

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-209БВ-24

Студент: Лисов Д.С.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_

Дата: 12.11.25

Москва, 2025

# Постановка задачи

## Вариант 15.

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы. Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в File Mapping. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в Error File Mapping выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода

## Общий метод и алгоритм решения

Использованные системные вызовы:

- open() - открытие файла;
- ftruncate() - установка размера файла;
- mmap() - отображение файла в память;
- munmap() - удаление отображения;
- fork() - создание нового процесса;
- execv() - замена образа процесса;
- wait() - ожидание завершения процесса;
- kill() - отправка сигналов;
- signal() - обработка сингалов;
- fstat() - получение информации о файле;
- pause() - ожидание сигнала;
- close() - закрытие файлового дескриптора

Алгоритм аналогичен описанному в условии задачи: создаётся File Mapping для обмена строк между процессами и Error File Mapping для переправки ошибок от дочернего процесса.

## Код программы

### parent.c

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <string.h>
#include <sys/wait.h>
#include <signal.h>

#define MAX_LEN 4096
```

```

volatile sig_atomic_t child_done = 0;

typedef struct {
    char message[MAX_LEN];
    int has_message;
    int terminate;
} shared_data_t;

void handler_parent(int sig) {
    child_done = 1;
}

int main() {
    char *filename = "FileMapping";

    int fd = open(filename, O_RDWR | O_CREAT, 0666);
    int fd_errors = open("Errors File Mapping", O_CREAT | O_RDWR, 0666);

    if (fd == -1 || fd_errors == -1) {
        perror("open");
        exit(1);
    }

    if (ftruncate(fd, sizeof(shared_data_t)) == -1) {
        perror("ftruncate");
        close(fd);
        exit(1);
    }

    if (ftruncate(fd_errors, sizeof(shared_data_t)) == -1) {
        perror("ftruncate");
        close(fd_errors);
        exit(1);
    }

    shared_data_t *data = mmap(NULL, sizeof(shared_data_t), PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0);
    shared_data_t *errors = mmap(NULL, sizeof(shared_data_t), PROT_READ |
PROT_WRITE, MAP_SHARED, fd_errors, 0);
    if (data == MAP_FAILED || errors == MAP_FAILED) {
        perror("mmap");
        close(fd);
        exit(1);
    }
}

```

```

data->message[0] = '\0';
data->has_message = 0;
data->terminate = 0;

char *s = NULL;
size_t len = 0;
getline(&s, &len, stdin);

if (s[strlen(s) - 1] == '\n') {
    s[strlen(s) - 1] = '\0';
}
strncpy(data->message, s, 255);

signal(SIGUSR2, handler_parent);
pid_t pid = fork();

if (pid == -1) {
    perror("fork");
    munmap(data, sizeof(shared_data_t));
    close(fd);
    exit(1);
}

if (pid > 0) {
    while (1) {
        fflush(stdout);

        ssize_t read_bytes = getline(&s, &len, stdin);
        if (read_bytes == -1) {
            break;
        }

        if (s[read_bytes - 1] == '\n') {
            s[read_bytes - 1] = '\0';
            --read_bytes;
        }

        if (strcmp(s, "exit") == 0) {
            data->has_message = 1;
            data->terminate = 1;
            kill(pid, SIGUSR1);
            break;
        }
    }
}

```

```

        while (data->has_message) {
            usleep(10000);
        }

        strncpy(data->message, s, MAX_LEN - 1);
        data->message[MAX_LEN - 1] = '\0';
        data->has_message = 1;

        printf("Отправлено через File Mapping: %s\n", s);
        kill(pid, SIGUSR1);

        /*while (!errors->has_message && !errors->terminate) {
            usleep(10000);
        }*/
        child_done = 0;
        while (!child_done) {
            pause();
        }

        if (errors->has_message)
        {
            printf("Дочерний процесс ответил: %s\n", errors->message);
            errors->has_message = 0;
        }
        errors->terminate = 0;
    }
    free(s);

    wait(NULL);
} else {
    char *args[] = { "./child", filename, NULL };
    execv("./child", args);
    perror("execv");
    exit(1);
}

munmap(data, sizeof(shared_data_t));
close(fd);
return 0;
}

```

### **child.c**

```

#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>

```

```

#include <string.h>
#include <fcntl.h>
#include <ctype.h>
#include <sys/stat.h>
#include <signal.h>

#define MAX_LEN 4096

volatile sig_atomic_t has_signal = 0;

typedef struct {
    char message[MAX_LEN];
    int has_message;
    int terminate;
} shared_data_t;

void handler_child(int sig) {
    has_signal = 1;
}

int main(int argc, char* argv[]) {
    signal(SIGUSR1, handler_child);
    char *mapping_filename = argv[1];

    char *error_filename = "Errors File Mapping";
    char output_filename[256];

    int fd = open(mapping_filename, O_RDWR);
    int fd_errors = open(error_filename, O_RDWR);

    if (fd == -1 || fd_errors == -1) {
        perror("open");
        exit(1);
    }

    struct stat st;
    fstat(fd, &st);
    if (st.st_size < sizeof(shared_data_t)) {
        fprintf(stderr, "File too small: %ld < %zu\n", st.st_size,
        sizeof(shared_data_t));
        exit(1);
    }

    shared_data_t *shared_data = mmap(NULL, sizeof(shared_data_t),
        PROT_WRITE | PROT_READ, MAP_SHARED, fd, 0);

```

```
shared_data_t *error_data = mmap(NULL, sizeof(shared_data_t), PROT_WRITE  
| PROT_READ, -MAP_SHARED, fd_errors, 0);
```

```
if (shared_data == MAP_FAILED || error_data == MAP_FAILED) {  
    perror("mmap");  
    close(fd);  
    exit(1);  
}
```

```
strncpy(output_filename, shared_data->message, 255);
```

```
FILE* output_file = fopen(output_filename, "w");
```

```
printf("Открыт файл %s для записи\n", output_filename);
```

```
while(1) {  
    while (!has_signal) {  
        pause();  
    }
```

```
    has_signal = 0;  
    printf("Получено через File Mapping: %s\n", shared_data->message);
```

```
    if (shared_data->terminate) {  
        printf("Получен сигнал завершения\n");  
        break;  
    }
```

```
    if (isupper((unsigned char)shared_data->message[0])) {  
        fprintf(output_file, "%s\n", shared_data->message);  
        fflush(output_file);  
        error_data->terminate = 1;  
        error_data->has_message = 0;  
    } else {  
        strncpy(error_data->message, "ERROR!", strlen("ERROR!"));  
        // error_data->terminate = 0;  
        error_data->has_message = 1;  
        kill(getppid(), SIGUSR2);  
    }
```

```
    shared_data->has_message = 0;
```

```
    kill(getppid(), SIGUSR2);  
    // usleep(10000);  
}
```

```

fclose(output_file);
munmap(shared_data, sizeof(shared_data_t));
munmap(error_data, sizeof(shared_data_t));

close(fd_errors);
close(fd);

return 0;
}

```

## Протокол работы программы

### Тестирование:

```

$ ./parent
example.txt
Открыт файл example.txt для записи
A
Отправлено через File Mapping: A
Получено через File Mapping: A
Aboba
Отправлено через File Mapping: Aboba
Получено через File Mapping: Aboba
E
Отправлено через File Mapping: E
Получено через File Mapping: E
error
Отправлено через File Mapping: error
Получено через File Mapping: error
Дочерний процесс ответил: ERROR!
buffer
Отправлено через File Mapping: buffer
Получено через File Mapping: buffer
Дочерний процесс ответил: ERROR!
exit
Получено через File Mapping: buffer
Получен сигнал завершения

```

### Strace:

```

$ strace -f ./parent
execve("./parent", [ "./parent" ], 0x7fff13d0c108 /* 63 vars */) = 0
brk(NULL)                               = 0x63f913080000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ef1a2c36000
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=153815, ...}) = 0

```

```
mmap(NULL, 153815, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ef1a2c10000  
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\3\0>\n\n\220\243\2\0\0\0\0\0...\r832")=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..."...,  
784, 64) = 784  
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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..."...,  
784, 64) = 784  
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =  
0x7ef1a2800000  
mmap(0x7ef1a2828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|  
MAP_DENYWRITE, 3, 0x28000) = 0x7ef1a2828000  
mmap(0x7ef1a29b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,  
0x1b0000) = 0x7ef1a29b0000  
mmap(0x7ef1a29ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|  
MAP_DENYWRITE, 3, 0x1fe000) = 0x7ef1a29ff000  
mmap(0x7ef1a2a05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|  
MAP_ANONYMOUS, -1, 0) = 0x7ef1a2a05000  
close(3)                                = 0  
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =  
0x7ef1a2cd0000  
arch_prctl(ARCH_SET_FS, 0x7ef1a2cd740) = 0  
set_tid_address(&tidptr)                = 7743  
set_robust_list(&rlist, 24)             = 0  
rseq(0x7ef1a2ce0e60, 0x20, 0, 0x53053053) = 0  
mprotect(0x7ef1a29ff000, 16384, PROT_READ) = 0  
mprotect(0x63f8dcb0e000, 4096, PROT_READ) = 0  
mprotect(0x7ef1a2c6e000, 8192, PROT_READ) = 0  
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,  
rlim_max=RLIM64_INFINITY}) = 0  
munmap(0x7ef1a2c10000, 153815)           = 0  
openat(AT_FDCWD, "FileMapping", O_RDWR|O_CREAT, 0666) = 3  
openat(AT_FDCWD, "Errors File Mapping", O_RDWR|O_CREAT, 0666) = 4  
ftruncate(3, 4104)                       = 0  
ftruncate(4, 4104)                       = 0  
mmap(NULL, 4104, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7ef1a2c34000  
mmap(NULL, 4104, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7ef1a2c32000  
getrandom("\x04\x02\xe5\x64\x87\x84\xeb\x31", 8, GRND_NONBLOCK) = 8  
brk(NULL)                                 = 0x63f913080000  
brk(0x63f9130a1000)                     = 0x63f9130a1000  
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0  
read(0, example.txt  
example.txt\n", 1024)                    = 12  
sigaction(SIGUSR2, {sa_handler=0x63f8dcb0c3e9, sa_mask=[USR2],  
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=&0x7ef1a284530}, {sa_handler=SIG_DFL,  
sa_mask=[], sa_flags=0}), 8) = 0  
(child_stack=NULL, flags=CLONE_CHILD_CLEAR_TID|CLONE_CHILD_SETTID|SIGCHLD, trace:  
Process 7745 attached)  
[pid 7745] set_robust_list(&rlist, 24)    = 0  
[pid 7743] read(0, <unfinished ...>  
vars [pid 7745]) execve("./child", ["./child", "FileMapping"], 0x7ffe23ff5b88 /* 63  
[pid 7745] brk(NULL)                      = 0x5c926e158000  
[pid 7745] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -  
1, 0) = 0x7aa540e95000
```

```
[pid 7745] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)
[pid 7745] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 5
[pid 7745] fstat(5, {st_mode=S_IFREG|0644, st_size=153815, ...}) = 0
[pid 7745] mmap(NULL, 153815, PROT_READ, MAP_PRIVATE, 5, 0) = 0x7aa540e6f000
[pid 7745] close(5) = 0
[pid 7745] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 5
[pid 7745] read(5, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\>\0\1\0\0\0\220\243\2\0\0\0\0\0"...; 832) = 832
[pid 7745] pread64(5, "\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"...; 784, 64) = 784
[pid 7745] fstat(5, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 7745] pread64(5, "\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"...; 784, 64) = 784
[pid 7745] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 5, 0) = 0x7aa540c00000
[pid 7745] mmap(0x7aa540c28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x28000) = 0x7aa540c28000
[pid 7745] mmap(0x7aa540db0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x1b0000) = 0x7aa540db0000
[pid 7745] mmap(0x7aa540dff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x1fe000) = 0x7aa540dff000
[pid 7745] mmap(0x7aa540e05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7aa540e05000
[pid 7745] close(5) = 0
[pid 7745] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7aa540e6c000
[pid 7745] arch_prctl(ARCH_SET_FS, 0x7aa540e6c740) = 0
[pid 7745] set_tid_address(0x7aa540e6ca10) = 7745
[pid 7745] set_robust_list(0x7aa540e6ca20, 24) = 0
[pid 7745] rseq(0x7aa540e6d060, 0x20, 0, 0x53053053) = 0
[pid 7745] mprotect(0x7aa540dff000, 16384, PROT_READ) = 0
[pid 7745] mprotect(0x5c92424f1000, 4096, PROT_READ) = 0
[pid 7745] mprotect(0x7aa540ecd000, 8192, PROT_READ) = 0
[pid 7745] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 7745] munmap(0x7aa540e6f000, 153815) = 0
[pid 7745] rt_sigaction(SIGUSR1, {sa_handler=0x5c92424ef3a9, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7aa540c43550}, {sa_handler=sig_DFL, sa_mask=[], sa_flags=0}, 8) = 0
[pid 7745] openat(AT_FDCWD, "FileMapping", O_RDWR) = 5
[pid 7745] openat(AT_FDCWD, "Errors File Mapping", O_RDWR) = 6
[pid 7745] fstat(5, {st_mode=S_IFREG|0664, st_size=4104, ...}) = 0
[pid 7745] mmap(NULL, 4104, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7aa540e93000
[pid 7745] mmap(NULL, 4104, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0) = 0x7aa540e91000
[pid 7745] getrandom("\x05\x7e\xc4\xae\x67\x49\x57\xb7", 8, GRND_NONBLOCK) = 8
[pid 7745] brk(NULL) = 0x5c926e158000
[pid 7745] brk(0x5c926e179000) = 0x5c926e179000
[pid 7745] openat(AT_FDCWD, "example.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 7
[pid 7745] fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
[pid 7745] write(1, "\320\336\321\202\320\272\321\200\321\213\321\202 \321\204\320\260\320\271\320\273 example.tx...; 540)крыт файл example.txt для записи) = 54
[pid 7745] pause>Hello
<unfinished ...>
[pid 7743] <... read resumed>"Hello\n", 1024) = 6
[pid 7743] fstat(1, {st mode=S IFCHR|0620, st rdev=makedev(0x88, 0), ...}) = 0
```

```

320\256\321\262\260\267\321\200\320\260\320\262\320\273\320\265\320\275\320\276\Hello
321\207\320\265\321\200\320\265\320\267"...; 520Правлено через File Mapping: Hello
) = 52
[pid 7743] kill(7745, SIGUSR1) = 0
no handler) [pid 7745] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if
[pid 7743] pause( <unfinished ...>
si_uid=1000) [pid 7745] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=7743,
[pid 7745] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
320\256\321\262\260\267\321\200\320\260\320\262\320\273\320\265\320\275\320\276\
321\207\320\265\321\200\320\265\320\267 File"...; 48Получено через File Mapping:
Hello
) = 48
[pid 7745] fstat(7, {st_mode=S_IFREG|0664, st_size=0, ...}) = 0
[pid 7745] write(7, "Hello\n", 6) = 6
[pid 7745] getppid() = 7743
[pid 7745] kill(7743, SIGUSR2) = 0
no handler) [pid 7743] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if
[pid 7745] pause( <unfinished ...>
si_uid=1000) [pid 7743] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=7745,
[pid 7743] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
[pid 7743] read(0, Abo abof
"Abo abof\n", 1024) = 9
320\256\321\262\260\267\321\200\320\260\320\262\320\273\320\265\320\275\320\276\
321\207\320\265\321\200\320\265\320\267 File"...; 550Правлено через File Mapping: Abo
abof
) = 55
[pid 7743] kill(7745, SIGUSR1) = 0
[pid 7743] pause( <unfinished ...>
no handler) [pid 7745] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if
si_uid=1000) [pid 7745] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=7743,
[pid 7745] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
320\256\321\262\260\267\321\200\320\260\320\262\320\273\320\265\320\275\320\276\
321\207\320\265\321\200\320\265\320\267 File"...; 51Получено через File Mapping: Abo
abof
) = 51
[pid 7745] write(7, "Abo abof\n", 9) = 9
[pid 7745] getppid() = 7743
[pid 7745] kill(7743, SIGUSR2) = 0
no handler) [pid 7743] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if
[pid 7745] pause( <unfinished ...>
si_uid=1000) [pid 7743] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=7745,
[pid 7743] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
[pid 7743] read(0, hi
"hi\n", 1024) = 3
320\256\321\262\260\267\321\200\320\260\320\262\320\273\320\265\320\275\320\276\
321\207\320\265\321\200\320\265\320\267 File"...; 490Правлено через File Mapping: hi
) = 49
[pid 7743] kill(7745, SIGUSR1) = 0
no handler) [pid 7745] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if

```

```

[pid 7743] pause( <unfinished ...>
[pid 7743] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=7743,
si_uid=1000}
[pid 7745] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
[pid 7745] write(1, "\203\321\207\320\265\320\275\320\276 \n", 45) = 45
[pid 7745] getppid() = 7743
[pid 7745] kill(7743, SIGUSR2) = 0
[pid 7743] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if no handler)
[pid 7745] getppid( <unfinished ...>
[pid 7743] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=7745,
si_uid=1000}
[pid 7745] <... getppid resumed> = 7743
[pid 7743] rt_sigreturn({mask=[]} <unfinished ...>
[pid 7745] kill(7743, SIGUSR2 <unfinished ...>
[pid 7743] <... rt_sigreturn resumed> = -1 EINTR (Прерван системный вызов)
[pid 7745] <... kill resumed> = 0
[pid 7743] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=7745,
si_uid=1000}
[pid 7745] pause( <unfinished ...>
[pid 7743] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
[pid 7743] write(1, "\265\321\200\320\275\320\270\320\271 \n", 55) = 55
[pid 7743] read(0, Hewfrwq
"Hewfrwq\n", 1024) = 8
[pid 7743] write(1, "\265\321\200\320\269\320\262\320\273\320\265\320\275\320\276 \n", 54) = 54
[pid 7743] kill(7745, SIGUSR1) = 0
[pid 7745] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if no handler)
[pid 7743] pause( <unfinished ...>
[pid 7745] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=7743,
si_uid=1000}
[pid 7745] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
[pid 7745] write(1, "\203\321\207\320\265\320\275\320\276 \n", 50) = 50
[pid 7745] write(7, "Hewfrwq\n", 8) = 8
[pid 7745] getppid() = 7743
[pid 7745] kill(7743, SIGUSR2) = 0
[pid 7743] <... pause resumed> = ? ERESTARTNOHAND (To be restarted if no handler)
[pid 7745] pause( <unfinished ...>
[pid 7743] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=7745,
si_uid=1000}
[pid 7743] rt_sigreturn({mask=[]}) = -1 EINTR (Прерван системный вызов)
[pid 7743] read(0, exit
"exit\n", 1024) = 5
[pid 7743] kill(7745, SIGUSR1) = 0

```

```

no handler) [pid 7745] <... pause resumed>          = ? ERESTARTNOHAND (To be restarted if
    [pid 7743] wait4(-1, <unfinished ...>
si_uid=1000) [pid 7745] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=7743,
    [pid 7745] rt_sigreturn({mask=[]})              = -1 EINTR (Прерван системный вызов)
320\259\320\276\320\278\321\203\321\207\320\265\320\275\320\276\
321\207\320\276\320\263\320\275\320\260\320\273\320\267\320\260"...., 49Получен
Hewfrwq File "..., 50Получено через File Mapping:
    ) = 50
320\259\320\276\320\278\321\203\321\207\320\265\320\275\320\276\
321\207\320\276\320\263\320\275\320\260\320\273\320\267\320\260"...., 49Получен
сигнал завершения
    ) = 49
    [pid 7745] close(7)                               = 0
    [pid 7745] munmap(0x7aa540e93000, 4104)            = 0
    [pid 7745] munmap(0x7aa540e91000, 4104)            = 0
    [pid 7745] close(6)                               = 0
    [pid 7745] close(5)                               = 0
    [pid 7745] exit_group(0)                           = ?
    [pid 7745] +++ exited with 0 +++
    <... wait4 resumed>NULL, 0, NULL)                  = 7745
    SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=7745, si_uid=1000,
si_status=0, si_utime=0, si_stime=0}
    munmap(0x7ef1a2c34000, 4104)                       = 0
    close(3)                                           = 0
    exit_group(0)                                     = ?
    +++ exited with 0 +++

```

## Вывод

Лабораторная работа показалась достаточно интересной и важной, поскольку научился работать с разделяемой памятью и отображением файла в память процесса (File Mapping).

Освоил системные вызовы `mmap`, `munmap`, `ftruncate`. Реализована синхронизация процессов через сигналы (SIGUSR1, SIGUSR2).

Сталкивался со сложностью работы со строками произвольной длины, поскольку `mmap` создаёт фиксированное отображение.