Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 4 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**

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

lab4\_test.cpp - тесты к лабораторной работе

**Общий метод и алгоритм решения**main перенаправляет ввод в родительский процесс, родительский процесс создает дочерний процесс с помощью fork, дочерний процесс запускает отдельно программу. Процессы взаимодействуют с друг другом через файлы, отображаемые в память. Чтобы действие по варианту происходило построчно, использовал примитив синхронизации семафор. 1 семафор на ввод данных, 2 семафор на обработку данных.

**Исходный код**

**CMakeLists.txt**

**add\_executable(lab4**

**main.cpp**

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

**target\_include\_directories(lab4 PRIVATE include)**

**add\_executable(child4**

**src/child.cpp**

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

**target\_include\_directories(child4 PRIVATE include)**

**add\_dependencies(lab4 child4)**

**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>**

**#include <iterator>**

**#include <pthread.h>**

**#include <algorithm>**

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

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

**#include <fcntl.h>**

**#include <semaphore.h>**

**#include <cstring>**

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

**#endif**

**parent.cpp**

**#include "parent.h"**

**#include <algorithm>**

**#include <cstring>**

**#include <semaphore.h>**

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

**#include <unistd.h>**

**constexpr auto SHARED\_MEMORY\_OBJECT\_NAME = "shared\_memory";**

**constexpr auto SHARED\_MEMORY\_SEMAPHORE\_INPUT\_NAME = "shared\_semaphore\_input";**

**constexpr auto SHARED\_MEMORY\_SEMAPHORE\_OUTPUT\_NAME = "shared\_semaphore\_output";**

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

**// Clear();**

**std::string nameOutputFile;**

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

**/\* shared memory file descriptor \*/**

**int sfd;**

**int semInFd;**

**int semOutFd;**

**/\* create the shared memory object \*/**

**if ((sfd = shm\_open(SHARED\_MEMORY\_OBJECT\_NAME, O\_CREAT | O\_RDWR, S\_IRWXU)) == -1) {**

**std::cout << "Shm\_open error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if ((semInFd = shm\_open(SHARED\_MEMORY\_SEMAPHORE\_INPUT\_NAME, O\_CREAT | O\_RDWR, S\_IRWXU)) == -1) {**

**std::cout << "Shm\_open error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if ((semOutFd = shm\_open(SHARED\_MEMORY\_SEMAPHORE\_OUTPUT\_NAME, O\_CREAT | O\_RDWR, S\_IRWXU)) == -1) {**

**std::cout << "Shm\_open error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**/\* configure the size of the shared memory object \*/**

**ftruncate(sfd, getpagesize());**

**ftruncate(semInFd, getpagesize());**

**ftruncate(semOutFd, getpagesize());**

**auto \*semInput = (sem\_t\*)mmap(nullptr, getpagesize(), PROT\_WRITE | PROT\_READ, MAP\_SHARED, semInFd, 0);**

**auto \*semOutput = (sem\_t\*)mmap(nullptr, getpagesize(), PROT\_WRITE | PROT\_READ, MAP\_SHARED, semOutFd, 0);**

**if (semInput == MAP\_FAILED) {**

**std::cout << "Mmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (semOutput == MAP\_FAILED) {**

**std::cout << "Mmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**sem\_init(semInput, 1, 1);**

**sem\_init(semOutput, 1, 0);**

**int pid = fork();**

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

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

**exit(EXIT\_FAILURE);**

**}**

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

**/\* memory map the shared memory object \*/**

**char\* ptr = (char\*)mmap(nullptr, getpagesize(), PROT\_WRITE | PROT\_READ, MAP\_SHARED, sfd, 0);**

**if (ptr == MAP\_FAILED) {**

**std::cout << "Mmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**std::string stringNumbers;**

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

**sem\_wait(semInput);**

**if (std::string(ptr) == "Division by zero.") {**

**sem\_post(semInput);**

**break;**

**}**

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

**sprintf((char \*) ptr, "%s", stringNumbers.c\_str());**

**sem\_post(semOutput);**

**}**

**sem\_wait(semInput);**

**sprintf((char \*) ptr, "%s", "");**

**sem\_post(semOutput);**

**wait(nullptr);**

**if (sem\_destroy(semInput) == -1) {**

**std::cout << "Sem\_destroy error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (sem\_destroy(semOutput) == -1) {**

**std::cout << "Sem\_destroy error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (munmap(semInput, getpagesize()) == -1) {**

**std::cout << "Munmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (munmap(semOutput, getpagesize()) == -1) {**

**std::cout << "Munmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (munmap(ptr, getpagesize()) == -1) {**

**std::cout << "Munmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (shm\_unlink(SHARED\_MEMORY\_OBJECT\_NAME) == -1) {**

**std::cout << "Shm\_unlink error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (shm\_unlink(SHARED\_MEMORY\_SEMAPHORE\_INPUT\_NAME) == -1) {**

**std::cout << "Shm\_unlink error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (shm\_unlink(SHARED\_MEMORY\_SEMAPHORE\_OUTPUT\_NAME) == -1) {**

**std::cout << "Shm\_unlink error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**}**

**else { // дочерний процесс**

**if(execl(pathToChild, pathToChild, nameOutputFile.data(),**

**SHARED\_MEMORY\_OBJECT\_NAME, SHARED\_MEMORY\_SEMAPHORE\_INPUT\_NAME,**

**SHARED\_MEMORY\_SEMAPHORE\_OUTPUT\_NAME, nullptr) == -1) {**

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

**exit(EXIT\_FAILURE);**

**}**

**}**

**}**

**child.cpp**

**#include <istream>**

**#include <ostream>**

**#include <vector>**

**#include <iostream>**

**#include <string>**

**#include <unistd.h>**

**#include <cstdlib>**

**#include <fstream>**

**#include <cstdio>**

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

**#include <pthread.h>**

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

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

**#include <fcntl.h>**

**#include <cstring>**

**#include <semaphore.h>**

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

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

**if (argc != 5) {**

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

**exit(EXIT\_FAILURE);**

**}**

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

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

**int sfd;**

**int semInFd;**

**int semOutFd;**

**if ((sfd = shm\_open(argv[2], O\_RDWR, S\_IRWXU)) == -1) {**

**std::cout << "shm\_open error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if ((semInFd = shm\_open(argv[3], O\_RDWR, S\_IRWXU)) == -1) {**

**std::cout << "Shm\_open error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if ((semOutFd = shm\_open(argv[4], O\_RDWR, S\_IRWXU)) == -1) {**

**std::cout << "Shm\_open error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**char\* ptr = (char\*)mmap(nullptr, getpagesize(), PROT\_READ | PROT\_WRITE, MAP\_SHARED, sfd, 0);**

**if (ptr == MAP\_FAILED) {**

**std::cout << "error mmap func" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**auto \*semInput = (sem\_t\*)mmap(nullptr, getpagesize(), PROT\_WRITE | PROT\_READ, MAP\_SHARED, semInFd, 0);**

**auto \*semOutput = (sem\_t\*)mmap(nullptr, getpagesize(), PROT\_WRITE | PROT\_READ, MAP\_SHARED, semOutFd, 0);**

**if (semInput == MAP\_FAILED) {**

**std::cout << "Mmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (semOutput == MAP\_FAILED) {**

**std::cout << "Mmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**while (true) {**

**sem\_wait(semOutput);**

**std::string stringNumbers = ptr;**

**if (stringNumbers.empty()) {**

**sem\_post(semInput);**

**break;**

**}**

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

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

**for (size\_t i = 1; i < numbers.size(); i++) {**

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

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

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

**out.close();**

**sprintf((char \*) ptr, "%s", "Division by zero.");**

**sem\_post(semInput);**

**exit(EXIT\_FAILURE);**

**}**

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

**}**

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

**sem\_post(semInput);**

**}**

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

**out.close();**

**if (munmap(ptr, getpagesize()) == -1) {**

**std::cout << "Munmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (munmap(semInput, getpagesize()) == -1) {**

**std::cout << "Munmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**if (munmap(semOutput, getpagesize()) == -1) {**

**std::cout << "Munmap error" << std::endl;**

**exit(EXIT\_FAILURE);**

**}**

**return EXIT\_SUCCESS;**

**}**

**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;**

**}**

**lab4\_test.cpp**

**#include <cstdio>**

**#include <cstdlib>**

**#include <fstream>**

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

**#include <string>**

**#include "parent.h"**

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

**TEST(Lab4Test, 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(Lab4Test, 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\_CHILD4"));**

**}**

**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**

➜ lab4 git:(main) ls

child4 CMakeFiles lab4 test.txt

cmake\_install.cmake CTestTestfile.cmake Makefile

➜ lab4 git:(main) cat test.txt

out.txt

100 2 5

5 5 1

500 5 100

➜ lab4 git:(main) ./lab4 <test.txt

➜ lab4 git:(main) ls

child4 CMakeFiles lab4 out.txt

cmake\_install.cmake CTestTestfile.cmake Makefile test.txt

➜ lab4 git:(main) cat out.txt

10 1 1

**Выводы:**

Приобрел практические навыки в:

* Освоение принципов работы с файловыми системами
* Обеспечение обмена данных между процессами посредством технологии «File mapping»