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-214БВ-25

Студент: Лоскутова А. Д.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 12.11.25

Москва, 2025

Постановка задачи

Вариант 17.

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы удаляют все гласные из строк

Общий метод и алгоритм решения

В лабораторной работе реализовано взаимодействие между процессами с использованием разделяемой памяти (shared memory) и именованных семафоров. Родительский процесс принимает строки с клавиатуры и, в зависимости от их длины, передаёт их одному из двух дочерних процессов через shared memory:

- если длина строки ≤ 10 символов — первому дочернему процессу,
- иначе — второму.

Дочерние процессы читают данные из общей памяти, удаляют из строк все гласные буквы и выводят результат в свои файлы.

Для синхронизации используется несколько семафоров: один — для управления доступом к общей памяти, два других — для уведомления дочерних процессов о готовности данных.
Родитель и дочерние процессы работают последовательно через блокирующие операции sem_wait() и sem_post(), что исключает одновременный доступ к памяти и обеспечивает корректный обмен данными.

Использованные системные вызовы:

● pid_t fork(void); – создает дочерний процесс.

● **int execv(const char *path, char const argv[]); – заменяет текущий процесс на новый, передавая параметры запуска.

● **int shm_open(const char name, int oflag, mode_t mode); – создает или открывает именованный объект разделяемой памяти.

● int ftruncate(int fd, off_t length); – задает размер объекта shared memory.

● **void *mmap(void addr, size_t length, int prot, int flags, int fd, off_t offset); – отображает объект shared memory в адресное пространство процесса.

● **sem_t *sem_open(const char name, int oflag, mode_t mode, unsigned int value); – создает или открывает именованный семафор.

● **int sem_wait(sem_t sem); – операция ожидания (уменьшение значения семафора, блокирует процесс при 0).

● **int sem_post(sem_t sem); – операция освобождения (увеличивает значение семафора).

● **pid_t waitpid(pid_t pid, int status, int options); – ожидает завершения дочернего процесса.

Код программы

Server.c

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <stdio.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <stdint.h>
#include <time.h>

#define SHM_SIZE 8192
#define BUFFER_SIZE 4096
#define MAX_LINE_LEN 4096

struct shared_data {
    uint32_t length1;
    char data1[BUFFER_SIZE];
    uint32_t length2;
    char data2[BUFFER_SIZE];
};

ssize_t read_line(int fd, char *buffer, size_t max_len) {
    ssize_t total_read = 0;
    char c;
```

```
    while (total_read < max_len - 1 && read(fd, &c, 1) == 1) {
        if (c == '\n') {
            break;
        }
        buffer[total_read++] = c;
    }
    if (total_read == 0) {
        return -1;
    }
    buffer[total_read] = '\0';
    return total_read;
}

int main() {
    char shm_name[64];
    char sem_access_name[64];
    char sem_avail1_name[64];
    char sem_avail2_name[64];

    int pid = getpid();
    snprintf(shm_name, sizeof(shm_name), "/lab_shm_%d", pid);
    snprintf(sem_access_name, sizeof(sem_access_name), "/lab_access_%d", pid);
    snprintf(sem_avail1_name, sizeof(sem_avail1_name), "/lab_avail1_%d", pid);
    snprintf(sem_avail2_name, sizeof(sem_avail2_name), "/lab_avail2_%d", pid);

    int shm_fd = shm_open(shm_name, O_RDWR | O_CREAT | O_EXCL, 0666);
    if (shm_fd == -1) {
        perror("shm_open failed");
        exit(EXIT_FAILURE);
```

```
    }

    if (ftruncate(shm_fd, SHM_SIZE) == -1) {
        perror("ftruncate failed");
        shm_unlink(shm_name);
        close(shm_fd);
        exit(EXIT_FAILURE);
    }

    struct shared_data *shm_buf = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if (shm_buf == MAP_FAILED) {
        perror("mmap failed");
        shm_unlink(shm_name);
        close(shm_fd);
        exit(EXIT_FAILURE);
    }
    memset(shm_buf, 0, sizeof(struct shared_data));

    sem_t *sem_access = sem_open(sem_access_name, O_CREAT | O_EXCL, 0666, 1);
    sem_t *sem_avail1 = sem_open(sem_avail1_name, O_CREAT | O_EXCL, 0666, 0);
    sem_t *sem_avail2 = sem_open(sem_avail2_name, O_CREAT | O_EXCL, 0666, 0);

    if (sem_access == SEM_FAILED || sem_avail1 == SEM_FAILED || sem_avail2 == SEM_FAILED) {
        perror("sem_open failed");
        goto cleanup;
    }

    pid_t pid1 = fork();
    if (pid1 == -1) {
```

```
    perror("fork child1 failed");
    goto cleanup;
}

if (pid1 == 0) {
    char client_id_str[2] = "1";
    char *args[] = {"client", shm_name, sem_access_name, sem_avail1_name, client_id_str, NULL};
    execv("client", args);
    perror("execv child1 failed");
    exit(EXIT_FAILURE);
}

pid_t pid2 = fork();
if (pid2 == -1) {
    perror("fork child2 failed");
    goto cleanup;
}

if (pid2 == 0) {
    char client_id_str[2] = "2";
    char *args[] = {"client", shm_name, sem_access_name, sem_avail2_name, client_id_str, NULL};
    execv("client", args);
    perror("execv child2 failed");
    exit(EXIT_FAILURE);
}

sleep(1);

printf("Enter lines (empty line to finish):\n");
char line[MAX_LINE_LEN];
```

```
ssize_t len;

while ((len = read_line(STDIN_FILENO, line, sizeof(line))) != -1) {
    if (len == 0) {
        break;
    }

    if (sem_wait(sem_access) == -1) {
        perror("sem_wait access failed");
        break;
    }

    uint32_t *target_length;
    char *target_data;
    sem_t *target_sem_avail;

    if (len > 10) {
        target_length = &shm_buf->length2;
        target_data = shm_buf->data2;
        target_sem_avail = sem_avail2;
    } else {
        target_length = &shm_buf->length1;
        target_data = shm_buf->data1;
        target_sem_avail = sem_avail1;
    }

    memcpy(target_data, line, len);
    *target_length = len;

    if (sem_post(sem_access) == -1) {
```

```
            perror("sem_post access failed");
            break;
        }

        if (sem_post(target_sem_avail) == -1) {
            perror("sem_post avail failed");
            break;
        }
    }
    if (sem_wait(sem_access) == -1) {
        perror("sem_wait access failed");
    } else {
        shm_buf->length1 = UINT32_MAX;
        shm_buf->length2 = UINT32_MAX;
        sem_post(sem_access);
        sem_post(sem_avail1);
        sem_post(sem_avail2);
    }

    waitpid(pid1, NULL, 0);
    waitpid(pid2, NULL, 0);

    printf("Processing complete. Results saved to child1_output.txt and child2_output.txt\n");

cleanup:
    if (sem_access != SEM_FAILED) {
        sem_close(sem_access);
        sem_unlink(sem_access_name);
    }
    if (sem_avail1 != SEM_FAILED) {
```

```
        sem_close(sem_avail1);
        sem_unlink(sem_avail1_name);
    }
    if (sem_avail2 != SEM_FAILED) {
        sem_close(sem_avail2);
        sem_unlink(sem_avail2_name);
    }
    munmap(shm_buf, SHM_SIZE);
    shm_unlink(shm_name);
    close(shm_fd);

    return 0;
}
```

Client.c

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <stdio.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <stdint.h>

#define SHM_SIZE 8192
#define BUFFER_SIZE 4096

struct shared_data {
```

```
    uint32_t length1;
    char data1[BUFFER_SIZE];
    uint32_t length2;
    char data2[BUFFER_SIZE];
};

void remove_vowels(char *s) {
    char *src = s, *dst = s;
    while (*src) {
        char c = *src;
        if (!(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||
              c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U' ||
              c == 'y' || c == 'Y')) {
            *dst++ = c;
        }
        src++;
    }
    *dst = '\0';
}

int main(int argc, char **argv) {
    if (argc != 5) {
        const char msg[] = "Usage: client <shm_name> <sem_access_name> <sem_avail_name>
<client_id>\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        exit(EXIT_FAILURE);
    }

    const char *shm_name = argv[1];
    const char *sem_access_name = argv[2];
```

```
    const char *sem_avail_name = argv[3];
    int client_id = atoi(argv[4]);

    int shm_fd = shm_open(shm_name, O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("shm_open failed");
        exit(EXIT_FAILURE);
    }

    struct shared_data *shm_buf = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if (shm_buf == MAP_FAILED) {
        perror("mmap failed");
        close(shm_fd);
        exit(EXIT_FAILURE);
    }

    sem_t *sem_access = sem_open(sem_access_name, O_RDWR);
    sem_t *sem_avail = sem_open(sem_avail_name, O_RDWR);

    if (sem_access == SEM_FAILED || sem_avail == SEM_FAILED) {
        perror("sem_open failed");
        munmap(shm_buf, SHM_SIZE);
        close(shm_fd);
        exit(EXIT_FAILURE);
    }

    char filename[256];
    snprintf(filename, sizeof(filename), "child%d_output.txt", client_id);
    int fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
```

```
if (fd == -1) {
    perror("open file failed");
    sem_close(sem_access);
    sem_close(sem_avail);
    munmap(shm_buf, SHM_SIZE);
    close(shm_fd);
    exit(EXIT_FAILURE);
}
int running = 1;
while (running) {
    if (sem_wait(sem_avail) == -1) {
        perror("sem_wait avail failed");
        break;
    }

    if (sem_wait(sem_access) == -1) {
        perror("sem_wait access failed");
        break;
    }

    uint32_t *length = (client_id == 1) ? &shm_buf->length1 : &shm_buf->length2;
    char *data = (client_id == 1) ? shm_buf->data1 : shm_buf->data2;

    if (*length == UINT32_MAX) {
        running = 0;
    } else if (*length > 0) {
        char buffer[BUFFER_SIZE];
        memcpy(buffer, data, *length);
        buffer[*length] = '\0';
```

```
            remove_vowels(buffer);

            if (write(fd, buffer, strlen(buffer)) == -1 || write(fd, "\n", 1) == -1) {
                perror("write to file failed");
            }

            *length = 0;
        }

        if (sem_post(sem_access) == -1) {
            perror("sem_post access failed");
            break;
        }
    }

    close(fd);
    sem_close(sem_access);
    sem_close(sem_avail);
    munmap(shm_buf, SHM_SIZE);
    close(shm_fd);

    return 0;
}
```

Протокол работы программы

Тесты

```
kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3$ cd src
kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3/src$ gcc -o server server.c -lrt -lpthread
kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3/src$ gcc -o server server.c -lrt -lpthread
kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3/src$ ./server
Enter lines (empty line to finish):
abc def
auoiye
bcd
hehehe okokokok rararamza pupupu lalalalalalalala
roza
aaaaabbbbbbbccccccddddddddeeeeeeeeeffffffffkkkk

Processing complete. Results saved to child1_output.txt and child2_output.txt
kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3/src$ cat child1_output.txt
bc df

bcd
rz
kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3/src$ cat child2_output.txt
hhh kkkk rrrmz ppp llllllll
bbbbbbbccccccddddddddfffffffkkkk
kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3/src$
```

Strace

kriasatri@kriasa2006:/mnt/f/2_kurs/1_sem/os/lab_3/src$ strace -f ./server

execve("./server", ["./server"], 0x7ffc9e4e5388 /* 37 vars */) = 0

brk(NULL) = 0x59051e3f9000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x795873e92000

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=27643, ...}) = 0

mmap(NULL, 27643, PROT_READ, MAP_PRIVATE, 3, 0) = 0x795873e8b000

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\220\243\2\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

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\0\0\0"..., 784, 64) = 784

mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x795873c00000

mmap(0x795873c28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x795873c28000

mmap(0x795873db0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x795873db0000

mmap(0x795873dff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x795873dff000

mmap(0x795873e05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x795873e05000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x795873e88000

arch_prctl(ARCH_SET_FS, 0x795873e88740) = 0

set_tid_address(0x795873e88a10) = 49275

set_robust_list(0x795873e88a20, 24) = 0

rseq(0x795873e89060, 0x20, 0, 0x53053053) = 0

mprotect(0x795873dff000, 16384, PROT_READ) = 0

mprotect(0x5904e531c000, 4096, PROT_READ) = 0

mprotect(0x795873eca000, 8192, PROT_READ) = 0

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

munmap(0x795873e8b000, 27643) = 0

getpid() = 49275

openat(AT_FDCWD, "/dev/shm/lab_shm_49275", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

ftruncate(3, 8192) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x795873e90000

getrandom("\xb0\xf0\xed\x41\xf8\xf8\xa1\x0a", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.uwMQUB", 0x7fff51657fa0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.uwMQUB",
O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0666) = 4

write(4, "\1\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x795873e8f000

link("/dev/shm/sem.uwMQUB", "/dev/shm/sem.lab_access_49275") = 0

fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0

getrandom("\x02\x76\x7b\x98\x1f\x3b\x70\x9d", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x59051e3f9000

brk(0x59051e41a000) = 0x59051e41a000

unlink("/dev/shm/sem.uwMQUB") = 0

close(4) = 0

getrandom("\xd6\xd4\x02\xbe\x14\xa7\xe9\xf9", 8, GRND_NONBLOCK) = 8

getrandom("\xea\x44\x76\x8d\x6e\x45\x50\x4d", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.qPFuFf", 0x7fff51657fa0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.qPFuFf",
O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0666) = 4

write(4, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x795873e8e000

link("/dev/shm/sem.qPFuFf", "/dev/shm/sem.lab_avail1_49275") = 0

fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0

unlink("/dev/shm/sem.qPFuFf") = 0

close(4) = 0

getrandom("\xea\x34\xc9\xa7\x7b\x1a\xeb\x91", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.AgZGR7", 0x7fff51657fa0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.AgZGR7",
O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0666) = 4

write(4, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x795873e8d000

link("/dev/shm/sem.AgZGR7", "/dev/shm/sem.lab_avail2_49275") = 0

fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0

unlink("/dev/shm/sem.AgZGR7") = 0

close(4) = 0

clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 49276 attached

, child_tidptr=0x795873e88a10) = 49276

[pid 49276] set_robust_list(0x795873e88a20, 24 <unfinished ...>

[pid 49275] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>

[pid 49276] <... set_robust_list resumed>) = 0

[pid 49276] execve("client", ["client", "/lab_shm_49275", "/lab_access_49275", "/lab_avail1_49275", "1"], 0x7fff516595e8 /* 37 vars */strace: Process 49277 attached

<unfinished ...>

[pid 49275] **<... clone resumed>, child_tidptr=0x795873e88a10) = 49277**

[pid 49275] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, <unfinished ...>

[pid 49277] set_robust_list(0x795873e88a20, 24) = 0

[pid 49277] execve("client", ["client", "/lab_shm_49275", "/lab_access_49275", "/lab_avail2_49275", "2"], 0x7fff516595e8 /* 37 vars */) = 0

[pid 49276] <... execve resumed>) = 0

[pid 49277] brk(NULL <unfinished ...>

[pid 49276] brk(NULL <unfinished ...>

[pid 49277] <... brk resumed>) = 0x5b7c68d70000

[pid 49276] <... brk resumed>) = 0x5be646ad4000

[pid 49277] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 49276] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 49277] <... mmap resumed>) = 0x739ef7831000

[pid 49276] <... mmap resumed>) = 0x7b48ddca2000

[pid 49277] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 49276] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 49277] <... access resumed>) = -1 ENOENT (No such file or directory)

[pid 49276] <... access resumed>)   = -1 ENOENT (No such file or directory)

[pid 49277] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 49276] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 49277] <... openat resumed>)   = 3

[pid 49277] fstat(3, <unfinished ...>

[pid 49276] <... openat resumed>)   = 3

[pid 49277] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=27643, ...}) = 0

[pid 49276] fstat(3, <unfinished ...>

[pid 49277] mmap(NULL, 27643, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

[pid 49276] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=27643, ...}) = 0

[pid 49277] <... mmap resumed>)   = 0x739ef782a000

[pid 49276] mmap(NULL, 27643, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

[pid 49277] close(3 <unfinished ...>

[pid 49276] <... mmap resumed>)   = 0x7b48ddc9b000

[pid 49277] <... close resumed>)   = 0

[pid 49276] close(3 <unfinished ...>

[pid 49277] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>

[pid 49276] <... close resumed>)   = 0

[pid 49277] <... openat resumed>)   = 3

[pid 49276] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>

[pid 49277] read(3, <unfinished ...>

[pid 49276] <... openat resumed>)   = 3

[pid 49277] <... read resumed>"\177ELF\2\1\1\3\0\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 49277] 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 49276] read(3, <unfinished ...>

[pid 49277] fstat(3, <unfinished ...>

[pid 49276] <... read resumed>"\177ELF\2\1\1\3\0\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 49277] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

[pid 49276] pread64(3, <unfinished ...>

[pid 49277] pread64(3, <unfinished ...>

[pid 49276] <... 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 49277] <... 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 49276] fstat(3, <unfinished ...>

[pid 49277] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 49276] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

[pid 49277] <... mmap resumed>)   = 0x739ef7600000

[pid 49276] pread64(3, <unfinished ...>

[pid 49277] mmap(0x739ef7628000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

[pid 49276] <... 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 49277] <... mmap resumed>)   = 0x739ef7628000

[pid 49276] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 49277] mmap(0x739ef77b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000 <unfinished ...>

[pid 49276] <... mmap resumed>)   = 0x7b48dda00000

[pid 49276] mmap(0x7b48dda28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

[pid 49277] <... mmap resumed>)   = 0x739ef77b0000

[pid 49276] <... mmap resumed>)   = 0x7b48dda28000

[pid 49277] mmap(0x739ef77ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000 <unfinished ...>

[pid 49276] mmap(0x7b48ddbb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000 <unfinished ...>

[pid 49277] <... mmap resumed>)   = 0x739ef77ff000

[pid 49276] <... mmap resumed>)   = 0x7b48ddbb0000

[pid 49277] mmap(0x739ef7805000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 49276] mmap(0x7b48ddbff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000 <unfinished ...>

[pid 49277] <... mmap resumed>) = 0x739ef7805000

[pid 49276] <... mmap resumed>) = 0x7b48ddbff000

[pid 49277] close(3 <unfinished ...>

[pid 49276] mmap(0x7b48ddc05000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 49277] <... close resumed>) = 0

[pid 49276] <... mmap resumed>) = 0x7b48ddc05000

[pid 49277] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 49276] close(3 <unfinished ...>

[pid 49277] <... mmap resumed>) = 0x739ef7827000

[pid 49276] <... close resumed>) = 0

[pid 49277] arch_prctl(ARCH_SET_FS, 0x739ef7827740 <unfinished ...>

[pid 49276] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 49277] <... arch_prctl resumed>) = 0

[pid 49276] <... mmap resumed>) = 0x7b48ddc98000

[pid 49277] set_tid_address(0x739ef7827a10 <unfinished ...>

[pid 49276] arch_prctl(ARCH_SET_FS, 0x7b48ddc98740 <unfinished ...>

[pid 49277] <... set_tid_address resumed>) = 49277

[pid 49276] <... arch_prctl resumed>) = 0

[pid 49277] set_robust_list(0x739ef7827a20, 24 <unfinished ...>

[pid 49276] set_tid_address(0x7b48ddc98a10 <unfinished ...>

[pid 49277] <... set_robust_list resumed>) = 0

[pid 49276] <... set_tid_address resumed>) = 49276

[pid 49277] rseq(0x739ef7828060, 0x20, 0, 0x53053053 <unfinished ...>

[pid 49276] set_robust_list(0x7b48ddc98a20, 24 <unfinished ...>

[pid 49277] <... rseq resumed>) = 0

[pid 49276] <... set_robust_list resumed>) = 0

[pid 49276] rseq(0x7b48ddc99060, 0x20, 0, 0x53053053 <unfinished ...>

[pid 49277] mprotect(0x739ef77ff000, 16384, PROT_READ <unfinished ...>

[pid 49276] <... rseq resumed>) = 0

[pid 49277] <... mprotect resumed>) = 0

[pid 49277] mprotect(0x5b7c5db54000, 4096, PROT_READ <unfinished ...>

[pid 49276] mprotect(0x7b48ddbff000, 16384, PROT_READ <unfinished ...>

[pid 49277] <... mprotect resumed>) = 0

[pid 49276] <... mprotect resumed>) = 0

[pid 49277] mprotect(0x739ef7869000, 8192, PROT_READ <unfinished ...>

[pid 49276] mprotect(0x5be641c5c000, 4096, PROT_READ <unfinished ...>

[pid 49277] <... mprotect resumed>) = 0

[pid 49276] <... mprotect resumed>) = 0

[pid 49277] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>

[pid 49276] mprotect(0x7b48ddcda000, 8192, PROT_READ <unfinished ...>

[pid 49277] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 49276] <... mprotect resumed>) = 0

[pid 49277] munmap(0x739ef782a000, 27643) = 0

[pid 49276] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>

[pid 49277] **openat(AT_FDCWD, "/dev/shm/lab_shm_49275", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>**

[pid 49276] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 49277] <... openat resumed>) = 3

[pid 49276] munmap(0x7b48ddc9b000, 27643 <unfinished ...>

[pid 49277] **mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>**

[pid 49276] <... munmap resumed>) = 0

[pid 49277] <... mmap resumed>) = 0x739ef782f000

[pid 49276] **openat(AT_FDCWD, "/dev/shm/lab_shm_49275", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>**

[pid 49277] **openat(AT_FDCWD, "/dev/shm/sem.lab_access_49275", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>**

[pid 49276] <... openat resumed>) = 3

[pid 49277] <... openat resumed>) = 4

[pid 49276] **mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>**

[pid 49277] fstat(4, <unfinished ...>

[pid 49276] <... mmap resumed>) = 0x7b48ddca0000

[pid 49277] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}) = 0

[pid 49276] **openat(AT_FDCWD, "/dev/shm/sem.lab_access_49275", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>**

[pid 49277] getrandom(<unfinished ...>

[pid 49276] <... openat resumed>) = 4

[pid 49277] <... getrandom resumed>"\x03\x28\x70\x8a\xe4\xac\xf3\x16", 8, GRND_NONBLOCK) = 8

[pid 49276] fstat(4, <unfinished ...>

[pid 49277] brk(NULL <unfinished ...>

[pid 49276] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}) = 0

[pid 49277] <... brk resumed>) = 0x5b7c68d70000

[pid 49276] getrandom(<unfinished ...>

[pid 49277] brk(0x5b7c68d91000 <unfinished ...>

[pid 49276] <... getrandom resumed>"\xbf\xe5\x3c\x73\x46\x53\xe0\x03", 8, GRND_NONBLOCK) = 8

[pid 49277] <... brk resumed>) = 0x5b7c68d91000

[pid 49276] brk(NULL <unfinished ...>

[pid 49277**] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>**

[pid 49276] <... brk resumed>) = 0x5be646ad4000

[pid 49277] <... mmap resumed>) = 0x739ef782e000

[pid 49276] brk(0x5be646af5000 <unfinished ...>

[pid 49277] close(4 <unfinished ...>

[pid 49276] <... brk resumed>) = 0x5be646af5000

[pid 49277] <... close resumed>) = 0

[pid 49276] **mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>**

[pid 49277] **openat(AT_FDCWD, "/dev/shm/sem.lab_avail2_49275", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>**

[pid 49276] <... mmap resumed>) = 0x7b48ddc9f000

[pid 49277] <... openat resumed>) = 4

[pid 49277] fstat(4, <unfinished ...>

[pid 49276] close(4 <unfinished ...>

[pid 49277] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}) = 0

[pid 49276] <... close resumed>) = 0

[pid 49277] **mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>**

[pid 49276] **openat(AT_FDCWD, "/dev/shm/sem.lab_avail1_49275", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>**

[pid 49277] <... mmap resumed>) = 0x739ef782d000

[pid 49276] <... openat resumed>) = 4

[pid 49277] close(4 <unfinished ...>

[pid 49276] fstat(4, <unfinished ...>

[pid 49277] <... close resumed>) = 0

[pid 49276] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}) = 0

[pid 49277] openat(AT_FDCWD, "child2_output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644 <unfinished ...>

[pid 49276] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7b48ddc9e000

[pid 49276] close(4) = 0

[pid 49276] openat(AT_FDCWD, "child1_output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644 <unfinished ...>

[pid 49277] <... openat resumed>) = 4

[pid 49277] **futex(0x739ef782d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>**

[pid 49276] <... openat resumed>) = 4

[pid 49276] **futex(0x7b48ddc9e000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>**

[pid 49275] <... clock_nanosleep resumed>0x7fff516582e0) = 0

[pid 49275] fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}) = 0

[pid 49275] write(1, "Enter lines (empty line to finis"..., 36Enter lines (empty line to finish):

) = 36

[pid 49275] read(0, abc

"a", 1) = 1

[pid 49275] read(0, "b", 1) = 1

[pid 49275] read(0, "c", 1) = 1

[pid 49275] read(0, "\n", 1) = 1

[pid 49275] **futex(0x795873e8e000, FUTEX_WAKE, 1) = 1**

[pid 49276] <... futex resumed>) = 0

[pid 49275] read(0, <unfinished ...>

[pid 49276] write(4, "bc", 2) = 2

[pid 49276] write(4, "\n", 1) = 1

[pid 49276] **futex(0x7b48ddc9e000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANYaaaaabbbbbccccaaaaa oiuat**

<unfinished ...>

[pid 49275] <... read resumed>"a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "b", 1) = 1

[pid 49275] read(0, "b", 1) = 1

[pid 49275] read(0, "b", 1) = 1

[pid 49275] read(0, "b", 1) = 1

[pid 49275] read(0, "b", 1) = 1

[pid 49275] read(0, "c", 1) = 1

[pid 49275] read(0, "c", 1) = 1

[pid 49275] read(0, "c", 1) = 1

[pid 49275] read(0, "c", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, " ", 1) = 1

[pid 49275] read(0, "o", 1) = 1

[pid 49275] read(0, "i", 1) = 1

[pid 49275] read(0, "u", 1) = 1

[pid 49275] read(0, "a", 1) = 1

[pid 49275] read(0, "t", 1) = 1

[pid 49275] read(0, "\n", 1) = 1

[pid 49275] **futex(0x795873e8d000, FUTEX_WAKE, 1) = 1**

[pid 49277] <... futex resumed>) = 0

[pid 49275] read(0, <unfinished ...>

[pid 49277] write(4, "bbbbbcccc t", 11) = 11

[pid 49277] write(4, "\n", 1) = 1

[pid 49277] **futex(0x739ef782d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANYhello world**

<unfinished ...>

[pid 49275] <... read resumed>"h", 1) = 1

[pid 49275] read(0, "e", 1) = 1

[pid 49275] read(0, "l", 1) = 1

[pid 49275] read(0, "l", 1) = 1

[pid 49275] read(0, "o", 1) = 1

[pid 49275] read(0, " ", 1) = 1

[pid 49275] read(0, "w", 1) = 1

[pid 49275] read(0, "o", 1) = 1

[pid 49275] read(0, "r", 1) = 1

[pid 49275] read(0, "l", 1) = 1

[pid 49275] read(0, "d", 1) = 1

[pid 49275] read(0, "\n", 1) = 1

[pid 49275] **futex(0x795873e8d000, FUTEX_WAKE, 1) = 1**

[pid 49277] <... futex resumed>) = 0

[pid 49275] read(0, <unfinished ...>

[pid 49277] write(4, "hll wrld", 8) = 8

[pid 49277] write(4, "\n", 1) = 1

[pid 49277] **futex(0x739ef782d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY**

<unfinished ...>

[pid 49275] <... read resumed>"\n", 1) = 1

[pid 49275] **futex(0x795873e8e000, FUTEX_WAKE, 1) =** 1

[pid 49276] <... futex resumed>) = 0

[pid 49275] **futex(0x795873e8d000, FUTEX_WAKE, 1 <unfinished ...>**

[pid 49276] close(4 <unfinished ...>

[pid 49275] <... futex resumed>) = 1

[pid 49277] <... futex resumed>) = 0

[pid 49275] wait4(49276, <unfinished ...>

[pid 49277] close(4 <unfinished ...>

[pid 49276] <... close resumed>) = 0

[pid 49276] munmap(0x7b48ddc9f000, 32) = 0

[pid 49276] munmap(0x7b48ddc9e000, 32) = 0

[pid 49276] munmap(0x7b48ddca0000, 8192) = 0

[pid 49276] close(3) = 0

[pid 49276] exit_group(0) = ?

[pid 49276] +++ exited with 0 +++

[pid 49275] <... wait4 resumed>NULL, 0, NULL) = 49276

[pid 49277] <... close resumed>) = 0

[pid 49275] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=49276, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[pid 49275] wait4(49277, <unfinished ...>

[pid 49277] munmap(0x739ef782e000, 32) = 0

[pid 49277] munmap(0x739ef782d000, 32) = 0

[pid 49277] munmap(0x739ef782f000, 8192) = 0

[pid 49277] close(3) = 0

[pid 49277] exit_group(0) = ?

[pid 49277] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 49277

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=49277, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

write(1, "Processing complete. Results sav"..., 78Processing complete. Results saved to child1_output.txt and child2_output.txt

) = 78

munmap(0x795873e8f000, 32) = 0

unlink("/dev/shm/sem.lab_access_49275") = 0

munmap(0x795873e8e000, 32) = 0

unlink("/dev/shm/sem.lab_avail1_49275") = 0

munmap(0x795873e8d000, 32) = 0

unlink("/dev/shm/sem.lab_avail2_49275") = 0

munmap(0x795873e90000, 8192) = 0

unlink("/dev/shm/lab_shm_49275") = 0

close(3) = 0

exit_group(0) = ?

+++ exited with 0 +++

Вывод

В ходе выполнения лабораторной работы была разработана программа на языке C, использующая механизмы межпроцессорного взаимодействия (общая память, отображение памяти), с использованием семафоров для синхронизации в среде Linux.