Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №X по курсу**

**«Операционные системы»**

Студент: Казанцев Данила Игоревич

Группа: М8О-207Б-21

Вариант: 3

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/thgdanilaya/mai\_os\_labs

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в управлении процессов в С++ и обеспечении обмена данных между процессами посредством каналов.

**Задание**

Пользователь вводит команды вида: «число число число». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс производит деление первого числа, на последующие, а результат выводит в файл. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int. Количество чисел может быть произвольным

**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: unistd.h, fstream, vector, string В программе используются следующие системные вызовы:

1. pipe() - …

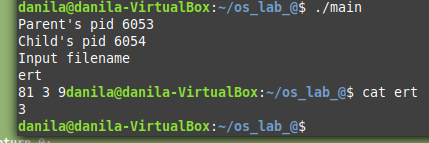
**Общий метод и алгоритм решения**

C помощью системного вызова fork создаются родительский и дочерний процессы, родительский процесс считывает название output файла и какое-то количество чисел, все это передается через pipe в дочерний процесс. Он же создает output файл и записывает в него результат деления.

**Исходный код**

|  |
| --- |
| #include <iostream> |
|  | #include "unistd.h" |
|  | #include "vector" |
|  | #include "fstream" |
|  | #include "string" |
|  |  |
|  | using namespace std; |
|  |  |
|  | int main() { |
|  | int pipefd[2]; |
|  | pipe(pipefd); |
|  | int id = fork(); |
|  |  |
|  | if (id == -1) { |
|  | return -1; |
|  | } |
|  | if (id == 0) { |
|  | string filename; |
|  | int lengthFilename; |
|  | read(pipefd[0], &lengthFilename, sizeof(int)); |
|  | for (int i = 0; i < lengthFilename; ++i) { |
|  | char c; |
|  | read(pipefd[0], &c, sizeof(char)); |
|  | filename.push\_back(c); |
|  | } |
|  | ofstream outfile(filename); |
|  | int numbersSize; |
|  | read(pipefd[0], &numbersSize, sizeof(int)); |
|  | int result; |
|  | int buff; |
|  | int current; |
|  | for (int i = 0; i < numbersSize; ++i) { |
|  | if (i == 0) { |
|  | read(pipefd[0], &buff, sizeof(int)); |
|  | } else { |
|  | read(pipefd[0], &current, sizeof(int)); |
|  | if (current == 0) { |
|  | exit(-1); |
|  | } else { |
|  | result = buff / current; |
|  | buff = current; |
|  | } |
|  | } |
|  | } |
|  | outfile << result << "\n"; |
|  | outfile.close(); |
|  | close(pipefd[0]); |
|  | close(pipefd[1]); |
|  | } else { |
|  | cout << "Parent's pid " << getpid() << "\n"; |
|  | cout << "Child's pid " << id << "\n"; |
|  | vector<int> nums; |
|  | string filename; |
|  | cout << "Input filename\n"; |
|  | cin >> filename; |
|  | int lengthFilename = filename.length(); |
|  | int number; |
|  | while (cin >> number) { |
|  | nums.push\_back(number); |
|  | } |
|  | int numbersSize = nums.size(); |
|  | write(pipefd[1], &lengthFilename, sizeof(int)); |
|  | for (int i = 0; i < lengthFilename; ++i) { |
|  | write(pipefd[1], &filename[i], sizeof(char)); |
|  | } |
|  | write(pipefd[1], &numbersSize, sizeof(int)); |
|  | for (int i = 0; i < numbersSize; ++i) { |
|  | write(pipefd[1], &nums[i], sizeof(int)); |
|  | } |
|  | close(pipefd[1]); |
|  | close(pipefd[0]); |
|  |  |
|  |  |
|  | } |
|  | return 0; |
|  | } |

**Демонстрация работы программы**



**Выводы**

Я приобрел навыки управления процесами в С++ и обеспечении обмена данных между процессами через каналы.