Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу

«Операционные системы»

Студент: Ветошкина София Владимировна
Группа: М8О-203Б-23
Вариант: 6
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Содержание

Репозиторий

https://github.com/sofiavetoshkina/os_labs/tree/main

Постановка задачи

Составить и отладить программу, 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.

Общие сведения о программе

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс представлены разными программами.

В файле записаны команды вида: «число число число<endline>». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип int. Количество чисел может быть произвольным.

Программа собирается системой сборки CMake.

Реализованы тесты для проверки корректности программы с помощью Google Test.

Исходный код

lab1/main.cpp:

```
#include <iostream>
#include <string>
#include "parent.hpp"

int main() {
   std:: string fileName;
   std::cout << "Введите название файла, где необходимо посчитать сумму
чисел: ";
```

```
    std::cin >> fileName;

    ParentRoutine(fileName, std::cout);

    return 0;
}
```

lab1/include/parent.hpp:

```
#ifndef OS_LABS_PARENT_H
#define OS_LABS_PARENT_H

#include <iostream>
#include <string>

void ParentRoutine(const std::string& fileName, std::ostream& output);

#endif //OS_LABS_PARENT_H
```

lab1/src /parent.cpp:

```
#include <fcntl.h>
#include <iostream>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <cstring>
#include <string>
#include "parent.hpp"

void ParentRoutine(const std::string& fileName, std::ostream& output) {
    int pipe1[2];

    if (pipe(pipe1) == -1) {
        perror("Pipe error");
        return;
    }

    pid_t pid = fork();

    switch (pid) {

        case -1: {
            perror("Fork error");
            return;
```

```
}

case 0: {
    // Дочерний процесс
    close(pipe1[0]); // Закрываем конец канала для чтения

    // Получение дескриптора файла
    int fileFd = open(fileName.c_str(), O_RDONLY);
    if (fileFd < 0) {
        perror("Open file error");
        return;
    }

    // Стандартный поток ввода переопределен открытым файлом
    if (dup2(fileFd, STDIN_FILENO) == -1) {
        perror("dup2 error");
        close(fileFd);
        return;
    }

    // Стандартный поток вывода перенаправляется в pipe1 на запись
    if (dup2(pipe1[1], STDOUT_FILENO) == -1) {
        perror("dup2 error");
        close(pipe1[1]);
        return;
    }

    const char* pathToChild = getenv("PATH_TO_EXEC_CHILD");
    if (pathToChild == nullptr) {
        perror("Path error");
        return;
    }
    // Запуск дочерней программы
    execl(pathToChild, pathToChild, nullptr);
    break;
}

default: {
    // Родительский процесс
    close(pipe1[1]); // Закрываем конец канала для записи

    //Читаем данные и выводим в стандартный поток вывода
    char buffer[128];
    size_t bytes = read(pipe1[0], buffer, sizeof(buffer));
    if (bytes > 0){
```

```
            output.write(buffer, bytes);
        } else {
            perror("Read error");
            return;
        }

        close(pipe1[0]); // Закрываем для чтения

        // Ожидаем завершения дочернего процесса
        wait(nullptr);
    }
  }
}
```

lab1/src /child.cpp:

```
#include <iostream>

int main() {
  int number {0};
  long long sum {0};
  while (std::cin >> number) {
    sum += number;
  }
  std::cout << sum << std::endl;

  return 0;
}
```

tests/lab1_test.cpp:

```
#include <gtest/gtest.h>
#include <fstream>
#include <string>
#include "parent.hpp"

TEST(ParentRoutineTest, CalculatesSumCorrectly) {
  std::ostringstream outputStream;
  const char* fileName = getenv("PATH_TO_TEST_FILE");
  if (fileName == nullptr) {
      perror("Переменная PATH_TO_TEST_FILE не установлена");
      exit(1);
  }

  //Содержимое test.txt:
```

```
    //100 10 50
    //40 -10 10

    const int expectedOutput = 200;

    ParentRoutine(fileName, outputStream);

    std::string output = outputStream.str();
    std::istringstream iss(output);
    int realOutput = 0;
    iss >> realOutput;

    EXPECT_EQ(realOutput, expectedOutput);
}

TEST(ParentRoutineTest, EmptyFile) {
    std::ostringstream outputStream;
    const char* fileName = getenv("PATH_TO_EMPTY_TEST_FILE");
    if (fileName == nullptr) {
        perror("Переменная PATH_TO_EMPTY_TEST_FILE не установлена");
        exit(1);
    }

    const int expectedOutput = 0;

    ParentRoutine(fileName, outputStream);

    std::string output = outputStream.str();
    std::istringstream iss(output);
    int realOutput = 0;
    iss >> realOutput;

    EXPECT_EQ(realOutput, expectedOutput);
}

TEST(ParentRoutineTest, CalculatesSumCorrectly2) {
    std::ostringstream outputStream;
    const char* fileName = getenv("PATH_TO_TEST_FILE2");
    if (fileName == nullptr) {
        perror("Переменная PATH_TO_TEST_FILE2 не установлена");
        exit(1);
    }

    //Содержимое test2.txt:
    //100 10 50
```

```
    //40 -10 10 100000 0

    const int expectedOutput = 100200;

    ParentRoutine(fileName, outputStream);

    std::string output = outputStream.str();
    std::istringstream iss(output);
    int realOutput = 0;
    iss >> realOutput;

    EXPECT_EQ(realOutput, expectedOutput);
}
```

Демонстрация работы программы

Содержимое файла test.txt: 100 10 50\n40 -10 10

getz66@getz1165-nettop:~/OS/os_labs/build$ export PATH_TO_EXEC_CHILD='/home/getz66/OS/os_labs/build/lab1/child'

getz66@getz1165-nettop:~/OS/os_labs/build$ ./lab1/parent

Введите название файла, где необходимо посчитать сумму чисел: test.txt

200

Выводы

В ходе выполнения данной лабораторной работы были получены знания и навыки использования системных вызовов Linux для работы с процессами и межпроцессным взаимодействием. Программа успешно реализует создание дочернего процесса (fork, execl), перенаправление стандартных потоков ввода и вывода (dup2), а также взаимодействие между процессами через канал (pipe). Была изучена система сборки CMake, а также библиотека для тестирования Google Test.