Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу
«Операционные системы»

Студент: Скрипачев Фёдор Михайлович
Группа: М8О-209Б-23
Вариант: 16
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2025

Содержание

Репозиторий

https://github.com/gthcbr25/osi/tree/main/oslab1

Постановка задачи

Цель работы

Приобретение практических навыков в:
Управление процессами в ОС
Обеспечение обмена данных между процессами посредством каналов

Задание

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 оканчиваться на «.» или «;».

Исходный код

parent.cpp

```
#include <iostream>

#include <unistd.h>

#include <string.h>

#include <string>

#include <sys/wait.h>

#include <fcntl.h>

#include <vector>


int main() {

    std::string file_name1;

    std::string file_name2;

    char c = 1;
```

```
write(STDOUT_FILENO, "Enter file name 1: ", 20);
while (c != '\n') {
    read(STDIN_FILENO, &c, sizeof(char));
    if (c != '\n') {
        file_name1 += c;
    }
}
c = 1;

write(STDOUT_FILENO, "Enter file name 2: ", 20);
while (c != '\n') {
    read(STDIN_FILENO, &c, sizeof(char));
    if (c != '\n') {
        file_name2 += c;
    }
}



int fd1[2], fd2[2];
int tmp = pipe(fd1);
if (tmp == -1) {
    write(STDERR_FILENO, "An error occured with creating pipe1", 37);
    return 1;
}
tmp = pipe(fd2);
if (tmp == -1) {
    write(STDERR_FILENO, "An error occured with creating pipe2", 37);
    return 1;
}
```

```
    pid_t process_id1 = fork();
    pid_t process_id2 = fork();
    if (process_id1 == -1) {
        write(STDERR_FILENO, "An error occured with creating child process 1",
47);
        return 1;
    }
    if (process_id2 == -1) {
        write(STDERR_FILENO, "An error occured with creating child process 2",
47);
        return 1;
    }
    if (process_id1 == 0 && process_id2 > 0) {  //child process 1
        close(fd1[1]);
        close(fd2[0]);
        close(fd2[1]);
        tmp = dup2(fd1[0], STDIN_FILENO);
        if (tmp == -1) {
            write(STDERR_FILENO, "An error occured with redirecting input 1", 40);
            return 1;
        }

        tmp = execl("child1", "child1", file_name1.c_str(), NULL);
        if (tmp == -1) {
            write(STDERR_FILENO, "An error occured with runing program from a
child process 1", 59);
            return 1;
        }
```

```
        exit(EXIT_FAILURE);
    }

    if (process_id1 > 0 && process_id2 == 0) {  //child process 2
        close(fd1[0]);
        close(fd1[1]);
        close(fd2[1]);
        tmp = dup2(fd2[0], STDIN_FILENO);
        if (tmp == -1) {
            write(STDERR_FILENO, "An error occured with redirecting input 2", 40);
            return 1;
        }

        tmp = execl("child2", "child2", file_name2.c_str(), NULL);
        if (tmp == -1) {
            write(STDERR_FILENO, "An error occured with runing program from a
child process 2", 58);
            return 1;
        }
        exit(EXIT_FAILURE);
    }

    if (process_id1 > 0 && process_id2 > 0) {    //parent process
        close(fd1[0]);
        close(fd2[0]);
        char c;
        char prev = '?';
        std::vector<char> vec;
        while(read(STDIN_FILENO, &c, 1)) {
```

```
            vec.push_back(c);
            if (c == '\n') {
                if (vec.size() > 11){
                    for (int i = 0; i < vec.size(); ++i) {
                        write(fd2[1], &vec[i], 1);
                    }
                } else {
                    for (int i = 0; i < vec.size(); ++i) {
                        write(fd1[1], &vec[i], 1);
                    }
                }
                vec.clear();
            }
            prev = c;
        }
        close(fd1[1]);
        close(fd2[1]);
        wait(nullptr);
    }
    return 0;
}
```

Child.cpp

```
#include <iostream>
#include <unistd.h>
#include <vector>
#include <fcntl.h>
#include <string>

int main(int argc, char* argv[]) {
```

```
    int file = open(argv[1], O_CREAT | O_WRONLY, S_IRWXU);
    if (file == -1) {
        write(STDERR_FILENO, "An error occured with opening file", 35);
        return 1;
    }
    int temp = ftruncate(file, 0);
    if (temp == -1) {
        write(STDERR_FILENO, "An error occured with clearing file", 36);
        return 1;
    }

    int tmp = dup2(file, STDOUT_FILENO);
    if (tmp == -1) {
        write(STDERR_FILENO, "An error occured with redirecting stdout to file",
48);
        close(file);
        return 1;
    }

    std::vector<char> vec;
    char c;
    while (read(STDIN_FILENO, &c, 1)) {
        vec.push_back(c);
        if (c == '\n') {
                for (int i = vec.size() - 2; i >= 0; --i) {
                    write(STDOUT_FILENO, &vec[i], 1);
                }
                write(STDOUT_FILENO, &vec.back(), 1);
```

```
            vec.clear();
        }
    }
    close(file);
    return 0;
}
```

Демонстрация работы программы

```
Enter file name 1: a.txt
Enter file name 2: b.txt
Hello.
qwerty
Hi;
a.txt
Hello.
Hi;
b.txt
qwerty
```

Выводы

Я научился использовать fork и обмениваться данными при помощи pipe-ов. При использовании fork создается копия вашего текущего процесса, и неправильная работа может привести к неожиданным результатам, однако создание процессов крайне полезно, когда вам нужно выполнять несколько действий параллельно. Также важно работать с чтением и записью из канала, помня, что read, write возвращает количество байт и оно необязательно равно тому значению, которое вы указали.