Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Группа: М8О-215Б-23

Студент: Голосов Г.С.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 31.10.24

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 pipe(int \*fd); – создаёт канал (pipe) для межпроцессного взаимодействия.
* int dup2(int oldfd, int newfd); - дублирует файловый дескриптор oldfd и заменяет им дескриптор newfd.
* 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 файл
* open(const char \*pathname, int flags, mode\_t mode) - открытие\создание файла
* exit(int status) - завершения выполнения процесса
* write();– записывает данные в канал.
* read();– читает данные из канала.

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

С помощью pipe() создаются два канала: fd1 для передачи данных от родительского процесса дочернему и fd2 для передачи данных от дочернего процесса родительскому. Далее с помощью fork() создается дочерний процесс.

После вызова fork(), родительский процесс закрывает ненужные конец чтения первого pipe и конец записи второго pipe. Выводится сообщение о необходимости ввода числа, после чего происходит считывание числа, введенного через стандартный ввод, и число передается дочернему процессу. После передачи числа дочернему процессу, родительский процесс ожидает завершения его работы с помощью waitpid. Если число оказалось простым или отрицательным, родительский процесс выводит соответствующее сообщение. Если же число составное, то закрывается запись в fd1 и чтение из fd2, программа завершает работу.

В свою очередь дочерний процесс после вызова fork(), закрывает запись в fd1 и чтение из fd2. Затем с помощью dup2 стандартный ввод stdin перенаправляется на чтение в fd1, а стандартный вывод stdout на запись в fd2. Далее с помощью execl запускается код программы child, которая делает проверку числа: если число составное, то оно записывается в файл result.txt. Результат выполнения программы передается родительскому процессу.

**Код программы**

**main.cpp**

#include <iostream>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

void to\_pipe(int fd[], int number) {

    write(fd[1], &number, sizeof(number));

}

int main() {

    int fd1[2], fd2[2];

    if (pipe(fd1) < 0 || pipe(fd2) < 0) {

        std::cout << "Ошибка при создании pipe" << std::endl;

        return 1;

    }

    pid\_t pid = fork();

    if (pid < 0) {

        std::cout << "Ошибка при создании дочернего процесса" << std::endl;

        return 1;

    }

    if (pid > 0) { // Родительский процесс

        close(fd1[0]); // Закрываем чтение из первого pipe

        close(fd2[1]); // Закрываем запись во второй pipe

        std::cout << "Введите данные в формате: «число<endline>»." << std::endl;

        int number;

        std::cin >> number;

        if (write(fd1[1], &number, sizeof(int)) == -1) {

            std::cout << "Ошибка при передаче числа";

            return 1;

        }

        // Ожидание завершения дочернего процесса

        int status;

        waitpid(pid, &status, 0);

        if (WIFEXITED(status) && WEXITSTATUS(status) == 1) {

            std::cerr << "Число отрицательное, либо простое. Завершение работы." << std::endl;

            return 1;

        }

        close(fd1[1]); // Закрываем запись в первый pipe

        close(fd2[0]); // Закрываем чтение из второго pipe

    } else { // Дочерний процесс

        close(fd1[1]); // Закрываем запись в первый pipe

        close(fd2[0]); // Закрываем чтение из второго pipe

        // Перенаправление stdin и stdout на pipe

        if (dup2(fd1[0], STDIN\_FILENO) == -1) {

            std::cout << "Ошибка при перенаправлении stdin";

            return 1;

        }

        if (dup2(fd2[1], STDOUT\_FILENO) == -1) {

            std::cout << "Ошибка при перенаправлении stdout";

            return 1;

        }

        // Запуск дочернего процесса

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

        std::cout << "Ошибка при вызове execl" << std::endl;

        return 1;

    }

    return 0;

}

**child.cpp**

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <cmath>

#include <cstring>

bool is\_composite(int n) {

    if (n < 2) return false; // Отрицательное или 0, 1

    for (int i = 2; i <= sqrt(n); ++i) {

        if (n % i == 0) return true; // Число составное

    }

    return false; // Число простое

}

int main() {

    int number;

    if (read(STDIN\_FILENO, &number, sizeof(number)) == -1) {

        std::cerr << "Ошибка при чтении числа" << std::endl;

        return 1;

    }

    if (number < 0) {

        return 1; // Число отрицательное

    }

    if (is\_composite(number)) {

        // Если число составное, записываем его в файл

        int file = open("result.txt", O\_WRONLY | O\_CREAT | O\_TRUNC);

        if (file == -1) {

            std::cerr << "Ошибка при открытии файла" << std::endl;

            return 1;

        }

        // Форматирование результата в строку

        char buffer[16];

        int n = snprintf(buffer, sizeof(buffer), "%d\n", number);

        if (n <= 0) {

            std::cerr << "Ошибка при форматировании строки" << std::endl;

            close(file);

            return 1;

        }

        // Запись строки в файл

        if (write(file, buffer, n) == -1) {

            std::cerr << "Ошибка при записи в файл" << std::endl;

            close(file);

            return 1;

        }

        close(file);

    } else {

        return 1;

    }

    return 0;

}

**Протокол работы программы**

**Тестирование:**

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build# ./main

Введите данные в формате: «число<endline>».

24

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build# cat result.txt

24

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build# ./main

Введите данные в формате: «число<endline>».

-15

Число отрицательное, либо простое. Завершение работы.

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build# ./main

Введите данные в формате: «число<endline>».

13

Число отрицательное, либо простое. Завершение работы.

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build# ./main

Введите данные в формате: «число<endline>».

168

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build# cat result.txt

168

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build#

**Strace:**

root@563c03e04e79:/workspaces/Labs\_OS/lab1/build# strace ./main

**execve**("./main", ["./main"], 0x7ffec114d028 /\* 26 vars \*/) = 0

brk(NULL) = 0xe86000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fadfee03000

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=25258, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fadfedfc000

**close**(3) = 0

**openat**(AT\_FDCWD, "/usr/local/lib64/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\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fadfeb8e000

mmap(0x7fadfec33000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000) = 0x7fadfec33000

mmap(0x7fadfed5c000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7fadfed5c000

mmap(0x7fadfedea000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000) = 0x7fadfedea000

mmap(0x7fadfedf8000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fadfedf8000

**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\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fadfeaaf000

mmap(0x7fadfeabf000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000) = 0x7fadfeabf000

mmap(0x7fadfeb32000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000) = 0x7fadfeb32000

mmap(0x7fadfeb8c000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000) = 0x7fadfeb8c000

**close**(3) = 0

**openat**(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fadfea82000

mmap(0x7fadfea86000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7fadfea86000

mmap(0x7fadfeaa9000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000) = 0x7fadfeaa9000

mmap(0x7fadfeaad000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000) = 0x7fadfeaad000

**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\20t\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

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, 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, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fadfe8a1000

mmap(0x7fadfe8c7000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000) = 0x7fadfe8c7000

mmap(0x7fadfea1c000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000) = 0x7fadfea1c000

mmap(0x7fadfea6f000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7fadfea6f000

mmap(0x7fadfea75000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fadfea75000

**close**(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fadfe89f000

arch\_prctl(ARCH\_SET\_FS, 0x7fadfe8a0480) = 0

set\_tid\_address(0x7fadfe8a0750) = 21095

set\_robust\_list(0x7fadfe8a0760, 24) = 0

rseq(0x7fadfe8a0da0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fadfea6f000, 16384, PROT\_READ) = 0

mprotect(0x7fadfeaad000, 4096, PROT\_READ) = 0

mprotect(0x7fadfeb8c000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fadfe89d000

mprotect(0x7fadfedea000, 45056, PROT\_READ) = 0

mprotect(0x403000, 4096, PROT\_READ) = 0

mprotect(0x7fadfee35000, 8192, PROT\_READ) = 0

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

munmap(0x7fadfedfc000, 25258) = 0

futex(0x7fadfedf873c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

getrandom("\x06\x56\xf6\x12\x7c\xc9\x9b\x43", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0xe86000

brk(0xea7000) = 0xea7000

**pipe2**([3, 4], 0) = 0

**pipe2**([5, 6], 0) = 0

**clone**(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 21096 attached

, child\_tidptr=0x7fadfe8a0750) = 21096

[pid 21096] set\_robust\_list(0x7fadfe8a0760, 24 <unfinished ...>

[pid 21095] **close**(3 <unfinished ...>

[pid 21096] <... set\_robust\_list resumed>) = 0

[pid 21095] <... close resumed>) = 0

[pid 21095] **close**(6) = 0

[pid 21096] **close**(4 <unfinished ...>

[pid 21095] newfstatat(1, "", <unfinished ...>

[pid 21096] <... close resumed>) = 0

[pid 21095] <... newfstatat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x1), ...}, AT\_EMPTY\_PATH) = 0

[pid 21096] **close**(5 <unfinished ...>

[pid 21095] **write**(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\264\320\260\320\275\320\275\321\213\320\265 \320\262 \321"..., 72 <unfinished ...>

Введите данные в формате: «число<endline>».

[pid 21096] <... close resumed>) = 0

[pid 21095] <... write resumed>) = 72

[pid 21096] **dup2**(3, 0 <unfinished ...>

[pid 21095] newfstatat(0, "", <unfinished ...>

[pid 21096] <... dup2 resumed>) = 0

[pid 21095] <... newfstatat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x1), ...}, AT\_EMPTY\_PATH) = 0

[pid 21095] **read**(0, <unfinished ...>

[pid 21096] **dup2**(6, 1) = 1

[pid 21096] **execve**("./child", ["child"], 0x7fff19a36e68 /\* 26 vars \*/) = 0

[pid 21096] brk(NULL) = 0x1db4000

[pid 21096] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f480ba1c000

[pid 21096] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 21096] **openat**(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 4

[pid 21096] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=25258, ...}, AT\_EMPTY\_PATH) = 0

[pid 21096] mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 4, 0) = 0x7f480ba15000

[pid 21096] **close**(4) = 0

[pid 21096] **openat**(AT\_FDCWD, "/usr/local/lib64/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 4

[pid 21096] **read**(4, "\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\0"..., 832) = 832

[pid 21096] newfstatat(4, "", {st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

[pid 21096] mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7f480b7a7000

[pid 21096] mmap(0x7f480b84c000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0xa5000) = 0x7f480b84c000

[pid 21096] mmap(0x7f480b975000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1ce000) = 0x7f480b975000

[pid 21096] mmap(0x7f480ba03000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x25c000) = 0x7f480ba03000

[pid 21096] mmap(0x7f480ba11000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f480ba11000

[pid 21096] **close**(4) = 0

[pid 21096] **openat**(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 4

[pid 21096] **read**(4, "\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\0"..., 832) = 832

[pid 21096] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

[pid 21096] mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7f480b6c8000

[pid 21096] mmap(0x7f480b6d8000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x10000) = 0x7f480b6d8000

[pid 21096] mmap(0x7f480b74b000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x83000) = 0x7f480b74b000

[pid 21096] mmap(0x7f480b7a5000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0xdc000) = 0x7f480b7a5000

[pid 21096] **close**(4) = 0

[pid 21096] **openat**(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 4

[pid 21096] **read**(4, "\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 21096] newfstatat(4, "", {st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

[pid 21096] mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7f480b69b000

[pid 21096] mmap(0x7f480b69f000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x4000) = 0x7f480b69f000

[pid 21096] mmap(0x7f480b6c2000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x27000) = 0x7f480b6c2000

[pid 21096] mmap(0x7f480b6c6000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x2b000) = 0x7f480b6c6000

[pid 21096] **close**(4) = 0

[pid 21096] **openat**(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 4

[pid 21096] **read**(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) = 832

[pid 21096] 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 21096] newfstatat(4, "", {st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, AT\_EMPTY\_PATH) = 0

[pid 21096] 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 21096] mmap(NULL, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7f480b4ba000

[pid 21096] mmap(0x7f480b4e0000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x26000) = 0x7f480b4e0000

[pid 21096] mmap(0x7f480b635000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x17b000) = 0x7f480b635000

[pid 21096] mmap(0x7f480b688000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1ce000) = 0x7f480b688000

[pid 21096] mmap(0x7f480b68e000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f480b68e000

[pid 21096] **close**(4) = 0

[pid 21096] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f480b4b8000

[pid 21096] arch\_prctl(ARCH\_SET\_FS, 0x7f480b4b9480) = 0

[pid 21096] set\_tid\_address(0x7f480b4b9750) = 21096

[pid 21096] set\_robust\_list(0x7f480b4b9760, 24) = 0

[pid 21096] rseq(0x7f480b4b9da0, 0x20, 0, 0x53053053) = 0

[pid 21096] mprotect(0x7f480b688000, 16384, PROT\_READ) = 0

[pid 21096] mprotect(0x7f480b6c6000, 4096, PROT\_READ) = 0

[pid 21096] mprotect(0x7f480b7a5000, 4096, PROT\_READ) = 0

[pid 21096] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f480b4b6000

[pid 21096] mprotect(0x7f480ba03000, 45056, PROT\_READ) = 0

[pid 21096] mprotect(0x403000, 4096, PROT\_READ) = 0

[pid 21096] mprotect(0x7f480ba4e000, 8192, PROT\_READ) = 0

[pid 21096] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 21096] munmap(0x7f480ba15000, 25258) = 0

[pid 21096] futex(0x7f480ba1173c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

[pid 21096] getrandom("\x86\xdf\xc7\x78\x1e\x4d\xd2\x1b", 8, GRND\_NONBLOCK) = 8

[pid 21096] brk(NULL) = 0x1db4000

[pid 21096] brk(0x1dd5000) = 0x1dd5000

[pid 21096] **read**(0, 21

<unfinished ...>

[pid 21095] <... read resumed>"21\n", 1024) = 3

[pid 21095] **write**(4, "\25\0\0\0", 4) = 4

[pid 21096] <... read resumed>"\25\0\0\0", 4) = 4

[pid 21095] **wait4**(21096, <unfinished ...>

[pid 21096] **openat**(AT\_FDCWD, "result.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 000) = 4

[pid 21096] **write**(4, "21\n", 3) = 3

[pid 21096] **close**(4) = 0

[pid 21096] exit\_group(0) = ?

[pid 21096] **+++ exited with 0 +++**

<... **wait4** resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 21096

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=21096, si\_uid=0, si\_status=0, si\_utime=0, si\_stime=0} ---

**close**(4) = 0

**close**(5) = 0

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

exit\_group(0) = ?

**+++ exited with 0 +++**

**Вывод**

В ходе лабораторной работы получилось изучить создание процессов с помощью fork(), а также работу других системных вызовов. Изучение инструментов для работы с процессами и потоками было очень интересным. Главной проблемой стало то, что в strace не отображался запуск dup2, что было очень странным, однако после некоторых изменений всё заработало.