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: Корепанов И.А.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 05.12.2025

Москва, 2025

Постановка задачи

Вариант 10.

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).
В файле записаны команды вида: «число». Дочерний процесс производит проверку этого числа на простоту. Если число составное, то дочерний процесс пишет это число в стандартный поток вывода. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются.

Общий метод и алгоритм решения

Использованные системные вызовы:

• shm_open(name, oflag, mode) - создает или открывает объект POSIX разделяемой памяти с указанным именем, флагами и правами доступа

• ftruncate(fd, length) - устанавливает размер объекта разделяемой памяти до указанной длины

• mmap(addr, length, prot, flags, fd, offset) - отображает объект разделяемой памяти в адресное пространство процесса для прямого доступа через указатели

• munmap(addr, length) - удаляет отображение области памяти из адресного пространства процесса

• shm_unlink(name) - удаляет объект разделяемой памяти из системы по имени

• fork() - создает новый дочерний процесс как точную копию родительского процесса

• execl(path, arg0, arg1, ..., NULL) - заменяет образ текущего процесса новым исполняемым файлом с передачей аргументов командной строки

• waitpid(pid, status, options) - ожидает завершения указанного дочернего процесса и получает его статус завершения

• exit(status) - завершает выполнение текущего процесса с указанным кодом возврата

Алгоритм работы программы:

Сначала родительский процесс создает необходимые переменные для работы с разделяемой памятью, включая файловый дескриптор shm_fd и указатель shm_ptr. С помощью системного вызова shm_open() создается объект разделяемой памяти с именем "/prime_shm", который будет использоваться для обмена данными между процессами. Затем с помощью ftruncate() устанавливается размер области памяти. С помощью системного вызова fork() создается точная копия родительского процесса — дочерний процесс. Дочерний процесс открывает указанный файл для чтения и подключается к существующей разделяемой памяти

через shm_open() и mmap(). После вывода результатов родительский процесс освобождает все используемые ресурсы: с помощью munmap()

Код программы

parent.c

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

#define SHM_SIZE 1024

int main() {
    char filename[256];
    printf("файл: ");
    if (scanf("%255s", filename) != 1) {
        fprintf(stderr, "Неверное имя файла\n");
        exit(EXIT_FAILURE);
    }
    int shm_fd = shm_open("/prime_shm", O_CREAT | O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(EXIT_FAILURE);
    }
    if (ftruncate(shm_fd, SHM_SIZE) == -1) {
        perror("ftruncate");
```

```
        exit(EXIT_FAILURE);
    }
    char *shm_ptr = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if (shm_ptr == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }

    shm_ptr[0] = '\0';

    pid_t pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        execl("./child", "child", filename, (char *)NULL);
        perror("execl");
        exit(EXIT_FAILURE);
    } else {
        wait(NULL);
        printf("Результат от дочернего процесса:\n%s", shm_ptr);
        munmap(shm_ptr, SHM_SIZE);
        close(shm_fd);
        shm_unlink("/prime_shm");
    }

    return 0;
```

```
}
```

child.c

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

#define INITIAL_SIZE 1024

int isPrime(int n) {
    if (n <= 1) return 0;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}
int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Использование: %s <имя файла>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    FILE *file = fopen(argv[1], "r");
    if (!file) {
        perror("fopen");
        exit(EXIT_FAILURE);
```

```
}
int shm_fd = shm_open("/prime_shm", O_RDWR, 0666);
if (shm_fd == -1) {
    perror("shm_open");
    exit(EXIT_FAILURE);
}
size_t buffer_size = INITIAL_SIZE;
char *local_buffer = malloc(buffer_size);
if (!local_buffer) {
    perror("malloc");
    exit(EXIT_FAILURE);
}
local_buffer[0] = '\0';
size_t used = 0;
char line[256];
while (fgets(line, sizeof(line), file)) {
    int num = atoi(line);
    if (num <= 0) {
        break;
    }
    if (!isPrime(num)) {
        char temp[20];
        int len = snprintf(temp, sizeof(temp), "%d\n", num);
        if (used + len + 1 >= buffer_size) {
            buffer_size *= 2;
            char *new_buffer = realloc(local_buffer, buffer_size);
            if (!new_buffer) {
                perror("realloc");
                free(local_buffer);
```

```
                exit(EXIT_FAILURE);
            }
            local_buffer = new_buffer;
        }
        strcat(local_buffer, temp);
        used += len;
    } else {
        break;
    }
}
fclose(file);
size_t needed_size = used + 1;
if (ftruncate(shm_fd, needed_size) == -1) {
    perror("ftruncate");
    free(local_buffer);
    exit(EXIT_FAILURE);
}
char *shm_ptr = mmap(NULL, needed_size, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
if (shm_ptr == MAP_FAILED) {
    perror("mmap");
    free(local_buffer);
    exit(EXIT_FAILURE);
}
memcpy(shm_ptr, local_buffer, needed_size);
munmap(shm_ptr, needed_size);
close(shm_fd);
free(local_buffer);
return 0;
}
```

Протокол работы программы

Тестирование:

```
./parent

файл: input.txt

Результат от дочернего процесса:

44

6
```

Strace:

strace -f ./parent

execve("./parent", ["./parent"], 0x7fffe7e3ce48 /* 33 vars */) = 0

brk(NULL) = 0x57060324d000

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

fstat(3, {st_mode=S_IFREG|0644, st_size=21784, ...}) = 0

mmap(NULL, 21784, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7dd0d81ff000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/librt.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\260#\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=35808, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7dd0d81fd000

mmap(NULL, 39904, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7dd0d81f3000

mmap(0x7dd0d81f5000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7dd0d81f5000

mmap(0x7dd0d81f9000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7dd0d81f9000

mmap(0x7dd0d81fb000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7dd0d81fb000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260>\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=1901536, ...}) = 0

mmap(NULL, 1914496, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7dd0d801f000

mmap(0x7dd0d8041000, 1413120, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7dd0d8041000

mmap(0x7dd0d819a000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17b000) = 0x7dd0d819a000

mmap(0x7dd0d81e9000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c9000) = 0x7dd0d81e9000

mmap(0x7dd0d81ef000, 13952, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7dd0d81ef000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", 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 I\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=149520, ...}) = 0

mmap(NULL, 136304, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7dd0d7ffd000

mmap(0x7dd0d8003000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7dd0d8003000

mmap(0x7dd0d8013000, 24576, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16000) = 0x7dd0d8013000

mmap(0x7dd0d8019000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b000) = 0x7dd0d8019000

mmap(0x7dd0d801b000, 13424, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7dd0d801b000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7dd0d7ffa000

arch_prctl(ARCH_SET_FS, 0x7dd0d7ffa740) = 0

mprotect(0x7dd0d81e9000, 16384, PROT_READ) = 0

mprotect(0x7dd0d8019000, 4096, PROT_READ) = 0

mprotect(0x7dd0d81fb000, 4096, PROT_READ) = 0

mprotect(0x5705da866000, 4096, PROT_READ) = 0

mprotect(0x7dd0d822f000, 4096, PROT_READ) = 0

munmap(0x7dd0d81ff000, 21784) = 0

set_tid_address(0x7dd0d7ffaa10) = 5772

set_robust_list(0x7dd0d7ffaa20, 24) = 0

rt_sigaction(SIGRTMIN, {sa_handler=0x7dd0d8003690, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7dd0d8010140}, NULL, 8) = 0

rt_sigaction(SIGRT_1, {sa_handler=0x7dd0d8003730, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7dd0d8010140}, NULL, 8) = 0

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

brk(NULL) = 0x57060324d000

brk(0x57060326e000) = 0x57060326e000

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

write(1, "\321\204\320\260\320\271\320\273: ", 10файл:) = 10

read(0, inpput.txt

"inpput.txt\n", 1024) = 11

statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=16384, f_bfree=16384, f_bavail=16384, f_files=973465, f_ffree=973464, f_fsid={val=[969116940, 50866251]}, f_namelen=255, f_frsize=4096, f_flags=ST_VALID|ST_NOSUID|ST_NODEV|ST_NOEXEC|ST_RELATIME}) = 0

futex(0x7dd0d801e410, FUTEX_WAKE_PRIVATE, 2147483647) = 0

openat(AT_FDCWD, "/dev/shm/prime_shm", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

ftruncate(3, 1024) = 0

mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7dd0d822e000

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 5858 attached

, child_tidptr=0x7dd0d7ffaa10) = 5858

[pid 5858] set_robust_list(0x7dd0d7ffaa20, 24 <unfinished ...>

[pid 5772] wait4(-1, <unfinished ...>

[pid 5858] <... set_robust_list resumed>) = 0

[pid 5858] execve("./child", ["child", "inpput.txt"], 0x7fff032f3098 /* 33 vars */) = 0

[pid 5858] brk(NULL) = 0x5ff73f70a000

[pid 5858] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

[pid 5858] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

[pid 5858] fstat(3, {st_mode=S_IFREG|0644, st_size=21784, ...}) = 0

[pid 5858] mmap(NULL, 21784, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7e48750c0000

[pid 5858] close(3) = 0

[pid 5858] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/librt.so.1", O_RDONLY|O_CLOEXEC) = 3

[pid 5858] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260#\0\0\0\0\0\0"..., 832) = 832

[pid 5858] fstat(3, {st_mode=S_IFREG|0644, st_size=35808, ...}) = 0

[pid 5858] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7e48750be000

[pid 5858] mmap(NULL, 39904, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7e48750b4000

[pid 5858] mmap(0x7e48750b6000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7e48750b6000

[pid 5858] mmap(0x7e48750ba000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7e48750ba000

[pid 5858] mmap(0x7e48750bc000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7e48750bc000

[pid 5858] close(3) = 0

[pid 5858] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

[pid 5858] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260>\2\0\0\0\0\0"..., 832) = 832

[pid 5858] fstat(3, {st_mode=S_IFREG|0755, st_size=1901536, ...}) = 0

[pid 5858] mmap(NULL, 1914496, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7e4874ee0000

[pid 5858] mmap(0x7e4874f02000, 1413120, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7e4874f02000

[pid 5858] mmap(0x7e487505b000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17b000) = 0x7e487505b000

[pid 5858] mmap(0x7e48750aa000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c9000) = 0x7e48750aa000

[pid 5858] mmap(0x7e48750b0000, 13952, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7e48750b0000

[pid 5858] close(3)   = 0

[pid 5858] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3

[pid 5858] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 I\0\0\0\0\0\0"..., 832) = 832

[pid 5858] fstat(3, {st_mode=S_IFREG|0755, st_size=149520, ...}) = 0

[pid 5858] mmap(NULL, 136304, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7e4874ebe000

[pid 5858] mmap(0x7e4874ec4000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7e4874ec4000

[pid 5858] mmap(0x7e4874ed4000, 24576, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16000) = 0x7e4874ed4000

[pid 5858] mmap(0x7e4874eda000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b000) = 0x7e4874eda000

[pid 5858] mmap(0x7e4874edc000, 13424, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7e4874edc000

[pid 5858] close(3)   = 0

[pid 5858] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7e4874ebb000

[pid 5858] arch_prctl(ARCH_SET_FS, 0x7e4874ebb740) = 0

[pid 5858] mprotect(0x7e48750aa000, 16384, PROT_READ) = 0

[pid 5858] mprotect(0x7e4874eda000, 4096, PROT_READ) = 0

[pid 5858] mprotect(0x7e48750bc000, 4096, PROT_READ) = 0

[pid 5858] mprotect(0x5ff725e57000, 4096, PROT_READ) = 0

[pid 5858] mprotect(0x7e48750f0000, 4096, PROT_READ) = 0

[pid 5858] munmap(0x7e48750c0000, 21784) = 0

[pid 5858] set_tid_address(0x7e4874ebba10) = 5858

[pid 5858] set_robust_list(0x7e4874ebba20, 24) = 0

[pid 5858] rt_sigaction(SIGRTMIN, {sa_handler=0x7e4874ec4690, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7e4874ed1140}, NULL, 8) = 0

[pid 5858] rt_sigaction(SIGRT_1, {sa_handler=0x7e4874ec4730, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7e4874ed1140}, NULL, 8) = 0

```
[pid  5858] rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
[pid  5858] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid  5858] brk(NULL)                   = 0x5ff73f70a000
[pid  5858] brk(0x5ff73f72b000)         = 0x5ff73f72b000
[pid  5858] openat(AT_FDCWD, "inpput.txt", O_RDONLY) = -1 ENOENT (No such file or directory)
[pid  5858] dup(2)                      = 3
[pid  5858] fcntl(3, F_GETFL)           = 0x2 (flags O_RDWR)
[pid  5858] fstat(3, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
[pid  5858] write(3, "fopen: No such file or directory"..., 33fopen: No such file or directory
) = 33
[pid  5858] close(3)                    = 0
[pid  5858] exit_group(1)               = ?
[pid  5858] +++ exited with 1 +++
<... wait4 resumed>NULL, 0, NULL)       = 5858
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5858, si_uid=1000, si_status=1,
si_utime=0, si_stime=1} ---
write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202 \320\276\321\202
\320\264\320\276\321\207\320\265"..., 61Результат от дочернего процесса:
) = 61
munmap(0x7dd0d822e000, 1024)            = 0
close(3)                                = 0
unlink("/dev/shm/prime_shm")            = 0
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были освоены принципы межпроцессного взаимодействия с использованием технологии File Mapping (Memory-Mapped Files). На практике реализован механизм разделяемой памяти POSIX, позволяющий нескольким процессам обмениваться данными через общую область памяти без необходимости копирования.