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: Постнов Александр Вячеславович

Группа: М8О-201Б-21

Вариант: 4

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

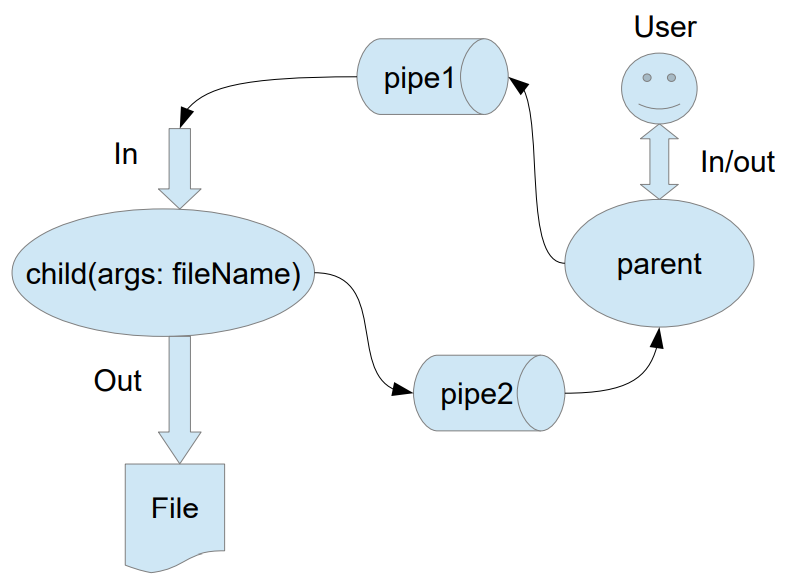
1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/kappaprideonly/mai-os-labs

**Постановка задачи**

Составить и отладить программу на языке C/C++, 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.



Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

Пользователь вводит команды вида: «число число число<endline>». 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 float. Количество чисел может быть произвольным.

**Общие сведения о программе**

CMakeLists.txt - описание процесса сборки проекта  
  
main.cpp - перенаправление потока ввода в функцию ParentRoutine  
  
parent.h - заголовочный файл, в котором описана функция родительского  
  
string\_to\_vector.h - сигнатура функции, которая преобразует строку в вектор float

string\_to\_vector.cpp - реализация функции

parent.cpp - реализация функции родительского процесса

child.cpp - отдельная программа дочернего процесса

lab2\_test.cpp - тесты к лабораторной работе

**Общий метод и алгоритм решения**В родительском процессе создается канал(pipe) и дочерний процесс с помощью системного вызова fork, дочерний процесс получает данные с помощью pipe и запускает программу child.cpp с помощью execl. А уже в child.cpp выполняется задание по варианту и запись в файл.

**Исходный код**

**CMakeLists.txt**

**add\_executable(lab2**

**main.cpp**

**include/parent.h src/parent.cpp)**

**target\_include\_directories(lab2 PRIVATE include)**

**add\_executable(child**

**src/child.cpp**

**include/string\_to\_vector.h src/string\_to\_vector.cpp)**

**target\_include\_directories(child PRIVATE include)**

**add\_dependencies(lab2 child)**

**main.cpp**

**#include "parent.h"**

**int main() {**

**ParentRoutine(std::cin, getenv("PATH\_TO\_CHILD"));**

**return 0;**

**}**

**parent.h**

**#ifndef PARENT\_H**

**#define PARENT\_H**

**#include <istream>**

**#include <vector>**

**#include <iostream>**

**#include <string>**

**#include <unistd.h>**

**#include <cstdlib>**

**#include <fstream>**

**#include <cstdio>**

**#include <sys/wait.h>**

**#include <array>**

**void ParentRoutine(std::istream& stream, const char\* pathToChild);**

**#endif**

**parent.cpp**

**#include "parent.h"**

**void ParentRoutine(std::istream& stream, const char\* pathToChild) {**

**std::string nameOutputFile;**

**std::getline(stream, nameOutputFile);**

**std::array <int, 2> parentPipe; //0 - read 1 - write**

**if (pipe(parentPipe.data()) == -1) {**

**std::cout << "Error creating pipe\n";**

**exit(EXIT\_FAILURE);**

**}**

**int pid = fork();**

**if (pid == -1) {**

**std::cout << "Error creating process\n";**

**exit(EXIT\_FAILURE);**

**}**

**if (pid != 0) { // родительский процесс**

**close(parentPipe[0]);**

**std::string stringNumbers;**

**while (std::getline(stream, stringNumbers)) {**

**stringNumbers += "\n";**

**write(parentPipe[1], stringNumbers.data(), stringNumbers.size());**

**}**

**close(parentPipe[1]);**

**wait(nullptr);**

**}**

**else { // дочерний процесс**

**close(parentPipe[1]);**

**dup2(parentPipe[0], 0);**

**if(execl(pathToChild, pathToChild, nameOutputFile.data(), nullptr) == -1) {**

**std::cout << "Failed to exec\n";**

**exit(EXIT\_FAILURE);**

**}**

**close(parentPipe[0]);**

**}**

**}**

**child.cpp**

**#include <cstdlib>**

**#include <iostream>**

**#include <string>**

**#include <vector>**

**#include <fstream>**

**#include "string\_to\_vector.h"**

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

**if (argc != 2) {**

**std::cout << "Invalid arguments.\n";**

**exit(EXIT\_FAILURE);**

**}**

**auto \*nameOutputFile = argv[1];**

**std::ofstream out(nameOutputFile);**

**std::string stringNumbers;**

**while (std::getline(std::cin, stringNumbers)) {**

**std::vector <float> numbers = StringToVectorFloats(stringNumbers);**

**float firstNumber = numbers[0];**

**for (unsigned long long i = 1; i < numbers.size(); i++) {**

**if (numbers[i] == 0) {**

**std::cout << "Division by zero.\n";**

**out << "\n";**

**out.close();**

**exit(EXIT\_FAILURE);**

**}**

**firstNumber /= numbers[i];**

**}**

**out << firstNumber << " ";**

**}**

**out << "\n";**

**out.close();**

**return 0;**

**}**

**string\_to\_vector.h**

**#ifndef STRING\_TO\_VECTOR\_H**

**#define STRING\_TO\_VECTOR\_H**

**#include <vector>**

**#include <iostream>**

**#include <string>**

**#include <cstring>**

**#include <cstdlib>**

**#include <algorithm>**

**std::vector<float> StringToVectorFloats(std::string const& stringNumbers, char separator=' ');**

**#endif//STRING\_TO\_VECTOR\_H**

**string\_to\_vector.cpp**

**#include "string\_to\_vector.h"**

**std::vector<float> StringToVectorFloats(std::string const& stringNumbers, char separator) {**

**std::vector<float> results;**

**auto start = stringNumbers.begin();**

**auto end = stringNumbers.end();**

**auto next = std::find(start, end, separator);**

**while (next != end) {**

**results.push\_back(stof(std::string(start, next)));**

**start = next + 1;**

**next = std::find(start, end, separator);**

**}**

**results.push\_back(stof(std::string(start, next)));**

**return results;**

**}**

**lab2\_test.cpp**

**#include <cstdio>**

**#include <cstdlib>**

**#include <fstream>**

**#include <gtest/gtest.h>**

**#include <string>**

**#include "parent.h"**

**#include "string\_to\_vector.h"**

**TEST(Lab2Test, StringToVectorTest) {**

**std::vector <std::vector <float>> expectedVectors = {**

**{1.5, 2.5, 3.5},**

**{1.5, 2, 3, 4, 5, 0},**

**{1}**

**};**

**std::vector <std::string> inputStrings = {**

**"1.5 2.5 3.5",**

**"1.5 2 3 4 5 0",**

**"1"**

**};**

**long unsigned int countTests = 3;**

**for (long unsigned int i = 0; i < countTests; i++) {**

**std::vector <float> outputVector = StringToVectorFloats(inputStrings[i]);**

**ASSERT\_EQ(expectedVectors[i].size(), outputVector.size());**

**for (long unsigned int j = 0; i < expectedVectors[i].size(); i++) {**

**EXPECT\_FLOAT\_EQ(expectedVectors[i][j], outputVector[j]);**

**}**

**}**

**}**

**TEST(Lab2Test, ParentTest) {**

**std::vector <std::string> namesOutputFile = {**

**"checker.txt",**

**"output.txt",**

**"jambo.tea"**

**};**

**std::vector <std::string> stringsNumbers = {**

**"1 0.5 0.5 0.5\n100 8\n1\n90 2",**

**"1 0.5 0.5\n100 0\n1\n90 2\n1 1",**

**"1 0 0.5 0.5\n100 0\n1\n90 2"**

**};**

**std::vector <std::string> expectedStrings = {**

**"8 12.5 1 45 ",**

**"4 ",**

**""**

**};**

**long unsigned int countTests = 3;**

**for (long unsigned int i = 0; i < countTests; i++) {**

**{**

**std::ofstream fOut("input.txt");**

**fOut << namesOutputFile[i] << "\n";**

**fOut << stringsNumbers[i] << "\n";**

**}**

**{**

**std::ifstream fIn("input.txt");**

**ParentRoutine(fIn, getenv("PATH\_TO\_CHILD"));**

**}**

**remove("input.txt");**

**{**

**std::ifstream fInCheckOutput = std::ifstream(namesOutputFile[i]);**

**ASSERT\_TRUE(fInCheckOutput.good());**

**std::string outputString;**

**std::getline(fInCheckOutput, outputString);**

**EXPECT\_EQ(outputString, expectedStrings[i]);**

**}**

**remove(namesOutputFile[i].data());**

**}**

**}**

**Демонстрация работы программы**

➜ lab2 git:(main) ✗ ls

child CMakeFiles lab2 test.txt

cmake\_install.cmake CTestTestfile.cmake Makefile

➜ lab2 git:(main) ✗ cat test.txt

checker.txt

1 2 3 4 56

100 2 5

200 2 10

300 4 2

1 1 1

4 5 6

➜ lab2 git:(main) ✗ ./lab2 <test.txt

➜ lab2 git:(main) ✗ ls

checker.txt cmake\_install.cmake CTestTestfile.cmake Makefile

child CMakeFiles lab2 test.txt

➜ lab2 git:(main) ✗ cat checker.txt

0.000744048 10 10 37.5 1 0.133333

**Выводы**Я приобрел практические навыки в:

1. Управление процессами в ОС
2. Обеспечение данными между процессами посредством каналов