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: Закарейшвили Г. М.

Преподаватель: Миронов Е.С. (ПМИ)

Оценка: _____

Дата: 05.03.24

Москва, 2024

Постановка задачи

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).
11 вариант) Child1 переводит строки в верхний регистр. Child2 превращает все пробельные символы в символ «_».

Общий метод и алгоритм решения

Использованные системные вызовы:

Родительский процесс (parent.c):

● Системные вызовы:

○ fork(): Создает дочерние процессы.

○ waitpid(): Ожидает завершения дочерних процессов.

mmap(): Отображает файл shared memory в память.

○ munmap(): Освобождает отображенную память.

○ open(): Открывает файл shared memory.

○ close(): Закрывает файловый дескриптор.

○ ftruncate(): Устанавливает размер файла shared memory.

○ sem_init(): Инициализирует семафоры.

○ sem_destroy(): Уничтожает семафоры.

○ sem_post(): Увеличивает значение семафора.

○ sem_wait(): Уменьшает значение семафора.

Дочерний процесс 1 (child1.c):

● Системные вызовы:

○ mmap(): Отображает файл shared memory в память.

○ munmap(): Освобождает отображенную память.

○ open(): Открывает файл shared memory.

○ close(): Закрывает файловый дескриптор.

○ sem_wait(): Ожидает, пока данные не будут готовы.

○ sem_post(): Сигнализирует о завершении обработки.

○ sem_trywait(): Проверяет, завершена ли работа.

Дочерний процесс 2 (child2.c):

● Системные вызовы:

○ mmap(): Отображает файл shared memory в память.

○ munmap(): Освобождает отображенную память.

○ open(): Открывает файл shared memory.

○ close(): Закрывает файловый дескриптор.

○ sem_wait(): Ожидает, пока данные не будут готовы.

○ sem_post(): Сигнализирует о завершении обработки.

○ sem_trywait(): Проверяет, завершена ли работа.

Алгоритм работы программы

1. Инициализация и ввод данных:
   - Ввод аргументов с командной строки. Максимальное количество потоков, которые будут использоваться для сортировки.
   - Ввод размера массива.
   - Выделение памяти для массива с помощью malloc.
   - Инициализация генератора случайных чисел с помощью функции srand и заполнение массива случайными числами с помощью функции rand.
2. Вывод исходного массива.
3. Измерение времени выполнения:
   - Записываем текущее время с помощью функции gettimeofday.
4. Многопоточная сортировка:
   - Вызов функции evenodd: создание потоков.
   - Сортировка в потоках. Их завершение pthread_exit.
   - После создания всех потоков программа ожидает их завершения с помощью функции pthread_join.

Работа программы: аналогична первой лабораторной, вместо pipe-ов используется MMF

Код программы

parent.c

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

#define MAX_LINE 1000
#define SHARED_FILE "shared_memory.bin"

typedef struct {
```

```
    char data[MAX_LINE];
    sem_t sem_data_ready;       // Семафор: данные готовы для обработки
    sem_t sem_data_processed_by_1;  // Семафор: данные обработаны
    sem_t sem_data_processed_by_2;  // Семафор: данные обработаны
} SharedData;

int main() {
    int fd;
    SharedData *shared;
    pid_t child1, child2;

    // Создаем файл для shared memory
    fd = open(SHARED_FILE, O_RDWR | O_CREAT | O_TRUNC, 0666);
    if (fd == -1) {
        perror("Ошибка открытия файла");
        exit(1);
    }

    // Устанавливаем размер файла
    if (ftruncate(fd, sizeof(SharedData)) == -1) {
        perror("Ошибка ftruncate");
        close(fd);
        exit(1);
    }

    // Отображаем файл в память
    shared = (SharedData *)mmap(NULL, sizeof(SharedData), PROT_READ |
     PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared == MAP_FAILED) {
        perror("Ошибка mmap");
        close(fd);
        exit(1);
    }

    close(fd);  // Файловый дескриптор больше не нужен

    // Инициализация семафоров
    sem_init(&shared->sem_data_ready, 1, 0);       // Изначально данные не
     готовы
    sem_init(&shared->sem_data_processed_by_1, 1, 0);  // Изначально данные не
     обработаны
    sem_init(&shared->sem_data_processed_by_2, 1, 0);  // Изначально данные не
     обработаны

    // Создаем первый дочерний процесс
    if ((child1 = fork()) == -1) {
        perror("Ошибка создания первого дочернего процесса");
```

```
        munmap(shared, sizeof(SharedData));
        exit(1);
    }

    if (child1 == 0) {
        // Код для первого дочернего процесса (child1)
        execl("./child1", "child1", SHARED_FILE, NULL);
        perror("Ошибка execl для child1");
        exit(1);
    }

    // Создаем второй дочерний процесс
    if ((child2 = fork()) == -1) {
        perror("Ошибка создания второго дочернего процесса");
        munmap(shared, sizeof(SharedData));
        exit(1);
    }

    if (child2 == 0) {
        // Код для второго дочернего процесса (child2)
        execl("./child2", "child2", SHARED_FILE, NULL);
        perror("Ошибка execl для child2");
        exit(1);
    }

    // Родительский процесс
    printf("Введите строки (Ctrl+D для завершения):\n");

    while (fgets(shared->data, MAX_LINE, stdin) != NULL) {
        // Проверяем что не отправляем пустую строку
        if (shared->data[0] != '\0') {
            // Сигнализируем, что данные готовы для обработки
            sem_post(&shared->sem_data_ready);

            // Ждем, пока данные будут обработаны
            sem_wait(&shared->sem_data_processed_by_2);
        }

        // Выводим результат
        printf("\n");
        printf("Результат: %s\n", shared->data);
    }

    // Сигнализируем дочерним процессам о завершении (отправляем пустую строку)
    shared->data[0] = '\0';
    sem_post(&shared->sem_data_ready);
    sem_wait(&shared->sem_data_processed_by_2);
```

```
    // Ожидание завершения дочерних процессов
    int status;
    waitpid(child1, &status, 0); // Блокируется до завершения child1
    if (WIFEXITED(status) && WEXITSTATUS(status) != 0) { //завершен через
     exit() или main() / код завершения доч проц
        printf("Дочерний процесс 1 завершился с ошибкой: %i",
     WEXITSTATUS(status));
    }
    waitpid(child2, &status, 0); // Ждем завершения child2
    if (WIFEXITED(status) && WEXITSTATUS(status) != 0) {
        printf("Дочерний процесс 2 завершился с ошибкой: %i",
     WEXITSTATUS(status));
    }

    // Уничтожение семафоров
    sem_destroy(&shared->sem_data_ready);
    sem_destroy(&shared->sem_data_processed_by_1);
    sem_destroy(&shared->sem_data_processed_by_2);

    // Освобождаем shared memory
    munmap(shared, sizeof(SharedData));

    printf("\nВсе процессы завершены.\n");

    return 0;
}
```

clild1.c

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

#define MAX_LINE 1000
#define SHARED_FILE "shared_memory.bin"

typedef struct {
    char data[MAX_LINE];
    sem_t sem_data_ready;      // Семафор: данные готовы для обработки
```

```c
    sem_t sem_data_processed_by_1;  // Семафор: данные обработаны
    sem_t sem_data_processed_by_2;  // Семафор: данные обработаны
} SharedData;

int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Не указан файл shared memory\n");
        exit(1);
    }

    const char *shared_file = argv[1];
    int fd;
    SharedData *shared;

    // Открываем файл shared memory
    fd = open(shared_file, O_RDWR);
    if (fd == -1) {
        perror("Ошибка открытия файла");
        exit(1);
    }

    // Отображаем файл в память
    shared = (SharedData *)mmap(NULL, sizeof(SharedData), PROT_READ |
     PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared == MAP_FAILED) {
        perror("Ошибка mmap");
        close(fd);
        exit(1);
    }

    close(fd);  // Файловый дескриптор больше не нужен

    while (1) {
        // Ждем, пока данные будут готовы для обработки
        sem_wait(&shared->sem_data_ready);

        // Проверяем, завершена ли работа
        if (shared->data[0] == '\0') {
            sem_post(&shared->sem_data_processed_by_1);
            break;
        }

        // Преобразуем строку в верхний регистsp
        for (int i = 0; shared->data[i]; i++) {
            shared->data[i] = toupper(shared->data[i]);
        }
```

```
        // Сигнализируем, что данные обработаны
        sem_post(&shared->sem_data_processed_by_1);
    }

    // Освобождаем shared memory
    munmap(shared, sizeof(SharedData));

    return 0;
}
```

child2.c

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

#define MAX_LINE 1000
#define SHARED_FILE "shared_memory.bin"

typedef struct {
    char data[MAX_LINE];
    sem_t sem_data_ready;       // Семафор: данные готовы для обработки
    sem_t sem_data_processed_by_1;  // Семафор: данные обработаны
    sem_t sem_data_processed_by_2;  // Семафор: данные обработаны
} SharedData;

void remove_extra_spaces(char *str) {
    int i = 0;

    while (str[i]) {
        if (str[i] == ' ') {
            str[i] = '_';
        }
        i++;
    }
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Не указан файл shared memory\n");
        exit(1);
    }
```

```
    const char *shared_file = argv[1];
    int fd;
    SharedData *shared;

    // Открываем файл shared memory
    fd = open(shared_file, O_RDWR);
    if (fd == -1) {
        perror("Ошибка открытия файла");
        exit(1);
    }

    // Отображаем файл в память
    shared = (SharedData *)mmap(NULL, sizeof(SharedData), PROT_READ |
     PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared == MAP_FAILED) {
        perror("Ошибка mmap");
        close(fd);
        exit(1);
    }

    close(fd);  // Файловый дескриптор больше не нужен

    while (1) {
        // Ждем, пока данные будут готовы для обработки
        sem_wait(&shared->sem_data_processed_by_1);

        // Проверяем, завершена ли работа
        if (shared->data[0] == '\0') {
            sem_post(&shared->sem_data_processed_by_2);
            break;
        }

        // Удаляем лишние пробелы
        remove_extra_spaces(shared->data);

        // Сигнализируем, что данные обработаны
        sem_post(&shared->sem_data_processed_by_2);
    }

    // Освобождаем shared memory
    munmap(shared, sizeof(SharedData));

    return 0;
}
```

Протокол работы программы

root@LAPTOP-CGCBKBHR:/home/OS/OS_labs/lab3/src# ./parent
Введите строки (Ctrl+D для завершения):
text privet fedor and vladimir
Результат: TEXT_PRIVET_FEDOR_AND_VLADIMIR
Все процессы завершены.

Strace:

```
root@LAPTOP-CGCBKBHR:/home/OS/OS_labs/lab3/src# strace -f ./parent
execve("./parent", ["./parent"], 0x7fffe01bb868 /* 19 vars */) = 0
brk(NULL)                               = 0x7fffbaf66000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffc3554ce0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fba783c0000
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=16555, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fba783cb000
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\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 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) = 0x7fba78190000
mprotect(0x7fba781b8000, 2023424, PROT_NONE) = 0
mmap(0x7fba781b8000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
0x7fba781b8000
mmap(0x7fba7834d000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7fba7834d000
mmap(0x7fba783a6000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) =
0x7fba783a6000
mmap(0x7fba783ac000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fba783ac000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fba78180000
arch_prctl(ARCH_SET_FS, 0x7fba78180740) = 0
set_tid_address(0x7fba78180a10)         = 917
set_robust_list(0x7fba78180a20, 24)     = 0
rseq(0x7fba781810e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)
mprotect(0x7fba783a6000, 16384, PROT_READ) = 0
mprotect(0x7fba78411000, 4096, PROT_READ) = 0
mprotect(0x7fba78408000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
munmap(0x7fba783cb000, 16555)           = 0
openat(AT_FDCWD, "shared_memory.bin", O_RDWR|O_CREAT|O_TRUNC, 0666) = 3
ftruncate(3, 1096)                      = 0
mmap(NULL, 1096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fba7840d000
close(3)                                = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 918 attached
, child_tidptr=0x7fba78180a10) = 918
[pid   918] set_robust_list(0x7fba78180a20, 24 <unfinished ...>
[pid   917] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid   918] <... set_robust_list resumed>) = 0
[pid   918] execve("./child1", ["child1", "shared_memory.bin"], 0x7fffc3554eb8 /* 19 vars */strace: Process 919
attached
 <unfinished ...>
[pid   917] <... clone resumed>, child_tidptr=0x7fba78180a10) = 919
[pid   919] set_robust_list(0x7fba78180a20, 24 <unfinished ...>
[pid   917] newfstatat(1, "",  <unfinished ...>
[pid   919] <... set_robust_list resumed>) = 0
[pid   917] <... newfstatat resumed>{st_mode=S_IFCHR|0640, st_rdev=makedev(0x4, 0x5), ...}, AT_EMPTY_PATH) = 0
[pid   919] execve("./child2", ["child2", "shared_memory.bin"], 0x7fffc3554eb8 /* 19 vars */ <unfinished ...>
[pid   917] ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
```

```
[pid   918] <... execve resumed>)      = 0
[pid   917] getrandom( <unfinished ...>
[pid   918] brk(NULL <unfinished ...>
[pid   917] <... getrandom resumed>"\x40\x82\x2b\x1b\x9b\xf6\x04\x78", 8, GRND_NONBLOCK) = 8
[pid   918] <... brk resumed>)         = 0x7fffc844e000
[pid   917] brk(NULL <unfinished ...>
[pid   918] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffd010f160 <unfinished ...>
[pid   917] <... brk resumed>)         = 0x7fffbaf66000
[pid   918] <... arch_prctl resumed>)  = -1 EINVAL (Invalid argument)
[pid   917] brk(0x7fffbaf87000 <unfinished ...>
[pid   918] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   917] <... brk resumed>)         = 0x7fffbaf87000
[pid   918] <... mmap resumed>)        = 0x7f8b09fa0000
[pid   917] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\320\270 (Ctr"..., 66Введите строки (Ctrl+D для завершения):
 <unfinished ...>
[pid   918] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid   917] <... write resumed>)       = 66
[pid   918] <... access resumed>)      = -1 ENOENT (No such file or directory)
[pid   917] newfstatat(0, "",  <unfinished ...>
[pid   918] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   917] <... newfstatat resumed>{st_mode=S_IFCHR|0640, st_rdev=makedev(0x4, 0x5), ...}, AT_EMPTY_PATH) = 0
[pid   918] <... openat resumed>)      = 3
[pid   917] ioctl(0, TCGETS <unfinished ...>
[pid   918] newfstatat(3, "",  <unfinished ...>
[pid   917] <... ioctl resumed>, {B38400 opost isig icanon echo ...}) = 0
[pid   918] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=16555, ...}, AT_EMPTY_PATH) = 0
[pid   917] read(0,  <unfinished ...>
[pid   918] mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8b09fab000
[pid   918] close(3)                   = 0
[pid   919] <... execve resumed>)      = 0
[pid   918] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   919] brk(NULL <unfinished ...>
[pid   918] <... openat resumed>)      = 3
[pid   919] <... brk resumed>)         = 0x7fffd8fba000
[pid   918] read(3,  <unfinished ...>
[pid   919] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffe15005a0 <unfinished ...>
[pid   918] <... 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"..., 832) = 832
[pid   919] <... arch_prctl resumed>)  = -1 EINVAL (Invalid argument)
[pid   918] pread64(3,  <unfinished ...>
[pid   919] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   918] <... 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   919] <... mmap resumed>)        = 0x7f81c0bc0000
[pid   918] pread64(3,  <unfinished ...>
[pid   919] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid   918] <... pread64 resumed>"\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   919] <... access resumed>)      = -1 ENOENT (No such file or directory)
[pid   918] pread64(3,  <unfinished ...>
[pid   919] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   918] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68,
896) = 68
[pid   919] <... openat resumed>)      = 3
[pid   918] newfstatat(3, "",  <unfinished ...>
[pid   919] newfstatat(3, "",  <unfinished ...>
[pid   918] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid   919] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=16555, ...}, AT_EMPTY_PATH) = 0
[pid   918] pread64(3,  <unfinished ...>
[pid   919] mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid   918] <... 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   919] <... mmap resumed>)        = 0x7f81c0bcb000
[pid   918] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid   919] close(3 <unfinished ...>
[pid   918] <... mmap resumed>)        = 0x7f8b09d70000
[pid   919] <... close resumed>)       = 0
[pid   918] mprotect(0x7f8b09d98000, 2023424, PROT_NONE <unfinished ...>
[pid   919] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   918] <... mprotect resumed>)    = 0
[pid   919] <... openat resumed>)      = 3
[pid   918] mmap(0x7f8b09d98000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000 <unfinished ...>
[pid   919] read(3,  <unfinished ...>
```

```
[pid   918] <... mmap resumed>)         = 0x7f8b09d98000
[pid   919] <... 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"..., 832) = 832
[pid   918] mmap(0x7f8b09f2d000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000
<unfinished ...>
[pid   919] pread64(3,  <unfinished ...>
[pid   918] <... mmap resumed>)         = 0x7f8b09f2d000
[pid   919] <... 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   918] mmap(0x7f8b09f86000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000 <unfinished ...>
[pid   919] pread64(3,  <unfinished ...>
[pid   918] <... mmap resumed>)         = 0x7f8b09f86000
[pid   919] <... pread64 resumed>"\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   918] mmap(0x7f8b09f8c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid   919] pread64(3,  <unfinished ...>
[pid   918] <... mmap resumed>)         = 0x7f8b09f8c000
[pid   919] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68,
896) = 68
[pid   918] close(3 <unfinished ...>
[pid   919] newfstatat(3, "",  <unfinished ...>
[pid   918] <... close resumed>)        = 0
[pid   919] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid   918] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   919] pread64(3,  <unfinished ...>
[pid   918] <... mmap resumed>)         = 0x7f8b09d60000
[pid   919] <... 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   918] arch_prctl(ARCH_SET_FS, 0x7f8b09d60740 <unfinished ...>
[pid   919] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid   918] <... arch_prctl resumed>)   = 0
[pid   919] <... mmap resumed>)         = 0x7f81c0990000
[pid   918] set_tid_address(0x7f8b09d60a10 <unfinished ...>
[pid   919] mprotect(0x7f81c09b8000, 2023424, PROT_NONE <unfinished ...>
[pid   918] <... set_tid_address resumed>) = 918
[pid   919] <... mprotect resumed>)     = 0
[pid   918] set_robust_list(0x7f8b09d60a20, 24 <unfinished ...>
[pid   919] mmap(0x7f81c09b8000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000 <unfinished ...>
[pid   918] <... set_robust_list resumed>) = 0
[pid   919] <... mmap resumed>)         = 0x7f81c09b8000
[pid   918] rseq(0x7f8b09d610e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid   919] mmap(0x7f81c0b4d000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000
<unfinished ...>
[pid   918] <... rseq resumed>)         = -1 ENOSYS (Function not implemented)
[pid   919] <... mmap resumed>)         = 0x7f81c0b4d000
[pid   918] mprotect(0x7f8b09f86000, 16384, PROT_READ <unfinished ...>
[pid   919] mmap(0x7f81c0ba6000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000 <unfinished ...>
[pid   918] <... mprotect resumed>)     = 0
[pid   919] <... mmap resumed>)         = 0x7f81c0ba6000
[pid   918] mprotect(0x7f8b09fef000, 4096, PROT_READ <unfinished ...>
[pid   919] mmap(0x7f81c0bac000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid   918] <... mprotect resumed>)     = 0
[pid   919] <... mmap resumed>)         = 0x7f81c0bac000
[pid   918] mprotect(0x7f8b09fe8000, 8192, PROT_READ <unfinished ...>
[pid   919] close(3 <unfinished ...>
[pid   918] <... mprotect resumed>)     = 0
[pid   919] <... close resumed>)        = 0
[pid   918] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid   919] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   918] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
[pid   919] <... mmap resumed>)         = 0x7f81c0980000
[pid   918] munmap(0x7f8b09fab000, 16555 <unfinished ...>
[pid   919] arch_prctl(ARCH_SET_FS, 0x7f81c0980740 <unfinished ...>
[pid   918] <... munmap resumed>)       = 0
[pid   919] <... arch_prctl resumed>)   = 0
[pid   918] openat(AT_FDCWD, "shared_memory.bin", O_RDWR <unfinished ...>
[pid   919] set_tid_address(0x7f81c0980a10) = 919
[pid   919] set_robust_list(0x7f81c0980a20, 24) = 0
[pid   919] rseq(0x7f81c09810e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)
[pid   919] mprotect(0x7f81c0ba6000, 16384, PROT_READ <unfinished ...>
```

```
[pid   918] <... openat resumed>)       = 3
[pid   919] <... mprotect resumed>)     = 0
[pid   918] mmap(NULL, 1096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>
[pid   919] mprotect(0x7f81c0c13000, 4096, PROT_READ <unfinished ...>
[pid   918] <... mmap resumed>)         = 0x7f8b09fe7000
[pid   919] <... mprotect resumed>)     = 0
[pid   918] close(3 <unfinished ...>
[pid   919] mprotect(0x7f81c0c08000, 8192, PROT_READ <unfinished ...>
[pid   918] <... close resumed>)        = 0
[pid   919] <... mprotect resumed>)     = 0
[pid   918] futex(0x7f8b09fe73e8, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY
<unfinished ...>
[pid   919] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
[pid   919] munmap(0x7f81c0bcb000, 16555) = 0
[pid   919] openat(AT_FDCWD, "shared_memory.bin", O_RDWR) = 3
[pid   919] mmap(NULL, 1096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f81c0c0f000
[pid   919] close(3)                    = 0
[pid   919] futex(0x7f81c0c0f408, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANYtext text <unfinished ...>
[pid   917] <... read resumed>"text text", 4096) = 9
[pid   917] read(0, text  pr"text  pr", 4096)   = 8
[pid   917] read(0, "", 4096)           = 0
[pid   917] futex(0x7fba7840d3e8, FUTEX_WAKE, 1) = 1
[pid   918] <... futex resumed>)        = 0
[pid   917] futex(0x7fba7840d428, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY
<unfinished ...>
[pid   918] futex(0x7f8b09fe7408, FUTEX_WAKE, 1) = 1
[pid   919] <... futex resumed>)        = 0
[pid   918] futex(0x7f8b09fe73e8, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY
<unfinished ...>
[pid   919] futex(0x7f81c0c0f428, FUTEX_WAKE, 1 <unfinished ...>
[pid   917] <... futex resumed>)        = 0
[pid   919] <... futex resumed>)        = 1
[pid   917] write(1, "\n", 1
 <unfinished ...>
[pid   919] futex(0x7f81c0c0f408, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY
<unfinished ...>
[pid   917] <... write resumed>)        = 1
[pid   917] write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202: TEXT_TEXTTEX"...,
38Результат: TEXT_TEXTTEXT__PR
) = 38
[pid   917] futex(0x7fba7840d3e8, FUTEX_WAKE, 1) = 1
[pid   918] <... futex resumed>)        = 0
[pid   917] futex(0x7fba7840d428, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY
<unfinished ...>
[pid   918] futex(0x7f8b09fe7408, FUTEX_WAKE, 1) = 1
[pid   919] <... futex resumed>)        = 0
[pid   918] munmap(0x7f8b09fe7000, 1096 <unfinished ...>
[pid   919] futex(0x7f81c0c0f428, FUTEX_WAKE, 1 <unfinished ...>
[pid   918] <... munmap resumed>)       = 0
[pid   917] <... futex resumed>)        = 0
[pid   919] <... futex resumed>)        = 1
[pid   917] wait4(918,  <unfinished ...>
[pid   918] exit_group(0 <unfinished ...>
[pid   919] munmap(0x7f81c0c0f000, 1096 <unfinished ...>
[pid   918] <... exit_group resumed>)   = ?
[pid   919] <... munmap resumed>)       = 0
[pid   917] <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 918
[pid   918] +++ exited with 0 +++
[pid   917] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=918, si_uid=0, si_status=0, si_utime=0, si_stime=0}
---
[pid   919] exit_group(0 <unfinished ...>
[pid   917] wait4(919,  <unfinished ...>
[pid   919] <... exit_group resumed>)   = ?
[pid   919] +++ exited with 0 +++
<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 919
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=919, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---
munmap(0x7fba7840d000, 1096)            = 0
write(1, "\n", 1
)                       = 1
write(1, "\320\222\321\201\320\265 \320\277\321\200\320\276\321\206\320\265\321\201\321\201\321\213
\320\267\320\260\320\262\320\265"..., 44Все процессы завершены.
) = 44
```

```
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