

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-211Б-23

Студент: Ласточкин М.В.

Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_

Дата: 18.11.24

Москва, 2024

# 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.

**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

Использованные системные вызовы:

- `pid_t fork(void)`; – создает дочерний процесс.
- `int pipe(int *fd)`; – создает канал и помещает дескрипторы файла для чтения и записи в `fd[0]` и `fd[1]`.
- `pid_t getpid(void)`; – возвращает ID вызывающего процесса.
- `int open(const char * __file, int __oflag, ...)`; – используется для открытия файла для чтения, записи или и того, и другого.
- `ssize_t write(int __fd, const void * __buf, size_t __n)`; – Записывает N байт из буфера(BUF) в файл (FD). Возвращает количество записанных байт или -1.
- `void exit(int __status)`; – выполняет немедленное завершение программы. Все используемые программой потоки закрываются, и временные файлы удаляются, управление возвращается ОС или другой программе.
- `int close(int __fd)`; – сообщает операционной системе об окончании работы с файловым дескриптором, и закрывает файл(FD).
- `int dup2(int __fd, int __fd2)`; – копирует FD в FD2, закрыв FD2 если это требуется.
- `int execv(const char * __path, char *const * __argv)`; – заменяет образ текущего процесса на образ нового процесса, определённого в пути path.
- `ssize_t read(int __fd, void * __buf, size_t __nbytes)`; – считывает указанное количество байт из файла(FD) в буфер(BUF).
- `pid_t wait(int * __stat_loc)`; – используются для ожидания изменения состояния процесса-потомка вызвавшего процесса и получения информации о потомке, чьё состояние изменилось.

Для выполнения данной лабораторной работы я изучил указанные выше системные вызовы, а также пример выполнения подобного задания.

Программа `server.c` получает на вход два аргумента – пути к файлам, в которые требуется записать результат работы. С помощью `readlink()` сохраняем полный путь до файла. После создаём два канала с помощью `pipe` для общения с двумя дочерними процессами. Далее выполняется `fork()`

для создания первого дочернего процесса, с помощью конструкции switch/case определяем в каком процессе мы находимся.

Если процесс дочерний, то используем dup2() для копирования файлового дескриптора канала и с помощью execl() подменяем образ текущего процесса на новый(client).

Если процесс – родитель, то делаем ещё один fork(), далее повторяем те же действия, если мы в дочернем процессе. Если же мы родитель, то начинаем читать строки из потока ввода и по очереди передавать то первому дочернему процессу, то второму. После окончания ввода ждём завершения обоих дочерних процессов и программа завершается.

Программа client открывает переданный в качестве аргумента файл, после этого считывает строки из потока ввода(подменён на вывод канала сервера), переворачивает и записывает в открытый файл. При окончании ввода строк файл закрывается, программа завершается.

## Код программы

### client.c

```
#include <stdint.h>
#include <stdbool.h>

#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>

void delete(char *str) {
    if (str == NULL) {
        return;
    }

    char vowels[] = "AEIOUYaeiouy";
    size_t len = strlen(str);
    size_t j = 0;

    for (size_t i = 0; i < len; ++i) {
        if (strchr(vowels, str[i]) == NULL) {
            str[j++] = str[i];
        }
    }
}
```

```
    str[j] = '\\0';
}

int main(int argc, char **argv)
{
    char buf[4096];

    ssize_t bytes;

    pid_t pid = getpid();

    // NOTE: `O_WRONLY` only enables file for writing
    // NOTE: `O_CREAT` creates the requested file if absent
    // NOTE: `O_TRUNC` empties the file prior to opening
    // NOTE: `O_APPEND` subsequent writes are being appended instead of
    overwritten

    int32_t file = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC | O_APPEND,
0600);

    if (file == -1)
    {
        const char msg[] = "error: failed to open requested file\\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    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 - 1] = '\0';
        delete(buf);
        size_t len = strlen(buf);
        buf[len] = '\n';
        int32_t written = write(file, buf, len + 1);
        if (written != len + 1)
        {
            const char msg[] = "error: client failed to write to file\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }
    }
}

if (close(file) == -1)
{
    const char msg[] = "error: client failed to close file\n";
    write(STDERR_FILENO, msg, sizeof(msg));
    exit(EXIT_FAILURE);
}

return 0;
}

```

### server.c

```

#include <stdint.h>
#include <stdbool.h>

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

static char CLIENT_PROGRAM_NAME[] = "client";

```

```

size_t my_strlen(char *str) {
    char *end = str;
    while(*end++);
    return end - str - 1;
}

int main(int argc, char **argv)
{
    if (argc == 1)
    {
        char msg[1024];
        uint32_t len = snprintf(msg, sizeof(msg) - 1, "usage: %s filename\n",
argv[0]);
        write(STDERR_FILENO, msg, len);
        exit(EXIT_SUCCESS);
    }

    char progbath[1024];
    {
        ssize_t len = readlink("/proc/self/exe", progbath,
                                sizeof(progbath) - 1);

        if (len == -1)
        {
            const char msg[] = "error: failed to read full program path\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }

        while (progbath[len] != '/')
            --len;

        progbath[len] = '\0';
    }

    int channel_1[2], channel_2[2];

```

```
if (pipe(channel_1) == -1)
{
    const char msg[] = "error: failed to create pipe\n";
    write(STDERR_FILENO, msg, sizeof(msg));
    exit(EXIT_FAILURE);
}

if (pipe(channel_2) == -1)
{
    const char msg[] = "error: failed to create pipe\n";
    write(STDERR_FILENO, msg, sizeof(msg));
    exit(EXIT_FAILURE);
}

const pid_t child_1 = fork();

switch (child_1)
{
    case -1:
    {
        const char msg[] = "error: failed to spawn new process\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    break;

    case 0:
    {
        pid_t pid = getpid();

        if (dup2(channel_1[STDIN_FILENO], STDIN_FILENO) == -1)
        {
            char msg[128];
            snprintf(msg, sizeof(msg), "%d: failed to use dup2\n", pid);
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }
    }
}
```

```

    }

    if (close(channel_1[STDOUT_FILENO]) == -1)
    {
        char msg[128];
        snprintf(msg, sizeof(msg), "%d: failed to close pipe\n",
pid);

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

    {
        char msg[64];
        const int32_t length = snprintf(msg, sizeof(msg),
                                         "%d: I'm a child1\n", pid);
        write(STDOUT_FILENO, msg, length);
    }

    {
        char path[1024];
        snprintf(path, sizeof(path) - 1, "%s/%s", progbath,
CLIENT_PROGRAM_NAME);

        char *const args[] = {CLIENT_PROGRAM_NAME, argv[1], NULL};

        int32_t status = execv(path, args);

        if (status == -1)
        {
            const char msg[] = "error: failed to exec into new
executable image\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }
    }
}

break;

```



```

default:
{
    const pid_t child_2 = fork();

    switch (child_2)
    {
        case -1:
        {
            const char msg[] = "error: failed to spawn new
process\n";

            write(STDERR_FILENO, msg, sizeof(msg));

            exit(EXIT_FAILURE);
        }

        break;

        case 0:
        {
            pid_t pid = getpid();

            if (dup2(channel_2[STDIN_FILENO], STDIN_FILENO) == -1)
            {
                char msg[128];
                snprintf(msg, sizeof(msg), "%d: failed to use
dup2\n", pid);

                write(STDERR_FILENO, msg, sizeof(msg));

                exit(EXIT_FAILURE);
            }

            if (close(channel_2[STDOUT_FILENO]) == -1)
            {
                char msg[128];
                snprintf(msg, sizeof(msg), "%d: failed to close
pipe\n", pid);

                write(STDERR_FILENO, msg, sizeof(msg));

                exit(EXIT_FAILURE);
            }
        }
    }
}

```

```

        {
            char msg[64];
            const int32_t length = snprintf(msg, sizeof(msg),
                                           "%d: I'm a child2\n",
pid);

            write(STDOUT_FILENO, msg, length);
        }

        {
            char path[1024];
            snprintf(path, sizeof(path) - 1, "%s/%s", proppath,
CLIENT_PROGRAM_NAME);

            char *const args[] = {CLIENT_PROGRAM_NAME, argv[2],
NULL};

            int32_t status = execv(path, 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;

default:
{
    pid_t pid = getpid();

    {
        char msg[128];
        const int32_t length = snprintf(msg, sizeof(msg),

```

```

                                "%d: I'm a parent, my
child1 & child2 has PID %d %d\n", pid, child_1, child_2);

        write(STDOUT_FILENO, msg, length);

    }

    if (close(channel_1[STDIN_FILENO]) == -1 ||
close(channel_2[STDIN_FILENO]) == -1)

    {

        const char msg[] = "error: server failed to close
pipe\n";

        write(STDERR_FILENO, msg, sizeof(msg));

        exit(EXIT_FAILURE);

    }

    char buf[4096];
    ssize_t bytes;

    {

        sleep(1);

        const char msg[] = "Input strings:\n";
        write(STDOUT_FILENO, msg, sizeof(msg));

    }

    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 - 1] = '\0';

            int32_t written;

```

```

        size_t len = my_strlen(buf);

        if (len > 10)

            written = write(channel_1[STDOUT_FILENO],
buf, bytes);

        else

            written = write(channel_2[STDOUT_FILENO],
buf, bytes);

        if (written != bytes)
        {

            const char msg[] = "error: client failed to
write to file\n";

            write(STDERR_FILENO, msg, sizeof(msg));

            exit(EXIT_FAILURE);

        }

    }

    buf[0] = '\n';

    if (write(channel_1[STDOUT_FILENO], buf, sizeof(char)) ==
-1 || write(channel_2[STDOUT_FILENO], buf, sizeof(char)) == -1)
    {

        const char msg[] = "error: server failed to write to
file\n";

        write(STDERR_FILENO, msg, sizeof(msg));

        exit(EXIT_FAILURE);

    }

    if (close(channel_1[STDOUT_FILENO]) == -1 ||
close(channel_2[STDOUT_FILENO] == -1))
    {

        const char msg[] = "error: server failed to close
pipe\n";

        write(STDERR_FILENO, msg, sizeof(msg));

        exit(EXIT_FAILURE);

    }

    // NOTE: `wait` blocks the parent until childs exits

    int child_status;

    pid_t wpid;

    while ((wpid = wait(&child_status) > 0))

```

```

        {
            if (child_status != EXIT_SUCCESS)
            {
                const char msg[] = "error: child exited with
error\n";

                write(STDERR_FILENO, msg, sizeof(msg));

                exit(child_status);
            }
        }
    }

    break;

}

break;

}

}

return 0;
}

```

## Протокол работы программы

```
max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/Osi/laba1$ ./server file1.txt
file2.txt
```

```
24224: I'm a child1
```

```
24223: I'm a parent, my child1 & child2 has PID 24224 24225
```

```
24225: I'm a child2
```

```
Input strings:
```

```
hahahajaha
```

```
adsuaiuiuwradacvcz213
```

```
9833ajshdjnnzxm
```

```
alsk334jf
```

```
baba12
```

```
dalk1
```

```
max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/Osi/laba1$ strace -f ./server
file1.txt file2.txt
```

```
execve("./server", [".server", "file1.txt", "file2.txt"], 0x7fff6a9a04f8 /* 26 vars
*/) = 0
```

```
brk(NULL) = 0x5581a1685000
```

```
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe0e06b6b0) = -1 EINVAL (Invalid argument)
```

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f23d5455000
```

```

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=18383, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 18383, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f23d5450000
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\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"... , 48,
848) = 48
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"... , 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) = 0x7f23d5227000
mprotect(0x7f23d524f000, 2023424, PROT_NONE) = 0
mmap(0x7f23d524f000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7f23d524f000
mmap(0x7f23d53e4000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f23d53e4000
mmap(0x7f23d543d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x215000) = 0x7f23d543d000
mmap(0x7f23d5443000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f23d5443000
close(3) = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f23d5224000
arch_prctl(ARCH_SET_FS, 0x7f23d5224740) = 0
set_tid_address(0x7f23d5224a10) = 24814
set_robust_list(0x7f23d5224a20, 24) = 0
rseq(0x7f23d52250e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f23d543d000, 16384, PROT_READ) = 0
mprotect(0x5581852f9000, 4096, PROT_READ) = 0
mprotect(0x7f23d548f000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f23d5450000, 18383) = 0
readlink("/proc/self/exe", "/mnt/c/Users/lasto/CLionProjects"... , 1023) = 49
pipe2([3, 4], 0) = 0
pipe2([5, 6], 0) = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 24815 attached
, child_tidptr=0x7f23d5224a10) = 24815
[pid 24815] set_robust_list(0x7f23d5224a20, 24 <unfinished ...>
[pid 24814] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>

```

```

[pid 24815] <... set_robust_list resumed>) = 0
[pid 24815] getpid() = 24815
[pid 24815] dup2(3, 0strace: Process 24816 attached
<unfinished ...>
[pid 24814] <... clone resumed>, child_tidptr=0x7f23d5224a10) = 24816
[pid 24815] <... dup2 resumed>) = 0
[pid 24814] getpid( <unfinished ...>
[pid 24816] set_robust_list(0x7f23d5224a20, 24 <unfinished ...>
[pid 24814] <... getpid resumed>) = 24814
[pid 24815] close(4 <unfinished ...>
[pid 24814] write(1, "24814: I'm a parent, my child1 &"..., 60 <unfinished ...>
[pid 24816] <... set_robust_list resumed>) = 0
24814: I'm a parent, my child1 & child2 has PID 24815 24816
[pid 24814] <... write resumed>) = 60
[pid 24815] <... close resumed>) = 0
[pid 24814] close(3 <unfinished ...>
[pid 24816] getpid( <unfinished ...>
[pid 24815] write(1, "24815: I'm a child1\n", 20 <unfinished ...>
[pid 24814] <... close resumed>) = 0
24815: I'm a child1
[pid 24816] <... getpid resumed>) = 24816
[pid 24814] close(5 <unfinished ...>
[pid 24815] <... write resumed>) = 20
[pid 24814] <... close resumed>) = 0
[pid 24816] dup2(5, 0 <unfinished ...>
[pid 24815] execve("/mnt/c/Users/lasto/CLionProjects/Osi/lab1/client", ["client",
"file1.txt"], 0x7ffe0e06b898 /* 26 vars */ <unfinished ...>
[pid 24814] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, <unfinished ...>
[pid 24816] <... dup2 resumed>) = 0
[pid 24816] close(6) = 0
[pid 24816] write(1, "24816: I'm a child2\n", 2024816: I'm a child2
) = 20
[pid 24816] execve("/mnt/c/Users/lasto/CLionProjects/Osi/lab1/client", ["client",
"file2.txt"], 0x7ffe0e06b898 /* 26 vars */) = 0
[pid 24815] <... execve resumed>) = 0
[pid 24816] brk(NULL <unfinished ...>
[pid 24815] brk(NULL <unfinished ...>
[pid 24816] <... brk resumed>) = 0x558f08b9d000
[pid 24815] <... brk resumed>) = 0x561f2c93d000
[pid 24816] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffb4e98af0 <unfinished ...>
[pid 24815] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc29fcff30 <unfinished ...>
[pid 24816] <... arch_prctl resumed>) = -1 EINVAL (Invalid argument)
[pid 24815] <... arch_prctl resumed>) = -1 EINVAL (Invalid argument)
[pid 24816] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid 24815] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid 24816] <... mmap resumed>) = 0x7fc9e5dc9000
[pid 24816] access("/etc/ld.so.preload", R_OK <unfinished ...>

```

```

[pid 24815] <... mmap resumed>)          = 0x7f10c8201000
[pid 24816] <... access resumed>)         = -1 ENOENT (No such file or directory)
[pid 24815] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 24816] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 24815] <... access resumed>)         = -1 ENOENT (No such file or directory)
[pid 24816] <... openat resumed>)         = 6
[pid 24815] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 24816] newfstatat(6, "", <unfinished ...>
[pid 24815] <... openat resumed>)         = 4
[pid 24816] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=18383, ...},
AT_EMPTY_PATH) = 0
[pid 24815] newfstatat(4, "", <unfinished ...>
[pid 24816] mmap(NULL, 18383, PROT_READ, MAP_PRIVATE, 6, 0 <unfinished ...>
[pid 24815] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=18383, ...},
AT_EMPTY_PATH) = 0
[pid 24816] <... mmap resumed>)          = 0x7fc9e5dc4000
[pid 24816] close(6 <unfinished ...>
[pid 24815] mmap(NULL, 18383, PROT_READ, MAP_PRIVATE, 4, 0 <unfinished ...>
[pid 24816] <... close resumed>)         = 0
[pid 24815] <... mmap resumed>)          = 0x7f10c81fc000
[pid 24816] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 24815] close(4 <unfinished ...>
[pid 24816] <... openat resumed>)         = 6
[pid 24815] <... close resumed>)         = 0
[pid 24816] read(6, <unfinished ...>
[pid 24815] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid 24816] <... 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"..., 832) =
832
[pid 24815] <... openat resumed>)         = 4
[pid 24816] pread64(6, <unfinished ...>
[pid 24815] read(4, <unfinished ...>
[pid 24816] <... 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 24815] <... 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"..., 832) =
832
[pid 24816] pread64(6, <unfinished ...>
[pid 24815] pread64(4, <unfinished ...>
[pid 24816] <... 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"..., 48, 848) = 48
[pid 24815] <... 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 24816] pread64(6,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68,

```



```

896) = 68
[pid 24815] pread64(4, <unfinished ...>
[pid 24816] newfstatat(6, "", <unfinished ...>
[pid 24815] <... 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"... , 48, 848) = 48
[pid 24816] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0
[pid 24815] pread64(4, <unfinished ...>
[pid 24816] pread64(6, <unfinished ...>
[pid 24815] <... pread64
resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"...
, 68,
896) = 68
[pid 24816] <... 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 24815] newfstatat(4, "", <unfinished ...>
[pid 24816] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 6, 0 <unfinished
...>
[pid 24815] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0
[pid 24816] <... mmap resumed>          = 0x7fc9e5b9b000
[pid 24815] pread64(4, <unfinished ...>
[pid 24816] mprotect(0x7fc9e5bc3000, 2023424, PROT_NONE <unfinished ...>
[pid 24815] <... 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 24816] <... mprotect resumed>      = 0
[pid 24815] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0 <unfinished
...>
[pid 24816] mmap(0x7fc9e5bc3000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 6, 0x28000 <unfinished ...>
[pid 24815] <... mmap resumed>          = 0x7f10c7fd3000
[pid 24816] <... mmap resumed>          = 0x7fc9e5bc3000
[pid 24815] mprotect(0x7f10c7ffb000, 2023424, PROT_NONE <unfinished ...>
[pid 24816] mmap(0x7fc9e5d58000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 6, 0x1bd000 <unfinished ...>
[pid 24815] <... mprotect resumed>      = 0
[pid 24816] <... mmap resumed>          = 0x7fc9e5d58000
[pid 24815] mmap(0x7f10c7ffb000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x28000 <unfinished ...>
[pid 24816] mmap(0x7fc9e5db1000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 6, 0x215000 <unfinished ...>
[pid 24815] <... mmap resumed>          = 0x7f10c7ffb000
[pid 24816] <... mmap resumed>          = 0x7fc9e5db1000
[pid 24815] mmap(0x7f10c8190000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1bd000 <unfinished ...>
[pid 24816] mmap(0x7fc9e5db7000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

```

```

[pid 24815] <... mmap resumed>          = 0x7f10c8190000
[pid 24816] <... mmap resumed>          = 0x7fc9e5db7000
[pid 24815] mmap(0x7f10c81e9000, 24576, PROT_READ|PROT_WRITE,
  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x215000 <unfinished ...>
[pid 24816] close(6 <unfinished ...>
[pid 24815] <... mmap resumed>          = 0x7f10c81e9000
[pid 24816] <... close resumed>         = 0
[pid 24815] mmap(0x7f10c81ef000, 52816, PROT_READ|PROT_WRITE,
  MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 24816] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fc9e5b98000
[pid 24815] <... mmap resumed>          = 0x7f10c81ef000
[pid 24816] arch_prctl(ARCH_SET_FS, 0x7fc9e5b98740 <unfinished ...>
[pid 24815] close(4 <unfinished ...>
[pid 24816] <... arch_prctl resumed>    = 0
[pid 24815] <... close resumed>         = 0
[pid 24816] set_tid_address(0x7fc9e5b98a10 <unfinished ...>
[pid 24815] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
  <unfinished ...>
[pid 24816] <... set_tid_address resumed> = 24816
[pid 24815] <... mmap resumed>          = 0x7f10c7fd0000
[pid 24816] set_robust_list(0x7fc9e5b98a20, 24 <unfinished ...>
[pid 24815] arch_prctl(ARCH_SET_FS, 0x7f10c7fd0740 <unfinished ...>
[pid 24816] <... set_robust_list resumed> = 0
[pid 24815] <... arch_prctl resumed>    = 0
[pid 24816] rseq(0x7fc9e5b990e0, 0x20, 0, 0x53053053) = 0
[pid 24815] set_tid_address(0x7f10c7fd0a10 <unfinished ...>
[pid 24816] mprotect(0x7fc9e5db1000, 16384, PROT_READ <unfinished ...>
[pid 24815] <... set_tid_address resumed> = 24815
[pid 24816] <... mprotect resumed>      = 0
[pid 24815] set_robust_list(0x7f10c7fd0a20, 24 <unfinished ...>
[pid 24816] mprotect(0x558ee91be000, 4096, PROT_READ <unfinished ...>
[pid 24815] <... set_robust_list resumed> = 0
[pid 24816] <... mprotect resumed>      = 0
[pid 24815] rseq(0x7f10c7fd10e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 24816] mprotect(0x7fc9e5e03000, 8192, PROT_READ <unfinished ...>
[pid 24815] <... rseq resumed>          = 0
[pid 24816] <... mprotect resumed>      = 0
[pid 24816] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>
[pid 24815] mprotect(0x7f10c81e9000, 16384, PROT_READ <unfinished ...>
[pid 24816] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY} = 0
[pid 24815] <... mprotect resumed>      = 0
[pid 24816] munmap(0x7fc9e5dc4000, 18383) = 0
[pid 24815] mprotect(0x561f1e00e000, 4096, PROT_READ <unfinished ...>
[pid 24816] getpid( <unfinished ...>
[pid 24815] <... mprotect resumed>      = 0
[pid 24816] <... getpid resumed>        = 24816
[pid 24815] mprotect(0x7f10c823b000, 8192, PROT_READ <unfinished ...>
[pid 24816] openat(AT_FDCWD, "file2.txt", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600

```

```

<unfinished ...>
[pid 24815] <... mprotect resumed>      = 0
[pid 24815] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
    rlim_max=RLIM64_INFINITY}) = 0
[pid 24815] munmap(0x7f10c81fc000, 18383) = 0
[pid 24815] getpid()                    = 24815
[pid 24815] openat(AT_FDCWD, "file1.txt", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600
    <unfinished ...>
[pid 24816] <... openat resumed>         = 6
[pid 24816] read(0, <unfinished ...>
[pid 24815] <... openat resumed>         = 4
[pid 24815] read(0, <unfinished ...>
[pid 24814] <... clock_nanosleep resumed>0x7ffe0e06a240) = 0
[pid 24814] write(1, "Input strings:\n\0", 16Input strings:
) = 16
[pid 24814] read(0, hahahajaha
    "hahahajaha\n", 4096) = 11
[pid 24814] write(6, "hahahajaha\0", 11) = 11
[pid 24816] <... read resumed>"hahahajaha\0", 4096) = 11
[pid 24814] read(0, <unfinished ...>
[pid 24816] write(6, "hhhjh\n", 6)      = 6
[pid 24816] read(0, adsuaiuiuwradacvcz213
    <unfinished ...>
[pid 24814] <... read resumed>"adsuaiuiuwradacvcz213\n", 4096) = 22
[pid 24814] write(4, "adsuaiuiuwradacvcz213\0", 22) = 22
[pid 24815] <... read resumed>"adsuaiuiuwradacvcz213\0", 4096) = 22
[pid 24814] read(0, <unfinished ...>
[pid 24815] write(4, "dswrdvcz213\n", 13) = 13
[pid 24815] read(0, 9833ajshdjnnzxm
    <unfinished ...>
[pid 24814] <... read resumed>"9833ajshdjnnzxm\n", 4096) = 17
[pid 24814] write(4, "9833ajshdjnnzxm\0", 17) = 17
[pid 24815] <... read resumed>"9833ajshdjnnzxm\0", 4096) = 17
[pid 24814] read(0, <unfinished ...>
[pid 24815] write(4, "9833jshdjnnzxm\n", 16) = 16
[pid 24815] read(0, alsk334jf
    <unfinished ...>
[pid 24814] <... read resumed>"alsk334jf\n", 4096) = 10
[pid 24814] write(6, "alsk334jf\0", 10) = 10
[pid 24816] <... read resumed>"alsk334jf\0", 4096) = 10
[pid 24814] read(0, <unfinished ...>
[pid 24816] write(6, "lsk334jf\n", 9)   = 9
[pid 24816] read(0, baba12
    <unfinished ...>
[pid 24814] <... read resumed>"baba12\n", 4096) = 7
[pid 24814] write(6, "baba12\0", 7)     = 7
[pid 24816] <... read resumed>"baba12\0", 4096) = 7
[pid 24814] read(0, <unfinished ...>
[pid 24816] write(6, "bb12\n", 5)      = 5

```

```

[pid 24816] read(0, dalk1
<unfinished ...>
[pid 24814] <... read resumed>"dalk1\n", 4096) = 6
[pid 24814] write(6, "dalk1\0", 6) = 6
[pid 24816] <... read resumed>"dalk1\0", 4096) = 6
[pid 24814] read(0, <unfinished ...>
[pid 24816] write(6, "dlk1\n", 5) = 5
[pid 24816] read(0, <unfinished ...>
[pid 24814] <... read resumed>"", 4096) = 0
[pid 24814] write(4, "\n", 1) = 1
[pid 24815] <... read resumed>"\n", 4096) = 1
[pid 24814] write(6, "\n", 1) = 1
[pid 24815] close(4 <unfinished ...>
[pid 24816] <... read resumed>"\n", 4096) = 1
[pid 24814] close(4 <unfinished ...>
[pid 24816] close(6 <unfinished ...>
[pid 24814] <... close resumed>) = 0
[pid 24814] close(0) = 0
[pid 24814] wait4(-1, <unfinished ...>
[pid 24815] <... close resumed>) = 0
[pid 24816] <... close resumed>) = 0
[pid 24815] exit_group(0 <unfinished ...>
[pid 24816] exit_group(0 <unfinished ...>
[pid 24815] <... exit_group resumed>) = ?
[pid 24816] <... exit_group resumed>) = ?
[pid 24815] +++ exited with 0 +++
[pid 24814] <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) =
24815
[pid 24814] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=24815,
si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
[pid 24816] +++ exited with 0 +++
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=24816, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 24816
wait4(-1, 0x7ffe0e06a290, 0, NULL) = -1 ECHILD (No child processes)
exit_group(0) = ?
+++ exited with 0 +++

```

## Вывод

В ходе написания данной лабораторной работы я научился работать с системными вызовами в СИ. Научился создавать программы, состоящие из нескольких процессов, и передавать данные между процессами по каналам. Во время отладки программы я познакомился с утилитой strace, она оказалась достаточно удобной для получения информации о работе многопоточных программ. Лабораторная работа была довольно интересна, так как я раньше не создавал программы на СИ, которые запускают несколько процессов параллельно.