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-215Б-23

Студент: Матвеев Семен Романович

Преподаватель: Миронов Евгений Сергеевич

Оценка: в

Дата: 13.03.25

Москва, 2025

# Постановка задачи

## Вариант 6.

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).

Родительский процесс получает через пользовательский ввод путь к файлу, из которого построчно считываются данные, загружает его в memory-mapped file, дочерний процесс считывает построчно считывает данные из memory-mapped file, после чего записывает результат своей работы в memory-mapped file, а родительский процесс его получает и выводит в консоль.

# Общий метод и алгоритм решения

1. Родительский процесс (parent.c):

Запрашивает путь к файлу у пользователя. Открывает файл для чтения. Создает разделяемую память. Открывает файл разделяемой памяти (SHARED\_FILENAME). Устанавливает размер разделяемой памяти (ftruncate). Отображает файл в память с помощью mmap. Инициализирует семафоры для синхронизации:sem\_parent — для ожидания завершения работы дочернего процесса,sem\_child — для ожидания завершения работы родительского процесса.Создает дочерний процесс с помощью fork и запускает программу child.c через execl.

Читает файл построчно: Передает строку в разделяемую память. Ожидает, пока дочерний процесс обработает строку и вернет результат. Построчно выводит результат на экран. Этот этап синхронизируется посредством семафора

Завершает работу: Ожидает завершения дочернего процесса (wait). Освобождает ресурсы (семафоры, разделяемую память, файл).

2. Дочерний процесс (child.c):

Открывает разделяемую память: Открывает файл разделяемой памяти (SHARED\_FILENAME). Отображает файл в память с помощью mmap. Ожидает данных от родительского процесса:Ожидает сигнала от семафора sem\_parent. Если данных нет (data\_ready == 0), завершает работу. Вычисляет сумму чисел в строке: Использует функцию get\_line\_sum. Формирует ответ: Добавляет сумму чисел к строке. Записывает результат в разделяемую память. Сигнализирует родительскому процессу через семафор sem\_child. Освобождает ресурсы (разделяемую память).

# Код программы

## shared.h

#ifndef SHARED\_H

#define SHARED\_H

#include <semaphore.h>

#define BUFFER\_SIZE 4096

#define SHARED\_FILENAME "shared\_memory.dat"

typedef struct {

char buffer[BUFFER\_SIZE];

int data\_ready;

sem\_t sem\_parent;

sem\_t sem\_child;

} SharedMemory;

#endif

**parent.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

#include "shared.h"

int main() {

char filepath[BUFFER\_SIZE];

printf("Введите путь к файлу:\n");

scanf("%s", filepath);

FILE \*file = fopen(filepath, "r");

if (!file) {

perror("Ошибка fopen");

exit(1);

}

int fd = open(SHARED\_FILENAME, O\_CREAT | O\_RDWR, 0666); // 0666 — права доступа (чтение и запись для всех)

if (fd == -1) {

perror("Ошибка open");

exit(1);

}

ftruncate(fd, sizeof(SharedMemory));

SharedMemory \*shared = (NULL, sizeof(SharedMemory), PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

if (shared == MAP\_FAILED) {

perror("Ошибка mmap");

exit(1);

}

close(fd);

sem\_init(&shared->sem\_parent, 1, 0);

sem\_init(&shared->sem\_child, 1, 1);

pid\_t child = fork();

if (child == -1) {

perror("Ошибка fork");

exit(1);

}

if (child == 0) {

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

perror("Ошибка execl");

exit(1);

}

char line[BUFFER\_SIZE];

while (fgets(line, BUFFER\_SIZE, file)) {

sem\_wait(&shared->sem\_child);

strncpy(shared->buffer, line, BUFFER\_SIZE);

shared->data\_ready = 1;

sem\_post(&shared->sem\_parent);

sem\_wait(&shared->sem\_child);

printf("%s", shared->buffer);

sem\_post(&shared->sem\_parent);

}

sem\_wait(&shared->sem\_child);

shared->data\_ready = 0;

sem\_post(&shared->sem\_parent);

wait(NULL);

sem\_destroy(&shared->sem\_parent);

sem\_destroy(&shared->sem\_child);

munmap(shared, sizeof(SharedMemory));

remove(SHARED\_FILENAME);

fclose(file);

return 0;

}

**child.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include "shared.h"

long long get\_line\_sum(char\* line) {

long long line\_sum = 0;

int number = 0;

int digit\_power = 1;

int i = strlen(line);

while (i >= 0) {

if ((line[i] >= '0') && (line[i] <= '9')) {

number += (line[i] - '0') \* digit\_power;

digit\_power \*= 10;

} else {

if (line[i] == '-') {

number \*= -1;

}

line\_sum += number;

number = 0;

digit\_power = 1;

}

i--;

}

if (number != 0) {

line\_sum += number;

}

return line\_sum;

}

int main() {

int fd = open(SHARED\_FILENAME, O\_RDWR, 0666); // 0666 — права доступа (чтение и запись для всех)

if (fd == -1) {

perror("Ошибка open");

exit(1);

}

SharedMemory \*shm = mmap(NULL, sizeof(SharedMemory), PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

if (shm == MAP\_FAILED) {

perror("Ошибка mmap");

exit(1);

}

close(fd);

while (1) {

sem\_wait(&shm->sem\_parent);

if (!shm->data\_ready) break;

long long line\_sum = get\_line\_sum(shm->buffer);

// Обрезаем строку, если она слишком длинная

size\_t len = strnlen(shm->buffer, BUFFER\_SIZE);

if (len > BUFFER\_SIZE - 32) len = BUFFER\_SIZE - 32; // Оставляем место под " -> %lld\n"

char answer[BUFFER\_SIZE];

snprintf(answer, BUFFER\_SIZE, "%.\*s -> %lld\n", (int)len, shm->buffer, line\_sum);

strncpy(shm->buffer, answer, BUFFER\_SIZE);

shm->buffer[BUFFER\_SIZE - 1] = '\0'; // Гарантируем null-терминированную строку

sem\_post(&shm->sem\_child);

}

munmap(shm, sizeof(SharedMemory));

return 0;

}

# Протокол работы программы

## Тестирование:

./parent

Введите путь к файлу:

file.txt

17 21 67

-> 105

12 50 0

-> 62

7 -21 314

-> 300

-201 77

-> -124

## Strace:

strace -f ./parent

execve("./parent", ["./parent"], 0x7ffcfc9b3528 /\* 77 vars \*/) = 0

brk(NULL) = 0x56496f582000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffd30cfc100) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9b51240000

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=167639, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 167639, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f9b51217000

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\315A\vq\17\17\tLh2\355\331Y1\0m"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, 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, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f9b50fee000

mprotect(0x7f9b51016000, 2023424, PROT\_NONE) = 0

mmap(0x7f9b51016000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f9b51016000

mmap(0x7f9b511ab000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f9b511ab000

mmap(0x7f9b51204000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7f9b51204000

mmap(0x7f9b5120a000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f9b5120a000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9b50feb000

arch\_prctl(ARCH\_SET\_FS, 0x7f9b50feb740) = 0

set\_tid\_address(0x7f9b50feba10) = 707844

set\_robust\_list(0x7f9b50feba20, 24) = 0

rseq(0x7f9b50fec0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f9b51204000, 16384, PROT\_READ) = 0

mprotect(0x56494c338000, 4096, PROT\_READ) = 0

mprotect(0x7f9b5127a000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f9b51217000, 167639) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x7), ...}, AT\_EMPTY\_PATH) = 0

getrandom("\xf3\xb4\x2b\x89\xdf\xcf\x43\xe0", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x56496f582000

brk(0x56496f5a3000) = 0x56496f5a3000

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\277\321\203\321\202\321\214 \320\272 \321\204\320\260\320"..., 39Введите путь к файлу:

) = 39

newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x7), ...}, AT\_EMPTY\_PATH) = 0

read(0, 0x56496f5826b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, 0x56496f5826b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, 0x56496f5826b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, 0x56496f5826b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, 0x56496f5826b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, 0x56496f5826b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, 0x56496f5826b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, file.txt

"file.txt\n", 1024) = 9

openat(AT\_FDCWD, "file.txt", O\_RDONLY) = 3

openat(AT\_FDCWD, "shared\_memory.dat", O\_RDWR|O\_CREAT, 0666) = 4

ftruncate(4, 4168) = 0

mmap(NULL, 4168, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7f9b5123e000

close(4) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 708166 attached

, child\_tidptr=0x7f9b50feba10) = 708166

[pid 708166] set\_robust\_list(0x7f9b50feba20, 24) = 0

[pid 707844] newfstatat(3, "", <unfinished ...>

[pid 708166] execve("./child", ["child"], 0x7ffd30cfc2d8 /\* 77 vars \*/ <unfinished ...>

[pid 707844] <... newfstatat resumed>{st\_mode=S\_IFREG|0664, st\_size=35, ...}, AT\_EMPTY\_PATH) = 0

[pid 707844] read(3, "17 21 67\n12 50 0\n7 -21 314\n-201 "..., 4096) = 35

[pid 707844] futex(0x7f9b5123f028, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 708166] <... execve resumed>) = 0

[pid 708166] brk(NULL) = 0x556cf4bbd000

[pid 708166] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffeca628be0) = -1 EINVAL (Invalid argument)

[pid 708166] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff93aba2000

[pid 708166] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 708166] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 4

[pid 708166] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=167639, ...}, AT\_EMPTY\_PATH) = 0

[pid 708166] mmap(NULL, 167639, PROT\_READ, MAP\_PRIVATE, 4, 0) = 0x7ff93ab79000

[pid 708166] close(4) = 0

[pid 708166] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 4

[pid 708166] read(4, "\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 708166] pread64(4, "\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 708166] pread64(4, "\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 708166] pread64(4, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\315A\vq\17\17\tLh2\355\331Y1\0m"..., 68, 896) = 68

[pid 708166] newfstatat(4, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, AT\_EMPTY\_PATH) = 0

[pid 708166] pread64(4, "\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 708166] mmap(NULL, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7ff93a950000

[pid 708166] mprotect(0x7ff93a978000, 2023424, PROT\_NONE) = 0

[pid 708166] mmap(0x7ff93a978000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x28000) = 0x7ff93a978000

[pid 708166] mmap(0x7ff93ab0d000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1bd000) = 0x7ff93ab0d000

[pid 708166] mmap(0x7ff93ab66000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x215000) = 0x7ff93ab66000

[pid 708166] mmap(0x7ff93ab6c000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7ff93ab6c000

[pid 708166] close(4) = 0

[pid 708166] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff93a94d000

[pid 708166] arch\_prctl(ARCH\_SET\_FS, 0x7ff93a94d740) = 0

[pid 708166] set\_tid\_address(0x7ff93a94da10) = 708166

[pid 708166] set\_robust\_list(0x7ff93a94da20, 24) = 0

[pid 708166] rseq(0x7ff93a94e0e0, 0x20, 0, 0x53053053) = 0

[pid 708166] mprotect(0x7ff93ab66000, 16384, PROT\_READ) = 0

[pid 708166] mprotect(0x556ccdfc9000, 4096, PROT\_READ) = 0

[pid 708166] mprotect(0x7ff93abdc000, 8192, PROT\_READ) = 0

[pid 708166] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 708166] munmap(0x7ff93ab79000, 167639) = 0

[pid 708166] openat(AT\_FDCWD, "shared\_memory.dat", O\_RDWR) = 4

[pid 708166] mmap(NULL, 4168, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7ff93aba0000

[pid 708166] close(4) = 0

[pid 708166] futex(0x7ff93aba1028, FUTEX\_WAKE, 1) = 1

[pid 707844] <... futex resumed>) = 0

[pid 708166] futex(0x7ff93aba1008, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 707844] write(1, "17 21 67\n -> 105\n", 1717 21 67

-> 105

) = 17

[pid 707844] futex(0x7f9b5123f008, FUTEX\_WAKE, 1) = 1

[pid 708166] <... futex resumed>) = 0

[pid 707844] futex(0x7f9b5123f028, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 708166] futex(0x7ff93aba1028, FUTEX\_WAKE, 1 <unfinished ...>

[pid 707844] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 708166] <... futex resumed>) = 0

[pid 707844] futex(0x7f9b5123f028, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 708166] futex(0x7ff93aba1028, FUTEX\_WAKE, 1 <unfinished ...>

[pid 707844] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 708166] <... futex resumed>) = 0

[pid 707844] write(1, "12 50 0\n -> 62\n", 15 <unfinished ...>

[pid 708166] futex(0x7ff93aba1008, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY12 50 0

-> 62

<unfinished ...>

[pid 707844] <... write resumed>) = 15

[pid 707844] futex(0x7f9b5123f008, FUTEX\_WAKE, 1) = 1

[pid 708166] <... futex resumed>) = 0

[pid 707844] futex(0x7f9b5123f028, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 708166] futex(0x7ff93aba1028, FUTEX\_WAKE, 1 <unfinished ...>

[pid 707844] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 708166] <... futex resumed>) = 0

[pid 707844] futex(0x7f9b5123f028, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 708166] futex(0x7ff93aba1028, FUTEX\_WAKE, 1 <unfinished ...>

[pid 707844] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 708166] <... futex resumed>) = 0

[pid 707844] write(1, "7 -21 314\n -> 300\n", 18 <unfinished ...>

[pid 708166] futex(0x7ff93aba1008, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY7 -21 314

-> 300

<unfinished ...>

[pid 707844] <... write resumed>) = 18

[pid 707844] futex(0x7f9b5123f008, FUTEX\_WAKE, 1) = 1

[pid 708166] <... futex resumed>) = 0

[pid 707844] futex(0x7f9b5123f028, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 708166] futex(0x7ff93aba1028, FUTEX\_WAKE, 1 <unfinished ...>

[pid 707844] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 708166] <... futex resumed>) = 0

[pid 707844] futex(0x7f9b5123f028, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 708166] futex(0x7ff93aba1028, FUTEX\_WAKE, 1) = 0

[pid 708166] futex(0x7ff93aba1008, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 707844] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 707844] write(1, "-201 77\n -> -124\n", 17-201 77

-> -124

) = 17

[pid 707844] futex(0x7f9b5123f008, FUTEX\_WAKE, 1) = 1

[pid 707844] read(3, <unfinished ...>

[pid 708166] <... futex resumed>) = 0

[pid 707844] <... read resumed>"", 4096) = 0

[pid 707844] futex(0x7f9b5123f008, FUTEX\_WAKE, 1 <unfinished ...>

[pid 708166] futex(0x7ff93aba1008, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 707844] <... futex resumed>) = 0

[pid 707844] wait4(-1, <unfinished ...>

[pid 708166] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 708166] munmap(0x7ff93aba0000, 4168) = 0

[pid 708166] exit\_group(0) = ?

[pid 708166] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 708166

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=708166, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

munmap(0x7f9b5123e000, 4168) = 0

unlink("shared\_memory.dat") = 0

close(3) = 0

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Лабораторная работа демонстрирует использование разделяемой памяти (shared memory) и семафоров для взаимодействия между процессами в Linux. Программа корректно обрабатывает ввод пользователя, синхронизирует работу процессов и завершается без утечек ресурсов.