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-213Б-23

Студент: Пономарев А.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 18.10.24

Москва, 2024

Постановка задачи

Вариант 13.

Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода. Child1 переводит строки в нижний регистр. Child2 превращает все пробельные символы в символ «_».

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void) — используется для создания дочернего процесса.
- int pipe(int fd) — создает канал для однонаправленной связи между процессами. fd[0] используется для чтения из канала, а fd[1] — для записи в него.
- ssize_t write(int fd, const void buf, size_t count) — записывает данные из буфера buf в файл, связанный с файловым дескриптором fd, в количестве байтов, указанном в count.
- ssize_t read(int fd, void buf, size_t count) — читает данные из файла или канала, связанного с файловым дескриптором fd, в буфер buf в количестве байтов, указанном в count.
- int execv(const char path, char const argv[]) — заменяет текущий процесс новым процессом, запускающим указанную программу.
- int32_t open(const char*file, int oflag, …); – открывает файл и возвращает файловый дескриптор.
- int close(int fd) – закрывает файл.
- int dup2(int oldfd, int newfd) — дублирует файловый дескриптор oldfd, заменяя им дескриптор newfd. Перенаправление стандартного ввода дочернего процесса на канал.
- int wait(int status) — приостанавливает выполнение родительского процесса до завершения дочернего процесса.

Алгоритм решения:

Во время выполнения лабораторной работы я разрабатывал программу, в которой родительский процесс создает два дочерних процесса для обработки строк, получаемых от пользователя. Сначала я организовал механизм передачи данных между процессами, используя каналы (pipes), что позволило мне отправлять введенные строки в первый дочерний процесс, а затем получать обработанные данные от второго дочернего процесса.

Я создал два отдельных исполняемых файла для дочерних процессов: первый отвечает за преобразование строк в нижний регистр, а второй заменяет пробелы на символы подчеркивания. В родительском процессе я использовал функции fork() и execv() для создания и запуска дочерних процессов, а также перенаправил стандартные потоки ввода-вывода с помощью dup2(), чтобы установить каналы между процессами.

В процессе разработки я учел обработку ошибок: проверял результаты вызовов функций, таких как pipe(), fork() и execv(), и выводил соответствующие сообщения об ошибках в стандартный поток ошибок. Я реализовал цикл, в котором родительский процесс считывал ввод от пользователя, отправлял его через pipe1, а затем ожидал результаты от второго дочернего процесса через pipe3. После получения результатов я выводил их на экран.

Я также добавил возможность завершить ввод, нажав клавишу Enter или сочетание клавиш CTRL+D. В завершение работы программы я закрыл открытые каналы и дождался завершения обоих дочерних процессов с помощью wait().

Код программы

server.c

```c
#include <stdint.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
#include <string.h>

static char CHILD1_PROGRAM_NAME[] = "./child1";
static char CHILD2_PROGRAM_NAME[] = "./child2";

int main(int argc, char **argv) {
    if (argc != 1) {
        char msg[] =  "usage: ./{filename}\n";
        write(STDOUT_FILENO, msg, strlen(msg));
        exit(EXIT_SUCCESS);
    }

    // Get full path to the directory, where program resides
```

```
    char progpath[1024];
    {
        // Read full program path, including its name
        ssize_t len = readlink("/proc/self/exe", progpath,
                               sizeof(progpath) - 1);
        if (len == -1) {
            const char msg[] = "error: failed to read full program path\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }

        // Trim the path to first slash from the end
        while (progpath[len] != '/')
            --len;

        progpath[len + 1] = '\0';
    }

    // Open pipe
    int pipe1[2], pipe2[2], pipe3[2];
    if (pipe(pipe1) == -1 || pipe(pipe2) == -1 || pipe(pipe3) == -1) {
        const char msg[] = "error: failed to create pipe\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    const pid_t child1 = fork();

    switch (child1) {
        case -1: {
            const char msg[] = "error: failed to spawn new process\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        } break;

        case 0: {
            dup2(pipe1[STDIN_FILENO], STDIN_FILENO);
            dup2(pipe2[STDOUT_FILENO], STDOUT_FILENO);

            close(pipe1[STDOUT_FILENO]);
            close(pipe2[STDIN_FILENO]);
            close(pipe3[STDIN_FILENO]);
            close(pipe3[STDOUT_FILENO]);

            {
                char *const args[] = {CHILD1_PROGRAM_NAME, NULL};

                int32_t status = execv(CHILD1_PROGRAM_NAME, args);

                if (status == -1) {
                    const char msg[] = "error: failed to exec into new exectuable
image\n";

                    write(STDERR_FILENO, msg, sizeof(msg));
                    exit(EXIT_FAILURE);
                }
            }
        } break;
    }

    const pid_t child2 = fork();
    switch (child2) {
        case -1: {
            const char msg[] = "error: failed to spawn new process\n";
```

```
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        } break;

        case 0: {

            dup2(pipe2[STDIN_FILENO], STDIN_FILENO);
            dup2(pipe3[STDOUT_FILENO], STDOUT_FILENO);

            close(pipe1[STDIN_FILENO]);
            close(pipe1[STDOUT_FILENO]);

            close(pipe2[STDOUT_FILENO]);
            close(pipe3[STDIN_FILENO]);


            {
                char *const args[] = {CHILD2_PROGRAM_NAME, NULL};

                int32_t status = execv(CHILD2_PROGRAM_NAME, args);

                if (status == -1) {
                    const char msg[] = "error: failed to exec into new exectuable
image\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                    exit(EXIT_FAILURE);
                }
            }
        } break;
    }
    // closing useless
    close(pipe1[0]);
    close(pipe2[0]);
    close(pipe3[1]);

    ssize_t bytes;
    char buf[1024];

    char msg_of_hint[] = "Enter your string or (Enter / CTRL + D) for stop: \n";
    int len_of_msg_of_hint = strlen(msg_of_hint);
    write(STDOUT_FILENO, msg_of_hint, len_of_msg_of_hint);
    while (bytes = read(STDIN_FILENO, buf, sizeof(buf))) {
        if (bytes < 0) {
                const char msg[] = "error: failed to read from stdin\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
         } else if (buf[0] == '\n') {
                break;
         }
        buf[bytes] = '\0';
        // Write into pipe1 for child1 input
        write(pipe1[1], buf, strlen(buf));

        // read from pipe3
        char result[1024];
        ssize_t bytes_read = read(pipe3[0], result, sizeof(result) - 1);
        if (bytes_read > 0) {
            result[bytes_read] = '\0';
            char msg[] = "Processed result: ";
            write(STDOUT_FILENO, msg, strlen(msg));
            write(STDOUT_FILENO, result, bytes_read - 1);
            write(STDOUT_FILENO, "\n\n", 2);
            write(STDOUT_FILENO, msg_of_hint, len_of_msg_of_hint);
```

```
        }

    }
    close(pipe1[1]);
    close(pipe3[0]);
    close(pipe2[1]);

    wait(NULL);
    wait(NULL);
    return 0;
}
```

child1.c

```
#include <ctype.h>
#include <unistd.h>
#include <string.h>

int main() {
    char input[1024];
    ssize_t bytes_read;

    while ((bytes_read = read(STDIN_FILENO, input, sizeof(input))) > 0) {
        input[bytes_read] = '\0';

        for (int i = 0; i < bytes_read; i++) {
            input[i] = tolower(input[i]);
        }
        write(STDOUT_FILENO, input, bytes_read);
    }
    return 0;
}
```

child2.c:

```
#include <unistd.h>
#include <string.h>

int main() {
    char input[1024];
    ssize_t bytes_read;

    while ((bytes_read = read(STDIN_FILENO, input, sizeof(input))) > 0) {
        input[bytes_read] = '\0';
        for (int i = 0; i < bytes_read; i++) {
            if (input[i] == ' ') {
                input[i] = '_';
            }
        }
        write(STDOUT_FILENO, input, bytes_read);
    }
    return 0;
}
```

Протокол работы программы

Тестирование:

./server

Enter your string or (Enter / CTRL + D) for stop:

HHE LLL ll 23 4 4444444

Processed result: hhe_lll_ll_23_4_4444444

Enter your string or (Enter / CTRL + D) for stop:

IT is Me MARRRRRio OOOOOO

Processed result: it_is_me_marrrrrio_oooooo___

Enter your string or (Enter / CTRL + D) for stop:

HELLO Brother11111!

Processed result: _____hello_brother11111!

Enter your string or (Enter / CTRL + D) for stop:

Strace:

artem@artem-VirtualBox:~/MAI_OS/lab01/src$ strace -f ./server

execve("./server", ["./server"], 0x7ffc99039ad8 /* 49 vars */) = 0

brk(NULL) = 0x562f8c688000

arch_prctl(0x3001 /* ARCH_??? */, 0x7fff79b2cf50) = -1 EINVAL (Недопустимый аргумент)

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=67464, ...}) = 0

mmap(NULL, 67464, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f4a4fa70000

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\300A\2\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68

fstat(3, {st_mode=S_IFREG|0755, st_size=2029592, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f4a4fa6e000

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\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68

mmap(NULL, 2037344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f4a4f87c000

mmap(0x7f4a4f89e000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7f4a4f89e000

mmap(0x7f4a4fa16000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19a000) = 0x7f4a4fa16000

mmap(0x7f4a4fa64000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f4a4fa64000

mmap(0x7f4a4fa6a000, 13920, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f4a4fa6a000

close(3) = 0

arch_prctl(ARCH_SET_FS, 0x7f4a4fa6f540) = 0

mprotect(0x7f4a4fa64000, 16384, PROT_READ) = 0

mprotect(0x562f8a7f2000, 4096, PROT_READ) = 0

mprotect(0x7f4a4faae000, 4096, PROT_READ) = 0

munmap(0x7f4a4fa70000, 67464) = 0

readlink("/proc/self/exe", "/home/artem/MAI_OS/lab01/src/ser"..., 1023) = 35

pipe([3, 4]) = 0

pipe([5, 6]) = 0

pipe([7, 8]) = 0

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 108786 attached

<unfinished ...>

[pid 108786] dup2(3, 0 <unfinished ...>

[pid 108785] <... clone resumed>, child_tidptr=0x7f4a4fa6f810) = 108786

[pid 108786] <... dup2 resumed>) = 0

[pid 108785] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f4a4fa6f810) = 108787

[pid 108785] close(3) = 0

[pid 108785] close(5) = 0

[pid 108785] close(8)   = 0

[pid 108785] write(1, "Enter your string or (Enter / CT"..., 51Enter your string or (Enter / CTRL + D) for stop:

) = 51

[pid 108785] read(0, strace: Process 108787 attached

 <unfinished ...>

[pid 108787] dup2(5, 0)   = 0

[pid 108787] dup2(8, 1)   = 1

[pid 108787] close(3)   = 0

[pid 108787] close(4 <unfinished ...>

[pid 108786] dup2(6, 1 <unfinished ...>

[pid 108787] <... close resumed>)   = 0

[pid 108787] close(6 <unfinished ...>

[pid 108786] <... dup2 resumed>)   = 1

[pid 108787] <... close resumed>)   = 0

[pid 108786] close(4 <unfinished ...>

[pid 108787] close(7 <unfinished ...>

[pid 108786] <... close resumed>)   = 0

[pid 108787] <... close resumed>)   = 0

[pid 108787] execve("./child2", ["./child2"], 0x7fff79b2d038 /* 49 vars */ <unfinished ...>

[pid 108786] close(5)   = 0

[pid 108787] <... execve resumed>)   = 0

[pid 108786] close(7 <unfinished ...>

[pid 108787] brk(NULL <unfinished ...>

[pid 108786] <... close resumed>)   = 0

[pid 108787] <... brk resumed>)   = 0x561f0d096000

[pid 108786] close(8 <unfinished ...>

[pid 108787] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc8a9ae8d0 <unfinished ...>

[pid 108786] <... close resumed>)   = 0

[pid 108787] <... arch_prctl resumed>) = -1 EINVAL (Недопустимый аргумент)

[pid 108786] execve("./child1", ["./child1"], 0x7fff79b2d038 /* 49 vars */ <unfinished ...>

[pid 108787] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)

[pid 108787] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 108786] <... execve resumed>) = 0

[pid 108787] <... openat resumed>) = 3

[pid 108786] brk(NULL <unfinished ...>

[pid 108787] fstat(3, <unfinished ...>

[pid 108786] <... brk resumed>) = 0x563c82dcf000

[pid 108787] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=67464, ...}) = 0

[pid 108787] mmap(NULL, 67464, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

[pid 108786] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe23c675b0 <unfinished ...>

[pid 108787] <... mmap resumed>) = 0x7ff1efb0a000

[pid 108787] close(3 <unfinished ...>

[pid 108786] <... arch_prctl resumed>) = -1 EINVAL (Недопустимый аргумент)

[pid 108787] <... close resumed>) = 0

[pid 108786] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 108787] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 108786] <... access resumed>) = -1 ENOENT (Нет такого файла или каталога)

[pid 108787] <... openat resumed>) = 3

[pid 108787] read(3, <unfinished ...>

[pid 108786] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 108787] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300A\2\0\0\0\0\0"..., 832) = 832

[pid 108786] <... openat resumed>) = 4

[pid 108787] pread64(3, <unfinished ...>

[pid 108786] fstat(4, <unfinished ...>

[pid 108787] <... 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 108786] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=67464, ...}) = 0

[pid 108787] pread64(3, <unfinished ...>

[pid 108786] mmap(NULL, 67464, PROT_READ, MAP_PRIVATE, 4, 0 <unfinished ...>

[pid 108787] <... pread64 resumed>"\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0", 32, 848) = 32

[pid 108786] <... mmap resumed>) = 0x7f276a2f8000

[pid 108787] pread64(3, <unfinished ...>

[pid 108786] close(4 <unfinished ...>

[pid 108787] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68

[pid 108787] fstat(3, <unfinished ...>

[pid 108786] <... close resumed>)   = 0

[pid 108787] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2029592, ...}) = 0

[pid 108787] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 108786] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 108787] <... mmap resumed>)   = 0x7ff1efb08000

[pid 108786] <... openat resumed>)   = 4

[pid 108787] pread64(3, <unfinished ...>

[pid 108786] read(4, <unfinished ...>

[pid 108787] <... 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 108786] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300A\2\0\0\0\0\0"..., 832) = 832

[pid 108787] pread64(3, <unfinished ...>

[pid 108786] pread64(4, <unfinished ...>

[pid 108787] <... pread64 resumed>"\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

[pid 108786] <... 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 108787] pread64(3, <unfinished ...>

[pid 108786] pread64(4, <unfinished ...>

[pid 108787] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68

[pid 108786] <... pread64 resumed>"\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

[pid 108787] mmap(NULL, 2037344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 108786] pread64(4, <unfinished ...>

[pid 108787] <... mmap resumed>)   = 0x7ff1ef916000

[pid 108787] mmap(0x7ff1ef938000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000 <unfinished ...>

[pid 108786] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68

[pid 108787] <... mmap resumed>) = 0x7ff1ef938000

[pid 108787] mmap(0x7ff1efab0000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19a000 <unfinished ...>

[pid 108786] fstat(4, <unfinished ...>

[pid 108787] <... mmap resumed>) = 0x7ff1efab0000

[pid 108786] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2029592, ...}) = 0

[pid 108787] mmap(0x7ff1efafe000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7ff1efafe000

[pid 108786] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 108787] mmap(0x7ff1efb04000, 13920, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 108786] <... mmap resumed>) = 0x7f276a2f6000

[pid 108787] <... mmap resumed>) = 0x7ff1efb04000

[pid 108786] 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 108786] pread64(4, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

[pid 108786] pread64(4, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68

[pid 108786] mmap(NULL, 2037344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) = 0x7f276a104000

[pid 108787] close(3) = 0

[pid 108787] arch_prctl(ARCH_SET_FS, 0x7ff1efb09540) = 0

[pid 108787] mprotect(0x7ff1efafe000, 16384, PROT_READ) = 0

[pid 108787] mprotect(0x561f0ca5a000, 4096, PROT_READ) = 0

[pid 108786] mmap(0x7f276a126000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x22000 <unfinished ...>

[pid 108787] mprotect(0x7ff1efb48000, 4096, PROT_READ <unfinished ...>

[pid 108786] <... mmap resumed>) = 0x7f276a126000

[pid 108787] <... mprotect resumed>) = 0

[pid 108786] mmap(0x7f276a29e000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x19a000) = 0x7f276a29e000

[pid 108787] munmap(0x7ff1efb0a000, 67464) = 0

[pid 108786] mmap(0x7f276a2ec000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1e7000) = 0x7f276a2ec000

[pid 108786] mmap(0x7f276a2f2000, 13920, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 108787] read(0, <unfinished ...>

[pid 108786] <... mmap resumed>)   = 0x7f276a2f2000

[pid 108786] close(4)   = 0

[pid 108786] arch_prctl(ARCH_SET_FS, 0x7f276a2f7540) = 0

[pid 108786] mprotect(0x7f276a2ec000, 16384, PROT_READ) = 0

[pid 108786] mprotect(0x563c817e4000, 4096, PROT_READ) = 0

[pid 108786] mprotect(0x7f276a336000, 4096, PROT_READ) = 0

[pid 108786] munmap(0x7f276a2f8000, 67464) = 0

[pid 108786] read(0, Hello MAMRIRIRrioroo

 <unfinished ...>

[pid 108785] <... read resumed>"Hello MAMRIRIRrioroo\n", 1024) = 21

[pid 108785] write(4, "Hello MAMRIRIRrioroo\n", 21) = 21

[pid 108785] read(7, <unfinished ...>

[pid 108786] <... read resumed>"Hello MAMRIRIRrioroo\n", 1024) = 21

[pid 108786] write(1, "hello mamrirrioroo\n", 21 <unfinished ...>

[pid 108787] <... read resumed>"hello mamrirrioroo\n", 1024) = 21

[pid 108786] <... write resumed>)   = 21

[pid 108787] write(1, "hello_mamrirrioroo\n", 21 <unfinished ...>

[pid 108785] <... read resumed>"hello_mamrirrioroo\n", 1023) = 21

[pid 108787] <... write resumed>)   = 21

[pid 108786] read(0, <unfinished ...>

[pid 108785] write(1, "Processed result: ", 18Processed result: <unfinished ...>

[pid 108787] read(0, <unfinished ...>

[pid 108785] <... write resumed>)   = 18

[pid 108785] write(1, "hello_mamrirrioroo", 20hello_mamrirrioroo) = 20

[pid 108785] write(1, "\n\n", 2

)   = 2

```
[pid 108785] write(1, "Enter your string or (Enter / CT"..., 51Enter your string or (Enter / CTRL + D) for stop:
) = 51
[pid 108785] read(0,
"\n", 1024)       = 1
[pid 108785] close(4)                = 0
[pid 108785] close(7)                = 0
[pid 108786] <... read resumed>"", 1024) = 0
[pid 108785] close(6)                = 0
[pid 108785] wait4(-1,  <unfinished ...>
[pid 108786] exit_group(0)           = ?
[pid 108787] <... read resumed>"", 1024) = 0
[pid 108786] +++ exited with 0 +++
[pid 108785] <... wait4 resumed>NULL, 0, NULL) = 108786
[pid 108787] exit_group(0 <unfinished ...>
[pid 108785] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=108786, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
[pid 108785] wait4(-1,  <unfinished ...>
[pid 108787] <... exit_group resumed>)  = ?
[pid 108787] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 108787
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=108787, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
exit_group(0)                        = ?
+++ exited with 0 +++
```

Вывод

Во время выполнения лабораторной работы я разработал программу, которая использует несколько процессов для обработки строк, вводимых пользователем. Основная сложность возникла из-за не закрытых каналов (pipes), что приводило к зависанию процессов: дочерние процессы не завершались, поскольку продолжали ждать ввода. Я исправил это, убедившись, что все ненужные дескрипторы закрыты после их использования. В будущем хотелось бы уделить больше времени отладке и тестированию процессов, чтобы избежать подобных проблем. В целом, работа была полезной и помогла мне лучше понять взаимодействие между процессами.