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-214БВ-23

Студент: Богданов М.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 15.11.25

Москва, 2025

Постановка задачи

Нужно взять свою первую лабу и переделать её с использованием shared memory и memory mapping. Варианты остаются те же, что и у первой лабораторной. Так как блокирующего чтения из каналов у вас больше не будет, то для синхронизации чтения и записи из shared memory будем использовать семафор.

Общий метод и алгоритм решения

Использованные системные вызовы:

1) shm_open - создаёт/открывает объект разделяемой памяти
2) shm_unlink - удаляет именованный объект разделяемой памяти
3) ftruncate - устанавливает размер файла/разделяемой памяти
4) mmap - отображает файл/объект в память процесса
5) munmap - удаляет отображение памяти
6) sem_open - создаёт/открывает именованный семафор
7) sem_wait - уменьшает значение семафора (ожидание)
8) sem_post - увеличивает значение семафора (сигнал)
9) sem_unlink - удаляет именованный семафор
10) sem_close - закрывает открытый семафор

Основная идея была в том что теперь процессы вместо того чтобы обмениваться данными при помощи буфера(в данном случае пайпа) теперь просто обращаются к одной и той же области памяти. Это значительно ускоряет работу кода при большом объеме данных, потому что исчезает необходимость в копировании в пайп.

Код программы

lab3_client.c

```
        #include <fcntl.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <semaphore.h>
#include <sys/mman.h>
#include <errno.h>



#define SHM_SIZE 4096



static char SERVER_PROGRAM_NAME[] = "lab3_server";
const char SHM_NAME[] = "lab_shm";
const char SEM_NAME[] = "lab_sem";



int main(int argc, char **argv) {
   shm_unlink(SHM_NAME);
   sem_unlink(SEM_NAME);
   int shm = shm_open(SHM_NAME, O_RDWR | O_CREAT | O_EXCL, 0600);
```

```
    if (shm == -1) {
        const char msg[] = "error: failed to create SHM\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        exit(EXIT_FAILURE);
    }
    if (ftruncate(shm, SHM_SIZE) == -1) {
        const char msg[] = "error: failed to resize SHM\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        shm_unlink(SHM_NAME);
        close(shm);
        exit(EXIT_FAILURE);
    }
    char *shm_buf = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm, 0);
    if (shm_buf == MAP_FAILED) {
        const char msg[] = "error: failed to map SHM\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        shm_unlink(SHM_NAME);
        close(shm);
        exit(EXIT_FAILURE);
    }
    uint32_t *length = (uint32_t *)shm_buf;
    *length = 0;
    sem_t *sem = sem_open(SEM_NAME, O_CREAT | O_EXCL, 0600, 1);
    if (sem == SEM_FAILED) {
        const char msg[] = "error: failed to create semaphore\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        munmap(shm_buf, SHM_SIZE);
        shm_unlink(SHM_NAME);
        close(shm);
        exit(EXIT_FAILURE);
    }
    char filename[256];
    ssize_t bytes_read = read(STDIN_FILENO, filename, sizeof(filename) - 1);
    if (bytes_read <= 0) {
        const char msg[] = "error: failed to read filename\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        sem_unlink(SEM_NAME);
        sem_close(sem);
        munmap(shm_buf, SHM_SIZE);
        shm_unlink(SHM_NAME);
        close(shm);
        exit(EXIT_FAILURE);
    }
    filename[bytes_read] = '\0';
    filename[strcspn(filename, "\n")] = '\0';
    char progpath[1024];
    ssize_t len = readlink("/proc/self/exe", progpath, sizeof(progpath) - 1);
    if (len == -1) {
        const char msg[] = "error: failed to read full program path\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        sem_unlink(SEM_NAME);
        sem_close(sem);
        munmap(shm_buf, SHM_SIZE);
        shm_unlink(SHM_NAME);
        close(shm);
        exit(EXIT_FAILURE);
    }
    while (len > 0 && progpath[len] != '/')
        --len;

    if (len == 0) {
        const char msg[] = "error: invalid program path\n";
```

```
    write(STDERR_FILENO, msg, sizeof(msg) - 1);
    sem_unlink(SEM_NAME);
    sem_close(sem);
    munmap(shm_buf, SHM_SIZE);
    shm_unlink(SHM_NAME);
    close(shm);
    exit(EXIT_FAILURE);
}
progpath[len] = '\0';
const pid_t child = fork();
switch (child) {
    case -1: {
        const char msg[] = "error: failed to spawn new process\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        sem_unlink(SEM_NAME);
        sem_close(sem);
        munmap(shm_buf, SHM_SIZE);
        shm_unlink(SHM_NAME);
        close(shm);
        exit(EXIT_FAILURE);
    } break;

    case 0: {
        char path[1024];
        strcpy(path, progpath);
        strcat(path, "/");
        strcat(path, SERVER_PROGRAM_NAME);

        char *const args[] = {SERVER_PROGRAM_NAME, filename, NULL};
        int32_t status = execv(path, args);

        if (status == -1) {
            const char msg[] = "error: failed to exec into new executable image\n";
            write(STDERR_FILENO, msg, sizeof(msg) - 1);
            exit(EXIT_FAILURE);
        }
    } break;

    default: {
        char buf[SHM_SIZE - sizeof(uint32_t) - 1];
        ssize_t bytes;
        while ((bytes = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
            if (buf[0] == '\n') {
                break;
            }
            sem_wait(sem);
            uint32_t *length = (uint32_t *)shm_buf;
            char *text = shm_buf + sizeof(uint32_t);

            *length = bytes;
            memcpy(text, buf, bytes);
            text[bytes] = '\0';
            sem_post(sem);
        }

        if (bytes < 0) {
            const char msg[] = "error: failed to read from stdin\n";
            write(STDERR_FILENO, msg, sizeof(msg) - 1);
        }
        sem_wait(sem);
        uint32_t *length_final = (uint32_t *)shm_buf;
```

```
            *length_final = UINT32_MAX;
            sem_post(sem);
            sem_unlink(SEM_NAME);
            sem_close(sem);
            munmap(shm_buf, SHM_SIZE);
            shm_unlink(SHM_NAME);
            close(shm);
        } break;
    }

    return 0;
}
```

lab3_server.c

```
#include <fcntl.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <semaphore.h>
#include <sys/mman.h>
#include <errno.h>
#include <ctype.h>



#define SHM_SIZE 4096
#define MIN_LEN 256



const char SHM_NAME[] = "lab_shm";
const char SEM_NAME[] = "lab_sem";
float  buf_to_numbers(char * buf) {
    float n = 10, num2 = 0, result;
    int num_path, i = 0, flag = 0, min_flag = 0, first_flag = 0;
    while (true){
        if (buf[i] != '1' && buf[i] != '2' && buf[i] != '3' && buf[i] != '4' && buf[i] != '5' \
            && buf[i] != '6' && buf[i] != '7' && buf[i] != '8'&& buf[i] != '9' && buf[i] != '0' && buf[i] != '.' && buf[i] != ' ' \
            && buf[i] != '-' && buf[i] !='\n'){
                return -1;
        }
        if (buf[i] == ' '){
            n = 10;
            flag = 0;
            if (min_flag == 1){
                num2 *= -1;
            }
            min_flag = 0;
            if (first_flag == 0){
                result = num2;
                first_flag = 1;
            }
            else if (num2 != 0 && first_flag == 1){
                result /= num2;
            }
            else if (num2 == 0 && first_flag == 1){
                return -1;
```

```
            }
            num2 = 0;
        }
        else if (buf[i] == '\n' || buf[i] == '\0'){
            n = 10;
            flag = 0;
            if (min_flag == 1){
                num2 *= -1;
            }
            min_flag = 0;
            if (first_flag == 0){
                result = num2;
                first_flag = 1;
            }
            else if (num2 != 0 && first_flag == 1){
                result /= num2;
            }
            else if (num2 == 0 && first_flag == 1){
                return -1;
            }
            num2 = 0;
            break;
        }
        else if (buf[i] == '.'){
            flag = 1;
            n = 0.1;
        }
        else if (buf[i] == '-'){
            min_flag = 1;
        }
        else if (flag == 0) {
            num2 = (buf[i] - '0') + num2 * n;
        }
        else if (flag == 1) {
            num2 += (buf[i] - '0') * n;
            n/=10;
        }
        ++i;
    }
    return result;
}


int main(int argc, char **argv) {
    if (argc < 2) {
        const char msg[] = "error: no filename provided\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        exit(EXIT_FAILURE);
    }
    int shm = shm_open(SHM_NAME, O_RDWR, 0);
    if (shm == -1) {
        const char msg[] = "error: failed to open SHM\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        exit(EXIT_FAILURE);
    }
    char *shm_buf = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm, 0);
    if (shm_buf == MAP_FAILED) {
        const char msg[] = "error: failed to map SHM\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        close(shm);
        exit(EXIT_FAILURE);
```

```
    }
    sem_t *sem = sem_open(SEM_NAME, 0);
    if (sem == SEM_FAILED) {
        const char msg[] = "error: failed to open semaphore\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        munmap(shm_buf, SHM_SIZE);
        close(shm);
        exit(EXIT_FAILURE);
    }
    int32_t file = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0600);
    if (file == -1) {
        const char msg[] = "error: failed to open requested file\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        sem_close(sem);
        munmap(shm_buf, SHM_SIZE);
        close(shm);
        exit(EXIT_FAILURE);
    }
    bool running = true;
    float res = 0;
    char output_buf[MIN_LEN];
    while (running) {
        sem_wait(sem);
        uint32_t *length = (uint32_t *)shm_buf;
        char *text = shm_buf + sizeof(uint32_t);
        if (*length == UINT32_MAX) {
            sem_post(sem);
            running = false;
        } else if (*length > 0) {
            res = buf_to_numbers(text);
            if (res != -1) {
                int len = snprintf(output_buf, MIN_LEN, "%f\n", res);
                write(file, output_buf, len);
            } else {
                const char error_msg[] = "error: invalid input or division by zero\n";
                write(file, error_msg, sizeof(error_msg) - 1);
            }

            *length = 0;
            sem_post(sem);
        } else {
            sem_post(sem);
        }
    }
    sem_close(sem);
    munmap(shm_buf, SHM_SIZE);
    close(shm);
    close(file);

    return 0;
}
```

Протокол работы программы

Тестирование:

#1

bogdanoff@arch ~/s/O/l/src (main)> strace -o test1.log ./lab3_client

test.txt

12 12 12 12

1 1 1 1 1

121 1212 1212 1212

13313 13 1 1 1 3131

1333 333

1311 1

1311 0

0 1

Файл:
0.006944

error: invalid input or division by zero

0.000000

0.327077

4.003003

1311.000000

error: invalid input or division by zero

0.000000

#2

bogdanoff@arch ~/s/O/l/src (main)> strace -o test2.log ./lab3_client

test.txt

0 1 1

1 1 1212

1222 2222

1 1 1 1 1 1 1

Файл:
0.000000

0.000825

0.549955

1.000000

#3

bogdanoff@arch ~/s/O/l/src (main)> strace -o test3.log ./lab3_client

test.txt

1 0

0 1

0 0 0 0

Файл:
error: invalid input or division by zero

0.000000

error: invalid input or division by zero

Strace

Test1:

```
execve("./lab3_client", ["./lab3_client"], 0x7ffe58194130 /* 49 vars */) = 0
brk(NULL)                               = 0x5635bc1f3000
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=124823, ...}) = 0
mmap(NULL, 124823, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f248a8a3000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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\0000x\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"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f248a8a1000
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"..., 896, 64) = 896
mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f248a600000
mmap(0x7f248a624000, 1515520, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x24000) = 0x7f248a624000
mmap(0x7f248a796000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x196000) =
0x7f248a796000
mmap(0x7f248a805000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x204000) = 0x7f248a805000
mmap(0x7f248a80b000, 31792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7f248a80b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f248a89e000
arch_prctl(ARCH_SET_FS, 0x7f248a89e740) = 0
set_tid_address(0x7f248a89ea10)         = 700876
set_robust_list(0x7f248a89ea20, 24)     = 0
rseq(0x7f248a89e680, 0x20, 0, 0x53053053) = 0
mprotect(0x7f248a805000, 16384, PROT_READ) = 0
mprotect(0x5635ba73c000, 4096, PROT_READ) = 0
mprotect(0x7f248a903000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
getrandom("\x62\x19\xc5\xb5\xcd\xcd\x2d\xa9", 8, GRND_NONBLOCK) = 8
munmap(0x7f248a8a3000, 124823)          = 0
unlink("/dev/shm/lab_shm")              = 0
unlink("/dev/shm/sem.lab_sem")          = 0
openat(AT_FDCWD, "/dev/shm/lab_shm", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 3
ftruncate(3, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f248a8c1000
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_DROPPABLE|MAP_ANONYMOUS, -1, 0) = 0x7f248a8c0000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f248a8bf000
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
getrandom("\x68\x41\xf6\x6c\xbf\x31\x66\xae\x44\x3c\xb7\x3e\xb5\xe1\x6f\x45\x2c\xa6\xc6\x16\x08\x98\x56\x
```

```
80\x3f\xc8\xfb\x2d\x54\xd9\x14\x68", 32, 0) = 32
newfstatat(AT_FDCWD, "/dev/shm/sem.k6enCb", 0x7ffe5d9585d0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/sem.k6enCb", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 4
write(4, "\1\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\0", 32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f248a8be000
link("/dev/shm/sem.k6enCb", "/dev/shm/sem.lab_sem") = 0
fstat(4, {st_mode=S_IFREG|0600, st_size=32, ...}) = 0
brk(NULL)                               = 0x5635bc1f3000
brk(0x5635bc214000)                     = 0x5635bc214000
unlink("/dev/shm/sem.k6enCb")           = 0
close(4)                                = 0
read(0, "test.txt\n", 255)              = 9
readlink("/proc/self/exe", "/home/bogdanoff/study/OS/lab3/sr"..., 1023) = 45
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f248a89ea10) = 700893
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
read(0, "12 12 12 12\n", 4091)          = 12
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 1
read(0, "1 1 1 1 1 \n", 4091)           = 11
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 0
read(0, "121 1212 1212 1212\n", 4091)   = 19
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 0
read(0, "13313 13 1 1 1 3131\n", 4091)  = 20
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 0
read(0, "1333 333\n", 4091)             = 9
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 0
read(0, "1311 1\n", 4091)               = 7
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 0
read(0, "1311 0\n", 4091)               = 7
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
read(0, "0 1\n", 4091)                  = 4
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 1
read(0, "\n", 4091)                     = 1
futex(0x7f248a8be000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7f248a8be000, FUTEX_WAKE, 1)    = 0
unlink("/dev/shm/sem.lab_sem")          = 0
munmap(0x7f248a8be000, 32)              = 0
munmap(0x7f248a8c1000, 4096)            = 0
unlink("/dev/shm/lab_shm")              = 0
```

```
close(3)                                = 0
exit_group(0)                           = ?
```

Test2:

```
execve("./lab3_client", ["./lab3_client"], 0x7fff5eec00e0 /* 49 vars */) = 0
brk(NULL)                               = 0x563fd1845000
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=124823, ...}) = 0
mmap(NULL, 124823, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f22f462e000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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\0000x\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"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f22f462c000
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"..., 896, 64) = 896
mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f22f4400000
mmap(0x7f22f4424000, 1515520, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x24000) = 0x7f22f4424000
mmap(0x7f22f4596000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x196000) =
0x7f22f4596000
mmap(0x7f22f4605000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x204000) = 0x7f22f4605000
mmap(0x7f22f460b000, 31792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7f22f460b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f22f4629000
arch_prctl(ARCH_SET_FS, 0x7f22f4629740) = 0
set_tid_address(0x7f22f4629a10)         = 703795
set_robust_list(0x7f22f4629a20, 24)     = 0
rseq(0x7f22f4629680, 0x20, 0, 0x53053053) = 0
mprotect(0x7f22f4605000, 16384, PROT_READ) = 0
mprotect(0x563fbd502000, 4096, PROT_READ) = 0
mprotect(0x7f22f468e000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
getrandom("\xa7\xb4\x42\x35\xa1\x2c\xef\xd4", 8, GRND_NONBLOCK) = 8
munmap(0x7f22f462e000, 124823)          = 0
unlink("/dev/shm/lab_shm")              = -1 ENOENT (Нет такого файла или каталога)
unlink("/dev/shm/sem.lab_sem")          = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/lab_shm", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 3
ftruncate(3, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f22f464c000
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_DROPPABLE|MAP_ANONYMOUS, -1, 0) = 0x7f22f464b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f22f464a000
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
getrandom("\xc1\x26\x86\x13\x85\x21\xce\x28\x4c\x3c\xc6\xc1\x37\x8f\xf2\x7e\x64\x7c\xc2\x09\x3b\x49\x7a\x
8a\x8b\x5e\x2a\xad\xa2\x97\x1f\xc5", 32, 0) = 32
newfstatat(AT_FDCWD, "/dev/shm/sem.nCmRTT", 0x7ffc75b61550, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/sem.nCmRTT", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 4
write(4, "\1\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\0", 32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f22f4649000
```

```
link("/dev/shm/sem.nCmRTT", "/dev/shm/sem.lab_sem") = 0
fstat(4, {st_mode=S_IFREG|0600, st_size=32, ...}) = 0
brk(NULL)                               = 0x563fd1845000
brk(0x563fd1866000)                     = 0x563fd1866000
unlink("/dev/shm/sem.nCmRTT")           = 0
close(4)                                = 0
read(0, "test.txt\n", 255)              = 9
readlink("/proc/self/exe", "/home/bogdanoff/study/OS/lab3/sr"..., 1023) = 45
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f22f4629a10) = 703802
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
read(0, "0 1 1\n", 4091)                = 6
futex(0x7f22f4649000, FUTEX_WAKE, 1)    = 1
read(0, "1 1 1212\n", 4091)             = 9
futex(0x7f22f4649000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
read(0, "1222 2222\n", 4091)            = 10
futex(0x7f22f4649000, FUTEX_WAKE, 1)    = 0
read(0, "1 1 1 1 1 1 1\n", 4091)        = 14
read(0, "\n", 4091)                     = 1
futex(0x7f22f4649000, FUTEX_WAKE, 1)    = 0
unlink("/dev/shm/sem.lab_sem")          = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=703802, si_uid=1000, si_status=0, si_utime=2313 /*
23.13 s */, si_stime=0} ---
munmap(0x7f22f4649000, 32)              = 0
munmap(0x7f22f464c000, 4096)            = 0
unlink("/dev/shm/lab_shm")              = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Test3:

```
execve("./lab3_client", ["./lab3_client"], 0x7ffff9899d60 /* 49 vars */) = 0
brk(NULL)                               = 0x55b69145f000
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=124823, ...}) = 0
mmap(NULL, 124823, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbfe4a9b000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000x\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"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbfe4a99000
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"..., 896, 64) = 896
mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fbfe4800000
mmap(0x7fbfe4824000, 1515520, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x24000) = 0x7fbfe4824000
mmap(0x7fbfe4996000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x196000) =
0x7fbfe4996000
mmap(0x7fbfe4a05000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
```

```
0x204000) = 0x7fbfe4a05000
mmap(0x7fbfe4a0b000, 31792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fbfe4a0b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbfe4a96000
arch_prctl(ARCH_SET_FS, 0x7fbfe4a96740) = 0
set_tid_address(0x7fbfe4a96a10)         = 704435
set_robust_list(0x7fbfe4a96a20, 24)     = 0
rseq(0x7fbfe4a96680, 0x20, 0, 0x53053053) = 0
mprotect(0x7fbfe4a05000, 16384, PROT_READ) = 0
mprotect(0x55b652992000, 4096, PROT_READ) = 0
mprotect(0x7fbfe4afb000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
getrandom("\xe6\x3f\xdd\x04\x41\x13\x67\xe6", 8, GRND_NONBLOCK) = 8
munmap(0x7fbfe4a9b000, 124823)          = 0
unlink("/dev/shm/lab_shm")              = -1 ENOENT (Нет такого файла или каталога)
unlink("/dev/shm/sem.lab_sem")          = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/lab_shm", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 3
ftruncate(3, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fbfe4ab9000
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_DROPPABLE|MAP_ANONYMOUS, -1, 0) = 0x7fbfe4ab8000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbfe4ab7000
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
getrandom("\x8e\x94\x20\x00\xf5\x0a\xda\x6a\x97\xb3\xff\x44\xf7\xf8\x13\xde\x8e\x7d\xdf\x0b\xca\x4f\xad\x3
3\x02\x67\xe5\x07\x3b\x9b\x47\x7e", 32, 0) = 32
newfstatat(AT_FDCWD, "/dev/shm/sem.fI04XG", 0x7ffda72178c0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/sem.fI04XG", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 4
write(4, "\1\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\0", 32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fbfe4ab6000
link("/dev/shm/sem.fI04XG", "/dev/shm/sem.lab_sem") = 0
fstat(4, {st_mode=S_IFREG|0600, st_size=32, ...}) = 0
brk(NULL)                               = 0x55b69145f000
brk(0x55b691480000)                     = 0x55b691480000
unlink("/dev/shm/sem.fI04XG")           = 0
close(4)                                = 0
read(0, "test.txt\n", 255)              = 9
readlink("/proc/self/exe", "/home/bogdanoff/study/OS/lab3/sr"..., 1023) = 45
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fbfe4a96a10) = 704463
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
read(0, "1 0\n", 4091)                  = 4
futex(0x7fbfe4ab6000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY) = -1
EAGAIN (Ресурс временно недоступен)
futex(0x7fbfe4ab6000, FUTEX_WAKE, 1)    = 1
read(0, "0 1\n", 4091)                  = 4
read(0, "0 0 0 0 \n", 4091)             = 9
futex(0x7fbfe4ab6000, FUTEX_WAKE, 1)    = 0
read(0, "\n", 4091)                     = 1
futex(0x7fbfe4ab6000, FUTEX_WAKE, 1)    = 0
unlink("/dev/shm/sem.lab_sem")          = 0
munmap(0x7fbfe4ab6000, 32)              = 0
munmap(0x7fbfe4ab9000, 4096)            = 0
```

```
unlink("/dev/shm/lab_shm")              = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=704463, si_uid=1000, si_status=0, si_utime=886 /*
8.86 s */, si_stime=0} ---
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Использование shared memory значительно ускоряет работу кода, потому что позволяет процессам напрямую общаться между собой. Но при этом нужно работать с ней очень аккуратно потому что неправильная работа с ней может полностью нарушить работу программы (например, гонка данных).