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: Чувилов А.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 18.10.24

Москва, 2024

Постановка задачи

Вариант 21.

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: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние процессы инвертируют строки.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork() - создание дочернего процесса
- int execve(const char *filename, char *const argv[], char *const envp[]) (и другие вариации exec) - замена образа памяти процесса
- pid_t waitpid(pid_t pid, int *status, int options) - Ожидание завершения дочернего процесса
- void exit(int status) - завершения выполнения процесса и возвращение статуса
- int pipe(int pipefd[2]) - создание неименованного канала для передачи данных между процессами
- int dup2(int oldfd, int newfd) - переназначение файлового дескриптора
- int open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- int close(int fd) - закрыть файл
- int mkfifo(const char *pathname, mode_t mode) - создание именованного канала

Алгоритм решения:

Данный код реализует взаимодействие между процессами через каналы. Основной процесс создает два канала (pipe1 и pipe2) и порождает два дочерних процесса с помощью fork(). Каждый дочерний процесс получает имя файла от пользователя и перенаправляет стандартный ввод из соответствующего канала на запись в файл. Основной процесс принимает строки от пользователя, чередуя их отправку в первый или второй канал, а строки записываются в соответствующие файлы через дочерние процессы. Программа завершается, если пользователь вводит пустую строку или `exit`. Также реализована базовая обработка ошибок для создания каналов, порождения процессов и открытия файлов.

Код программы

Parent.c

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

#define BUFFER_SIZE 1024

void reverse_string(char *str) {
    int len = strlen(str);
    for (int i = 0; i < len / 2; i++) {
        char temp = str[i];
        str[i] = str[len - 1 - i];
        str[len - 1 - i] = temp;
    }
}

int main() {
    int pipe1[2], pipe2[2];
    char buffer[BUFFER_SIZE];
    int message_count = 1;
    char file1[BUFFER_SIZE];
    char file2[BUFFER_SIZE];

    const char* error_pipe = "Failed to create pipe.\n";
    const char* error_fork1 = "Failed to fork process 1.\n";
    const char* error_open1 = "Failed to open file in child 1.\n";
    const char* error_fork2 = "Failed to fork process 2.\n";
    const char* error_open2 = "Failed to open file in child 2.\n";

    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
        write(2, error_pipe, strlen(error_pipe));
        exit(EXIT_FAILURE);
    }

    write(1, "Enter filename for child 1: ", strlen("Enter filename for child 1: "));
    read(0, file1, BUFFER_SIZE);
```

```
file1[strcspn(file1, "\n")] = '\0';

write(1, "Enter filename for child 2: ", strlen("Enter filename for child 2: "));
read(0, file2, BUFFER_SIZE);
file2[strcspn(file2, "\n")] = '\0';

if (fork() == 0) {
    int fd1 = open(file1, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd1 == -1) {
        write(2, error_open1, strlen(error_open1));
        exit(EXIT_FAILURE);
    }

    close(pipe1[1]);
    dup2(pipe1[0], 0);
    dup2(fd1, 1);
    close(pipe1[0]);
    close(fd1);

    execl("./child1", "child1", NULL);
    write(2, "Execution failed for child 1.\n", strlen("Execution failed for child 1.\n"));
    exit(EXIT_FAILURE);
}

if (fork() == 0) {
    int fd2 = open(file2, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd2 == -1) {
        write(2, error_open2, strlen(error_open2));
        exit(EXIT_FAILURE);
    }

    close(pipe2[1]);
    dup2(pipe2[0], 0);
    dup2(fd2, 1);
    close(pipe2[0]);
    close(fd2);

    execl("./child2", "child2", NULL);
    write(2, "Execution failed for child 2.\n", strlen("Execution failed for child 2.\n"));
    exit(EXIT_FAILURE);
}

close(pipe1[0]);
close(pipe2[0]);

while (1) {
    write(1, "Enter a line: ", strlen("Enter a line: "));
    ssize_t bytes_read = read(0, buffer, BUFFER_SIZE);
    if (bytes_read <= 0) {
        break;
    }
    buffer[bytes_read - 1] = '\0';

    if (strlen(buffer) == 0 || strcmp(buffer, "exit") == 0) {
        break;
    }

    if (message_count % 2 == 0) {
        write(pipe2[1], buffer, bytes_read);
    } else {
```

```
            write(pipe1[1], buffer, bytes_read);
        }
        message_count++;
    }

    close(pipe1[1]);
    close(pipe2[1]);

    return 0;
}
```

child1.c

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

#define BUFFER_SIZE 1024

void reverse_string(char *str) {
    int len = strlen(str);
    for (int i = 0; i < len / 2; i++) {
        char temp = str[i];
        str[i] = str[len - 1 - i];
        str[len - 1 - i] = temp;
    }
}

int main() {
    char buffer[BUFFER_SIZE];

    while (1) {
        ssize_t bytes_read = read(0, buffer, BUFFER_SIZE);
        if (bytes_read <= 0) {
            exit(EXIT_SUCCESS);
        }
        buffer[bytes_read - 1] = '\0';

        if (strlen(buffer) == 0) {
            break;
        }

        reverse_string(buffer);
        write(1, buffer, strlen(buffer));
        write(1, "\n", 1);
    }

    return 0;
}
```

child2.c

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

#define BUFFER_SIZE 1024

void to_uppercase(char *str) {
    for (int i = 0; str[i] != '\0'; i++) {
```

```
        str[i] = toupper(str[i]);
    }
}

int main() {
    char buffer[BUFFER_SIZE];

    while (1) {
        ssize_t bytes_read = read(0, buffer, BUFFER_SIZE);
        if (bytes_read <= 0) {

            exit(EXIT_SUCCESS);
        }
        buffer[bytes_read - 1] = '\0';

        if (strlen(buffer) == 0) {
            break;
        }

        to_uppercase(buffer);
        write(1, buffer, strlen(buffer));
        write(1, "\n", 1);
    }

    return 0;
}
```

Протокол работы программы

Тестирование:

$ ./a.out

Enter filename for child 1: 1.txt

Enter filename for child 2: 2.txt

Enter a line: hello

Enter a line: lol

Enter a line: world

Enter a line: how are you?

Enter a line: ddd

Enter a line: exit

Вывод в файле 1.txt:

olleh

dlrow

ddd

Вывод в файле 2.txt:

LOL

HOW ARE YOU?

Strace:

```
traktor@traktor-MaiBook-X-series:~/OS/src$ strace -f ./a.out
execve("./a.out", ["./a.out"], 0x7fffe7ec8b28 /* 81 vars */) = 0
brk(NULL)                               = 0x5f23e2f9f000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x738da717c000
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=74519, ...}) = 0
mmap(NULL, 74519, PROT_READ, MAP_PRIVATE, 3, 0) = 0x738da7169000
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"..., 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) =
0x738da6e00000
mmap(0x738da6e28000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x738da6e28000
mmap(0x738da6fb0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x738da6fb0000
mmap(0x738da6fff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x738da6fff000
mmap(0x738da7005000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x738da7005000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x738da7166000
arch_prctl(ARCH_SET_FS, 0x738da7166740) = 0
set_tid_address(0x738da7166a10)         = 12821
set_robust_list(0x738da7166a20, 24)     = 0
rseq(0x738da7167060, 0x20, 0, 0x53053053) = 0
mprotect(0x738da6fff000, 16384, PROT_READ) = 0
mprotect(0x5f23ae43c000, 4096, PROT_READ) = 0
mprotect(0x738da71b4000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x738da7169000, 74519)           = 0
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
write(1, "Enter filename for child 1: ", 28Enter filename for child 1: ) = 28
read(0, 1.txt
"1.txt\n", 1024)                = 6
write(1, "Enter filename for child 2: ", 28Enter filename for child 2: ) = 28
read(0, 2.txt
"2.txt\n", 1024)                = 6
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x738da7166a10) = 12932
clone(child_stack=NULL,
```

```
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 12932
attached
strace: Process 12933 attached
, child_tidptr=0x738da7166a10) = 12933
[pid 12821] close(3 <unfinished ...>
[pid 12932] set_robust_list(0x738da7166a20, 24 <unfinished ...>
[pid 12821] <... close resumed>)        = 0
[pid 12933] set_robust_list(0x738da7166a20, 24 <unfinished ...>
[pid 12821] close(5 <unfinished ...>
[pid 12932] <... set_robust_list resumed>) = 0
[pid 12821] <... close resumed>)        = 0
[pid 12933] <... set_robust_list resumed>) = 0
[pid 12821] write(1, "Enter a line: ", 14Enter a line: ) = 14
[pid 12821] read(0,  <unfinished ...>
[pid 12933] openat(AT_FDCWD, "2.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644 <unfinished
...>
[pid 12932] openat(AT_FDCWD, "1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644 <unfinished
...>
[pid 12933] <... openat resumed>)       = 7
[pid 12932] <... openat resumed>)       = 7
[pid 12933] close(6 <unfinished ...>
[pid 12932] close(4 <unfinished ...>
[pid 12933] <... close resumed>)        = 0
[pid 12932] <... close resumed>)        = 0
[pid 12933] dup2(5, 0 <unfinished ...>
[pid 12932] dup2(3, 0 <unfinished ...>
[pid 12933] <... dup2 resumed>)         = 0
[pid 12933] dup2(7, 1 <unfinished ...>
[pid 12932] <... dup2 resumed>)         = 0
[pid 12933] <... dup2 resumed>)         = 1
[pid 12932] dup2(7, 1 <unfinished ...>
[pid 12933] close(5 <unfinished ...>
[pid 12932] <... dup2 resumed>)         = 1
[pid 12933] <... close resumed>)        = 0
[pid 12932] close(3 <unfinished ...>
[pid 12933] close(7 <unfinished ...>
[pid 12932] <... close resumed>)        = 0
[pid 12933] <... close resumed>)        = 0
[pid 12932] close(7 <unfinished ...>
[pid 12933] execve("./child2", ["child2"], 0x7fff9d672968 /* 81 vars */ <unfinished ...>
[pid 12932] <... close resumed>)        = 0
[pid 12932] execve("./child1", ["child1"], 0x7fff9d672968 /* 81 vars */) = 0
[pid 12933] <... execve resumed>)       = 0
[pid 12932] brk(NULL <unfinished ...>
[pid 12933] brk(NULL <unfinished ...>
[pid 12932] <... brk resumed>)          = 0x56caa8f79000
[pid 12933] <... brk resumed>)          = 0x61c00c9b0000
[pid 12933] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 12932] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 12933] <... mmap resumed>)         = 0x79b483544000
[pid 12932] <... mmap resumed>)         = 0x762dbe2bb000
[pid 12933] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 12932] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 12933] <... access resumed>)       = -1 ENOENT (Нет такого файла или каталога)
[pid 12932] <... access resumed>)       = -1 ENOENT (Нет такого файла или каталога)
```

```
[pid 12933] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 12932] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 12933] <... openat resumed>)       = 5
[pid 12933] fstat(5,  <unfinished ...>
[pid 12932] <... openat resumed>)       = 3
[pid 12933] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=74519, ...}) = 0
[pid 12932] fstat(3,  <unfinished ...>
[pid 12933] mmap(NULL, 74519, PROT_READ, MAP_PRIVATE, 5, 0 <unfinished ...>
[pid 12932] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=74519, ...}) = 0
[pid 12933] <... mmap resumed>)         = 0x79b483531000
[pid 12932] mmap(NULL, 74519, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 12933] close(5 <unfinished ...>
[pid 12932] <... mmap resumed>)         = 0x762dbe2a8000
[pid 12933] <... close resumed>)        = 0
[pid 12932] close(3 <unfinished ...>
[pid 12933] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 12932] <... close resumed>)        = 0
[pid 12933] <... openat resumed>)       = 5
[pid 12932] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 12933] read(5,  <unfinished ...>
[pid 12932] <... openat resumed>)       = 3
[pid 12933] <... 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"..., 832) = 832
[pid 12932] read(3,  <unfinished ...>
[pid 12933] pread64(5,  <unfinished ...>
[pid 12932] <... 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"..., 832) = 832
[pid 12933] <... 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 12932] pread64(3,  <unfinished ...>
[pid 12933] fstat(5,  <unfinished ...>
[pid 12932] <... 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 12933] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 12932] fstat(3,  <unfinished ...>
[pid 12933] pread64(5,  <unfinished ...>
[pid 12932] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 12933] <... 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 12932] pread64(3,  <unfinished ...>
[pid 12933] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 5, 0
<unfinished ...>
[pid 12932] <... 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 12933] <... mmap resumed>)         = 0x79b483200000
[pid 12932] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0
<unfinished ...>
[pid 12933] mmap(0x79b483228000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x28000 <unfinished ...>
[pid 12932] <... mmap resumed>)         = 0x762dbe000000
[pid 12932] mmap(0x762dbe028000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
[pid 12933] <... mmap resumed>)         = 0x79b483228000
[pid 12933] mmap(0x79b4833b0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x1b0000 <unfinished ...>
```

```
[pid 12932] <... mmap resumed>)         = 0x762dbe028000
[pid 12933] <... mmap resumed>)         = 0x79b4833b0000
[pid 12932] mmap(0x762dbe1b0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000 <unfinished ...>
[pid 12933] mmap(0x79b4833ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x1fe000 <unfinished ...>
[pid 12932] <... mmap resumed>)         = 0x762dbe1b0000
[pid 12933] <... mmap resumed>)         = 0x79b4833ff000
[pid 12932] mmap(0x762dbe1ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000 <unfinished ...>
[pid 12933] mmap(0x79b483405000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 12932] <... mmap resumed>)         = 0x762dbe1ff000
[pid 12933] <... mmap resumed>)         = 0x79b483405000
[pid 12932] mmap(0x762dbe205000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 12933] close(5 <unfinished ...>
[pid 12932] <... mmap resumed>)         = 0x762dbe205000
[pid 12933] <... close resumed>)        = 0
[pid 12932] close(3 <unfinished ...>
[pid 12933] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 12932] <... close resumed>)        = 0
[pid 12933] <... mmap resumed>)         = 0x79b48352e000
[pid 12933] arch_prctl(ARCH_SET_FS, 0x79b48352e740 <unfinished ...>
[pid 12932] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 12933] <... arch_prctl resumed>)   = 0
[pid 12933] set_tid_address(0x79b48352ea10 <unfinished ...>
[pid 12932] <... mmap resumed>)         = 0x762dbe2a5000
[pid 12933] <... set_tid_address resumed>) = 12933
[pid 12933] set_robust_list(0x79b48352ea20, 24 <unfinished ...>
[pid 12932] arch_prctl(ARCH_SET_FS, 0x762dbe2a5740 <unfinished ...>
[pid 12933] <... set_robust_list resumed>) = 0
[pid 12932] <... arch_prctl resumed>)   = 0
[pid 12933] rseq(0x79b48352f060, 0x20, 0, 0x53053053 <unfinished ...>
[pid 12932] set_tid_address(0x762dbe2a5a10 <unfinished ...>
[pid 12933] <... rseq resumed>)         = 0
[pid 12932] <... set_tid_address resumed>) = 12932
[pid 12932] set_robust_list(0x762dbe2a5a20, 24) = 0
[pid 12932] rseq(0x762dbe2a6060, 0x20, 0, 0x53053053) = 0
[pid 12933] mprotect(0x79b4833ff000, 16384, PROT_READ) = 0
[pid 12932] mprotect(0x762dbe1ff000, 16384, PROT_READ <unfinished ...>
[pid 12933] mprotect(0x61bff49d8000, 4096, PROT_READ <unfinished ...>
[pid 12932] <... mprotect resumed>)     = 0
[pid 12933] <... mprotect resumed>)     = 0
[pid 12932] mprotect(0x56caa898a000, 4096, PROT_READ <unfinished ...>
[pid 12933] mprotect(0x79b483582000, 8192, PROT_READ <unfinished ...>
[pid 12932] <... mprotect resumed>)     = 0
[pid 12933] <... mprotect resumed>)     = 0
[pid 12932] mprotect(0x762dbe2f3000, 8192, PROT_READ <unfinished ...>
[pid 12933] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid 12932] <... mprotect resumed>)     = 0
[pid 12933] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 12933] munmap(0x79b483531000, 74519 <unfinished ...>
[pid 12932] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
```

```
[pid 12933] <... munmap resumed>)      = 0
[pid 12933] read(0,  <unfinished ...>
[pid 12932] munmap(0x762dbe2a8000, 74519) = 0
[pid 12932] read(0, sddSAAS
 <unfinished ...>
[pid 12821] <... read resumed>"sddSAAS\n", 1024) = 8
[pid 12821] write(4, "sddSAAS\0", 8)    = 8
[pid 12932] <... read resumed>"sddSAAS\0", 1024) = 8
[pid 12932] write(1, "SAASdds", 7 <unfinished ...>
[pid 12821] write(1, "Enter a line: ", 14 <unfinished ...>
[pid 12932] <... write resumed>)       = 7
Enter a line: [pid 12932] write(1, "\n", 1 <unfinished ...>
[pid 12821] <... write resumed>)       = 14
[pid 12821] read(0,  <unfinished ...>
[pid 12932] <... write resumed>)       = 1
[pid 12932] read(0, dasda
 <unfinished ...>
[pid 12821] <... read resumed>"dasda\n", 1024) = 6
[pid 12821] write(6, "dasda\0", 6)      = 6
[pid 12933] <... read resumed>"dasda\0", 1024) = 6
[pid 12821] write(1, "Enter a line: ", 14 <unfinished ...>
[pid 12933] write(1, "DASDA", 5Enter a line:  <unfinished ...>
[pid 12821] <... write resumed>)       = 14
[pid 12933] <... write resumed>)       = 5
[pid 12933] write(1, "\n", 1 <unfinished ...>
[pid 12821] read(0,  <unfinished ...>
[pid 12933] <... write resumed>)       = 1
[pid 12933] read(0, sdadasd
 <unfinished ...>
[pid 12821] <... read resumed>"sdadasd\n", 1024) = 8
[pid 12821] write(4, "sdadasd\0", 8)    = 8
[pid 12932] <... read resumed>"sdadasd\0", 1024) = 8
[pid 12821] write(1, "Enter a line: ", 14Enter a line: ) = 14
[pid 12821] read(0,  <unfinished ...>
[pid 12932] write(1, "dsadads", 7)      = 7
[pid 12932] write(1, "\n", 1)           = 1
[pid 12932] read(0, sadasd
 <unfinished ...>
[pid 12821] <... read resumed>"sadasd\n", 1024) = 7
[pid 12821] write(6, "sadasd\0", 7)     = 7
[pid 12933] <... read resumed>"sadasd\0", 1024) = 7
[pid 12821] write(1, "Enter a line: ", 14 <unfinished ...>
Enter a line: [pid 12933] write(1, "SADASD", 6 <unfinished ...>
[pid 12821] <... write resumed>)       = 14
[pid 12821] read(0,  <unfinished ...>
[pid 12933] <... write resumed>)       = 6
[pid 12933] write(1, "\n", 1)          = 1
[pid 12933] read(0, asdasda
 <unfinished ...>
[pid 12821] <... read resumed>"asdasda\n", 1024) = 8
[pid 12821] write(4, "asdasda\0", 8)    = 8
[pid 12932] <... read resumed>"asdasda\0", 1024) = 8
[pid 12821] write(1, "Enter a line: ", 14Enter a line: ) = 14
[pid 12932] write(1, "adsadsa", 7 <unfinished ...>
[pid 12821] read(0,  <unfinished ...>
[pid 12932] <... write resumed>)       = 7
[pid 12932] write(1, "\n", 1)          = 1
```

```
[pid 12932] read(0, dasdasd
 <unfinished ...>
[pid 12821] <... read resumed>"dasdasd\n", 1024) = 8
[pid 12821] write(6, "dasdasd\0", 8)    = 8
[pid 12933] <... read resumed>"dasdasd\0", 1024) = 8
[pid 12821] write(1, "Enter a line: ", 14Enter a line: ) = 14
[pid 12933] write(1, "DASDASD", 7 <unfinished ...>
[pid 12821] read(0,  <unfinished ...>
[pid 12933] <... write resumed>)        = 7
[pid 12933] write(1, "\n", 1)           = 1
[pid 12933] read(0, exit
 <unfinished ...>
[pid 12821] <... read resumed>"exit\n", 1024) = 5
[pid 12821] close(4)                    = 0
[pid 12821] close(6)                    = 0
[pid 12821] exit_group(0)               = ?
[pid 12821] +++ exited with 0 +++
```

Вывод

В данной работе реализовано взаимодействие между процессами с использованием каналов (pipe) для передачи данных между основным и дочерними процессами. Основной процесс запрашивает строки у пользователя, чередует их отправку через два канала, а дочерние процессы записывают полученные данные в указанные пользователем файлы. В программе продемонстрированы ключевые аспекты межпроцессного взаимодействия: создание процессов с помощью fork(), перенаправление ввода/вывода через dup2(), обработка строк и базовая обработка ошибок для обеспечения надежности (проверка на ошибки при создании каналов, порождении процессов и открытии файлов). Гибкость программы обеспечивается возможностью пользователя задавать имена файлов и завершать работу с помощью команды "exit". Итогом является создание функциональной системы взаимодействия процессов, которая демонстрирует основы IPC в UNIX.