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: М80-206Б-22

Студент: Седов М.Д

Преподаватель: Миронов Е.С.

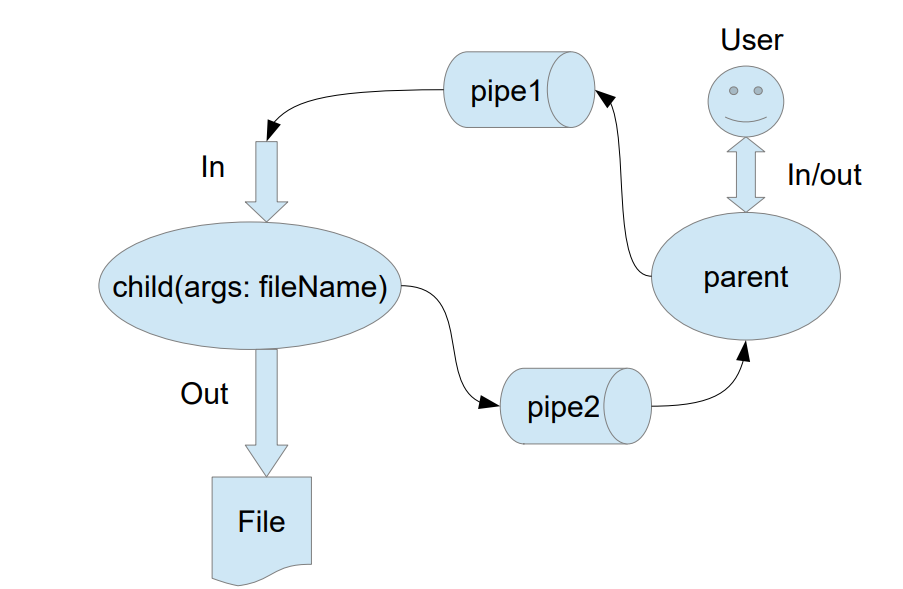
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 02.12.2023

Москва, 2023

**Постановка задачи**

**Вариант 5.**



Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс представлены разными программами. Пользователь вводит команды вида: «число<endline>». Родительский процесс передает команды пользователя через общую область памяти, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс производит проверку на простоту. Если число составное, то в это число записывается в созданный/открытый им файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются, при этом дочерний процесс передает сигнал в родительский процесс через общую область памяти.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* int pipe(int \*fd); – создает канал для связи процессов.
* int close(int fd); - закрывает файловый дескриптор.
* int execl(const char\* path, const char\* age, …); - запускает исполняемый файл
* int open(const char \*\_\_file, int \_\_oflag, ...); - открывает/создает файл.
* void\* mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset) создает новое отображение в виртуальном адресном пространстве вызывающего процесса.
* int munmap(void \*addr, size\_t length) – освобождает область памяти длиной length начиная с адресса addr.
* void \*memcpy(void \* dest, const void \* src, size\_t n) – копирует n байт в dest из src.
* int shm\_open(const char \* name, int flag, mode\_t mode) – открывает сегмент разделяемой памяти, возвращая дескриптор.
* sem\_t \*sem\_open(const char \* name, int oflag, ...) – открывает семафор для синхронизации, возвращая дескриптор.
* int sem\_post(sem\_t \* sem) – отправляет сигнал.
* int sem\_wait(sem\_t \* sem) – ожидает сигнал.
* int shm\_unlink(const char \* name) – удаляет сегмент разделяемой памяти.
* int sem\_close(sem\_t \* sem) – закрывает дескриптор семафора.
* int sem\_unlink(const char \* name) – удаляет имя семафора.

Cоздал дескриптор {fd} для взаимодействия с разделяемой памятью с помошью shm\_open() и отобразил в виртуальную память с помощью mmap() затем создал дочерний процесс в родительском с помощью fork() . Организовал ветвление с помощью идентификатора процесса {pid}. В родительском считал имя файла и создал дескриптор {fd\_fname}, затем отобразил его в виртуальную область памяти. Дальше создаю семафоры для синхронизации процессов для файла {fmem\_sem} и для синхронизации памяти “mem” {mem\_sem1}, {mem\_sem2}, копирую название файла в общую область памяти с помощью memcpy() и отправляю сигнал в семафор файла с помощью sem\_post(). В дочернем вызвал исполняемый файл child с помощью execl().Затем в нем создаю и отображаю в ту же область памяти для дескриптора {fd}, создаю те же семафоры и затем получаю сигнал в семафор файла с помощью sem\_wait(). Далее отображаю в ту же область памяти для дескриптора {fd\_fname} и получаю имя файла, затем открываю этот файл с помощью open(). В родительском процессе считываю числа в общую область памяти, затем отправляю сигнал семафору для продолжения работы уже в родительском процессе и ожидаю сигнал семафору, чтобы считать флаг об успехе/неуспехе от дочернего процесса . В это время в дочерний процесс получает сигнал семафора, проверяет число из общей памяти на то, что они составные и записывает число в файл в случае успеха, отправяя сигнал или отправляет сигнал в родительский. Дальше если ввод в консоль прекращен, или число оказалось не составным, дочерний и родительские процессы завершаются, освобождая все области памяти, закрывая все файловые дескрипторы и семафоры.

**Код программы**

**main.cpp**

#include <iostream>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <semaphore.h>

#include <string.h>

#include <stdio.h>

int create\_process() {

    pid\_t pid = fork();

    if (pid == -1) {

        perror("fork");

        exit(-1);

    }

    return pid;

}

int main() {

    const int mem\_size = 2 \* sizeof(int);

    const char\* mem\_name = "/mem";

    int fd = shm\_open(mem\_name, O\_CREAT | O\_RDWR, 0666);

    if (fd == -1) {

        perror("shm\_open");

        exit(-1);

    }

    ftruncate(fd, mem\_size);

    void\* addr = mmap(0, mem\_size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

    if (addr == MAP\_FAILED) {

        perror("mmap");

        exit(-1);

    }

    int pid = create\_process();

    if (pid == 0) {

        execl("./child", "./child", NULL);

        perror("execl");

        exit(-1);

    } else {

        int\* num\_ptr = (int\*)addr;

        int\* flag\_ptr = num\_ptr + 1;

        char fname[40];

        printf("Enter file name:\n");

        scanf("%39s", fname);

        const char\* fmem\_name = "/filename";

        int fd\_fname = shm\_open(fmem\_name, O\_CREAT | O\_RDWR, 0666);

        if (fd\_fname == -1) {

            perror("shm\_open (fname)");

            exit(-1);

        }

        ftruncate(fd\_fname, sizeof(fname));

        char\* faddr = (char\*)mmap(0, sizeof(fname), PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd\_fname, 0);

        if (faddr == MAP\_FAILED) {

            perror("mmap (fname)");

            exit(-1);

        }

        sem\_t\* fmem\_sem = sem\_open("/fmem\_semaphore", O\_CREAT, 0666, 0);

        if (fmem\_sem == SEM\_FAILED) {

            perror("Unable to create fmem\_semaphore");

            exit(-1);

        }

        sem\_t\* mem\_sem1 = sem\_open("/mem\_semaphore1", O\_CREAT, 0666, 0);

        if (mem\_sem1 == SEM\_FAILED) {

            perror("Unable to create mem\_semaphore1");

            exit(-1);

        }

        sem\_t\* mem\_sem2 = sem\_open("/mem\_semaphore2", O\_CREAT, 0666, 0);

        if (mem\_sem2 == SEM\_FAILED) {

            perror("Unable to create mem\_semaphore2");

            exit(-1);

        }

        memcpy(faddr, fname, sizeof(fname));

        sem\_post(fmem\_sem);

        int num;

        printf("Enter num:\n");

        while (scanf("%d", &num) != EOF) {

            \*num\_ptr = num;

            sem\_post(mem\_sem2);

            sem\_wait(mem\_sem1);

            if (!(\*flag\_ptr)){

                printf("Finish processes\n");

                break;

            }

        }

        munmap(faddr, sizeof(fname));

        close(fd\_fname);

        shm\_unlink(fmem\_name);

        sem\_close(mem\_sem1);

        sem\_unlink("/mem\_semaphore1");

        sem\_close(mem\_sem2);

        sem\_unlink("/mem\_semaphore2");

        sem\_close(fmem\_sem);

        sem\_unlink("/fmem\_semaphore");

    }

    munmap(addr, mem\_size);

    close(fd);

    shm\_unlink(mem\_name);

    return 0;

}

**child.cpp**

#include <iostream>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <unistd.h>

#include <string.h>

#include <semaphore.h>

#include "composite.h"

int main() {

    const int mem\_size = 2 \* sizeof(int);

    const char\* mem\_name = "/mem";

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

    if (fd == -1) {

        perror("shm\_open child");

        exit(-1);

    }

    void\* addr = mmap(0, mem\_size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

    if (addr == MAP\_FAILED) {

        perror("mmap child");

        exit(-1);

    }

    int\* num\_ptr = (int\*)addr;

    int\* flag\_ptr = num\_ptr + 1;

    sem\_t\* fmem\_sem = sem\_open("/fmem\_semaphore", O\_CREAT, 0666, 0);

    if (fmem\_sem == SEM\_FAILED) {

        perror("Unable to open fmem\_semaphore");

        exit(-1);

    }

    sem\_t\* mem\_sem1 = sem\_open("/mem\_semaphore1", O\_CREAT, 0666, 0);

    if (mem\_sem1 == SEM\_FAILED) {

        perror("Unable to create mem\_semaphore1");

        exit(-1);

    }

    sem\_t\* mem\_sem2 = sem\_open("/mem\_semaphore2", O\_CREAT, 0666, 0);

        if (mem\_sem2 == SEM\_FAILED) {

            perror("Unable to create mem\_semaphore2");

            exit(-1);

        }

    sem\_wait(fmem\_sem);

    const char\* fmem\_name = "/filename";

    int fd\_fname = shm\_open(fmem\_name, O\_RDWR, 0666);

    if (fd\_fname == -1) {

        perror("shm\_open child (fname)");

        exit(-1);

    }

    char fname[40];

    char\* faddr = (char\*)mmap(0, sizeof(fname), PROT\_READ, MAP\_SHARED, fd\_fname, 0);

    if (faddr == MAP\_FAILED) {

        perror("mmap child (fname)");

        exit(-1);

    }

    memcpy(fname, faddr, sizeof(fname));

    int fdout = open(fname, O\_WRONLY | O\_CREAT | O\_TRUNC, 0777);

    if (fdout == -1) {

        perror("fdout child");

        exit(-1);

    }

    while (true) {

        sem\_wait(mem\_sem2);

        int num = \*num\_ptr;

        \*flag\_ptr = isComposite(num);

        if (!(\*flag\_ptr)) {

            sem\_post(mem\_sem1);

            break;

        }

        write(fdout, &num, sizeof(int));

        sem\_post(mem\_sem1);

    }

    sem\_close(mem\_sem1);

    sem\_unlink("/mem\_semaphore1");

    sem\_close(mem\_sem2);

    sem\_unlink("/mem\_semaphore2");

    munmap(addr, mem\_size);

    munmap(faddr, sizeof(fname));

    close(fd);

    close(fd\_fname);

    shm\_unlink(mem\_name);

    shm\_unlink(fmem\_name);

    close(fdout);

    return 0;

}

**composite.h**

#pragma once

#include <iostream>

int isComposite(int num);

**composite.cpp**

#include <iostream>

#include "composite.h"

int isComposite(int num){

    if(num <= 0){

        return 0;

    }

    if(num == 1){

        return 1;

    }

    for(int i = 2; i \* i <= num; i++){

        if(num % i == 0){

            return 1;

        }

    }

    return 0;

}

**Протокол работы программы**

**Тестирование:**

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./main

Enter file name:

toji

Enter num:

1000

26

64

3

Finish processes

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./flread

1000

26

64

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./main

Enter file name:

toji

Enter num:

46

86

90

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./flread

46

86

90

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./main

Enter file name:

toji

Enter num:

000000000000

Finish processes

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./flread

maksim@caseofpeace:~/course2/sem3/os/src/lab3$ ./main

Enter file name:

toji

Enter num:

-1000

Finish processes

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./flread

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./main

Enter file name:

toji

Enter num:

-+^#

Finish processes

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./flread

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./main

Enter file name:

toji

Enter num:

75

54

67

Finish processes

maksim@caseofpeace:~/course2/sem3/os/src/lab3 $ ./flread

75

54

**STRACE:**

maksim@caseofpeace:~/course2/sem3/os/lab3/src$ strace -f ./main

execve("./main", ["./main"], 0x7ffff1d4b568 /\* 31 vars \*/) = 0

brk(NULL) = 0x7fffd25fe000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffda1f5220) = -1 EINVAL (Invalid argument)

**mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f0b9a460000**

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=22607, ...}, AT\_EMPTY\_PATH) = 0

**mmap(NULL, 22607, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f0b9a46a000**

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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

**mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f0b9a230000**

mprotect(0x7f0b9a2ca000, 1576960, PROT\_NONE) = 0

**mmap(0x7f0b9a2ca000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7f0b9a2ca000**

**mmap(0x7f0b9a3db000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7f0b9a3db000**

**mmap(0x7f0b9a44b000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7f0b9a44b000**

**mmap(0x7f0b9a459000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f0b9a459000**

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\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 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) = 0x7f0b9a000000**

**mmap(0x7f0b9a028000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f0b9a028000**

**mmap(0x7f0b9a1bd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f0b9a1bd000**

**mmap(0x7f0b9a215000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f0b9a215000**

**mmap(0x7f0b9a21b000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f0b9a21b000**

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=940560, ...}, AT\_EMPTY\_PATH) = 0

**mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f0b99f10000**

**mmap(0x7f0b99f1e000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7f0b99f1e000**

**mmap(0x7f0b99f9a000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7f0b99f9a000**

**mmap(0x7f0b99ff5000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7f0b99ff5000**

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/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=125488, ...}, AT\_EMPTY\_PATH) = 0

**mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f0b99ef0000**

**mmap(0x7f0b99ef3000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f0b99ef3000**

**mmap(0x7f0b99f0a000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7f0b99f0a000**

**mmap(0x7f0b99f0e000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7f0b99f0e000**

close(3) = 0

**mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f0b99ee0000**

**mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f0b99ed0000**

arch\_prctl(ARCH\_SET\_FS, 0x7f0b99ed0740) = 0

set\_tid\_address(0x7f0b99ed0a10) = 9570

set\_robust\_list(0x7f0b99ed0a20, 24) = 0

rseq(0x7f0b99ed10e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)

mprotect(0x7f0b9a215000, 16384, PROT\_READ) = 0

mprotect(0x7f0b99f0e000, 4096, PROT\_READ) = 0

mprotect(0x7f0b99ff5000, 4096, PROT\_READ) = 0

mprotect(0x7f0b9a44b000, 45056, PROT\_READ) = 0

mprotect(0x7f0b9a4b2000, 4096, PROT\_READ) = 0

mprotect(0x7f0b9a4a8000, 8192, PROT\_READ) = 0

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

**munmap(0x7f0b9a46a000, 22607) = 0**

getrandom("\x99\x0b\x0b\x86\x4f\xe0\xa9\xd0", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x7fffd25fe000

brk(0x7fffd261f000) = 0x7fffd261f000

futex(0x7f0b9a45977c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

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

ftruncate(3, 8) = 0

**mmap(NULL, 8, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f0b9a4ae000**

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 9571 attached

, child\_tidptr=0x7f0b99ed0a10) = 9571

[pid 9571] set\_robust\_list(0x7f0b99ed0a20, 24 <unfinished ...>

[pid 9570] newfstatat(1, "", <unfinished ...>

[pid 9571] <... set\_robust\_list resumed>) = 0

[pid 9570] <... newfstatat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

[pid 9571] execve("./child", ["./child"], 0x7fffda1f53f8 /\* 31 vars \*/ <unfinished ...>

[pid 9570] write(1, "Enter file name:\n", 17Enter file name:

) = 17

[pid 9570] newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

[pid 9570] read(0, <unfinished ...>

[pid 9571] <... execve resumed>) = 0

[pid 9571] brk(NULL) = 0x7fffde766000

[pid 9571] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffe6df7ca0) = -1 EINVAL (Invalid argument)

**[pid 9571] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fbf1cf70000**

[pid 9571] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 9571] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 9571] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=22607, ...}, AT\_EMPTY\_PATH) = 0

[pid 9571] mmap(NULL, 22607, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fbf1cf7a000

[pid 9571] close(3) = 0

[pid 9571] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 9571] 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

[pid 9571] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

**[pid 9571] mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fbf1cd40000**

[pid 9571] mprotect(0x7fbf1cdda000, 1576960, PROT\_NONE) = 0

[pid 9571] mmap(0x7fbf1cdda000, 1118208, PROT\_READ|PROT\_EXEC, MAP**\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7fbf1cdda000**

**[pid 9571] mmap(0x7fbf1ceeb000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7fbf1ceeb000**

**[pid 9571] mmap(0x7fbf1cf5b000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7fbf1cf5b000**

**[pid 9571] mmap(0x7fbf1cf69000, 10432, PROT\_READ|PROT\_WRITE**, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fbf1cf69000

[pid 9571] close(3) = 0

[pid 9571] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 9571] 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

[pid 9571] 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

[pid 9571] 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

[pid 9571] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) = 68

[pid 9571] newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

[pid 9571] 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

**[pid 9571] mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fbf1cb10000**

**[pid 9571] mmap(0x7fbf1cb38000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fbf1cb38000**

**[pid 9571] mmap(0x7fbf1cccd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fbf1cccd000**

**[pid 9571] mmap(0x7fbf1cd25000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7fbf1cd25000**

**[pid 9571] mmap(0x7fbf1cd2b000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fbf1cd2b000**

[pid 9571] close(3) = 0

[pid 9571] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 9571] 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

[pid 9571] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

**[pid 9571] mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fbf1ca20000**

**[pid 9571] mmap(0x7fbf1ca2e000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7fbf1ca2e000**

**[pid 9571] mmap(0x7fbf1caaa000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7fbf1caaa000**

**[pid 9571] mmap(0x7fbf1cb05000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7fbf1cb05000**

[pid 9571] close(3) = 0

[pid 9571] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

[pid 9571] 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

[pid 9571] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

**[pid 9571] mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fbf1ca00000**

**[pid 9571] mmap(0x7fbf1ca03000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7fbf1ca03000**

**[pid 9571] mmap(0x7fbf1ca1a000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7fbf1ca1a000**

**[pid 9571] mmap(0x7fbf1ca1e000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7fbf1ca1e000**

[pid 9571] close(3) = 0

**[pid 9571] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fbf1c9f0000**

**[pid 9571] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fbf1c9e0000**

[pid 9571] arch\_prctl(ARCH\_SET\_FS, 0x7fbf1c9e0740) = 0

[pid 9571] set\_tid\_address(0x7fbf1c9e0a10) = 9571

[pid 9571] set\_robust\_list(0x7fbf1c9e0a20, 24) = 0

[pid 9571] rseq(0x7fbf1c9e10e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)

[pid 9571] mprotect(0x7fbf1cd25000, 16384, PROT\_READ) = 0

[pid 9571] mprotect(0x7fbf1ca1e000, 4096, PROT\_READ) = 0

[pid 9571] mprotect(0x7fbf1cb05000, 4096, PROT\_READ) = 0

[pid 9571] mprotect(0x7fbf1cf5b000, 45056, PROT\_READ) = 0

[pid 9571] mprotect(0x7fbf1cfc2000, 4096, PROT\_READ) = 0

[pid 9571] mprotect(0x7fbf1cfb8000, 8192, PROT\_READ) = 0

[pid 9571] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=8192\*1024}) = 0

**[pid 9571] munmap(0x7fbf1cf7a000, 22607) = 0**

[pid 9571] getrandom("\x4f\x8d\x78\xfe\x71\x87\x4d\xa0", 8, GRND\_NONBLOCK) = 8

[pid 9571] brk(NULL) = 0x7fffde766000

[pid 9571] brk(0x7fffde787000) = 0x7fffde787000

[pid 9571] futex(0x7fbf1cf6977c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

[pid 9571] openat(AT\_FDCWD, "/dev/shm/mem", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = 3

**[pid 9571] mmap(NULL, 8, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7fbf1cfbe000**

[pid 9571] openat(AT\_FDCWD, "/dev/shm/sem.fmem\_semaphore", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 9571] getrandom("\x45\xdd\x40\xd7\x37\xda\xde\x9a", 8, GRND\_NONBLOCK) = 8

[pid 9571] newfstatat(AT\_FDCWD, "/dev/shm/sem.1YJz3Y", 0x7fffe6df7950, AT\_SYMLINK\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 9571] openat(AT\_FDCWD, "/dev/shm/sem.1YJz3Y", O\_RDWR|O\_CREAT|O\_EXCL, 0666) = 4

[pid 9571] write(4, "\0\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

**[pid 9571] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7fbf1cfbd000**

[pid 9571] link("/dev/shm/sem.1YJz3Y", "/dev/shm/sem.fmem\_semaphore") = 0

[pid 9571] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=32, ...}, AT\_EMPTY\_PATH) = 0

[pid 9571] unlink("/dev/shm/sem.1YJz3Y") = 0

[pid 9571] close(4) = 0

[pid 9571] openat(AT\_FDCWD, "/dev/shm/sem.mem\_semaphore1", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 9571] getrandom("\x9c\x20\xc2\x2d\x0d\xe8\x98\xde", 8, GRND\_NONBLOCK) = 8

[pid 9571] newfstatat(AT\_FDCWD, "/dev/shm/sem.2RvZaW", 0x7fffe6df7950, AT\_SYMLINK\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 9571] openat(AT\_FDCWD, "/dev/shm/sem.2RvZaW", O\_RDWR|O\_CREAT|O\_EXCL, 0666) = 4

[pid 9571] write(4, "\0\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

**[pid 9571] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7fbf1cfbc000**

[pid 9571] link("/dev/shm/sem.2RvZaW", "/dev/shm/sem.mem\_semaphore1") = 0

[pid 9571] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=32, ...}, AT\_EMPTY\_PATH) = 0

[pid 9571] unlink("/dev/shm/sem.2RvZaW") = 0

[pid 9571] close(4) = 0

[pid 9571] openat(AT\_FDCWD, "/dev/shm/sem.mem\_semaphore2", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 9571] getrandom("\x38\xd1\xa8\x52\x4a\xa1\xe5\x29", 8, GRND\_NONBLOCK) = 8

[pid 9571] newfstatat(AT\_FDCWD, "/dev/shm/sem.miL3sW", 0x7fffe6df7950, AT\_SYMLINK\_NOFOLLOW) = -1 ENOENT (No such file or directory)

[pid 9571] openat(AT\_FDCWD, "/dev/shm/sem.miL3sW", O\_RDWR|O\_CREAT|O\_EXCL, 0666) = 4

[pid 9571] write(4, "\0\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

**[pid 9571] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7fbf1cfb7000**

[pid 9571] link("/dev/shm/sem.miL3sW", "/dev/shm/sem.mem\_semaphore2") = 0

[pid 9571] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=32, ...}, AT\_EMPTY\_PATH) = 0

[pid 9571] unlink("/dev/shm/sem.miL3sW") = 0

[pid 9571] close(4) = 0

[pid 9571] futex(0x7fbf1cfbd000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 9570] <... read resumed>0x7fffd26102c0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 9571] <... futex resumed>) = ? ERESTART\_RESTARTBLOCK (Interrupted by signal)

[pid 9570] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 9571] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 9570] read(0, <unfinished ...>

[pid 9571] restart\_syscall(<... resuming interrupted futex ...>toji

<unfinished ...>

[pid 9570] <... read resumed>"toji\n", 1024) = 5

[pid 9570] openat(AT\_FDCWD, "/dev/shm/filename", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 4

[pid 9570] ftruncate(4, 40) = 0

**[pid 9570] mmap(NULL, 40, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7f0b9a4ad000**

[pid 9570] openat(AT\_FDCWD, "/dev/shm/sem.fmem\_semaphore", O\_RDWR|O\_NOFOLLOW) = 5

[pid 9570] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=32, ...}, AT\_EMPTY\_PATH) = 0

**[pid 9570] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0x7f0b9a4ac000**

[pid 9570] close(5) = 0

[pid 9570] openat(AT\_FDCWD, "/dev/shm/sem.mem\_semaphore1", O\_RDWR|O\_NOFOLLOW) = 5

[pid 9570] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=32, ...}, AT\_EMPTY\_PATH) = 0

**[pid 9570] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0x7f0b9a4a7000**

[pid 9570] close(5) = 0

[pid 9570] openat(AT\_FDCWD, "/dev/shm/sem.mem\_semaphore2", O\_RDWR|O\_NOFOLLOW) = 5

[pid 9570] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=32, ...}, AT\_EMPTY\_PATH) = 0

**[pid 9570] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0x7f0b9a46f000**

[pid 9570] close(5) = 0

[pid 9570] futex(0x7f0b9a4ac000, FUTEX\_WAKE, 1) = 1

[pid 9571] <... restart\_syscall resumed>) = 0

[pid 9570] write(1, "Enter num:\n", 11Enter num:

<unfinished ...>

[pid 9571] openat(AT\_FDCWD, "/dev/shm/filename", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC <unfinished ...>

[pid 9570] <... write resumed>) = 11

[pid 9570] read(0, <unfinished ...>

[pid 9571] <... openat resumed>) = 4

**[pid 9571] mmap(NULL, 40, PROT\_READ, MAP\_SHARED, 4, 0) = 0x7fbf1cf7f000**

[pid 9571] openat(AT\_FDCWD, "toji", O\_WRONLY|O\_CREAT|O\_TRUNC, 0777) = 5

[pid 9571] futex(0x7fbf1cfb7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY10

<unfinished ...>

[pid 9570] <... read resumed>"10\n", 1024) = 3

[pid 9570] futex(0x7f0b9a46f000, FUTEX\_WAKE, 1) = 1

[pid 9571] <... futex resumed>) = 0

[pid 9570] futex(0x7f0b9a4a7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 9571] write(5, "\n\0\0\0", 4) = 4

[pid 9571] futex(0x7fbf1cfbc000, FUTEX\_WAKE, 1 <unfinished ...>

[pid 9570] <... futex resumed>) = 0

[pid 9571] <... futex resumed>) = 1

[pid 9570] read(0, <unfinished ...>

[pid 9571] futex(0x7fbf1cfb7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY20

<unfinished ...>

[pid 9570] <... read resumed>"20\n", 1024) = 3

[pid 9570] futex(0x7f0b9a46f000, FUTEX\_WAKE, 1) = 1

[pid 9571] <... futex resumed>) = 0

[pid 9570] futex(0x7f0b9a4a7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 9571] write(5, "\24\0\0\0", 4) = 4

[pid 9571] futex(0x7fbf1cfbc000, FUTEX\_WAKE, 1 <unfinished ...>

[pid 9570] <... futex resumed>) = 0

[pid 9571] <... futex resumed>) = 1

[pid 9570] read(0, <unfinished ...>

[pid 9571] futex(0x7fbf1cfb7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY34

<unfinished ...>

[pid 9570] <... read resumed>"34\n", 1024) = 3

[pid 9570] futex(0x7f0b9a46f000, FUTEX\_WAKE, 1) = 1

[pid 9570] futex(0x7f0b9a4a7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 9571] <... futex resumed>) = 0

[pid 9571] write(5, "\"\0\0\0", 4) = 4

[pid 9571] futex(0x7fbf1cfbc000, FUTEX\_WAKE, 1 <unfinished ...>

[pid 9570] <... futex resumed>) = 0

[pid 9571] <... futex resumed>) = 1

[pid 9571] futex(0x7fbf1cfb7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 9570] read(0, 46

"46\n", 1024) = 3

[pid 9570] futex(0x7f0b9a46f000, FUTEX\_WAKE, 1) = 1

[pid 9571] <... futex resumed>) = 0

[pid 9570] futex(0x7f0b9a4a7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 9571] write(5, ".\0\0\0", 4) = 4

[pid 9571] futex(0x7fbf1cfbc000, FUTEX\_WAKE, 1 <unfinished ...>

[pid 9570] <... futex resumed>) = 0

[pid 9571] <... futex resumed>) = 1

[pid 9570] read(0, <unfinished ...>

[pid 9571] futex(0x7fbf1cfb7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY2

<unfinished ...>

[pid 9570] <... read resumed>"2\n", 1024) = 2

[pid 9570] futex(0x7f0b9a46f000, FUTEX\_WAKE, 1) = 1

[pid 9570] futex(0x7f0b9a4a7000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 9571] <... futex resumed>) = 0

[pid 9571] futex(0x7fbf1cfbc000, FUTEX\_WAKE, 1 <unfinished ...>

[pid 9570] <... futex resumed>) = 0

[pid 9571] <... futex resumed>) = 1

[pid 9570] write(1, "Finish processes\n", 17 <unfinished ...>

[pid 9571] munmap(0x7fbf1cfbc000, 32Finish processes

<unfinished ...>

[pid 9570] <... write resumed>) = 17

[pid 9571] <... munmap resumed>) = 0

**[pid 9570] munmap(0x7f0b9a4ad000, 40) = 0**

[pid 9571] unlink("/dev/shm/sem.mem\_semaphore1" <unfinished ...>

[pid 9570] close(4) = 0

[pid 9570] unlink("/dev/shm/filename" <unfinished ...>

[pid 9571] <... unlink resumed>) = 0

**[pid 9571] munmap(0x7fbf1cfb7000, 32) = 0**

[pid 9571] unlink("/dev/shm/sem.mem\_semaphore2" <unfinished ...>

[pid 9570] <... unlink resumed>) = 0

**[pid 9570] munmap(0x7f0b9a4a7000, 32) = 0**

[pid 9570] unlink("/dev/shm/sem.mem\_semaphore1") = -1 ENOENT (No such file or directory)

[pid 9571] <... unlink resumed>) = 0

**[pid 9570] munmap(0x7f0b9a46f000, 32 <unfinished ...>**

**[pid 9571] munmap(0x7fbf1cfbe000, 8) = 0**

**[pid 9571] munmap(0x7fbf1cf7f000, 40) = 0**

[pid 9571] close(3 <unfinished ...>

[pid 9570] <... munmap resumed>) = 0

[pid 9570] unlink("/dev/shm/sem.mem\_semaphore2" <unfinished ...>

[pid 9571] <... close resumed>) = 0

[pid 9570] <... unlink resumed>) = -1 ENOENT (No such file or directory)

[pid 9571] close(4 <unfinished ...>

**[pid 9570] munmap(0x7f0b9a4ac000, 32) = 0**

[pid 9570] unlink("/dev/shm/sem.fmem\_semaphore" <unfinished ...>

[pid 9571] <... close resumed>) = 0

[pid 9571] unlink("/dev/shm/mem" <unfinished ...>

[pid 9570] <... unlink resumed>) = 0

**[pid 9570] munmap(0x7f0b9a4ae000, 8) = 0**

[pid 9570] close(3) = 0

[pid 9570] unlink("/dev/shm/mem") = -1 ENOENT (No such file or directory)

[pid 9571] <... unlink resumed>) = 0

[pid 9571] unlink("/dev/shm/filename") = -1 ENOENT (No such file or directory)

[pid 9570] lseek(0, -1, SEEK\_CUR <unfinished ...>

[pid 9571] close(5 <unfinished ...>

[pid 9570] <... lseek resumed>) = -1 ESPIPE (Illegal seek)

[pid 9570] exit\_group(0 <unfinished ...>

[pid 9571] <... close resumed>) = 0

[pid 9570] <... exit\_group resumed>) = ?

[pid 9571] exit\_group(0) = ?

[pid 9570] +++ exited with 0 +++

+++ exited with 0 +++

**Вывод**

Разобрался в новом способе межпроцессорного взаимодействия через создание и взаимодействие с общей памятью. После первой лабораторной было не так трудно понять, что от меня требуется, скорее надо было переварить новый подход, требующий немного другой логики. Именно поэтому здесь я использовал семафоры для синхронизации взаимодействия, потому что память может затираться или обновляться, без синхронизации это было привело к неправильному выполнению. Поэтому большее время я занимался синхронизацией и отладкой.