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

# Лабораторная работа №2 по курсу
# «Операционные системы»

Студент: Соколов Арсений Игоревич
Группа: М8О-207Б-21
Вариант: 17
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

**Содержание**

## Репозиторий

https://github.com/valerasaray/os

## Постановка задачи

# Цель работы

Приобретение практических навыков в управлении процессами в ОС, обеспечение обмена данных между процессами посредством каналов.

# Задание

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).

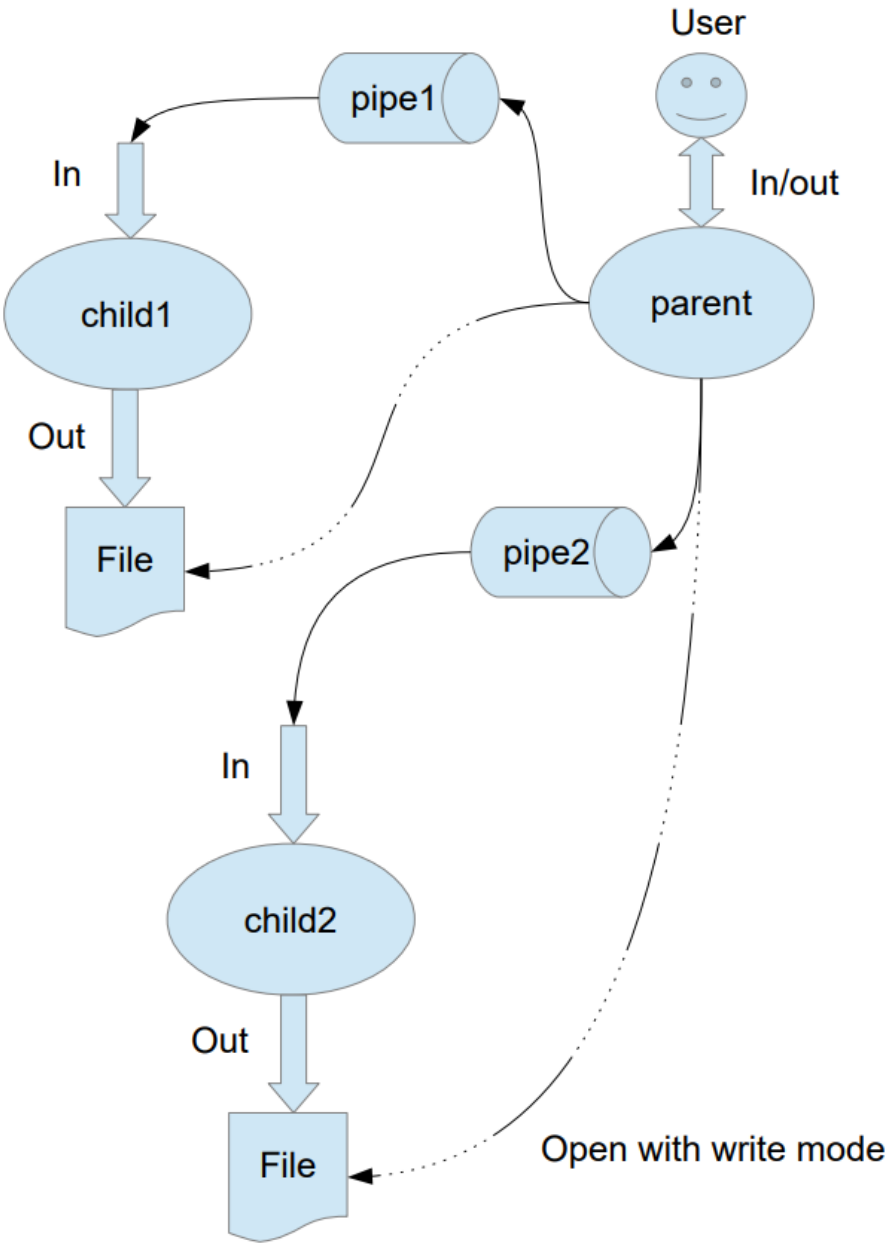
Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

**Группа вариантов 5**

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.

**Вариант 17**

Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы удаляют все гласные из строк.




## Общие сведения о программе

Программа компилируется из файла main.cpp, child.cpp. В программе используются следующие системные вызовы:

1. pipe() - существует для передачи информации между различными процессами.
2. fork() - создает новый процесс.
3. execpl() - передает процесс на исполнение другой программе.
4. read() - читает данные из файла.
5. write() - записывает данные в файл.
6. close() - закрывает файл.
7. exit() - завершает процесс

## Общий метод и алгоритм решения

Пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на запись для дочернего процесса. Аналогично для второй строки и второго дочернего процесса. Родительский процесс принимает от пользователя строки произвольной длины и пересылает строки длины больше 10 символов в pipe2, иначе в pipe1. Дочерние процессы удаляют все гласные из строк и пишут результаты своей работы в стандартный вывод.

## Исходный код

### main.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <sys/types.h>
#include <unistd.h>
#include <fstream>
#include <errno.h>
#include <signal.h>
#include <sys/wait.h>

using namespace std;

int main()
{
    string current_str; // текущая строка
    int child_tag;
    int fd[2];
    fstream res_file;
    string child1, child2;
    cout << "Введите имя для первого дочернего файла: ";
    cin >> child1;
    cout << "Введите имя для второго дочернего файла: ";
    cin >> child2;
    int fd1[2];
    int fd2[2];
```

```
    if (pipe(fd1) == -1) // проверка на ошибки
    {
        cout << "Произошла ошибка pipe" << endl;
        exit(EXIT_FAILURE);
    }
    if (pipe(fd2) == -1)
    {
        cout << "Произошла ошибка pipe" << endl;
        exit(EXIT_FAILURE);
    }

    pid_t f_id1 = fork(); // fork

    if (f_id1 == -1)  // проверка на ошибки
    {
        cout << "Ошибка fork с кодом -1, возвращенным в
родительском процессе, child1 не создан" << endl;
        exit(EXIT_FAILURE);
    }
    else if (f_id1 == 0)
    {
        fd[1] = fd1[1];
        fd[0] = fd1[0];
        string child = child1;
        execlp("./child", to_string(fd[0]).c_str(),
to_string(fd[1]).c_str(), child.c_str(), NULL);
        perror("Execlp error");
    }

    pid_t f_id2 = fork(); // fork

    if (f_id2 == -1) // проверка на ошибки
    {
        cout << "Ошибка fork с кодом -1, возвращенным в
родительском процессе, child2 не создан" << endl;
        exit(EXIT_FAILURE);
    }
    else if (f_id2 == 0)
```

```
    {
        fd[1] = fd2[1];
        fd[0] = fd2[0];
        string child = child2;
        execlp("./child", to_string(fd[0]).c_str(),
to_string(fd[1]).c_str(), child.c_str(), NULL);
        perror("Ошибка execlp");
    }

    else
    {
        while (getline(std::cin, current_str))
        {
            int s_size = current_str.size() + 1;
            if (current_str.size() <= 10)
            {
                write(fd1[1], &s_size, sizeof(int));
                write(fd1[1], current_str.c_str(), s_size);
            }
            else
            {
                write(fd2[1], &s_size, sizeof(int));
                write(fd2[1], current_str.c_str(), s_size);
            }
        }
    }

    close(fd2[1]);
    close(fd1[1]);
    close(fd2[0]);
    close(fd1[0]);
    return 0;
}
```

**child.cpp**

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <sys/types.h>
#include <unistd.h>
#include <fstream>
#include <errno.h>
#include <string.h>
#include <set>
#include <algorithm>

using namespace std;

int main(int argc, char const *argv[])
{
    std::string vovels = "aoueiy";
    std::set<char> volSet(vovels.begin(), vovels.end());
    string filename = argv[2];
    int fd[2];
    fd[0] = stoi(argv[0]);
    fd[1] = stoi(argv[1]);
    fstream cur_file;
    cur_file.open(filename, fstream::in | fstream::out |
fstream::app);
    while (true)
    {
        int size_of_str;
        read(fd[0], &size_of_str, sizeof(int));
```

```
        char str_array[size_of_str];
        read(fd[0], &str_array, sizeof(char) * size_of_str);
        string result_str;
        for (int i = 0; i < size_of_str; i++) {
            if (volSet.find(std::tolower(str_array[i])) ==
volSet.cend()) {
                result_str.push_back(str_array[i]);
            }
        }
        cur_file << result_str << endl;
    }
    return 0;
}
```

## Демонстрация работы программы

```
[valerasaray@valerasaray build]$ ./main
Введите имя для первого дочернего файла: child1
Введите имя для второго дочернего файла: child2
dddfrrssd
aaaoedddkeeee
fffkfkaaasdkkkddk
dkfwoa
oaf
[valerasaray@valerasaray build]$ cat child1

dddfrrssd
dkfw
f
[valerasaray@valerasaray build]$ cat child2
dddk
fffkfksdkkkddk
[valerasaray@valerasaray build]$
```

## Выводы

В ходе выполнения лабораторной работы №2 я приобрел практические навыки в управлении процессами в ОС, и в обеспечении обмена данных между процессами посредством каналов.