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: Голосов Г.С.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 12.12.24

Москва, 2024

Постановка задачи

Вариант 5.

Пользователь вводит команды вида: «число». Далее это число передается от родительского процесса в дочерний. Дочерний процесс производит проверку на простоту. Если число составное, то в это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются.

Общий метод и алгоритм решения

Кратко опишите системные вызовы, которые вы использовали в лабораторной работе.

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int execl(const char *path, const char *arg, ...); - загружает и исполняет новый образ программы.
- int waitpid(pid_t pid, int *status, int options); - ожидает завершения дочернего процесса с идентификатором pid и получает его статус завершения.
- close(int fd) - закрыть файловый дескриптор
- open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- int shm_open (const char *name, int oflag, mode_t mode); - создать или открыть объект разделяемой памяти
- int shm_unlink (const char *name);- удалить объект разделяемой памяти по имени
- int ftruncate (int fd, off_t length); - изменить размер объекта разделяемой памяти
- void *mmap (void *addr, size_t length, int prot, int flags, int fd, off_t offset);-сопоставить область памяти с файлом. (Отображает объект разделяемой памяти в адресное пространство процесса)
- int munmap (void *addr, size_t length); - отменить сопоставление области памяти
- sem_t *sem_open (const char *name, int oflag); - создать или открыть именованный семафор
- int sem_post (sem_t *sem); - сигнализировать (разблокировать) семафор
- int sem_wait (sem_t *sem); - ожидать (заблокироваться) на семафоре
- int sem_unlink (const char *name); - удалить именованный семафор
- int sem_close (sem_t *sem) - закрывает именованный семафор

Далее описываете то, что вы делали в рамках лабораторной работы, а также то, как работает ваша программа и т.д..

Родительский процесс создает разделяемую память и 2 семафора. Затем вызывается fork для создания дочернего процесса, после чего родительский процесс принимает число и записывает его в разделяемую память. С помощью sem_post(sem_child) родительский процесс уведомляет дочерний о записи числа и ожидает его обработки с помощью sem_wait(sem_parent). Дочерний процесс ждет, пока родительский процесс запишет число в разделяемую память, с помощью sem_wait(sem_child), затем считывает число из разделяемой памяти и производит проверку на то, является ли число составным. Если число составное, то оно записывается в result.txt. По окончании проверки в разделяемую память записывается соответствующее сообщение (число составное или нет) и выполняется sem_post(sem_parent). Родительский процесс считывает сообщение об окончании обработки числа и выводит его.

Код программы

main.cpp

```
#include <iostream>
#include <sys/mman.h>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <unistd.h>
#include <cstring>

struct SharedData {
    int number;
    char message[256];
};

int main() {
    // Размер общей памяти
    size_t shared_size = sizeof(SharedData);

    // Создание разделяемой памяти
    int fd = shm_open("/my_shared_memory", O_CREAT | O_RDWR, 0666);
    if (fd == -1) {
        std::cerr << "Ошибка при создании разделяемой памяти" << std::endl;
        return 1;
    }
    ftruncate(fd, shared_size);

    // Отображение памяти
    SharedData* shared_data = (SharedData*)mmap(nullptr, shared_size, PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared_data == MAP_FAILED) {
        std::cerr << "Ошибка при отображении разделяемой памяти" << std::endl;
        return 1;
    }

    // Создание семафоров
    sem_t* sem_parent = sem_open("/sem_parent", O_CREAT, 0666, 0);
    sem_t* sem_child = sem_open("/sem_child", O_CREAT, 0666, 0);
    if (sem_parent == SEM_FAILED || sem_child == SEM_FAILED) {
        std::cerr << "Ошибка при создании семафоров" << std::endl;
        return 1;
    }

    // Создание дочернего процесса
    pid_t pid = fork();
    if (pid < 0) {
        std::cerr << "Ошибка при создании дочернего процесса" << std::endl;
        return 1;
    }

    if (pid > 0) { // Родительский процесс
        std::cout << "Введите число: ";
```

```
        int number;
        std::cin >> number;

        // Запись числа в разделяемую память
        shared_data->number = number;
        sem_post(sem_child); // Уведомляем дочерний процесс

        // Ожидание результата от дочернего процесса
        sem_wait(sem_parent);
        std::cout << "Результат: " << shared_data->message << std::endl;

        // Завершаем работу
        waitpid(pid, nullptr, 0);

        // Удаление ресурсов
        munmap(shared_data, shared_size);
        shm_unlink("/my_shared_memory");
        sem_close(sem_parent);
        sem_close(sem_child);
        sem_unlink("/sem_parent");
        sem_unlink("/sem_child");
    } else { // Дочерний процесс
        execl("./child", "child", nullptr);
        std::cerr << "Ошибка при вызове дочернего процесса" << std::endl;
        return 1;
    }

    return 0;
}
```

child.cpp

```
#include <iostream>
#include <sys/mman.h>
#include <fcntl.h>
#include <semaphore.h>
#include <unistd.h>
#include <cmath>
#include <cstring>

struct SharedData {
    int number;
    char message[256];
};

bool is_composite(int n) {
    if (n < 2) return false; // Отрицательное, 0 или 1
    for (int i = 2; i <= sqrt(n); ++i) {
        if (n % i == 0) return true; // Составное число
    }
    return false; // Простое число
}

int main() {
    // Открытие разделяемой памяти
```

```
    int fd = shm_open("/my_shared_memory", O_RDWR, 0666);
    if (fd == -1) {
        std::cerr << "Ошибка при открытии разделяемой памяти" << std::endl;
        return 1;
    }

    // Отображение памяти
    size_t shared_size = sizeof(SharedData);
    SharedData* shared_data = (SharedData*)mmap(nullptr, shared_size, PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared_data == MAP_FAILED) {
        std::cerr << "Ошибка при отображении разделяемой памяти" << std::endl;
        return 1;
    }

    // Открытие семафоров
    sem_t* sem_parent = sem_open("/sem_parent", 0);
    sem_t* sem_child = sem_open("/sem_child", 0);
    if (sem_parent == SEM_FAILED || sem_child == SEM_FAILED) {
        std::cerr << "Ошибка при открытии семафоров" << std::endl;
        return 1;
    }

    // Ожидание числа от родительского процесса
    sem_wait(sem_child);

    int number = shared_data->number;
    if (number < 0) {
        strncpy(shared_data->message, "Число отрицательное", sizeof(shared_data->message));
    } else {
        // Проверка числа
        if (is_composite(number)) {
            // Запись в файл, если число составное
            int file = open("result.txt", O_WRONLY | O_CREAT | O_TRUNC, 0666);
            if (file != -1) {
                dprintf(file, "%d\n", number);
                close(file);
                strncpy(shared_data->message, "Число составное, записано в файл",
sizeof(shared_data->message));
            } else {
                strncpy(shared_data->message, "Ошибка записи в файл", sizeof(shared_data-
>message));
            }
        } else {
            strncpy(shared_data->message, "Число простое", sizeof(shared_data->message));
        }
    }

    sem_post(sem_parent); // Уведомляем родительский процесс

    // Завершаем работу
    munmap(shared_data, shared_size);
    return 0;
}
```

Протокол работы программы

Тестирование:

tobiklosj@LAPTOP-C3C2PI9E:~/labs_OS/lab3/build$ ./main

Введите число: 24

Число составное, записано в файл

tobiklosj@LAPTOP-C3C2PI9E:~/labs_OS/lab3/build$ ./main

Введите число: 11

Число простое

tobiklosj@LAPTOP-C3C2PI9E:~/labs_OS/lab3/build$ ./main

Введите число: -2

Число отрицательное

tobiklosj@LAPTOP-C3C2PI9E:~/labs_OS/lab3/build$ ./main

Введите число: 333

Число составное, записано в файл

Strace:

```
tobiklosj@LAPTOP-C3C2PI9E:~/labs_OS/lab3/build$ strace -f ./main
execve("./main", ["./main"], 0x7fffb99b8a98 /* 20 vars */) = 0
brk(NULL)                               = 0x5593420dd000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff3be1af30) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f6010d1c000
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=36115, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 36115, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6010d13000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.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\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6010ae7000
mprotect(0x7f6010b81000, 1576960, PROT_NONE) = 0
mmap(0x7f6010b81000, 1118208, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x7f6010b81000
mmap(0x7f6010c92000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1ab000) = 0x7f6010c92000
mmap(0x7f6010d02000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x7f6010d02000
mmap(0x7f6010d10000, 10432, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f6010d10000
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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 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) = 0x7f60108be000
mprotect(0x7f60108e6000, 2023424, PROT_NONE) = 0
mmap(0x7f60108e6000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f60108e6000
mmap(0x7f6010a7b000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1bd000) = 0x7f6010a7b000
mmap(0x7f6010ad4000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f6010ad4000
mmap(0x7f6010ada000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f6010ada000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f60107d7000
mmap(0x7f60107e5000, 507904, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7f60107e5000
mmap(0x7f6010861000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x8a000) = 0x7f6010861000
mmap(0x7f60108bc000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7f60108bc000
```

```
  close(3)                              = 0
  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
  read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
  newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0
  mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f60107b7000
  mmap(0x7f60107ba000, 94208, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f60107ba000
  mmap(0x7f60107d1000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1a000) = 0x7f60107d1000
  mmap(0x7f60107d5000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x7f60107d5000
  close(3)                              = 0
  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f60107b5000
  arch_prctl(ARCH_SET_FS, 0x7f60107b63c0) = 0
  set_tid_address(0x7f60107b6690)       = 584
  set_robust_list(0x7f60107b66a0, 24)   = 0
  rseq(0x7f60107b6d60, 0x20, 0, 0x53053053) = 0
  mprotect(0x7f6010ad4000, 16384, PROT_READ) = 0
  mprotect(0x7f60107d5000, 4096, PROT_READ) = 0
  mprotect(0x7f60108bc000, 4096, PROT_READ) = 0
  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f60107b3000
  mprotect(0x7f6010d02000, 45056, PROT_READ) = 0
  mprotect(0x559341299000, 4096, PROT_READ) = 0
  mprotect(0x7f6010d56000, 8192, PROT_READ) = 0
  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
  munmap(0x7f6010d13000, 36115)         = 0
  getrandom("\x05\x4d\x1e\x71\xe1\x63\x61\x9c", 8, GRND_NONBLOCK) = 8
  brk(NULL)                             = 0x5593420dd000
  brk(0x5593420fe000)                   = 0x5593420fe000
  futex(0x7f6010d1077c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
  openat(AT_FDCWD, "/dev/shm/my_shared_memory",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3
  ftruncate(3, 260)                     = 0
  mmap(NULL, 260, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f6010d55000
  openat(AT_FDCWD, "/dev/shm/sem.sem_parent", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No
such file or directory)
  getrandom("\xf0\xc8\x8f\xb3\x77\x68\xbc\x37", 8, GRND_NONBLOCK) = 8
  newfstatat(AT_FDCWD, "/dev/shm/sem.2TTSSo", 0x7fff3be1ac40, AT_SYMLINK_NOFOLLOW) = -
1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/dev/shm/sem.2TTSSo", O_RDWR|O_CREAT|O_EXCL, 0666) = 4
  write(4, "\0\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", 32) = 32
  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f6010d1b000
  link("/dev/shm/sem.2TTSSo", "/dev/shm/sem.sem_parent") = 0
  newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
  unlink("/dev/shm/sem.2TTSSo")         = 0
  close(4)                              = 0
  openat(AT_FDCWD, "/dev/shm/sem.sem_child", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such
file or directory)
  getrandom("\x99\x7e\x2a\x53\x68\xf4\x55\x52", 8, GRND_NONBLOCK) = 8
  newfstatat(AT_FDCWD, "/dev/shm/sem.VZ4qfJ", 0x7fff3be1ac40, AT_SYMLINK_NOFOLLOW) = -1
ENOENT (No such file or directory)
  openat(AT_FDCWD, "/dev/shm/sem.VZ4qfJ", O_RDWR|O_CREAT|O_EXCL, 0666) = 4
  write(4, "\0\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", 32) = 32
```

```
  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f6010d1a000
  link("/dev/shm/sem.VZ4qfJ", "/dev/shm/sem.sem_child") = 0
  newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
  unlink("/dev/shm/sem.VZ4qfJ")           = 0
  close(4)                                = 0
  clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 585 attached
  , child_tidptr=0x7f60107b6690) = 585
  [pid   585] set_robust_list(0x7f60107b66a0, 24) = 0
  [pid   584] newfstatat(1, "",  <unfinished ...>
  [pid   585] execve("./child", ["child"], 0x7fff3be1b108 /* 20 vars */ <unfinished ...>
  [pid   584] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0
  [pid   584] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\276: ", 27Введите число: ) = 27
  [pid   584] newfstatat(0, "",  <unfinished ...>
  [pid   585] <... execve resumed>)       = 0
  [pid   584] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0
  [pid   585] brk(NULL <unfinished ...>
  [pid   584] read(0,  <unfinished ...>
  [pid   585] <... brk resumed>)          = 0x555c55d00000
  [pid   585] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffff71fb700) = -1 EINVAL (Invalid argument)
  [pid   585] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7fac6fa1f000
  [pid   585] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
  [pid   585] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
  [pid   585] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=36115, ...}, AT_EMPTY_PATH) = 0
  [pid   585] mmap(NULL, 36115, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fac6fa16000
  [pid   585] close(3)                    = 0
  [pid   585] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) =
3
  [pid   585] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
  [pid   585] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
  [pid   585] mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fac6f7ea000
  [pid   585] mprotect(0x7fac6f884000, 1576960, PROT_NONE) = 0
  [pid   585] mmap(0x7fac6f884000, 1118208, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x7fac6f884000
  [pid   585] mmap(0x7fac6f995000, 454656, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ab000) = 0x7fac6f995000
  [pid   585] mmap(0x7fac6fa05000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x7fac6fa05000
  [pid   585] mmap(0x7fac6fa13000, 10432, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fac6fa13000
  [pid   585] close(3)                    = 0
  [pid   585] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
  [pid   585] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
  [pid   585] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0
  [pid   585] mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fac6f703000
  [pid   585] mmap(0x7fac6f711000, 507904, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7fac6f711000
  [pid   585] mmap(0x7fac6f78d000, 372736, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x7fac6f78d000
```

```
  [pid   585] mmap(0x7fac6f7e8000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7fac6f7e8000
  [pid   585] close(3)                    = 0
  [pid   585] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
  [pid   585] 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
  [pid   585] 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   585] pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
  [pid   585] pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68
  [pid   585] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
  [pid   585] 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   585] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fac6f4da000
  [pid   585] mprotect(0x7fac6f502000, 2023424, PROT_NONE) = 0
  [pid   585] mmap(0x7fac6f502000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fac6f502000
  [pid   585] mmap(0x7fac6f697000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fac6f697000
  [pid   585] mmap(0x7fac6f6f0000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7fac6f6f0000
  [pid   585] mmap(0x7fac6f6f6000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fac6f6f6000
  [pid   585] close(3)                    = 0
  [pid   585] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) =
3
  [pid   585] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
  [pid   585] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0
  [pid   585] mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fac6f4ba000
  [pid   585] mmap(0x7fac6f4bd000, 94208, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7fac6f4bd000
  [pid   585] mmap(0x7fac6f4d4000, 16384, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) = 0x7fac6f4d4000
  [pid   585] mmap(0x7fac6f4d8000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x7fac6f4d8000
  [pid   585] close(3)                    = 0
  [pid   585] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7fac6f4b8000
  [pid   585] arch_prctl(ARCH_SET_FS, 0x7fac6f4b93c0) = 0
  [pid   585] set_tid_address(0x7fac6f4b9690) = 585
  [pid   585] set_robust_list(0x7fac6f4b96a0, 24) = 0
  [pid   585] rseq(0x7fac6f4b9d60, 0x20, 0, 0x53053053) = 0
  [pid   585] mprotect(0x7fac6f6f0000, 16384, PROT_READ) = 0
  [pid   585] mprotect(0x7fac6f4d8000, 4096, PROT_READ) = 0
  [pid   585] mprotect(0x7fac6f7e8000, 4096, PROT_READ) = 0
  [pid   585] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7fac6f4b6000
  [pid   585] mprotect(0x7fac6fa05000, 45056, PROT_READ) = 0
  [pid   585] mprotect(0x555c54303000, 4096, PROT_READ) = 0
  [pid   585] mprotect(0x7fac6fa59000, 8192, PROT_READ) = 0
  [pid   585] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
  [pid   585] munmap(0x7fac6fa16000, 36115) = 0
```

```
[pid   585] getrandom("\x03\x4f\x7b\x3c\x56\x21\x9d\x45", 8, GRND_NONBLOCK) = 8
[pid   585] brk(NULL)                   = 0x555c55d00000
[pid   585] brk(0x555c55d21000)         = 0x555c55d21000
[pid   585] futex(0x7fac6fa1377c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
[pid   585] openat(AT_FDCWD, "/dev/shm/my_shared_memory",
O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 3
[pid   585] mmap(NULL, 260, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fac6fa58000
[pid   585] openat(AT_FDCWD, "/dev/shm/sem.sem_parent", O_RDWR|O_NOFOLLOW) = 4
[pid   585] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
[pid   585] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fac6fa1e000
[pid   585] close(4)                    = 0
[pid   585] openat(AT_FDCWD, "/dev/shm/sem.sem_child", O_RDWR|O_NOFOLLOW) = 4
[pid   585] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
[pid   585] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fac6fa1d000
[pid   585] close(4)                    = 0
[pid   585] futex(0x7fac6fa1d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY24
 <unfinished ...>
[pid   584] <... read resumed>"24\n", 1024) = 3
[pid   584] futex(0x7f6010d1a000, FUTEX_WAKE, 1) = 1
[pid   585] <... futex resumed>)        = 0
[pid   584] futex(0x7f6010d1b000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid   585] openat(AT_FDCWD, "result.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4
[pid   585] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=0, ...}, AT_EMPTY_PATH) = 0
[pid   585] lseek(4, 0, SEEK_CUR)       = 0
[pid   585] write(4, "24\n", 3)         = 3
[pid   585] close(4)                    = 0
[pid   585] futex(0x7fac6fa1e000, FUTEX_WAKE, 1 <unfinished ...>
[pid   584] <... futex resumed>)        = 0
[pid   585] <... futex resumed>)        = 1
[pid   584] write(1, "\320\247\320\270\321\201\320\273\320\276
\321\201\320\276\321\201\321\202\320\260\320\262\320\275\320\276\320\265, \320"..., 60Число
составное, записано в файл
 <unfinished ...>
[pid   585] munmap(0x7fac6fa58000, 260 <unfinished ...>
[pid   584] <... write resumed>)        = 60
[pid   585] <... munmap resumed>)       = 0
[pid   584] wait4(585,  <unfinished ...>
[pid   585] exit_group(0)               = ?
[pid   585] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 585
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=585, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---
munmap(0x7f6010d55000, 260)             = 0
unlink("/dev/shm/my_shared_memory")     = 0
munmap(0x7f6010d1b000, 32)              = 0
munmap(0x7f6010d1a000, 32)              = 0
unlink("/dev/shm/sem.sem_parent")       = 0
unlink("/dev/shm/sem.sem_child")        = 0
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе лабораторной работы мне удалось изучить альтернативный pipe способ передачи данных между процессами – с помощью memory mapping. Также пришлось разобраться с синхронизацией процессов с помощью семафора. Работа была интересной и познавательной.