7fe8e7bdb000, 4096, PROT\_READ <unfinished ...>

<... mprotect resumed>) = 0

<... mprotect resumed>) = 0

mprotect(0x7fe8e7e39000, 45056, PROT\_READ <unfinished ...>

mprotect(0x7f2b0542f000, 45056, PROT\_READ <unfinished ...>

<... mprotect resumed>) = 0

<... mprotect resumed>) = 0

mprotect(0x405000, 4096, PROT\_READ <unfinished ...>

mprotect(0x405000, 4096, PROT\_READ <unfinished ...>

<... mprotect resumed>) = 0

<... mprotect resumed>) = 0

mprotect(0x7fe8e7e84000, 8192, PROT\_READ <unfinished ...>

mprotect(0x7f2b0547a000, 8192, PROT\_READ <unfinished ...>

<... mprotect resumed>) = 0

<... mprotect resumed>) = 0

prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

<... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

<... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7fe8e7e4b000, 25258 <unfinished ...>

munmap(0x7f2b05441000, 25258 <unfinished ...>

<... munmap resumed>) = 0

<... munmap resumed>) = 0

futex(0x7fe8e7e4773c, FUTEX\_WAKE\_PRIVATE, 2147483647 <unfinished ...>

futex(0x7f2b0543d73c, FUTEX\_WAKE\_PRIVATE, 2147483647 <unfinished ...>

<... futex resumed>) = 0

<... futex resumed>) = 0

getrandom( <unfinished ...>

getrandom( <unfinished ...>

<... getrandom resumed>"\x9d\xd4\xf3\x2b\x37\x8d\x93\x84", 8, GRND\_NONBLOCK) = 8

<... getrandom resumed>"\x17\x39\xdc\x73\x5a\xf9\xd8\x5b", 8, GRND\_NONBLOCK) = 8

brk(NULL <unfinished ...>

brk(NULL <unfinished ...>

<... brk resumed>) = 0x3531000

<... brk resumed>) = 0x20503000

brk(0x3552000 <unfinished ...>

brk(0x20524000 <unfinished ...>

<... brk resumed>) = 0x3552000

<... brk resumed>) = 0x20524000

openat(AT\_FDCWD, "/dev/shm/shm\_child2", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC <unfinished ...>

openat(AT\_FDCWD, "/dev/shm/shm\_child1", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC <unfinished ...>

<... openat resumed>) = 3

<... openat resumed>) = 3

mmap(NULL, 1096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0 <unfinished ...>

mmap(NULL, 1096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0 <unfinished ...>

<... mmap resumed>) = 0x7fe8e7e51000

<... mmap resumed>) = 0x7f2b05447000

futex(0x7fe8e7e51000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

futex(0x7f2b05447000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

<... read resumed>"1234567890345678\n", 1024) = 17

futex(0x7fdc4757c000, FUTEX\_WAKE, 1) = 1

<... futex resumed>) = 0

futex(0x7fdc4757c020, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

newfstatat(1, "", {st\_mode=S\_IFREG|0644, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

write(1, "8765430987654321\n", 17) = 17

futex(0x7fe8e7e51020, FUTEX\_WAKE, 1 <unfinished ...>

<... futex resumed>) = 0

<... futex resumed>) = 1

read(0, <unfinished ...>

futex(0x7fe8e7e51000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

<... read resumed>"123456\n", 1024) = 7

futex(0x7fdc4757d000, FUTEX\_WAKE, 1) = 1

<... futex resumed>) = 0

futex(0x7fdc4757d020, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

newfstatat(1, "", {st\_mode=S\_IFREG|0644, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

write(1, "654321\n", 7) = 7

futex(0x7f2b05447020, FUTEX\_WAKE, 1 <unfinished ...>

<... futex resumed>) = 0

<... futex resumed>) = 1

read(0, <unfinished ...>

futex(0x7f2b05447000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

<... read resumed>"\n", 1024) = 1

futex(0x7fdc4757d000, FUTEX\_WAKE, 1) = 1

<... futex resumed>) = 0

futex(0x7fdc4757c000, FUTEX\_WAKE, 1 <unfinished ...>

munmap(0x7f2b05447000, 1096 <unfinished ...>

<... futex resumed>) = 1

<... futex resumed>) = 0

<... munmap resumed>) = 0

wait4( , <unfinished ...>

munmap(0x7fe8e7e51000, 1096 <unfinished ...>

close(3 <unfinished ...>

<... munmap resumed>) = 0

<... close resumed>) = 0

close(3 <unfinished ...>

exit\_group(0 <unfinished ...>

<... close resumed>) = 0

<... exit\_group resumed>) = ?

exit\_group(0) = ?

+++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) =

+++ exited with 0 +++

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

wait4( , NULL, 0, NULL) =

munmap(0x7fdc4757d000, 1096) = 0

munmap(0x7fdc4757c000, 1096) = 0

close(5) = 0

close(6) = 0

unlink("/dev/shm/shm\_child1") = 0

unlink("/dev/shm/shm\_child2") = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Лабораторная работа демонстрирует использование разделяемой памяти (shared memory) и

семафоров для взаимодействия между процессами в Linux. Программа корректно обрабатывает ввод пользователя, синхронизирует работу процессов и завершается без утечек ресурсов.