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-210Б-23

Студент: Коростин Н.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 16.11..24

Москва, 2024

Постановка задачи

Вариант 15.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

Правило проверки: строка должна начинаться с заглавной буквы

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создает неименованный канал.
- int dup2(int oldfd, int newfd); – переназначение файлового дескриптора.
- int execv(const char *filename, char *const argv[]) – замена образа памяти процесса.
- int open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- int close(int fd) - закрыть файл

В родительском процессе (main.cpp) считывается имя файла для вывода и открывается этот файл. Далее создаются каналы с помощью **pipe()** и создается дочерний процесс вызовом **fork()**. Если мы остались в родительском процессе, то закрываются ненужные концы каналов и начинается считка строк, вводимых пользователем. Строки направляются (записываются) в **pipe1[1]** с помощью функции **write_to_file(int fd, const std::string&)** и попадают в дочерний процесс. Далее считываются все сообщения, пришедшие от дочернего процесса (строки, не начинающиеся с заглавной буквы) и выводятся в stdout пользователю. В конце закрываются все концы пайпов и файловый дескриптор.

В дочернем процессе (все еще в main.cpp) сначала производится переназначение файловых дескрипторов, чтобы дочерний процесс (в child.cpp) не знал о пайпах, а работал с обычными STD_OUT, STD_IN и STD_ERR. Строки считываются из потока ввода в цикле и обрабатываются функцией is_valid. Если строка начинается на заглавную букву, то она направляется в файл вывода (ребенок думает, что это STD_OUT). Если же не на заглавную букву, или в случае возникновения ошибок, child пишет в STD_ERR (обратно в родительский процесс). В конце ввода все концы пайпов и файловый дескриптор закрываются.

Код программы

main.c

```
#include <string>
#include <unistd.h>
#include "../include/io.h"
#include "../include/errors.h"
```

```
#include <sys/wait.h>
#include <fcntl.h>


int main(int argc, char* argv[]) {
    std::string file_name;
    print_to_stdout("Enter file name for output: ");
    read_from_stdin(file_name);

    int file_output = open(file_name.c_str(), O_WRONLY | O_CREAT | O_TRUNC | O_APPEND,
0600);
    if (file_output == -1) {
        log_errors(FILE_OPEN_ERROR);
        return 1;
    }

    int pipe1[2], pipe2[2];
    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
        log_errors(PIPE_ERROR);
        return 1;
    }

    pid_t child_pid = fork();

    if (child_pid == -1) {
        log_errors(FORK_ERROR);
        return 1;
    }

    if (child_pid == 0) {
        close(pipe1[1]);
        close(pipe2[0]);

        dup2(pipe1[0], STDIN_FILENO);
        dup2(pipe2[1], STDERR_FILENO);
        dup2(file_output, STDOUT_FILENO);

        close(pipe1[0]);
        close(pipe2[1]);
        close(file_output);

        char *args[] = {(char*)"child_exec", NULL};
        execv("./child_exec", args);
        log_errors(EXEC_ERROR);

        close(STDIN_FILENO);
        close(STDOUT_FILENO);
        close(STDERR_FILENO);
```

```
        return 1;
    }

    // Parent process

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

    std::string user_input;
    print_to_stdout("Enter strings (type 'exit' to quit):\n");
    while(read_input(STDIN_FILENO, user_input)) {
        if (user_input == "exit") break;
        user_input += "\n";
        write_to_file(pipe1[1], user_input);
    }

    close(pipe1[1]);

    std::string error_messages;

    while (read_input(pipe2[0], error_messages)) {
        print_to_stdout(error_messages + '\n');
        return 0;
    }

    close(pipe2[0]);
    close(file_output);
    return 0;
}
```

child.cpp

```
#include "../include/child.h"
#include <unistd.h>

int main() {
    std::string line;
    char buffer[1024];

    ssize_t bytes_read;
    while ((bytes_read = read_input(STDIN_FILENO, line)) > 0) {
        if (is_valid(line)) {
            print_to_stdout(line + '\n');
        } else {
            write_to_file(STDERR_FILENO, "Invalid input: " + line + '\n');
        }
    }

    return 0;
}
```

```
bool is_valid(std::string line) {
    return !line.empty() && std::isupper(line[0]);
}
```

errors.cpp

```
#include "../include/errors.h"

void log_errors(Error err) {
    ssize_t num;
    switch(err) {
        case EXEC_ERROR:
            write_to_file(STDERR_FILENO, "EXEC_ERROR\n");
            break;
        case FORK_ERROR:
            write_to_file(STDERR_FILENO, "FORK_ERROR\n");
            break;
        case PIPE_ERROR:
            write_to_file(STDERR_FILENO, "PIPE_ERROR\n");
            break;
        case STOP_ERROR:
            write_to_file(STDERR_FILENO, "STOP_ERROR\n");
            break;
        case READ_ERROR:
            write_to_file(STDERR_FILENO, "READ_FAILED\n");
            break;
        case WRITE_FAILED:
            write_to_file(STDERR_FILENO, "WRITE_FAILED\n");
            break;
        case INVALID_INPUT:
            write_to_file(STDERR_FILENO, "INVALID_INPUT\n");
            break;
        case FILE_OPEN_ERROR:
            write_to_file(STDERR_FILENO, "FILE_OPEN_ERROR\n");
            break;
        default:
            write_to_file(STDERR_FILENO, "UNKNOWN_ERROR\n");
            break;
    }
}
```

io.cpp

```
#include "../include/io.h"

ssize_t read_input(int input_file, std::string& message) {
    char buffer[BUFF_SIZE];
    ssize_t bytesRead;
```

```
    std::string output;

    while ((bytesRead = read(input_file, buffer, BUFF_SIZE)) > 0) {
        output.append(buffer, bytesRead);
        if (output.back() == '\n') {
            output.erase(output.length() - 1);
            break;
        }
    }

    if (bytesRead == -1) {
        return -1;  // Ошибка при чтении
    }

    message = output;
    return static_cast<ssize_t>(output.size());  // Возвращаем количество считанных символов
}

ssize_t read_from_stdin(std::string& message) {
    return read_input(STDIN_FILENO, message);
}

ssize_t write_to_file(int file_output, const std::string& message) {
    return write(file_output, message.c_str(), message.size());
}

void print_to_stdout(const std::string& message) {
    write_to_file(STDOUT_FILENO, message);
}
```

Протокол работы программы

```
$ ./main_exec
Enter file name for output: /home/nikita/operation_systems/lab1/src/output
Enter strings (type 'exit' to quit):
alksjflk
aaaaaaaaa
kkkkklksjdalksjdf;lkajs;lkdf
Alksdlkjlskd
VVVVVVVVVVVV
12345
exit
Invalid input: alksjflk
Invalid input: aaaaaaaaa
Invalid input: kkkkklksjdalksjdf;lkajs;lkdf
Invalid input: 12345
```

```
$ cat < ../src/output
Alksdlkjlskd
VVVVVVVVVVVV
$ strace -f ./main_exec
execve("./main_exec", ["./main_exec"], 0x7fff9c560c08 /* 56 vars */) = 0
brk(NULL)                               = 0x619a9b57e000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x723134c54000
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=69427, ...}) = 0
mmap(NULL, 69427, PROT_READ, MAP_PRIVATE, 3, 0) = 0x723134c43000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.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\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x723134800000
mmap(0x72313489d000, 1343488, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9d000) = 0x72313489d000
mmap(0x7231349e5000, 552960, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e5000) = 0x7231349e5000
mmap(0x723134a6c000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x26b000) = 0x723134a6c000
mmap(0x723134a7a000, 12608, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x723134a7a000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x723134c15000
mmap(0x723134c19000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x4000) = 0x723134c19000
mmap(0x723134c3d000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000)
= 0x723134c3d000
mmap(0x723134c41000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2b000) = 0x723134c41000
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\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) = 0x723134400000
mmap(0x723134428000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x723134428000
mmap(0x7231345b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7231345b0000
mmap(0x7231345ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7231345ff000
mmap(0x723134605000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x723134605000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
```

```
     mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x723134b2c000
     mmap(0x723134b3c000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x723134b3c000
     mmap(0x723134bbb000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x723134bbb000
     mmap(0x723134c13000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x723134c13000
     close(3)                           = 0
     mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x723134b2a000
     mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x723134b27000
     arch_prctl(ARCH_SET_FS, 0x723134b27740) = 0
     set_tid_address(0x723134b27a10)    = 10170
     set_robust_list(0x723134b27a20, 24)     = 0
     rseq(0x723134b28060, 0x20, 0, 0x53053053) = 0
     mprotect(0x7231345ff000, 16384, PROT_READ) = 0
     mprotect(0x723134c13000, 4096, PROT_READ) = 0
     mprotect(0x723134c41000, 4096, PROT_READ) = 0
     mprotect(0x723134a6c000, 45056, PROT_READ) = 0
     mprotect(0x619a9a13c000, 4096, PROT_READ) = 0
     mprotect(0x723134c8c000, 8192, PROT_READ) = 0
     prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
     munmap(0x723134c43000, 69427)      = 0
     futex(0x723134a7a7bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
     getrandom("\x6a\xd4\xdd\x45\x09\x46\x2a\xdb", 8, GRND_NONBLOCK) = 8
     brk(NULL)                          = 0x619a9b57e000
     brk(0x619a9b59f000)                = 0x619a9b59f000
     write(1, "Enter file name for output: ", 28Enter file name for output: ) = 28
     read(0, /home/nikita/operation_systems/lab1/src/output
     "/home/nikita/operation_systems/l"..., 100) = 47
     openat(AT_FDCWD, "/home/nikita/operation_systems/lab1/src/output",
O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600) = 3
     pipe2([4, 5], 0)                   = 0
     pipe2([6, 7], 0)                   = 0
     clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x723134b27a10) = 10173
     strace: Process 10173 attached
     [pid 10170] close(4)               = 0
     [pid 10170] close(7 <unfinished ...>
     [pid 10173] set_robust_list(0x723134b27a20, 24 <unfinished ...>
     [pid 10170] <... close resumed>)   = 0
     [pid 10173] <... set_robust_list resumed>) = 0
     [pid 10170] write(1, "Enter strings (type 'exit' to qu"..., 37Enter strings (type
'exit' to quit):
      <unfinished ...>
     [pid 10173] close(5 <unfinished ...>
     [pid 10170] <... write resumed>)   = 37
     [pid 10173] <... close resumed>)   = 0
     [pid 10170] read(0,  <unfinished ...>
     [pid 10173] close(6)               = 0
     [pid 10173] dup2(4, 0)             = 0
     [pid 10173] dup2(7, 2)             = 2
     [pid 10173] dup2(3, 1)             = 1
```

```
[pid 10173] close(4)                    = 0
[pid 10173] close(7)                    = 0
[pid 10173] close(3)                    = 0
[pid 10173] execve("./child_exec", ["child_exec"], 0x7ffc3cd139a8 /* 56 vars */) = 0
[pid 10173] brk(NULL)                   = 0x5cd4b2957000
[pid 10173] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ac546c22000
[pid 10173] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid 10173] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 10173] fstat(3, {st_mode=S_IFREG|0644, st_size=69427, ...}) = 0
[pid 10173] mmap(NULL, 69427, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ac546c11000
[pid 10173] close(3)                    = 0
[pid 10173] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6",
O_RDONLY|O_CLOEXEC) = 3
[pid 10173] read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
[pid 10173] fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
[pid 10173] mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7ac546800000
[pid 10173] mmap(0x7ac54689d000, 1343488, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9d000) = 0x7ac54689d000
[pid 10173] mmap(0x7ac5469e5000, 552960, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e5000) = 0x7ac5469e5000
[pid 10173] mmap(0x7ac546a6c000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26b000) = 0x7ac546a6c000
[pid 10173] mmap(0x7ac546a7a000, 12608, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ac546a7a000
[pid 10173] close(3)                    = 0
[pid 10173] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC)
= 3
[pid 10173] read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
[pid 10173] fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
[pid 10173] mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7ac546be3000
[pid 10173] mmap(0x7ac546be7000, 147456, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7ac546be7000
[pid 10173] mmap(0x7ac546c0b000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7ac546c0b000
[pid 10173] mmap(0x7ac546c0f000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2b000) = 0x7ac546c0f000
[pid 10173] close(3)                    = 0
[pid 10173] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid 10173] 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\0"..., 832) = 832
[pid 10173] pread64(3,
"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
[pid 10173] fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 10173] pread64(3,
"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
[pid 10173] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7ac546400000
[pid 10173] mmap(0x7ac546428000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7ac546428000
[pid 10173] mmap(0x7ac5465b0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7ac5465b0000
[pid 10173] mmap(0x7ac5465ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7ac5465ff000
[pid 10173] mmap(0x7ac546605000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ac546605000
[pid 10173] close(3)                    = 0
[pid 10173] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
```

```
[pid 10173] read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
[pid 10173] fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
[pid 10173] mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7ac546afa000
[pid 10173] mmap(0x7ac546b0a000, 520192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10000) = 0x7ac546b0a000
[pid 10173] mmap(0x7ac546b89000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8f000) = 0x7ac546b89000
[pid 10173] mmap(0x7ac546be1000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe7000) = 0x7ac546be1000
[pid 10173] close(3)                    = 0
[pid 10173] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ac546af8000
[pid 10173] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ac546af5000
[pid 10173] arch_prctl(ARCH_SET_FS, 0x7ac546af5740) = 0
[pid 10173] set_tid_address(0x7ac546af5a10) = 10173
[pid 10173] set_robust_list(0x7ac546af5a20, 24) = 0
[pid 10173] rseq(0x7ac546af6060, 0x20, 0, 0x53053053) = 0
[pid 10173] mprotect(0x7ac5465ff000, 16384, PROT_READ) = 0
[pid 10173] mprotect(0x7ac546be1000, 4096, PROT_READ) = 0
[pid 10173] mprotect(0x7ac546c0f000, 4096, PROT_READ) = 0
[pid 10173] mprotect(0x7ac546a6c000, 45056, PROT_READ) = 0
[pid 10173] mprotect(0x5cd4b2741000, 4096, PROT_READ) = 0
[pid 10173] mprotect(0x7ac546c5a000, 8192, PROT_READ) = 0
[pid 10173] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 10173] munmap(0x7ac546c11000, 69427) = 0
[pid 10173] futex(0x7ac546a7a7bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
[pid 10173] getrandom("\x68\x3b\x4e\xf0\x40\x7b\x3b\x83", 8, GRND_NONBLOCK) = 8
[pid 10173] brk(NULL)                   = 0x5cd4b2957000
[pid 10173] brk(0x5cd4b2978000)         = 0x5cd4b2978000
[pid 10173] read(0, alksdlkfja;j;k;j
 <unfinished ...>
[pid 10170] <... read resumed>"alksdlkfja;j;k;j\n", 100) = 17
[pid 10170] write(5, "alksdlkfja;j;k;j\n", 17) = 17
[pid 10170] read(0,  <unfinished ...>
[pid 10173] <... read resumed>"alksdlkfja;j;k;j\n", 100) = 17
[pid 10173] write(2, "Invalid input: alksdlkfja;j;k;j\n", 32) = 32
[pid 10173] read(0, LAKJLSKD
 <unfinished ...>
[pid 10170] <... read resumed>"LAKJLSKD\n", 100) = 9
[pid 10170] write(5, "LAKJLSKD\n", 9)   = 9
[pid 10173] <... read resumed>"LAKJLSKD\n", 100) = 9
[pid 10170] read(0,  <unfinished ...>
[pid 10173] write(1, "LAKJLSKD\n", 9)   = 9
[pid 10173] read(0, laksjl;fkja
 <unfinished ...>
[pid 10170] <... read resumed>"laksjl;fkja\n", 100) = 12
[pid 10170] write(5, "laksjl;fkja\n", 12) = 12
[pid 10170] read(0,  <unfinished ...>
[pid 10173] <... read resumed>"laksjl;fkja\n", 100) = 12
[pid 10173] write(2, "Invalid input: laksjl;fkja\n", 27) = 27
```

```
[pid 10173] read(0, exit
 <unfinished ...>
[pid 10170] <... read resumed>"exit\n", 100) = 5
[pid 10170] close(5)                    = 0
[pid 10170] read(6, "Invalid input: alksdlkfja;j;k;j\n"..., 100) = 59
[pid 10173] <... read resumed>"", 100)  = 0
[pid 10170] write(1, "Invalid input: alksdlkfja;j;k;j\n"..., 59Invalid input:
alksdlkfja;j;k;j
Invalid input: laksjl;fkja
) = 59
[pid 10173] exit_group(0 <unfinished ...>
[pid 10170] exit_group(0 <unfinished ...>
[pid 10173] <... exit_group resumed>)   = ?
[pid 10170] <... exit_group resumed>)   = ?
[pid 10173] +++ exited with 0 +++
+++ exited with 0 +++
```

Вывод

В результате выполнения работы я научился пользоваться системными вызовами в Си. Я понял как работают процессы и организовал простейшее общение родительского и дочернего процессов. Одной из сложностей (кроме изучения документации незнакомых мне до этого syscalls) был дебаг, так как родительский и дочерний процессы выводят данные в разные потоки, в том числе в зависимости от самих данных. В этом мне помогли ранее реализованные мной функции для чтения/записи из/в поток.