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-213БВ-24

Студент: Месропян А.Э.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 04.10.25

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 их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод. Вариант 17) 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

Общий метод: Разделение задач (фильтрация и обработка строк) между родительским и двумя дочерними процессами, используя неименованные каналы (pipes) для передачи данных.

Использованные системные вызовы:

1. `pid_t fork()`: Создание дочернего процесса.
2. `int execlp(const char *file, const char *arg, ...)`: Замена образа памяти дочернего процесса на программу `./lab1/child_run`.
3. `pid_t waitpid(pid_t pid, int *status, int options)`: Ожидание завершения дочернего процесса.
4. `int pipe(int pipefd[2])`: Создание неименованного канала для передачи данных между процессами.
5. `int dup2(int oldfd, int newfd)`: Переназначение файлового дескриптора (перенаправление конца чтения канала на `STDIN_FILENO` в дочерних процессах).
6. `FILE* fopen(const char *filename, const char *mode)`: Открытие/создание файла в дочернем процессе.
7. `int fclose(FILE *stream)`: Закрытие файла в дочернем процессе.
8. `ssize_t read(int fd, void *buf, size_t count)`: Чтение имени файла из `STDIN_FILENO` (в родительском процессе, используя низкоуровневый ввод).
9. `ssize_t write(int fd, const void *buf, size_t count)`: Вывод сообщений и запись данных в конвейеры.
10. `int close(int fd)`: Закрыть файловый дескриптор канала.

Алгоритм работы программы:

Инициализация

- Запрос и считывание у пользователя имен **двух выходных файлов** (file1_name, file2_name).

- Создание **двух неименованных каналов** (pipe1_fd, pipe2_fd) для асинхронной передачи данных.

Создание процессов

- **Родительский процесс** создает двух дочерних процессов (pid_1 и pid_2) через fork().
- **Дочерний процесс 1:**
  - Закрывает неиспользуемые концы каналов.
  - Перенаправляет конец чтения pipe1_fd[0] на свой стандартный ввод (STDIN_FILENO) с помощью dup2().
  - Запускает программу ./lab1/child_run (передавая ей имя file1_name) через execlp().
- **Дочерний процесс 2:**
  - Аналогично, перенаправляет конец чтения pipe2_fd[0] на свой STDIN_FILENO.
  - Запускает программу ./lab1/child_run (передавая ей имя file2_name) через execlp().
- **Родительский процесс** закрывает концы чтения каналов (pipe1_fd[0], pipe2_fd[0]).

Распределение данных (Фильтрация)

- Родительский процесс читает строки из своего стандартного ввода (stdin) с помощью fgets() до тех пор, пока не введена строка "QUIT".
- **Применяется правило фильтрации по длине (FILTER_LENGTH = 10):**
  - Если **длина строки >= 10**, она отправляется в **pipe1_fd** (для Child 1).
  - Если **длина строки <= 10**, она отправляется в **pipe2_fd** (для Child 2).
- Родительский процесс печатает в stdout информацию о том, в какой дочерний процесс была отправлена строка и её длину.

Обработка данных

- **Дочерние процессы (./lab1/child_run):**
  - Читают строки из своего перенаправленного стандартного ввода (stdin).
  - Вызывают функцию **remove_vowels(str)**, которая удаляет все латинские гласные (a, e, i, o, u, y).
  - Записывают обработанные строки в соответствующий выходной файл (указанный при запуске) с помощью fputs().

Завершение работы

- При вводе строки "QUIT" родительский процесс:
  - Закрывает концы записи обоих каналов (pipe1_fd[1], pipe2_fd[1]), что сигнализирует дочерним процессам об окончании потока данных (EOF).
  - Ожидает завершения обоих дочерних процессов (pid_1, pid_2) через waitpid().
- Программа завершается, возвращая STATUS_OK

Код программы

os_utils.h

```h
#ifndef OS_UTILS_H
#define OS_UTILS_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>

typedef enum {
    STATUS_OK = 0,
    PIPE_ERROR = 1,
    FORK_ERROR = 2,
    EXEC_ERROR = 3,
    INVALID_INPUT = 4,
    FILE_OPEN_ERROR = 5,
    IO_ERROR = 6
} StatusCode;

void remove_vowels(char* str);

#endif
```

parent.c

```c
#include "os_utils.h"
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

#define MAX_LINE_LENGTH 1024
#define MAX_FILENAME_LENGTH 256
#define FILTER_LENGTH 10

void start_child_process(int read_fd, const char* child_program, const char* output_name) {
    if (dup2(read_fd, STDIN_FILENO) == -1) {
        perror("Ошибка dup2");
        exit(IO_ERROR);
    }
    close(read_fd);
    execlp(child_program, child_program, (char* )output_name, (char* )NULL);
    perror("Ошибка execlp");
    exit(EXEC_ERROR);
}

int main() {
    int pipe1_fd[2];
    int pipe2_fd[2];
    pid_t pid_1, pid_2;
    char file1_name[MAX_FILENAME_LENGTH];
    char file2_name[MAX_FILENAME_LENGTH];
    char temp_buffer[MAX_LINE_LENGTH];

    StatusCode status = STATUS_OK;
    const char* str1 = "Введите имя файла для child 1: ";
    write(STDOUT_FILENO, "Введите имя файла для child 1: ", strlen(str1));
    fflush(stdout);

    ssize_t bytes_read = read(STDIN_FILENO, file1_name, MAX_FILENAME_LENGTH - 1);
    if (bytes_read <= 0) {
        return INVALID_INPUT;
    }
    if (file1_name[bytes_read - 1] == '\n') {
        file1_name[bytes_read - 1] = '\0';
    } else {
        file1_name[bytes_read] = '\0';
    }
```

```
const char* str2 = "Введите имя файла для child 2: ";
write(STDOUT_FILENO, "Введите имя файла для child 2: \n", strlen(str2));
fflush(stdout);

bytes_read = read(STDIN_FILENO, file2_name, MAX_FILENAME_LENGTH - 1);
if (bytes_read <= 0) {
    return INVALID_INPUT;
}
if (file2_name[bytes_read - 1] == '\n') {
    file2_name[bytes_read - 1] = '\0';
} else {
    file2_name[bytes_read] = '\0';
}

if (pipe(pipe1_fd) == -1 || pipe(pipe2_fd) == -1) {
    perror("Ошибка pipe");
    return PIPE_ERROR;
}

pid_1 = fork();
if (pid_1 == -1) {
    perror("Ошибка fork 1");
    status = FORK_ERROR;
    goto cleanup_pipes;
}

if (pid_1 == 0) {
    close(pipe1_fd[1]);
    close(pipe2_fd[0]);
    close(pipe2_fd[1]);
    start_child_process(pipe1_fd[0], "./lab1/child_run", file1_name);
}

pid_2 = fork();
if (pid_2 == -1) {
    perror("Ошибка fork 2");
    status = FORK_ERROR;
    goto cleanup_pipes;
}

if (pid_2 == 0) {
    close(pipe2_fd[1]);
    close(pipe1_fd[0]);
    close(pipe1_fd[1]);
    start_child_process(pipe2_fd[0], "./lab1/child_run", file2_name);
```

```
close(pipe1_fd[0]);
close(pipe2_fd[0]);
const char* par_start = "Родительский процесс начался \n";
write(STDOUT_FILENO, "Родительский процесс начался \n", strlen(par_start));
const char* input_str = "Введите строчки \n";
write(STDOUT_FILENO, "Введите строчки \n", strlen(input_str));
fflush(stdout);

char buffer[MAX_LINE_LENGTH];
while (fgets(buffer, sizeof(buffer), stdin) != NULL) {
    int len = strlen(buffer);
    if (len > 0 && buffer[len - 1] == '\n') {
        buffer[len - 1] = '\0';
        len--;
    }
    if (strcmp(buffer, "QUIT") == 0) {
        break;
    }
    int write_fd = -1;

    if (len > FILTER_LENGTH) {
        write_fd = pipe2_fd[1];
        int n = snprintf(temp_buffer, sizeof(temp_buffer), "Длина Child 2: %d\n", len);
        write(STDOUT_FILENO, temp_buffer, n);
    } else {
        write_fd = pipe1_fd[1];
        int n = snprintf(temp_buffer, sizeof(temp_buffer), "Длина Child 1: %d\n", len);
        write(STDOUT_FILENO, temp_buffer, n);
    }
    fflush(stdout);

    if (len < sizeof(buffer) - 1) {
        buffer[len] = '\n';
        buffer[len + 1] = '\0';
        len++;
    }

    if (write(write_fd, buffer, len) != len) {
        perror("Ошибка с pipe");
        return PIPE_ERROR;
    }
}

close (pipe1_fd[1]);
close(pipe2_fd[1]);
```

```
    waitpid(pid_1, NULL, 0);
    waitpid(pid_2, NULL, 0);
    const char* end_msg = "Родительский и детский процесс успешно завершены\n";
    write(STDOUT_FILENO, "Родительский и детский процесс успешно завершены\n", strlen(end_msg));
    fflush(stdout);

    return STATUS_OK;

cleanup_pipes:
    close(pipe1_fd[0]); close(pipe1_fd[1]);
    close(pipe2_fd[0]); close(pipe2_fd[1]);
    return status;
}
```

child.c

```
#include "os_utils.h"
#include <ctype.h>

void remove_vowels(char* str) {
    if (!str) {
        return;
    }
    int write_idx = 0;
    for (int read_idx = 0; str[read_idx] != '\0'; read_idx++) {
        char c = str[read_idx];
        char lower_c = tolower((unsigned char) c);
        if (lower_c != 'a' && lower_c != 'e' && lower_c != 'i' && lower_c != 'o' && lower_c != 'u' && lower_c != 'y') {
            str[write_idx++] = c;
        }
    }
    str[write_idx] = '\0';
}


int main(int argc, char* argv[]) {
    if (argc != 2) {
        return INVALID_INPUT;
    }
    const char* output_name = argv[1];
    FILE* output = NULL;
    char buffer[1024];

    output = fopen(output_name, "w");
    if (output == NULL) {
        return FILE_OPEN_ERROR;
    }
    while (fgets(buffer, sizeof(buffer), stdin) != NULL) {
        remove_vowels(buffer);
        if (fputs(buffer, output) == EOF) {
            return IO_ERROR;
        }
    }
    fclose(output);
    return STATUS_OK;
}
```

Протокол работы программы

Тестирование:

```
armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ ./lab1/parent_run
Введите имя файла для child 1: short_output.txt
Введите имя файла для child 2: long_output.txt
Родительский процесс начался
Введите строчки
Hello, Programming!
Длина Child 2: 19
WOWOWOOW GJFKDJFHDJKJHGFDHJDKFJHDJKS
Длина Child 2: 36
AAAAAhjf
Длина Child 1: 8
jggjgjgjgjaaaa
Длина Child 2: 14
QUIT
Родительский и детский процесс успешно завершены
armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ cat short_output.txt
hjf
armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ cat long_output.txt
Hll, Prgrmmng!
WWWW GJFKDJFHDJKJHGFDHJDKFJHDJKS
jggjgjgjgj
```

Strace:

```
armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ strace -f ./lab1/parent_run
execve("./lab1/parent_run", ["./lab1/parent_run"], 0x7ffe821b20c8 /* 37 vars */) = 0
brk(NULL)                               = 0x5cd634c18000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffccfcd3e70) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70f734b8f000
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=17368, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 17368, PROT_READ, MAP_PRIVATE, 3, 0) = 0x70f734b8a000
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\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 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) = 0x70f734800000
mprotect(0x70f734828000, 2023424, PROT_NONE) = 0
mmap(0x70f734828000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x70f734828000
mmap(0x70f7349bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x70f7349bd000
mmap(0x70f734a16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x70f734a16000
mmap(0x70f734a1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x70f734a1c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70f734b87000
arch_prctl(ARCH_SET_FS, 0x70f734b87740) = 0
set_tid_address(0x70f734b87a10)         = 22933
set_robust_list(0x70f734b87a20, 24)     = 0
rseq(0x70f734b880e0, 0x20, 0, 0x53053053) = 0
mprotect(0x70f734a16000, 16384, PROT_READ) = 0
mprotect(0x5cd623298000, 4096, PROT_READ) = 0
mprotect(0x70f734bc9000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x70f734b8a000, 17368)           = 0
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 49Введите имя файла для child 1: ) = 49
read(0, short_output.txt
"short_output.txt\n", 255)      = 17
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 49Введите имя файла для child 2: ) = 49
read(0, long_output.txt
"long_output.txt\n", 255)       = 16
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 23211 attached
, child_tidptr=0x70f734b87a10) = 23211
[pid 23211] set_robust_list(0x70f734b87a20, 24 <unfinished ...>
```

```
[pid 22933] close(3 <unfinished ...>
[pid 23211] close(5 <unfinished ...>
[pid 22933] <... close resumed>)        = 0
[pid 23212] <... set_robust_list resumed>) = 0
[pid 22933] close(5 <unfinished ...>
[pid 23211] <... close resumed>)        = 0
[pid 22933] <... close resumed>)        = 0
[pid 23212] close(6 <unfinished ...>
[pid 23211] close(6 <unfinished ...>
[pid 22933] write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271 \320\277\321\200\320\276\321"..., 56 <unfinished ...>
Родительский процесс начался
[pid 23212] <... close resumed>)        = 0
[pid 22933] <... write resumed>)        = 56
[pid 23211] <... close resumed>)        = 0
[pid 22933] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\201\321\202\321\200\320\276\321\207\320\272\320\270 \n", 31 <unfinished ...>
Введите строчки
[pid 23212] close(3 <unfinished ...>
[pid 22933] <... write resumed>)        = 31
[pid 23211] dup2(3, 0 <unfinished ...>
[pid 23212] <... close resumed>)        = 0
[pid 22933] newfstatat(0, "",  <unfinished ...>
[pid 23211] <... dup2 resumed>)         = 0
[pid 22933] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x6), ...}, AT_EMPTY_PATH) = 0
[pid 23212] close(4 <unfinished ...>
[pid 22933] getrandom( <unfinished ...>
[pid 23211] close(3 <unfinished ...>
[pid 22933] <... getrandom resumed>"\xdd\x6d\x22\x3d\x9d\xe0\xbf\x72", 8, GRND_NONBLOCK) = 8
[pid 23212] <... close resumed>)        = 0
[pid 22933] brk(NULL <unfinished ...>
[pid 23211] <... close resumed>)        = 0
[pid 22933] <... brk resumed>)          = 0x5cd634c18000
[pid 23212] dup2(5, 0 <unfinished ...>
[pid 22933] brk(0x5cd634c39000 <unfinished ...>
[pid 23211] execve("./lab1/child_run", ["./lab1/child_run", "short_output.txt"], 0x7ffccfcd4048 /* 37 vars */ <unfinished ...>
[pid 22933] <... brk resumed>)          = 0x5cd634c39000
[pid 23212] <... dup2 resumed>)         = 0
[pid 22933] read(0,  <unfinished ...>
[pid 23212] close(5)                    = 0
[pid 23212] execve("./lab1/child_run", ["./lab1/child_run", "long_output.txt"], 0x7ffccfcd4048 /* 37 vars */ <unfinished ...>
[pid 23211] <... execve resumed>)       = 0
[pid 23211] brk(NULL <unfinished ...>
[pid 23212] <... execve resumed>)       = 0
```

```
[pid 23211] <... brk resumed>)          = 0x57fd61572000
[pid 23212] brk(NULL <unfinished ...>
[pid 23211] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcb01ada00 <unfinished ...>
[pid 23212] <... brk resumed>)          = 0x55682a167000
[pid 23211] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)
[pid 23212] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdbddf2ce0) = -1 EINVAL (Invalid argument)
[pid 23211] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 23212] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 23211] <... mmap resumed>)         = 0x706f54297000
[pid 23212] <... mmap resumed>)         = 0x7ac9d2067000
[pid 23211] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 23212] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 23211] <... access resumed>)       = -1 ENOENT (No such file or directory)
[pid 23212] <... access resumed>)       = -1 ENOENT (No such file or directory)
[pid 23211] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 23212] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 23211] <... openat resumed>)       = 3
[pid 23212] <... openat resumed>)       = 3
[pid 23211] newfstatat(3, "",  <unfinished ...>
[pid 23212] newfstatat(3, "",  <unfinished ...>
[pid 23211] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=17368, ...}, AT_EMPTY_PATH) = 0
[pid 23212] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=17368, ...}, AT_EMPTY_PATH) = 0
[pid 23211] mmap(NULL, 17368, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 23212] mmap(NULL, 17368, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 23211] <... mmap resumed>)         = 0x706f54292000
[pid 23212] <... mmap resumed>)         = 0x7ac9d2062000
[pid 23211] close(3 <unfinished ...>
[pid 23212] close(3 <unfinished ...>
[pid 23211] <... close resumed>)        = 0
[pid 23212] <... close resumed>)        = 0
[pid 23211] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 23212] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 23211] <... openat resumed>)       = 3
[pid 23212] <... openat resumed>)       = 3
[pid 23211] read(3,  <unfinished ...>
[pid 23212] read(3,  <unfinished ...>
[pid 23211] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
[pid 23212] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
[pid 23211] pread64(3,  <unfinished ...>
[pid 23212] pread64(3,  <unfinished ...>
[pid 23211] <... 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 23212] <... 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 23211] pread64(3,  <unfinished ...>
[pid 23212] pread64(3,  <unfinished ...>
```

```
[pid 23212] pread64(3,  <unfinished ...>
[pid 23211] <... 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 23212] <... 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 23211] pread64(3,  <unfinished ...>
[pid 23212] pread64(3,  <unfinished ...>
[pid 23211] <... pread64 resumed>"\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
[pid 23212] <... pread64 resumed>"\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
[pid 23211] pread64(3,  <unfinished ...>
[pid 23212] pread64(3,  <unfinished ...>
[pid 23211] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68
[pid 23212] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68
[pid 23211] newfstatat(3, "",  <unfinished ...>
[pid 23212] newfstatat(3, "",  <unfinished ...>
[pid 23211] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid 23212] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid 23211] pread64(3,  <unfinished ...>
[pid 23212] 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 23211] <... 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 23212] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid 23211] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid 23212] <... mmap resumed>)         = 0x7ac9d1e00000
[pid 23211] <... mmap resumed>)         = 0x706f54000000
[pid 23212] mprotect(0x7ac9d1e28000, 2023424, PROT_NONE <unfinished ...>
[pid 23211] mprotect(0x706f54028000, 2023424, PROT_NONE <unfinished ...>
[pid 23212] <... mprotect resumed>)     = 0
[pid 23211] <... mprotect resumed>)     = 0
[pid 23212] mmap(0x7ac9d1e28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
[pid 23211] mmap(0x706f54028000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
[pid 23212] <... mmap resumed>)         = 0x7ac9d1e28000
[pid 23211] <... mmap resumed>)         = 0x706f54028000
[pid 23212] mmap(0x7ac9d1fbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>
[pid 23211] mmap(0x706f541bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>
[pid 23212] <... mmap resumed>)         = 0x7ac9d1fbd000
[pid 23211] <... mmap resumed>)         = 0x706f541bd000
[pid 23212] mmap(0x7ac9d2016000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000 <unfinished ...>
[pid 23211] mmap(0x706f54216000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000 <unfinished ...>
[pid 23212] <... mmap resumed>)         = 0x7ac9d2016000
[pid 23211] <... mmap resumed>)         = 0x706f54216000
[pid 23212] mmap(0x7ac9d201c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 23211] mmap(0x706f5421c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 23212] <... mmap resumed>)         = 0x7ac9d201c000
[pid 23211] <... mmap resumed>)         = 0x706f5421c000
[pid 23212] close(3 <unfinished ...>
[pid 23211] close(3 <unfinished ...>
```

```
[pid 22933] write(1, "\320\224\320\273\320\270\320\275\320\260 Child 1: 6\n", 22Длина Child 1: 6
) = 22
[pid 22933] write(4, "LONGGG\n", 7)     = 7
[pid 23211] <... read resumed>"LONGGG\n", 4096) = 7
[pid 22933] read(0,  <unfinished ...>
[pid 23211] read(0, GJFKDLKJHGFBHJK
 <unfinished ...>
[pid 22933] <... read resumed>"GJFKDLKJHGFBHJK\n", 1024) = 16
[pid 22933] write(1, "\320\224\320\273\320\270\320\275\320\260 Child 2: 15\n", 23Длина Child 2: 15
) = 23
[pid 22933] write(6, "GJFKDLKJHGFBHJK\n", 16) = 16
[pid 23212] <... read resumed>"GJFKDLKJHGFBHJK\n", 4096) = 16
[pid 22933] read(0,  <unfinished ...>
[pid 23212] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=0, ...}, AT_EMPTY_PATH) = 0
[pid 23212] read(0, QUIT
 <unfinished ...>
[pid 22933] <... read resumed>"QUIT\n", 1024) = 5
[pid 22933] close(4)                    = 0
[pid 23211] <... read resumed>"", 4096) = 0
[pid 22933] close(6 <unfinished ...>
[pid 23211] write(3, "gjvfkc\nLNGGG\n", 13 <unfinished ...>
[pid 22933] <... close resumed>)        = 0
[pid 23212] <... read resumed>"", 4096) = 0
[pid 22933] wait4(23211,  <unfinished ...>
[pid 23212] write(3, "GJFKDLKJHGFBHJK\n", 16 <unfinished ...>
[pid 23211] <... write resumed>)        = 13
[pid 23211] close(3 <unfinished ...>
[pid 23212] <... write resumed>)        = 16
[pid 23211] <... close resumed>)        = 0
[pid 23212] close(3 <unfinished ...>
[pid 23211] exit_group(0 <unfinished ...>
[pid 23212] <... close resumed>)        = 0
[pid 23211] <... exit_group resumed>)   = ?
[pid 23212] exit_group(0)               = ?
[pid 23211] +++ exited with 0 +++
[pid 22933] <... wait4 resumed>NULL, 0, NULL) = 23211
[pid 23212] +++ exited with 0 +++
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=23211, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
wait4(23212, NULL, 0, NULL)             = 23212
write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271 \320\270 \320\264\320\265"..., 92Родительский и детский процесс успешно завершены
) = 92
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были успешно освоены принципы организации межпроцессного взаимодействия с использованием неименованных каналов (pipes). Основные трудности возникли при корректной настройке перенаправления стандартного ввода дочерних процессов (dup2) и обеспечении безошибочной маршрутизации данных от родителя к соответствующему дочернему процессу на основе логического правила (фильтрация по длине строки). В процессе работы был детально изучен и применён набор системных вызовов: fork() для создания процессов, pipe() для организации каналов, dup2() для переназначения дескрипторов и execlp() для запуска внешнего кода. Это позволило получить глубокое понимание механизмов параллельной обработки данных в операционной системе и корректного управления жизненным циклом процессов. Полученные навыки будут полезны для выполнения последующих, более сложных задач, требующих асинхронной работы процессов.