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

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

**Тема работы**

**“Взаимодействие между процессами”**

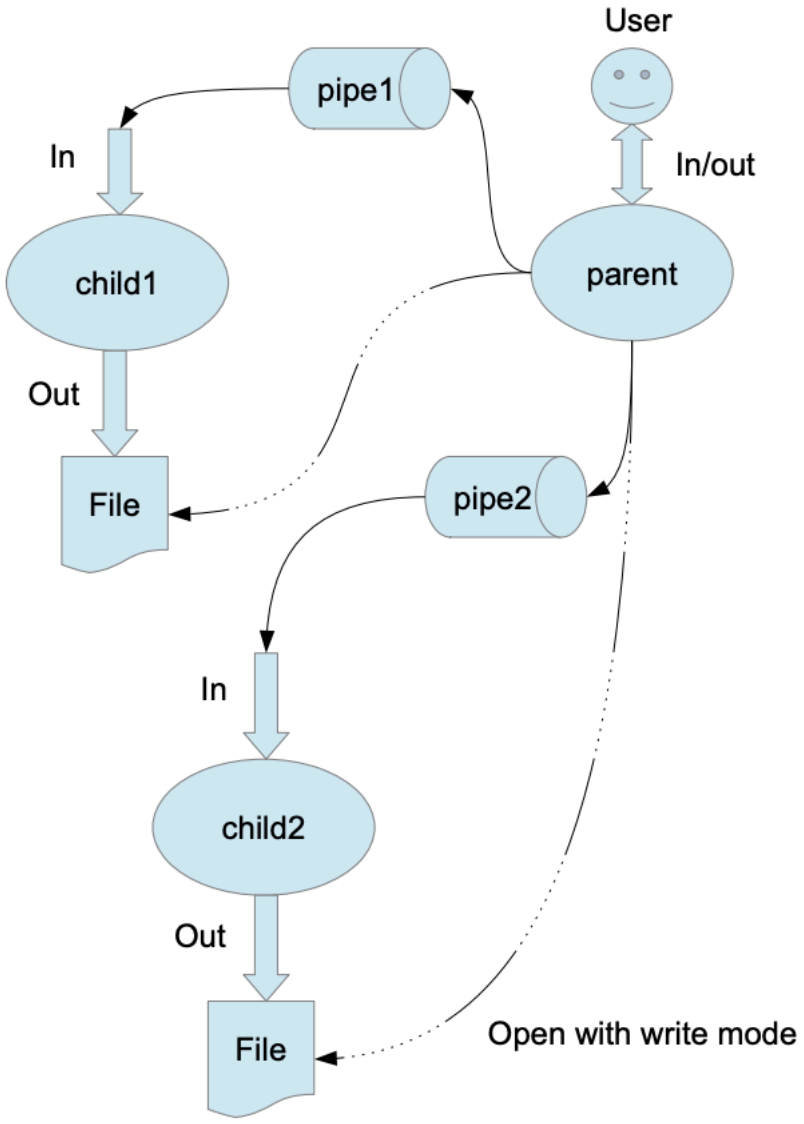
Студент: Слободин Никита Алексеевич
Группа: М8О-203Б-23
Вариант:21

Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

**Постановка задачи**

**Задача:** Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работ




**Вариант 21)** Правило фильтрации: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние процессы инвертируют строки.

## Общие сведения

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 дочерние 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.

Исходный код программы представлен в приложении.

**Вывод:**

При разработке программы я ознакомился с различными функциями работы с процессами. Функция fork используется для создания дочерних процессов, pipe - для создания каналов для передачи данных между процессами, dup2 - для перенаправления потоков ввода-вывода, а execv - для запуска новой программы в контексте дочернего процесса.

Приложение

src/child1.cpp

```
#include "utils.h"
#include <unistd.h>
#include <iostream>
#include <cstdio>

int main(int argc, char** argv) {
    if (argc != 1) {
        std::cerr << "Дочерний процесс 1: Необходимо передать имя файла для записи как
аргумент" << std::endl;
        return 1;
    }

    FILE* file = fopen(argv[0], "w");
    if (!file) {
        std::perror("Дочерний процесс 1: Не удалось открыть файл");
        return 1;
    }

    dup2(fileno(file), STDOUT_FILENO);

    ReadData([](const std::string& str) {
        std::string res = Modify(str);
        write(STDOUT_FILENO, res.c_str(), res.size());
    }, std::cin);

    fclose(file);

    return 0;
}
```

src/parent.cpp

```
#include "parent.h"
#include "utils.h"
#include <sys/wait.h>
#include <unistd.h>
#include <iostream>

bool FileExists(const char* filename) {
    return access(filename, F_OK) != -1;  // F_OK проверяет существование файла
}

bool StartProcess(int * pipe, const char* childPath, const char* filePath) {
    pid_t pid = fork();

    if (pid == -1){
        perror("Can't fork process");
        exit(-1);
```

```
    }

    if (pid == 0){
        close(pipe[WRITE_END]);
        dup2(pipe[READ_END], READ_END);

        close(pipe[READ_END]);

        if (execl(childPath, filePath, nullptr) == -1){
            std::cout << "Something went wrong when creating process " << childPath <<
std::endl;
        }
    }

    return pid == 0;
}

void ParentRoutine(const char* pathToChild1, const char* pathToChild2, std::istream&
input) {
    char filename1[256];
    char filename2[256];

    std::cout << "Enter filename for 1 process: " << std::endl;
    input.getline(filename1, 256);
    std::cout << "Enter filename for 2 process: " << std::endl;
    input.getline(filename2, 256);

    // Проверяем существование файлов
    if (!FileExists(filename1)) {
        std::cerr << "Error: File " << filename1 << " does not exist." << std::endl;
        return;  // Завершаем выполнение, если файл не существует
    }

    if (!FileExists(filename2)) {
        std::cerr << "Error: File " << filename2 << " does not exist." << std::endl;
        return;  // Завершаем выполнение, если файл не существует
    }

    int pipe1[2], pipe2[2];
    CreatePipe(pipe1);
    CreatePipe(pipe2);

    std::cout << "FILE NAMES: "<< filename1 << " " << filename2 << std::endl;

    if (StartProcess(pipe2, pathToChild2, filename2)) return;
    if (StartProcess(pipe1, pathToChild1, filename1)) return;

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

    size_t lineNumber = 1;
    std::cout << "Enter strings to process: " << std::endl;
```

```
    ReadData([pipe1, pipe2, &lineNumber](const std::string& str) {
        if (lineNumber % 2 == 1) { write(pipe1[WRITE_END], str.c_str(), str.size()); }
        else { write(pipe2[WRITE_END], str.c_str(), str.size()); }
        lineNumber++;
    }, input);

    write(pipe1[WRITE_END], "\n", 1); // Чтобы сработал выход в ReadData дочернего
процесса
    write(pipe2[WRITE_END], "\n", 1);

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

    // close(pipe1[READ_END]);
    // close(pipe2[READ_END]);

    wait(nullptr);
    wait(nullptr);
}
```

src/utils.cpp

```
#include "utils.h"
#include <unistd.h>
#include <cstring>
#include <cstdio>
#include <algorithm>

void CreatePipe(int pipeFd[2]) {
    if (pipe(pipeFd) == -1) {
        std::perror("Couldn't create pipe.");
        exit(EXIT_FAILURE);
    }
}

void ReadData(const std::function<void(const std::string&)>& handler, std::istream&
stream){
    std::string buff;

    while (std::getline(stream, buff)){
        if (buff.empty()){
            return;
        }
        handler(buff + '\n');
    }
}

std::string Modify(const std::string& str) {
    std::string result;
    if (!str.empty() && str.back() == '\n') {
        result = str.substr(0, str.size() - 1);
```

```
        std::reverse(result.begin(), result.end());
        result += '\n';
    } else {
        result = str;
        std::reverse(result.begin(), result.end());
    }
    return result;
}
```

Пример вывода:

```
root@c34508d80232:/workspaces/OS_MAI_Slobodin/build# ./lab1/parent

Enter filename for 1 process:

/workspaces/OS_MAI_Slobodin/file1.txt

Enter filename for 2 process:

/workspaces/OS_MAI_Slobodin/file2.txt

FILE NAMES: /workspaces/OS_MAI_Slobodin/file1.txt
/workspaces/OS_MAI_Slobodin/file2.txt

Enter strings to process:

abc

123

def

567


root@c34508d80232:/workspaces/OS_MAI_Slobodin/build#
```