

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-216БВ-24

Студент: Сальманов Э.Р.

Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_

Дата: 30.09.25

Москва, 2025

# Постановка задачи

## Вариант 22.

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.

Правило фильтрации: с вероятностью 80% строки отправляются в *pipe1*, иначе в *pipe2*. Дочерние процессы инвертируют строки.

## Общий метод и алгоритм решения

Использованные системные вызовы:

- *pid\_t fork(void)*; - создает дочерний процесс.
- *int pipe(int \*fd)*; - создаёт канал для межпроцессорного взаимодействия.
- *int dup2(int oldfd, int newfd)*; - создаёт копию старого файлового дескриптора в заданном новом дескрипторе.
- *int execl(const char \*path, const char \*arg, ...)*; - заменяет образ текущего процесса на указанный, принимая аргументы в качестве списка.
- *pid\_t waitpid(pid\_t pid, int \*status, int options)*; - ожидает изменения состояния входного процесса с заданными опциями.
- *int open(const char \*pathname, int flags, mode\_t mode)*; - открывает файл по входному пути с заданными флагами и правами доступа.
- *int close(int fd)*; - закрывает файловый дескриптор.
- *ssize\_t read(int fd, void \*buf, size\_t count)*; - читает данные из файлового дескриптора и записывает их в буфер.
- *ssize\_t write(int fd, const void \*buf, size\_t count)*; - записывает данные из буфера в файловый дескриптор.

В рамках лабораторной работы создавался родительский процесс, считывающий имена файлов, которые далее передаются как аргументы командной строки дочерним процессам, созданным с помощью *fork()* и *execl()*. Межпроцессорное взаимодействие осуществляется посредством каналов, созданных системным вызовом *pipe()*. У дочерних процессов с помощью вызова *dup2()* переопределяется стандартный поток ввода, который приходит из канала.

Родительский процесс построчно считывает со стандартного потока ввода текст с помощью метода *read()*, который затем случайным образом передаётся одному из дочерних процессов в соответствии с требованиями, описанными в условиях.

Дочерний процесс получает строку со стандартного потока ввода, инвертирует её и с помощью метода *write()* записывают в файл, который соответствует этому процессу.

При вводе слова *Stop* родительский процесс отправляет сообщение о завершении работы дочерним процессам и с помощью *waitpid()* ожидает их завершения.

Все выделенные ресурсы корректно освобождаются при завершении работы программы.

## Код программы

### parent.c

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

#include <lab1/messages.h>

#define MAX_BUFFER_SIZE 4096
#define PROBABILITY 80

ssize_t ReadFilename(char *buffer) {
    ssize_t size = read(STDIN_FILENO, buffer, MAX_BUFFER_SIZE);

    if (size == -1) {
        char message[] = "[ERROR] Can't read filename for process!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return size;
    }

    buffer[size - 1] = '\0';
    --size;

    return size;
}

int32_t ProcessInput(int child1_write_pipe,
                    int child2_write_pipe) {
    char buffer[MAX_BUFFER_SIZE];
    ssize_t bytes_read = 0;

    while ((bytes_read = read(STDIN_FILENO, buffer, MAX_BUFFER_SIZE)) > 0) {
        if (strncmp(buffer, EXIT_MESSAGE, sizeof(EXIT_MESSAGE) - 1) == 0) {
            break;
        }

        int pipe_to_send;

        if (rand() % 100 < PROBABILITY) {
            pipe_to_send = child1_write_pipe;
        } else {
            pipe_to_send = child2_write_pipe;
        }
    }
}
```

```

        ssize_t written = write(pipe_to_send, buffer, bytes_read);

        if (written != bytes_read) {
process!\n";
            char message[] = "[ERROR] Can't send all text from parent to child";
            write(STDOUT_FILENO, message, sizeof(message));

            break;
        }
    }

    return 0;
}

int main(void) {
    char filename1[MAX_BUFFER_SIZE], filename2[MAX_BUFFER_SIZE];

    if (ReadFilename(filename1) == -1) {
        return 1;
    }

    if (ReadFilename(filename2) == -1) {
        return 1;
    }

    int child1_pipes[2], child2_pipes[2];

    if (pipe(child1_pipes) == -1) {
        char message[] = "[ERROR] Can't create pipes for first child process!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return 1;
    }

    if (pipe(child2_pipes) == -1) {
        close(child1_pipes[0]);
        close(child1_pipes[1]);

        char message[] = "[ERROR] Can't create pipes for second child process!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return 1;
    }

    pid_t pid1, pid2;

    pid1 = fork();

```

```

if (pid1 == -1) {
    close(child1_pipes[0]);
    close(child1_pipes[1]);
    close(child2_pipes[0]);
    close(child2_pipes[1]);

    char message[] = "[ERROR] Can't create first child process!\n";
    write(STDOUT_FILENO, message, sizeof(message));

    return 1;
}

// child1
if (pid1 == 0) {
    dup2(child1_pipes[0], STDIN_FILENO);
    execl("./child", filename1, NULL);

    return 0;
}

pid2 = fork();

if (pid2 == -1) {
    write(child1_pipes[1], EXIT_MESSAGE, sizeof(EXIT_MESSAGE));

    waitpid(pid1, NULL, 0);

    close(child1_pipes[0]);
    close(child1_pipes[1]);
    close(child2_pipes[0]);
    close(child2_pipes[1]);

    char message[] = "[ERROR] Can't create second child process!\n";
    write(STDOUT_FILENO, message, sizeof(message));

    return 1;
}

// child2
if (pid2 == 0) {
    dup2(child2_pipes[0], STDIN_FILENO);
    execl("./child", filename2, NULL);

    return 0;
}

// parent
int32_t result = ProcessInput(child1_pipes[1],
                              child2_pipes[1]);

```

```

write(child1_pipes[1], EXIT_MESSAGE, sizeof(EXIT_MESSAGE));
write(child2_pipes[1], EXIT_MESSAGE, sizeof(EXIT_MESSAGE));

waitpid(pid1, NULL, 0);
waitpid(pid2, NULL, 0);

close(child1_pipes[0]);
close(child1_pipes[1]);
close(child2_pipes[0]);
close(child2_pipes[1]);

return result;
}

```

### **child.c**

```

#include <fcntl.h>
#include <stdint.h>
#include <string.h>
#include <unistd.h>

#include <lab1/messages.h>

#define MAX_BUFFER_SIZE 4096

void InvertString(char *buffer,
                  uint64_t size) {
    for (uint64_t i = 0; i < size / 2; ++i) {
        char tmp = buffer[size - i - 1];

        buffer[size - i - 1] = buffer[i];
        buffer[i] = tmp;
    }
}

int main(int argc,
         char **argv) {
    if (argc != 1) {
        char message[] = "[ERROR] Child process must start with output filename argument!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return 1;
    }

    int file = open(argv[0],
                    O_RDWR | O_CREAT,
                    S_IRWXU);

```

```

if (file == -1) {
    char message[] = "[ERROR] Can't create or open file in child process!\n";
    write(STDOUT_FILENO, message, sizeof(message));

    return 1;
}

char buffer[MAX_BUFFER_SIZE];
ssize_t bytes_read = 0;

while ((bytes_read = read(STDIN_FILENO, buffer, MAX_BUFFER_SIZE)) > 0) {
    if (strncmp(buffer, EXIT_MESSAGE, sizeof(EXIT_MESSAGE) - 1) == 0) {
        break;
    }

    InvertString(buffer, bytes_read - 1);

    ssize_t written = write(file, buffer, bytes_read);

    if (written != bytes_read) {
        char message[] = "[ERROR] Can't write all text to file!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        break;
    }
}

close(file);

return 0;
}

```

### **messages.h**

```

#ifndef MAI_OS_2025_MESSAGES_H
#define MAI_OS_2025_MESSAGES_H

static const char EXIT_MESSAGE[] = "Stop\n";

#endif //MAI_OS_2025_MESSAGES_H

```

## **Протокол работы программы**

### **Тестирование:**

```

$ ./parent
test1.txt
test2.txt
Hello, World!

```

```
Hi
Test messageeeeeeee
Finish
Stop
$ cat < test1.txt
eeeeeeegassem tseT
hsiniF
$ cat < test2.txt
!dlroW ,olleH
iH
```

### Strace:

```
$ strace -f ./parent
execve("./parent", [ "./parent" ], 0x7ffcf1324b98 /* 27 vars */) = 0
brk(NULL)                               = 0x61ae5e9ac000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7afe839c9000
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=22875, ...}) = 0
mmap(NULL, 22875, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7afe839c3000
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\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) = 0x7afe83600000
mmap(0x7afe83628000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7afe83628000
mmap(0x7afe837b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7afe837b0000
mmap(0x7afe837ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7afe837ff000
mmap(0x7afe83805000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7afe83805000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7afe839c0000
arch_prctl(ARCH_SET_FS, 0x7afe839c0740) = 0
set_tid_address(0x7afe839c0a10)         = 125301
set_robust_list(0x7afe839c0a20, 24)     = 0
rseq(0x7afe839c1060, 0x20, 0, 0x53053053) = 0
mprotect(0x7afe837ff000, 16384, PROT_READ) = 0
mprotect(0x61ae1ef0e000, 4096, PROT_READ) = 0
mprotect(0x7afe83a01000, 8192, PROT_READ) = 0
```



```

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7afe839c3000, 22875) = 0
read(0, test1.txt
"test1.txt\n", 4096) = 10
read(0, test2.txt
"test2.txt\n", 4096) = 10
pipe2([3, 4], 0) = 0
pipe2([5, 6], 0) = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 125302 attached
, child_tidptr=0x7afe839c0a10) = 125302
[pid 125302] set_robust_list(0x7afe839c0a20, 24 <unfinished ...>
[pid 125301] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid 125302] <... set_robust_list resumed>) = 0
[pid 125302] dup2(3, 0) = 0
[pid 125302] execve("./child", ["test1.txt"], 0x7fff8f444518 /* 27 vars */strace:
Process 125303 attached
<unfinished ...>
[pid 125301] <... clone resumed>, child_tidptr=0x7afe839c0a10) = 125303
[pid 125301] read(0, <unfinished ...>
[pid 125303] set_robust_list(0x7afe839c0a20, 24) = 0
[pid 125303] dup2(5, 0) = 0
[pid 125302] <... execve resumed>) = 0
[pid 125303] execve("./child", ["test2.txt"], 0x7fff8f444518 /* 27 vars */ <unfinished
...>
[pid 125302] brk(NULL) = 0x57a3bb4c6000
[pid 125302] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x79ec7f2fb000
[pid 125303] <... execve resumed>) = 0
[pid 125302] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 125303] brk(NULL <unfinished ...>
[pid 125302] <... access resumed>) = -1 ENOENT (No such file or directory)
[pid 125303] <... brk resumed>) = 0x5ba6635e2000
[pid 125302] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 7
[pid 125303] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid 125302] fstat(7, <unfinished ...>
[pid 125303] <... mmap resumed>) = 0x77bef6be8000
[pid 125302] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=22875, ...}) = 0
[pid 125303] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 125302] mmap(NULL, 22875, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>
[pid 125303] <... access resumed>) = -1 ENOENT (No such file or directory)
[pid 125303] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 125302] <... mmap resumed>) = 0x79ec7f2f5000
[pid 125303] <... openat resumed>) = 7
[pid 125302] close(7 <unfinished ...>
[pid 125303] fstat(7, <unfinished ...>
[pid 125302] <... close resumed>) = 0

```

```

[pid 125303] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=22875, ...}) = 0
[pid 125302] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 125303] mmap(NULL, 22875, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>
[pid 125302] <... openat resumed>) = 7
[pid 125303] <... mmap resumed>) = 0x77bef6be2000
[pid 125303] close(7 <unfinished ...>
[pid 125302] read(7, <unfinished ...>
[pid 125303] <... close resumed>) = 0
[pid 125302] <... read
resumed>"\177ELF\2\1\1\3\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 125303] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 125302] pread64(7, <unfinished ...>
[pid 125303] <... openat resumed>) = 7
[pid 125302] <... 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 125303] read(7, <unfinished ...>
[pid 125302] fstat(7, <unfinished ...>
[pid 125303] <... read
resumed>"\177ELF\2\1\1\3\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 125302] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 125303] pread64(7, <unfinished ...>
[pid 125302] pread64(7, <unfinished ...>
[pid 125303] <... 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 125302] <... 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 125303] fstat(7, <unfinished ...>
[pid 125302] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0 <unfinished
...>
[pid 125303] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 125302] <... mmap resumed>) = 0x79ec7f000000
[pid 125303] pread64(7, <unfinished ...>
[pid 125302] mmap(0x79ec7f028000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000 <unfinished ...>
[pid 125303] <... 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 125302] <... mmap resumed>) = 0x79ec7f028000
[pid 125303] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0 <unfinished
...>
[pid 125302] mmap(0x79ec7f1b0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1b0000 <unfinished ...>
[pid 125303] <... mmap resumed>) = 0x77bef6800000
[pid 125302] <... mmap resumed>) = 0x79ec7f1b0000
[pid 125303] mmap(0x77bef6828000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000) = 0x77bef6828000
[pid 125302] mmap(0x79ec7f1ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1fe000 <unfinished ...>

```

```

    [pid 125303] mmap(0x77bef69b0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1b0000 <unfinished ...>
    [pid 125302] <... mmap resumed>)          = 0x79ec7f1ff000
    [pid 125303] <... mmap resumed>)          = 0x77bef69b0000
    [pid 125303] mmap(0x77bef69ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1fe000 <unfinished ...>
    [pid 125302] mmap(0x79ec7f205000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
    [pid 125303] <... mmap resumed>)          = 0x77bef69ff000
    [pid 125302] <... mmap resumed>)          = 0x79ec7f205000
    [pid 125303] mmap(0x77bef6a05000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
    [pid 125302] close(7)                    = 0
    [pid 125303] <... mmap resumed>)          = 0x77bef6a05000
    [pid 125303] close(7 <unfinished ...>)
    [pid 125302] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
    [pid 125303] <... close resumed>)         = 0
    [pid 125302] <... mmap resumed>)          = 0x79ec7f2f2000
    [pid 125303] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
    [pid 125302] arch_prctl(ARCH_SET_FS, 0x79ec7f2f2740 <unfinished ...>
    [pid 125303] <... mmap resumed>)          = 0x77bef6bdf000
    [pid 125302] <... arch_prctl resumed>)    = 0
    [pid 125303] arch_prctl(ARCH_SET_FS, 0x77bef6bdf740 <unfinished ...>
    [pid 125302] set_tid_address(0x79ec7f2f2a10 <unfinished ...>
    [pid 125303] <... arch_prctl resumed>)    = 0
    [pid 125302] <... set_tid_address resumed>) = 125302
    [pid 125303] set_tid_address(0x77bef6bdffa10) = 125303
    [pid 125302] set_robust_list(0x79ec7f2f2a20, 24 <unfinished ...>
    [pid 125303] set_robust_list(0x77bef6bdffa20, 24 <unfinished ...>
    [pid 125302] <... set_robust_list resumed>) = 0
    [pid 125303] <... set_robust_list resumed>) = 0
    [pid 125302] rseq(0x79ec7f2f3060, 0x20, 0, 0x53053053 <unfinished ...>
    [pid 125303] rseq(0x77bef6be0060, 0x20, 0, 0x53053053 <unfinished ...>
    [pid 125302] <... rseq resumed>)          = 0
    [pid 125303] <... rseq resumed>)          = 0
    [pid 125302] mprotect(0x79ec7f1ff000, 16384, PROT_READ <unfinished ...>
    [pid 125303] mprotect(0x77bef69ff000, 16384, PROT_READ <unfinished ...>
    [pid 125302] <... mprotect resumed>)      = 0
    [pid 125303] <... mprotect resumed>)      = 0
    [pid 125302] mprotect(0x57a3b89bb000, 4096, PROT_READ <unfinished ...>
    [pid 125303] mprotect(0x5ba65dcdc000, 4096, PROT_READ <unfinished ...>
    [pid 125302] <... mprotect resumed>)      = 0
    [pid 125303] <... mprotect resumed>)      = 0
    [pid 125302] mprotect(0x79ec7f333000, 8192, PROT_READ <unfinished ...>
    [pid 125303] mprotect(0x77bef6c20000, 8192, PROT_READ <unfinished ...>
    [pid 125302] <... mprotect resumed>)      = 0
    [pid 125303] <... mprotect resumed>)      = 0

```

```

[pid 125302] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>
[pid 125303] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>
[pid 125302] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 125303] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 125302] munmap(0x79ec7f2f5000, 22875 <unfinished ...>
[pid 125303] munmap(0x77bef6be2000, 22875) = 0
[pid 125302] <... munmap resumed>) = 0
[pid 125303] openat(AT_FDCWD, "test2.txt", O_RDWR|O_CREAT, 0700 <unfinished ...>
[pid 125302] openat(AT_FDCWD, "test1.txt", O_RDWR|O_CREAT, 0700 <unfinished ...>
[pid 125303] <... openat resumed>) = 7
[pid 125302] <... openat resumed>) = 7
[pid 125303] read(0, <unfinished ...>
[pid 125302] read(0, Hello, World!
<unfinished ...>
[pid 125301] <... read resumed>"Hello, World!\n", 4096) = 14
[pid 125301] write(6, "Hello, World!\n", 14) = 14
[pid 125303] <... read resumed>"Hello, World!\n", 4096) = 14
[pid 125301] read(0, <unfinished ...>
[pid 125303] write(7, "!dlrow ,olleH\n", 14) = 14
[pid 125303] read(0, Hi
<unfinished ...>
[pid 125301] <... read resumed>"Hi\n", 4096) = 3
[pid 125301] write(6, "Hi\n", 3) = 3
[pid 125303] <... read resumed>"Hi\n", 4096) = 3
[pid 125301] read(0, <unfinished ...>
[pid 125303] write(7, "iH\n", 3) = 3
[pid 125303] read(0, Test messageeeeeeee
<unfinished ...>
[pid 125301] <... read resumed>"Test messageeeeeeee\n", 4096) = 19
[pid 125301] write(4, "Test messageeeeeeee\n", 19) = 19
[pid 125302] <... read resumed>"Test messageeeeeeee\n", 4096) = 19
[pid 125301] read(0, <unfinished ...>
[pid 125302] write(7, "eeeeeeegassem tseT\n", 19) = 19
[pid 125302] read(0, Finish
<unfinished ...>
[pid 125301] <... read resumed>"Finish\n", 4096) = 7
[pid 125301] write(4, "Finish\n", 7) = 7
[pid 125302] <... read resumed>"Finish\n", 4096) = 7
[pid 125301] read(0, <unfinished ...>
[pid 125302] write(7, "hsiniF\n", 7) = 7
[pid 125302] read(0, Stop
<unfinished ...>
[pid 125301] <... read resumed>"Stop\n", 4096) = 5
[pid 125301] write(4, "Stop\n\0", 6) = 6
[pid 125301] write(6, "Stop\n\0", 6 <unfinished ...>
[pid 125302] <... read resumed>"Stop\n\0", 4096) = 6
[pid 125301] <... write resumed>) = 6
[pid 125303] <... read resumed>"Stop\n\0", 4096) = 6
[pid 125302] close(7 <unfinished ...>

```

```

[pid 125301] wait4(125302, <unfinished ...>
[pid 125303] close(7) = 0
[pid 125302] <... close resumed>) = 0
[pid 125303] exit_group(0 <unfinished ...>
[pid 125302] exit_group(0 <unfinished ...>
[pid 125303] <... exit_group resumed>) = ?
[pid 125302] <... exit_group resumed>) = ?
[pid 125303] +++ exited with 0 +++
[pid 125302] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL) = 125302
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=125303, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
wait4(125303, NULL, 0, NULL) = 125303
close(3) = 0
close(4) = 0
close(5) = 0
close(6) = 0
exit_group(0) = ?
+++ exited with 0 +++

```

## Вывод

В ходе выполнения лабораторной работы были успешно приобретены практические навыки в управлении процессами в ОС, обеспечении обмена данных между процессами посредством каналов, обработке ошибок вызова системных вызовов ОС.

Возникли сложности со считыванием и записью строк посредством системных вызовов в консоль (особенности нулевого символа). Проблема была обнаружена с помощью программ *gdb* и *strace*. Была решена посредством незначительных изменений в обработке ввода-вывода из консоли.