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-210Б-23

Студент: Морозов Захар Олегович

Преподаватель: Бахарев В.Д.

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

Дата: 18.02.2025

Москва, 2025

**Постановка задачи**

**Вариант 19:**

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. Правило фильтрации: с вероятностью 80% строки отправляются в pipe1, иначе в pipe2. Дочерние процессы удаляют все гласные из строк.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* int pipe(int \*fd); – создаёт пайп и помещает дескрипторы в fd[0], fd[1], для чтения и записи.
* int write(int fd, const void\* buff, int count); – записывает по дескриптору fd count байт из buff.
* void exit(int number); – вызывает нормальное завершение программы с кодом number.
* int dup2(int fd1, int fd2); – делает эквиваентными дескрипторы fd1 и fd2.
* int exec(char\* path, const char\* argc); – заменяет текущий процесс на процесс path, с аргументами argc;
* int close(int fd); – закрывает дескриптор fd.
* pid\_t wait(int status) — функция, которая приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится,

Я создал два файла parent и child.

Программа создает неименованный канал (pipe) для передачи данных между родительским и дочерним процессами. Она запрашивает у пользователя ввод имени файла, после чего создаёт дочерний процесс с помощью fork(). Дочерний процесс перенаправляет стандартный ввод на чтение из канала и запускает другую программу (дочернюю программу), передавая ей имя файла. Родительский процесс читает данные из стандартного ввода до тех пор, пока пользователь не введёт пустую строку (нажатием Enter), и записывает эти данные в канал. После завершения записи родительский процесс закрывает канал и ожидает завершения дочернего процесса.

В файле child я обрабатываю получнные из родительского процесса данные и записываю их в файл, а тажке проверяю деление на 0, и ввод пустой строки.

**Код программы**

**Parent.c**

#include <unistd.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#define BUFFER\_SIZE 256

int main() {

    int pipe1[2], pipe2[2];

    char filename1[BUFFER\_SIZE];

    char filename2[BUFFER\_SIZE];

    ssize\_t bytes\_read;

    const char \*prompt1 = "Enter the file name for child1: ";

    write(STDOUT\_FILENO, prompt1, strlen(prompt1));

    bytes\_read = read(STDIN\_FILENO, filename1, BUFFER\_SIZE - 1);

    filename1[bytes\_read - 1] = '\0';

    const char \*prompt2 = "Enter the file name for child2: ";

    write(STDOUT\_FILENO, prompt2, strlen(prompt2));

    bytes\_read = read(STDIN\_FILENO, filename2, BUFFER\_SIZE - 1);

    filename2[bytes\_read - 1] = '\0';

    int file\_check1 = open(filename1, O\_WRONLY | O\_CREAT | O\_TRUNC, 0666);

    if (file\_check1 == -1) {

        const char \*error\_msg = "Error: could not open the file for child1\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        perror("open");

        return 1;

    }

    close(file\_check1);

    int file\_check2 = open(filename2, O\_WRONLY | O\_CREAT | O\_TRUNC, 0666);

    if (file\_check2 == -1) {

        const char \*error\_msg = "Error: could not open the file for child2\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        perror("open");

        return 1;

    }

    close(file\_check2);

    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {

        const char \*error\_msg = "Error during creation pipe\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        return 1;

    }

    pid\_t pid1 = fork();

    if (pid1 == -1) {

        const char \*error\_msg = "Error when creating a child process 1\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        return 1;

    } else if (pid1 == 0) {

        close(pipe1[1]);

        dup2(pipe1[0], STDIN\_FILENO);

        execl("./child.out", "./child.out", filename1, NULL);

        const char \*error\_msg = "Error when starting child1\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        exit(1);

    }

    pid\_t pid2 = fork();

    if (pid2 == -1) {

        const char \*error\_msg = "Error when creating a child process 2\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        return 1;

    } else if (pid2 == 0) {

        close(pipe2[1]);

        dup2(pipe2[0], STDIN\_FILENO);

        execl("./child.out", "./child.out", filename2, NULL);

        const char \*error\_msg = "Error when starting child2\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        exit(1);

    }

    close(pipe1[0]);

    close(pipe2[0]);

    char input[BUFFER\_SIZE];

    int randomInt = rand() % 5 + 1;

    while (1) {

        bytes\_read = read(STDIN\_FILENO, input, sizeof(input) - 1);

        if (bytes\_read <= 0) {

            const char \*error\_msg = "Error while reading the string\n";

            write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

            break;

        }

        if (bytes\_read > 0 && input[bytes\_read - 1] == '\n') {

            input[bytes\_read - 1] = '\0';

        }

        if (strcmp(input, "exit") == 0) {

            write(pipe2[1], input, strlen(input) + 1);

            write(pipe1[1], input, strlen(input) + 1);

            break;

        }

        randomInt = rand() % 5 + 1;

        if (randomInt == 5) {

            write(pipe2[1], input, strlen(input) + 1);

        } else {

            write(pipe1[1], input, strlen(input) + 1);

        }

    }

    close(pipe1[1]);

    close(pipe2[1]);

    waitpid(pid1, NULL, 0);

    waitpid(pid2, NULL, 0);

    return 0;

}

**Child.c**

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define BUFFER\_SIZE 256

bool is\_vowel(char c) {

    c = (c >= 'A' && c <= 'Z') ? c + 32 : c;

    return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';

}

void remove\_vowels(char \*str) {

    int write\_index = 0;

    for (int read\_index = 0; str[read\_index] != '\0'; read\_index++) {

        if (!is\_vowel(str[read\_index])) {

            str[write\_index++] = str[read\_index];

        }

    }

    str[write\_index] = '\0';

}

int main(int argc, char \*argv[]) {

    if (argc < 2) {

        const char \*error\_msg = "Error: The file name was not passed\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        return 1;

    }

    int file = open(argv[1], O\_WRONLY | O\_CREAT | O\_TRUNC, 0666);

    if (file == -1) {

        const char \*error\_msg = "Error: the file could not be opened for writing\n";

        write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

        perror("open");

        return 1;

    }

    char buffer[BUFFER\_SIZE];

    ssize\_t bytes\_read;

    while ((bytes\_read = read(STDIN\_FILENO, buffer, sizeof(buffer) - 1)) > 0) {

        buffer[bytes\_read - 1] = '\0';

        if(strcmp(buffer, "exit") == 0){

            close(file);

            return 0;

        }

        remove\_vowels(buffer);

        if (write(file, buffer, strlen(buffer)) == -1 || write(file, "\n", 1) == -1) {

            const char \*error\_msg = "Error: failed to write to a file\n";

            write(STDERR\_FILENO, error\_msg, strlen(error\_msg));

            close(file);

            return 1;

        }

    }

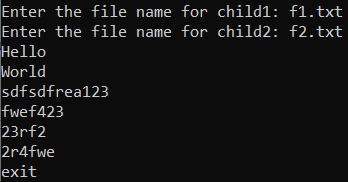
    close(file);

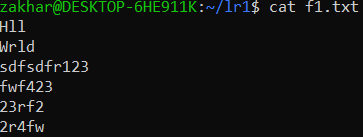
    return 0;

}

**Протокол работы программы**

Тестирование:



****

****

**Strace:**

execve("./parent.out", ["./parent.out"], 0x7ffc1a7fce28 /\* 22 vars \*/) = 0

brk(NULL) = 0x55c746fdd000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f002a248000

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=19871, ...}) = 0

mmap(NULL, 19871, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f002a243000

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\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) = 0x7f002a031000

mmap(0x7f002a059000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f002a059000

mmap(0x7f002a1e1000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7f002a1e1000

mmap(0x7f002a230000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7f002a230000

mmap(0x7f002a236000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f002a236000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f002a02e000

arch\_prctl(ARCH\_SET\_FS, 0x7f002a02e740) = 0

set\_tid\_address(0x7f002a02ea10) = 5697

set\_robust\_list(0x7f002a02ea20, 24) = 0

rseq(0x7f002a02f060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f002a230000, 16384, PROT\_READ) = 0

mprotect(0x55c745dee000, 4096, PROT\_READ) = 0

mprotect(0x7f002a280000, 8192, PROT\_READ) = 0

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

munmap(0x7f002a243000, 19871) = 0

write(1, "Enter the file name for child1: ", 32Enter the file name for child1: ) = 32

read(0, f1.txt

"f1.txt\n", 255) = 7

write(1, "Enter the file name for child2: ", 32Enter the file name for child2: ) = 32

read(0, f2.txt

"f2.txt\n", 255) = 7

openat(AT\_FDCWD, "f1.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666) = 3

close(3) = 0

openat(AT\_FDCWD, "f2.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666) = 3

close(3) = 0

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

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

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 5698 attached

, child\_tidptr=0x7f002a02ea10) = 5698

[pid 5698] set\_robust\_list(0x7f002a02ea20, 24 <unfinished ...>

[pid 5697] clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD <unfinished ...>

[pid 5698] <... set\_robust\_list resumed>) = 0

strace: Process 5699 attached

[pid 5698] close(4 <unfinished ...>

[pid 5697] <... clone resumed>, child\_tidptr=0x7f002a02ea10) = 5699

[pid 5699] set\_robust\_list(0x7f002a02ea20, 24 <unfinished ...>

[pid 5697] close(3) = 0

[pid 5699] <... set\_robust\_list resumed>) = 0

[pid 5698] <... close resumed>) = 0

[pid 5697] close(5 <unfinished ...>

[pid 5699] close(6 <unfinished ...>

[pid 5698] dup2(3, 0 <unfinished ...>

[pid 5697] <... close resumed>) = 0

[pid 5699] <... close resumed>) = 0

[pid 5698] <... dup2 resumed>) = 0

[pid 5697] read(0, <unfinished ...>

[pid 5698] execve("./child.out", ["./child.out", "f1.txt"], 0x7ffe184c7568 /\* 22 vars \*/ <unfinished ...>

[pid 5699] dup2(5, 0) = 0

[pid 5699] execve("./child.out", ["./child.out", "f2.txt"], 0x7ffe184c7568 /\* 22 vars \*/ <unfinished ...>

[pid 5698] <... execve resumed>) = 0

[pid 5698] brk(NULL) = 0x5597e8fea000

[pid 5698] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 5699] <... execve resumed>) = 0

[pid 5698] <... mmap resumed>) = 0x7f7b49aeb000

[pid 5699] brk(NULL) = 0x56371b5f2000

[pid 5698] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 5699] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 5698] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 5699] <... mmap resumed>) = 0x7f3af14fe000

[pid 5698] <... openat resumed>) = 4

[pid 5699] access("/etc/ld.so.preload", R\_OK <unfinished ...>

[pid 5698] fstat(4, <unfinished ...>

[pid 5699] <... access resumed>) = -1 ENOENT (No such file or directory)

[pid 5698] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=19871, ...}) = 0

[pid 5699] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 5698] mmap(NULL, 19871, PROT\_READ, MAP\_PRIVATE, 4, 0 <unfinished ...>

[pid 5699] <... openat resumed>) = 6

[pid 5698] <... mmap resumed>) = 0x7f7b49ae6000

[pid 5699] fstat(6, <unfinished ...>

[pid 5698] close(4 <unfinished ...>

[pid 5699] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=19871, ...}) = 0

[pid 5699] mmap(NULL, 19871, PROT\_READ, MAP\_PRIVATE, 6, 0 <unfinished ...>

[pid 5698] <... close resumed>) = 0

[pid 5699] <... mmap resumed>) = 0x7f3af14f9000

[pid 5698] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 5699] close(6 <unfinished ...>

[pid 5698] <... openat resumed>) = 4

[pid 5699] <... close resumed>) = 0

[pid 5698] read(4, <unfinished ...>

[pid 5699] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 5698] <... 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 5699] <... openat resumed>) = 6

[pid 5698] pread64(4, <unfinished ...>

[pid 5699] read(6, <unfinished ...>

[pid 5698] <... 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 5699] <... 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 5698] fstat(4, <unfinished ...>

[pid 5699] pread64(6, <unfinished ...>

[pid 5698] <... fstat resumed>{st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

[pid 5699] <... 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 5698] pread64(4, <unfinished ...>

[pid 5699] fstat(6, <unfinished ...>

[pid 5698] <... 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 5699] <... fstat resumed>{st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

[pid 5698] mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0 <unfinished ...>

[pid 5699] pread64(6, <unfinished ...>

[pid 5698] <... mmap resumed>) = 0x7f7b498d4000

[pid 5699] <... 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 5698] mmap(0x7f7b498fc000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x28000 <unfinished ...>

[pid 5699] mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 6, 0 <unfinished ...>

[pid 5698] <... mmap resumed>) = 0x7f7b498fc000

[pid 5699] <... mmap resumed>) = 0x7f3af12e7000

[pid 5698] mmap(0x7f7b49a84000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1b0000 <unfinished ...>

[pid 5699] mmap(0x7f3af130f000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x28000 <unfinished ...>

[pid 5698] <... mmap resumed>) = 0x7f7b49a84000

[pid 5699] <... mmap resumed>) = 0x7f3af130f000

[pid 5698] mmap(0x7f7b49ad3000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1fe000 <unfinished ...>

[pid 5699] mmap(0x7f3af1497000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x1b0000 <unfinished ...>

[pid 5698] <... mmap resumed>) = 0x7f7b49ad3000

[pid 5699] <... mmap resumed>) = 0x7f3af1497000

[pid 5699] mmap(0x7f3af14e6000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x1fe000 <unfinished ...>

[pid 5698] mmap(0x7f7b49ad9000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 5699] <... mmap resumed>) = 0x7f3af14e6000

[pid 5698] <... mmap resumed>) = 0x7f7b49ad9000

[pid 5699] mmap(0x7f3af14ec000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 5698] close(4 <unfinished ...>

[pid 5699] <... mmap resumed>) = 0x7f3af14ec000

[pid 5698] <... close resumed>) = 0

[pid 5699] close(6) = 0

[pid 5698] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 5699] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 5698] <... mmap resumed>) = 0x7f7b498d1000

[pid 5699] <... mmap resumed>) = 0x7f3af12e4000

[pid 5698] arch\_prctl(ARCH\_SET\_FS, 0x7f7b498d1740 <unfinished ...>

[pid 5699] arch\_prctl(ARCH\_SET\_FS, 0x7f3af12e4740 <unfinished ...>

[pid 5698] <... arch\_prctl resumed>) = 0

[pid 5699] <... arch\_prctl resumed>) = 0

[pid 5698] set\_tid\_address(0x7f7b498d1a10 <unfinished ...>

[pid 5699] set\_tid\_address(0x7f3af12e4a10 <unfinished ...>

[pid 5698] <... set\_tid\_address resumed>) = 5698

[pid 5699] <... set\_tid\_address resumed>) = 5699

[pid 5698] set\_robust\_list(0x7f7b498d1a20, 24 <unfinished ...>

[pid 5699] set\_robust\_list(0x7f3af12e4a20, 24 <unfinished ...>

[pid 5698] <... set\_robust\_list resumed>) = 0

[pid 5699] <... set\_robust\_list resumed>) = 0

[pid 5698] rseq(0x7f7b498d2060, 0x20, 0, 0x53053053 <unfinished ...>

[pid 5699] rseq(0x7f3af12e5060, 0x20, 0, 0x53053053 <unfinished ...>

[pid 5698] <... rseq resumed>) = 0

[pid 5699] <... rseq resumed>) = 0

[pid 5698] mprotect(0x7f7b49ad3000, 16384, PROT\_READ <unfinished ...>

[pid 5699] mprotect(0x7f3af14e6000, 16384, PROT\_READ <unfinished ...>

[pid 5698] <... mprotect resumed>) = 0

[pid 5699] <... mprotect resumed>) = 0

[pid 5698] mprotect(0x5597e791b000, 4096, PROT\_READ <unfinished ...>

[pid 5699] mprotect(0x563719e8d000, 4096, PROT\_READ <unfinished ...>

[pid 5698] <... mprotect resumed>) = 0

[pid 5699] <... mprotect resumed>) = 0

[pid 5698] mprotect(0x7f7b49b23000, 8192, PROT\_READ <unfinished ...>

[pid 5699] mprotect(0x7f3af1536000, 8192, PROT\_READ <unfinished ...>

[pid 5698] <... mprotect resumed>) = 0

[pid 5699] <... mprotect resumed>) = 0

[pid 5698] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 5699] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 5698] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 5699] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 5698] munmap(0x7f7b49ae6000, 19871 <unfinished ...>

[pid 5699] munmap(0x7f3af14f9000, 19871 <unfinished ...>

[pid 5698] <... munmap resumed>) = 0

[pid 5699] <... munmap resumed>) = 0

[pid 5698] openat(AT\_FDCWD, "f1.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666 <unfinished ...>

[pid 5699] openat(AT\_FDCWD, "f2.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666 <unfinished ...>

[pid 5698] <... openat resumed>) = 4

[pid 5699] <... openat resumed>) = 6

[pid 5698] read(0, <unfinished ...>

[pid 5699] read(0, wraewe

<unfinished ...>

[pid 5697] <... read resumed>"wraewe\n", 255) = 7

[pid 5697] write(4, "wraewe\0", 7) = 7

[pid 5698] <... read resumed>"wraewe\0", 255) = 7

[pid 5697] read(0, <unfinished ...>

[pid 5698] write(4, "wrw", 3) = 3

[pid 5698] write(4, "\n", 1) = 1

[pid 5698] read(0, geawef

<unfinished ...>

[pid 5697] <... read resumed>"geawef\n", 255) = 7

[pid 5697] write(4, "geawef\0", 7) = 7

[pid 5698] <... read resumed>"geawef\0", 255) = 7

[pid 5697] read(0, <unfinished ...>

[pid 5698] write(4, "gwf", 3) = 3

[pid 5698] write(4, "\n", 1) = 1

[pid 5698] read(0, farwfew

<unfinished ...>

[pid 5697] <... read resumed>"farwfew\n", 255) = 8

[pid 5697] write(4, "farwfew\0", 8) = 8

[pid 5698] <... read resumed>"farwfew\0", 255) = 8

[pid 5697] read(0, <unfinished ...>

[pid 5698] write(4, "frwfw", 5) = 5

[pid 5698] write(4, "\n", 1) = 1

[pid 5698] read(0, 123dew

<unfinished ...>

[pid 5697] <... read resumed>"123dew\n", 255) = 7

[pid 5697] write(4, "123dew\0", 7) = 7

[pid 5698] <... read resumed>"123dew\0", 255) = 7

[pid 5697] read(0, <unfinished ...>

[pid 5698] write(4, "123dw", 5) = 5

[pid 5698] write(4, "\n", 1) = 1

[pid 5698] read(0, exit

<unfinished ...>

[pid 5697] <... read resumed>"exit\n", 255) = 5

[pid 5697] write(6, "exit\0", 5) = 5

[pid 5699] <... read resumed>"exit\0", 255) = 5

[pid 5697] write(4, "exit\0", 5 <unfinished ...>

[pid 5699] close(6 <unfinished ...>

[pid 5697] <... write resumed>) = 5

[pid 5699] <... close resumed>) = 0

[pid 5698] <... read resumed>"exit\0", 255) = 5

[pid 5697] close(4 <unfinished ...>

[pid 5699] exit\_group(0 <unfinished ...>

[pid 5697] <... close resumed>) = 0

[pid 5698] close(4 <unfinished ...>

[pid 5697] close(6 <unfinished ...>

[pid 5699] <... exit\_group resumed>) = ?

[pid 5697] <... close resumed>) = 0

[pid 5699] +++ exited with 0 +++

[pid 5698] <... close resumed>) = 0

[pid 5697] --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=5699, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

[pid 5698] exit\_group(0 <unfinished ...>

[pid 5697] wait4(5698, <unfinished ...>

[pid 5698] <... exit\_group resumed>) = ?

[pid 5698] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 5698

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=5698, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=1 /\* 0.01 s \*/} ---

wait4(5699, NULL, 0, NULL) = 5699

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Выполнение лабораторной работы оказалось увлекательным занятием. В ходе работы я освоил использование различных системных вызовов, а также научился организовывать обмен данными между процессами с помощью каналов. Однако столкнулся с некоторыми сложностями, связанными с обработкой ошибок системных вызовов, что потребовало дополнительного внимания и усилий для корректной реализации программы.