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: Волков А.Д.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 17.11.2023

Москва, 2023

Постановка задачи

Вариант 14.

Child1 переводит строки в нижний регистр. Child2 убирает все задвоенные пробелы.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int open(const char *pathname, int flags, mode_t mode); – преобразует путь к файлу в файловый дескриптор, с учетом необходимых пользователю атрибутов.
- int ftruncate(int fd, off_t length); – устанавливает заданную длину файла в байт.
- void * mmap(void *start, size_t length, int prot , int flags, int fd, off_t offset); – отображает length байт из fd в память.
- int close(int fd); – закрывает файловый дескриптор, который после этого не ссылается ни на один и файл и может быть использован повторно.
- void *malloc(size_t size); – распределяет size байт и возвращает указатель на распределенную память.
- int execv(const char *path, const char *arg, ...); – заменяет текущий образ процесса новым образом процесса.
- pid_t wait(int *status); – приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится, или до появления сигнала, который либо завершает текущий процесс, либо требует вызвать функцию-обработчик.
- int munmap(void *start, size_t length); – снимает отображение length байт.

Родительский процесс создает файлы для хранения строки и количества элементов строки. Далее он определяет размер этих файлов и отображает их в память. После этого родительский процесс закрывает файловые дескрипторы и считывает пользовательскую строку и записывает саму строку и количество элементов в ней в отображенную память, после чего начинает ждать выполнение дочерних процессов, которым в свою очередь параметром запуска подается массив с названиями файлов для хранения счетчика и строки. Дочерние процессы выполняют абсолютно те же действия с файлами (открытие, задание размера, отображение в память, закрытие дескрипторов). После чего они выполняют необходимые действия со строкой, после чего в обоих процессах снимается отображение файла, и дочерние процессы завершаются. Ожидающий родительский процесс получает сигнал о завершении дочерних процессов и выводит получившуюся строку, после чего он, так же как и дочерние процессы, снимает отображение файла и завершает свою работу.

Код программы

string.h

```
#include <stdio.h>
#include <stdlib.h>

#define DEFAULT 80

typedef struct {
    char *str;
    int length;
} my_string;

my_string *create_string();
void read_string(my_string *mstr);
```

mstring.c

```
#include "string.h"

my_string *create_string() {
    my_string *tmp = (my_string *) malloc(sizeof(my_string));
    tmp->str = (char *) malloc(sizeof(char) * DEFAULT);
    tmp->length = 0; return tmp;
}

void read_string(my_string *mstr) {
    char c;
    while ((c = getchar()) != 10) {
        mstr->str[mstr->length] = c; mstr->length++;
    }
}
```

parent.c

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

#define MEM_SIZE 4096

pid_t create_process() {
    pid_t pid = fork();
    if (-1 == pid) {
        perror("fork");
        exit(-1);
    }
    return pid;
}
```

```
int main() {
    int fd, fd_count;
    char *file_in_memory; int *count_in_memory;
    if (((fd = open("file", O_RDWR | O_CREAT, 0666)) == -1) || ((fd_count =
open("count", O_RDWR | O_CREAT, 0666)) == -1)) {
        perror("open");
        return -1;
    }
    if ((ftruncate(fd, MEM_SIZE) == -1) || (ftruncate(fd_count, MEM_SIZE) == -1)) {
        perror("ftruncate");
        return -1;
    }
    if (((file_in_memory = mmap(NULL, MEM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
 0)) == MAP_FAILED) || ((count_in_memory = mmap(NULL, MEM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, fd_count, 0)) == MAP_FAILED)) {
        perror("mmap");
        return -1;
    }
    if ((close(fd) == -1) || (close(fd_count) == -1)) {
        perror("close");
        return -1;
    }
    char *argv[] = {"file", "count", (char *) NULL};
    my_string *p_mstr = create_string();
    printf("Enter your string: "); read_string(p_mstr);
    for (int i = 0; i < p_mstr->length; i++) {
        file_in_memory[i] = p_mstr->str[i];
    }
    count_in_memory[0] = p_mstr->length;
    pid_t cp1, cp2;
    if ((cp1 = create_process()) == 0) { //child1
        if (execv("../build/child1", argv) == -1) {
            perror("execv");
            return -1;
        }
    } else if (cp1 > 0 && (cp2 = create_process()) == 0) { //child2
        if (execv("../build/child2", argv) == -1) {
            perror("execv");
            return -1;
        }
    } else { //parent
        wait(NULL);
        wait(NULL);
        printf("Result: ");
        for (int i = 0; i < count_in_memory[0]; i++) {
            printf("%c", file_in_memory[i]);
        }
        printf("\n");
```

```
        if ((munmap(count_in_memory, MEM_SIZE) == -1) || (munmap(file_in_memory,
MEM_SIZE) == -1)) {
            perror("munmap");
            return -1;
        }
    }
    return 0;
}
```

child1.c

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

#define MEM_SIZE 4096

char tolowerc(char c) {
    if (c >= 'A' && c <= 'Z') {
        c += 32;
    }
    return c;
}

int main(int argc, char *argv[]) {
    int fd, fd_count;
    char *file_in_memory; int *count_in_memory;
    if (((fd = open(argv[0], O_RDWR, 0666)) == -1) || ((fd_count = open(argv[1],
O_RDWR, 0666)) == -1)) {
        perror("open");
        exit(-1);
    }
    if ((ftruncate(fd, MEM_SIZE) == -1) || (ftruncate(fd_count, MEM_SIZE) == -1)) {
        perror("ftruncate");
        exit(-1);
    }
    if (((file_in_memory = mmap(NULL, MEM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
 0)) == MAP_FAILED) || ((count_in_memory = mmap(NULL, MEM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, fd_count, 0)) == MAP_FAILED)) {
        perror("mmap");
        exit(-1);
    }
    if ((close(fd) == -1) || (close(fd_count) == -1)) {
        perror("close");
        exit(-1);
    }
    for (int i = 0; i < count_in_memory[0]; i++) {
        file_in_memory[i] = tolowerc(file_in_memory[i]);
```

```
    }
    if ((munmap(count_in_memory, MEM_SIZE) == -1) || (munmap(file_in_memory, MEM_SIZE)
 == -1)) {
        perror("munmap");
        exit(-1);
    }
    return 0;
}
```

child2.c

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

#define MEM_SIZE 4096

void delete_double_spaces(char *mstr, size_t size) {
    for (size_t i = 0; i < size; i++) {
        if (mstr[i] == 32 && mstr[i + 1] == 32) {
            mstr[i] = 0;
        }
    }
}

int main(int argc, char *argv[]) {
    int fd, fd_count;
    char *file_in_memory; int *count_in_memory;
    if (((fd = open(argv[0], O_RDWR, 0666)) == -1) || ((fd_count = open(argv[1],
O_RDWR, 0666)) == -1)) {
        perror("open");
        exit(-1);
    }
    if ((ftruncate(fd, MEM_SIZE) == -1) || (ftruncate(fd_count, MEM_SIZE) == -1)) {
        perror("ftruncate");
        exit(-1);
    }
    if (((file_in_memory = mmap(NULL, MEM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
0)) == MAP_FAILED) || ((count_in_memory = mmap(NULL, MEM_SIZE, PROT_READ | PROT_WRITE,
 MAP_SHARED, fd_count, 0)) == MAP_FAILED)) {
        perror("mmap");
        exit(-1);
    }
    if ((close(fd) == -1) || (close(fd_count) == -1)) {
        perror("close");
        exit(-1);
    }
```

```
    delete_double_spaces(file_in_memory, count_in_memory[0]);
    if ((munmap(count_in_memory, MEM_SIZE) == -1) || (munmap(file_in_memory, MEM_SIZE)
 == -1)) {
        perror("munmap");
        exit(-1);
    }
    return 0;
}
```

Протокол работы программы

Тестирование:

```
$ ./parent
Enter your string: ASDSADAS    asdasd
Result: asdsadas asdasd
$ ./parent
Enter your string:
Result:
$ ./parent
Enter your string: ASDSADASD
Result: asdsadasd
$ ./parent
Enter your string: asdasd
Result: asdasd
$ ./parent
Enter your string: asdsad     dfadasd
Result: asdsad dfadasd
$ ./parent
Enter your string: ASDSAD sadsad
Result: asdsad sadsad
```

Strace:

```
$ strace -f ./parent
execve("./parent", ["./parent"], 0x7ffedd737408 /* 76 vars */) = 0
brk(NULL)                               = 0x5572908a7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc0dcd8ee0) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7fdff32be000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=67971, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 67971, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdff32ad000
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"..., 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) = 0x7fdff3000000
mmap(0x7fdff3028000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7fdff3028000
mmap(0x7fdff31bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7fdff31bd000
mmap(0x7fdff3215000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
 3, 0x214000) = 0x7fdff3215000
mmap(0x7fdff321b000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
 -1, 0) = 0x7fdff321b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fdff32aa000
arch_prctl(ARCH_SET_FS, 0x7fdff32aa740) = 0
set_tid_address(0x7fdff32aaa10)         = 60497
set_robust_list(0x7fdff32aaa20, 24)     = 0
rseq(0x7fdff32ab0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7fdff3215000, 16384, PROT_READ) = 0
mprotect(0x55728edec000, 4096, PROT_READ) = 0
mprotect(0x7fdff32f8000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fdff32ad000, 67971)           = 0
openat(AT_FDCWD, "file", O_RDWR|O_CREAT, 0666) = 3
openat(AT_FDCWD, "count", O_RDWR|O_CREAT, 0666) = 4
ftruncate(3, 4096)                      = 0
```

```
ftruncate(4, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fdff32f7000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fdff32bd000
close(3)                                = 0
close(4)                                = 0
getrandom("\xf7\x79\x69\x52\x04\xea\x64\xf2", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5572908a7000
brk(0x5572908c8000)                     = 0x5572908c8000
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...},
AT_EMPTY_PATH) = 0
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...},
AT_EMPTY_PATH) = 0
write(1, "Enter your string: ", 19Enter your string: )     = 19
read(0, asdsad   ASDSAD
"asdsad   ASDSAD\n", 1024)      = 16
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
 Process 60534 attached
, child_tidptr=0x7fdff32aaa10) = 60534
[pid 60534] set_robust_list(0x7fdff32aaa20, 24) = 0
[pid 60497] clone(child_stack=NULL,
 flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid 60534] execve("../build/child1", ["file", "count"], 0x7ffc0dcd90b8 /* 76 vars
*/strace: Process 60535 attached
 <unfinished ...>
[pid 60497] <... clone resumed>, child_tidptr=0x7fdff32aaa10) = 60535
[pid 60497] wait4(-1,  <unfinished ...>
[pid 60535] set_robust_list(0x7fdff32aaa20, 24) = 0
[pid 60535] execve("../build/child2", ["file", "count"], 0x7ffc0dcd90b8 /* 76 vars */
<unfinished ...>
[pid 60534] <... execve resumed>)       = 0
[pid 60534] brk(NULL)                   = 0x55aa5e85b000
[pid 60534] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd69c13590 <unfinished ...>
[pid 60535] <... execve resumed>)       = 0
[pid 60534] <... arch_prctl resumed>)   = -1 EINVAL (Недопустимый аргумент)
[pid 60535] brk(NULL)                   = 0x556a7f71b000
[pid 60534] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fbdd86b4000
[pid 60535] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe8beb19a0) = -1 EINVAL
(Недопустимый
аргумент)
[pid 60534] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 60535] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid 60534] <... access resumed>)       = -1 ENOENT (Нет такого файла или каталога)
[pid 60535] <... mmap resumed>)         = 0x7f2340f7b000
[pid 60534] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 60535] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 60534] <... openat resumed>)       = 3
[pid 60535] <... access resumed>)       = -1 ENOENT (Нет такого файла или каталога)
```

```
[pid 60535] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 60534] newfstatat(3, "",  <unfinished ...>
[pid 60535] <... openat resumed>)       = 3
[pid 60534] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=67971, ...},
AT_EMPTY_PATH) = 0
[pid 60535] newfstatat(3, "",  <unfinished ...>
[pid 60534] mmap(NULL, 67971, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbdd86a3000
[pid 60534] close(3)                    = 0
[pid 60534] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 60535] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=67971, ...},
 AT_EMPTY_PATH) = 0
[pid 60534] <... openat resumed>)       = 3
[pid 60535] mmap(NULL, 67971, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 60534] read(3,  <unfinished ...>
[pid 60535] <... mmap resumed>)         = 0x7f2340f6a000
[pid 60534] <... read
 resumed>"\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 60535] close(3 <unfinished ...>
[pid 60534] pread64(3,  <unfinished ...>
[pid 60535] <... close resumed>)        = 0
[pid 60534] <... 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
[pid 60535] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 60534] 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 60535] <... openat resumed>)       = 3
[pid 60534] pread64(3,  <unfinished ...>
[pid 60535] read(3,  <unfinished ...>
[pid 60534] <... pread64
resumed>"\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) = 6
[pid 60535] <... read
resumed>"\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 60534] newfstatat(3, "",  <unfinished ...>
[pid 60535] 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 60534] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2216304, ...},
AT_EMPTY_PATH) = 0
[pid 60535] 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 60534] pread64(3,  <unfinished ...>
[pid 60535] pread64(3,  <unfinished ...>
[pid 60534] <... 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
```

```
[pid 60535] <... pread64
resumed>"\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 60534] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished
...>
[pid 60535] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...},
AT_EMPTY_PATH) = 0
[pid 60534] <... mmap resumed>)         = 0x7fbdd8400000
[pid 60535] pread64(3,  <unfinished ...>
[pid 60534] mmap(0x7fbdd8428000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
[pid 60535] <... 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
[pid 60535] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished
...>
[pid 60534] <... mmap resumed>)         = 0x7fbdd8428000
[pid 60535] <... mmap resumed>)         = 0x7f2340c00000
[pid 60534] mmap(0x7fbdd85bd000, 360448, PROT_READ,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>
[pid 60535] mmap(0x7f2340c28000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
[pid 60534] <... mmap resumed>)         = 0x7fbdd85bd000
[pid 60535] <... mmap resumed>)         = 0x7f2340c28000
[pid 60534] mmap(0x7fbdd8615000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000 <unfinished ...>
[pid 60535] mmap(0x7f2340dbd000, 360448, PROT_READ,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f2340dbd000
[pid 60534] <... mmap resumed>)         = 0x7fbdd8615000
[pid 60535] mmap(0x7f2340e15000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000 <unfinished ...>
[pid 60534] mmap(0x7fbdd861b000, 52816, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fbdd861b000
[pid 60535] <... mmap resumed>)         = 0x7f2340e15000
[pid 60535] mmap(0x7f2340e1b000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 60534] close(3)                    = 0
[pid 60535] <... mmap resumed>)         = 0x7f2340e1b000
[pid 60534] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid 60535] close(3)                    = 0
[pid 60534] <... mmap resumed>)         = 0x7fbdd86a0000
[pid 60535] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid 60534] arch_prctl(ARCH_SET_FS, 0x7fbdd86a0740) = 0
[pid 60535] <... mmap resumed>)         = 0x7f2340f67000
[pid 60534] set_tid_address(0x7fbdd86a0a10) = 60534
[pid 60535] arch_prctl(ARCH_SET_FS, 0x7f2340f67740 <unfinished ...>
[pid 60534] set_robust_list(0x7fbdd86a0a20, 24 <unfinished ...>
[pid 60535] <... arch_prctl resumed>)   = 0
[pid 60534] <... set_robust_list resumed>) = 0
```

```
[pid 60534] rseq(0x7fbdd86a10e0, 0x20, 0, 0x53053053) = 0
[pid 60534] mprotect(0x7fbdd8615000, 16384, PROT_READ) = 0
[pid 60534] mprotect(0x55aa5e38b000, 4096, PROT_READ <unfinished ...>
[pid 60535] set_tid_address(0x7f2340f67a10 <unfinished ...>
[pid 60534] <... mprotect resumed>)     = 0
[pid 60535] <... set_tid_address resumed>) = 60535
[pid 60534] mprotect(0x7fbdd86ee000, 8192, PROT_READ <unfinished ...>
[pid 60535] set_robust_list(0x7f2340f67a20, 24 <unfinished ...>
[pid 60534] <... mprotect resumed>)     = 0
[pid 60534] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 60534] munmap(0x7fbdd86a3000, 67971) = 0
[pid 60535] <... set_robust_list resumed>) = 0
[pid 60534] openat(AT_FDCWD, "file", O_RDWR <unfinished ...>
[pid 60535] rseq(0x7f2340f680e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 60534] <... openat resumed>)       = 3
[pid 60534] openat(AT_FDCWD, "count", O_RDWR) = 4
[pid 60534] ftruncate(3, 4096)          = 0
[pid 60534] ftruncate(4, 4096 <unfinished ...>
[pid 60535] <... rseq resumed>)         = 0
[pid 60534] <... ftruncate resumed>)    = 0
[pid 60534] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>
[pid 60534] <... mmap resumed>)         = 0x7fbdd86ed000
[pid 60534] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
[pid 60534] <... mmap resumed>)         = 0x7fbdd86b3000
[pid 60535] mprotect(0x7f2340e15000, 16384, PROT_READ <unfinished ...>
[pid 60534] close(3 <unfinished ...>
[pid 60535] <... mprotect resumed>)     = 0
[pid 60534] <... close resumed>)        = 0
[pid 60535] mprotect(0x556a7f107000, 4096, PROT_READ <unfinished ...>
[pid 60534] close(4 <unfinished ...>
[pid 60535] <... mprotect resumed>)     = 0
[pid 60534] <... close resumed>)        = 0
[pid 60535] mprotect(0x7f2340fb5000, 8192, PROT_READ <unfinished ...>
[pid 60535] <... mprotect resumed>)     = 0
[pid 60534] munmap(0x7fbdd86b3000, 4096 <unfinished ...>
[pid 60534] <... munmap resumed>)       = 0
[pid 60534] munmap(0x7fbdd86ed000, 4096 <unfinished ...>
[pid 60534] <... munmap resumed>)       = 0
[pid 60535] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 60535] munmap(0x7f2340f6a000, 67971 <unfinished ...>
[pid 60534] exit_group(0)               = ?
[pid 60535] <... munmap resumed>)       = 0
[pid 60535] openat(AT_FDCWD, "file", O_RDWR) = 3
[pid 60535] openat(AT_FDCWD, "count", O_RDWR) = 4
[pid 60535] ftruncate(3, 4096 <unfinished ...>
[pid 60534] +++ exited with 0 +++
[pid 60535] <... ftruncate resumed>)    = 0
```

```
[pid 60497] <... wait4 resumed>NULL, 0, NULL) = 60534
[pid 60535] ftruncate(4, 4096 <unfinished ...>
[pid 60497] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=60534,
si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
[pid 60535] <... ftruncate resumed>)    = 0
[pid 60497] wait4(-1,  <unfinished ...>
[pid 60535] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f2340fb4000
[pid 60535] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f2340f7a000
[pid 60535] close(3)                    = 0
[pid 60535] close(4)                    = 0
[pid 60535] munmap(0x7f2340f7a000, 4096) = 0
[pid 60535] munmap(0x7f2340fb4000, 4096) = 0
[pid 60535] exit_group(0)               = ?
[pid 60535] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 60535
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=60535, si_uid=1000,
 si_status=0, si_utime=0, si_stime=0} ---
write(1, "Result: asdsad\0\0 asdsad\n", 24Result: asdsad asdsad
) = 24
munmap(0x7fdff32bd000, 4096)            = 0
munmap(0x7fdff32f7000, 4096)            = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Главная проблема, с которой я столкнулся при выполнении этой лабораторной работы, была нехватка понятного кода, который бы выполнял нужные действия. Поэтому пришлось тестировать все самому, и писать программу “на ощупь”. Но благодаря этому я лучше усвоил тему отображения файлов в память. В остальном это была абсолютно та же ЛР№1, где так же процессы не могут записывать в отображаемую память структуру, что доставило кучу неприятностей при написании программы**!!!**