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-216Б-24

Студент: Котляр Д.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 22.10.25

Москва, 2025

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.

Вариант 15) Правило проверки: строка должна начинаться с заглавной буквы

Общий метод и алгоритм решения

В рамках лабораторной работы была разработана многопроцессная система для проверки строк на соответствие заданному правилу с использованием двух неименованных каналов. Программа состоит из двух отдельных модулей: родительского процесса (parent.c), который управляет взаимодействием с пользователем, и дочернего процесса (child.c), выполняющего проверку строк.

Использованные системные вызовы

- **pid_t fork(void)** – создание дочернего процесса
- **int pipe(int fd)* – создание неименованных каналов для межпроцессного взаимодействия
- **void exit(int status)** – завершение процесса с возвратом статуса
- **int execl(const char *path, const char arg, ...)* – замена образа памяти процесса
- **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)** – закрытие файлового дескриптора
- **pid_t wait(int status)* – ожидание завершения дочернего процесса
- **ssize_t read(int fd, void buf, size_t count)* – чтение данных из файлового дескриптора
- **ssize_t write(int fd, const void buf, size_t count)* – запись данных в файловый дескриптор

Код программы

parent.c

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

#define BUFFER_SIZE 1024

int main() {
    int pipe1_fd[2];
    int pipe2_fd[2];
    pid_t pid;
    char filename[BUFFER_SIZE];
    char buffer[BUFFER_SIZE];
    int file_fd;

    if (pipe(pipe1_fd) == -1 || pipe(pipe2_fd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    printf("Введите имя файла для записи: ");
    fflush(stdout);
    if (fgets(filename, BUFFER_SIZE, stdin) == NULL) {
        perror("fgets");
        exit(EXIT_FAILURE);
    }
    filename[strcspn(filename, "\n")] = 0;


    file_fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (file_fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {

        close(pipe1_fd[1]);
        close(pipe2_fd[0]);

        dup2(pipe1_fd[0], STDIN_FILENO);
        close(pipe1_fd[0]);
        dup2(pipe2_fd[1], STDOUT_FILENO);
        close(pipe2_fd[1]);
```

```
        execl("./child", "child", NULL);
        perror("execl");
        exit(EXIT_FAILURE);
    } else {
        close(pipe1_fd[0]);
        close(pipe2_fd[1]);

        printf("Вводите строки:\n");
        while (1) {
            printf("> ");
            fflush(stdout);
            if (fgets(buffer, BUFFER_SIZE, stdin) == NULL) {
                perror("fgets");
                break;
            }

            if (strcmp(buffer, "exit\n") == 0) {
                break;
            }

            write(pipe1_fd[1], buffer, strlen(buffer));

            ssize_t bytes_read = read(pipe2_fd[0], buffer, BUFFER_SIZE - 1);
            if (bytes_read > 0) {
                buffer[bytes_read] = '\0';
                if (strstr(buffer, "ERROR") != NULL) {
                    printf("Ошибка от child: %s", buffer);
                } else {
                    write(file_fd, buffer, strlen(buffer));
                }
            }
        }

        close(pipe1_fd[1]);
        close(pipe2_fd[0]);
        close(file_fd);
        wait(NULL);
    }

    return 0;
}
```

child.c

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

#define BUFFER_SIZE 1024

int main() {
    char buffer[BUFFER_SIZE];

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

```
            break;
        }
        buffer[bytes_read] = '\0';

        if (isupper((unsigned char)buffer[0])) {
            write(STDOUT_FILENO, buffer, bytes_read);
        } else {
            char error_msg[BUFFER_SIZE];
            char truncated_buffer[256];
            int len = bytes_read < 200 ? bytes_read : 200;
            strncpy(truncated_buffer, buffer, len);
            truncated_buffer[len] = '\0';

            snprintf(error_msg, BUFFER_SIZE,
                    "ERROR: Строка не начинается с заглавной буквы: %s\n",
                    truncated_buffer);
            write(STDOUT_FILENO, error_msg, strlen(error_msg));
        }
    }

    return 0;
}
```

Протокол работы программы

Strace

```
6143  execve("./parent", ["./parent"], 0x7fffe956d0b8 /* 64 vars */) = 0
6143  brk(NULL)                         = 0x5fb108e45000
6143  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fad09267000
6143  access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
6143  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
6143  fstat(3, {st_mode=S_IFREG|0644, st_size=74387, ...}) = 0
6143  mmap(NULL, 74387, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fad09254000
6143  close(3)                          = 0
6143  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
6143  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
6143  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
6143  fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
6143  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
6143  mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fad09000000
6143  mmap(0x7fad09028000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fad09028000
6143  mmap(0x7fad091b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7fad091b0000
6143  mmap(0x7fad091ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7fad091ff000
6143  mmap(0x7fad09205000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fad09205000
6143  close(3)                          = 0
6143  mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fad09251000
6143  arch_prctl(ARCH_SET_FS, 0x7fad09251740) = 0
6143  set_tid_address(0x7fad09251a10)   = 6143
6143  set_robust_list(0x7fad09251a20, 24) = 0
6143  rseq(0x7fad09252060, 0x20, 0, 0x53053053) = 0
6143  mprotect(0x7fad091ff000, 16384, PROT_READ) = 0
```

```
6143  mprotect(0x5fb0e5ec8000, 4096, PROT_READ) = 0
6143  mprotect(0x7fad092a5000, 8192, PROT_READ) = 0
6143  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
6143  munmap(0x7fad09254000, 74387)     = 0

6143  pipe2([3, 4], 0)                  = 0
6143  pipe2([5, 6], 0)                  = 0

6143  fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
6143  getrandom("\xb7\x63\xc7\x2e\x82\xa2\x0c\x6c", 8, GRND_NONBLOCK) = 8
6143  brk(NULL)                          = 0x5fb108e45000
6143  brk(0x5fb108e66000)                = 0x5fb108e66000
6143  write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217
\321\204\320\260\320\271\320\273\320\260"..., 54) = 54
6143  fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
6143  read(0, "file.txt\n", 1024)       = 9
6143  openat(AT_FDCWD, "file.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 7

6143  clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fad09251a10) = 6184

6184  set_robust_list(0x7fad09251a20, 24 <unfinished ...>
6143  close(3 <unfinished ...>
6184  <... set_robust_list resumed>)    = 0
6143  <... close resumed>)              = 0
6143  close(6)                          = 0
6184  close(4 <unfinished ...>
6143  write(1, "\320\222\320\262\320\276\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\320\270:\n", 29 <unfinished ...>
6184  <... close resumed>)              = 0
6143  <... write resumed>)              = 29
6143  write(1, "> ", 2 <unfinished ...>
6184  close(5 <unfinished ...>
6143  <... write resumed>)              = 2
6184  <... close resumed>)              = 0
6143  read(0,  <unfinished ...>
6184  dup2(3, 0)                        = 0
6184  close(3)                          = 0
6184  dup2(6, 1)                        = 1
6184  close(6)                          = 0
6184  execve("./child", ["child"], 0x7fffb63b87b8 /* 64 vars */) = 0
6184  brk(NULL)                         = 0x608fda30a000
6184  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7d32e50cc000
6184  access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
6184  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
6184  fstat(3, {st_mode=S_IFREG|0644, st_size=74387, ...}) = 0
6184  mmap(NULL, 74387, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7d32e50b9000
6184  close(3)                          = 0
6184  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
6184  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
6184  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
6184  fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
6184  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
6184  mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7d32e4e00000
6184  mmap(0x7d32e4e28000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7d32e4e28000
6184  mmap(0x7d32e4fb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7d32e4fb0000
6184  mmap(0x7d32e4fff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7d32e4fff000
6184  mmap(0x7d32e5005000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7d32e5005000
```

```
6184  close(3)                          = 0
6184  mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7d32e50b6000
6184  arch_prctl(ARCH_SET_FS, 0x7d32e50b6740) = 0
6184  set_tid_address(0x7d32e50b6a10)   = 6184
6184  set_robust_list(0x7d32e50b6a20, 24) = 0
6184  rseq(0x7d32e50b7060, 0x20, 0, 0x53053053) = 0
6184  mprotect(0x7d32e4fff000, 16384, PROT_READ) = 0
6184  mprotect(0x608fb8b4a000, 4096, PROT_READ) = 0
6184  mprotect(0x7d32e510a000, 8192, PROT_READ) = 0
6184  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
6184  munmap(0x7d32e50b9000, 74387)     = 0
6184  read(0,  <unfinished ...>
6143  <... read resumed>"Stroka 1\n", 1024) = 9
6143  write(4, "Stroka 1\n", 9)         = 9
6184  <... read resumed>"Stroka 1\n", 1023) = 9
6143  read(5,  <unfinished ...>
6184  write(1, "Stroka 1\n", 9 <unfinished ...>
6143  <... read resumed>"Stroka 1\n", 1023) = 9
6184  <... write resumed>)              = 9
6143  write(7, "Stroka 1\n", 9 <unfinished ...>
6184  read(0,  <unfinished ...>
6143  <... write resumed>)              = 9
6143  write(1, "> ", 2)                 = 2
6143  read(0, "stroka 2\n", 1024)       = 9
6143  write(4, "stroka 2\n", 9)         = 9
6184  <... read resumed>"stroka 2\n", 1023) = 9
6184  write(1, "ERROR: \320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\320\275\320\260\321\207\320"..., 90 <unfinished ...>
6143  read(5,  <unfinished ...>
6184  <... write resumed>)              = 90
6143  <... read resumed>"ERROR: \320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\320\275\320\260\321\207\320"..., 1023) = 90
6143  write(1, "\320\236\321\210\320\270\320\261\320\272\320\260 \320\276\321\202 child: ERROR: "..., 115
<unfinished ...>
6184  read(0,  <unfinished ...>
6143  <... write resumed>)              = 115
6143  write(1, "> ", 2)                 = 2
6143  read(0, "MEGA STRING\n", 1024)    = 12
6143  write(4, "MEGA STRING\n", 12)     = 12
6184  <... read resumed>"MEGA STRING\n", 1023) = 12
6143  read(5,  <unfinished ...>
6184  write(1, "MEGA STRING\n", 12)     = 12
6143  <... read resumed>"MEGA STRING\n", 1023) = 12
6184  read(0,  <unfinished ...>
6143  write(7, "MEGA STRING\n", 12)     = 12
6143  write(1, "> ", 2)                 = 2
6143  read(0, "popa\n", 1024)           = 5
6143  write(4, "popa\n", 5)             = 5
6184  <... read resumed>"popa\n", 1023) = 5
6143  read(5,  <unfinished ...>
6184  write(1, "ERROR: \320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\320\275\320\260\321\207\320"..., 86 <unfinished ...>
6143  <... read resumed>"ERROR: \320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\320\275\320\260\321\207\320"..., 1023) = 86
6184  <... write resumed>)              = 86
6143  write(1, "\320\236\321\210\320\270\320\261\320\272\320\260 \320\276\321\202 child: ERROR: "..., 111
<unfinished ...>
6184  read(0,  <unfinished ...>
6143  <... write resumed>)              = 111
6143  write(1, "> ", 2)                 = 2
6143  read(0, "", 1024)                 = 0
6143  dup(2)                            = 3
```

```
6143  fcntl(3, F_GETFL)                 = 0x2 (flags O_RDWR)
6143  fstat(3, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
6143  write(3, "fgets: Success\n", 15)  = 15
6143  close(3)                          = 0
6143  close(4)                          = 0
6184  <... read resumed>"", 1023)       = 0
6143  close(5)                          = 0
6143  close(7 <unfinished ...>
6184  exit_group(0 <unfinished ...>
6143  <... close resumed>)              = 0
6184  <... exit_group resumed>)         = ?
6143  wait4(-1,  <unfinished ...>
6184  +++ exited with 0 +++
6143  <... wait4 resumed>NULL, 0, NULL) = 6184
6143  --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=6184, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---
6143  exit_group(0)                     = ?
6143  +++ exited with 0 +++
```

Вывод

Программа корректно создает дочерний процесс и организует передачу данных между процессами с помощью pipe, что подтверждается анализом системных вызовов strace.

Основные проблемы при выполнении работы возникли с пониманием принципов работы с каналами pipe, особенно в области правильного закрытия файловых дескрипторов и организации двунаправленной коммуникации между процессами. Также сложности вызвало перенаправление стандартных потоков ввода-вывода и синхронизация работы родительского и дочернего процессов.