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: Дудовцев Андрей Андреевич
Группа: М8О-208Б-22
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

**Содержание**

## Репозиторий

AndreyDdvts/OS_LABS (github.com)

## Цель работы

Приобретение практических навыков в:

- Управлении процессами в ОС
- Обеспечении 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) должна 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.

## Описание работы программы

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Родительский и дочерний процесс представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода. В ходе выполнения лабораторной работы я использовал следующие системные вызовы:

- fork() - создание нового процесса
- pipe() - создание канала

- dup2() - создание копии файлового дескриптора, используя для нового дескриптора самый маленький свободный номер файлового дескриптора.
- execlp() - запуск файла на исполнение

## Исходный код

```
========================= parent.hpp ========================

#pragma once

#include "utils.hpp"

void parentProcess(const char *pathToChild);

========================= utils.hpp ========================

#pragma once

#include <iostream>
#include <string>
#include <sstream>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <ext/stdio_filebuf.h>

void createPipe(int fd[2]);
pid_t createChildProcess();
std::stringstream readFromPipe (int fd);
bool checkString(const std::string &str);

========================= parent.cpp ========================

#include "parent.hpp"
```

```
void parentProcess(const char *pathToChild) {
    std::string fileName;
    getline(std::cin, fileName);

    int fd1[2], fd2[2];
    createPipe(fd1);
    createPipe(fd2);

    int pid = createChildProcess();
    if (pid != 0) { // Parent process
        close(fd1[0]);
        close(fd2[1]);

        std::string str;
        while (getline(std::cin, str)) {
            str += "\n";
            write(fd1[1], str.c_str(), str.length()); // from str to
fd1[1]
        }
        close(fd1[1]);

        std::stringstream output = readFromPipe(fd2[0]);
        while(std::getline(output, str)) {
            std::cout << str << std::endl;
        }
        close(fd2[0]);
    } else { // Child process
        close(fd1[1]);
        close(fd2[0]);

        if (dup2(fd1[0], STDIN_FILENO) == -1 || dup2(fd2[1],
STDOUT_FILENO) == -1) {
            perror("Error with dup2");
```

```
            exit(EXIT_FAILURE);
        }

        if (execlp(pathToChild, pathToChild, fileName.c_str(),
nullptr) == -1) { // to child.cpp
            perror("Error with execlp");
            exit(EXIT_FAILURE);
        }
    }
}

========================== utils.cpp ======================

#include "utils.hpp"

void createPipe(int fd[2]) {
    if (pipe(fd) == -1) {
        perror("Couldn't create pipe");
        exit(EXIT_FAILURE);
    }
}

pid_t createChildProcess() {
    pid_t pid = fork();
    if (pid == -1) {
        perror("Couldn't create child process");
        exit(EXIT_FAILURE);
    }
    return pid;
}

std::stringstream readFromPipe (int fd) {
    constexpr int BUFFER_SIZE = 256;
    char buffer[BUFFER_SIZE] = "";
```

```
    // char c;
    std::stringstream stream;
    while(true) {
        int t = read(fd, &buffer, BUFFER_SIZE);
        // int t = read(fd, &c, sizeof(char));
        if (t == -1) {
            perror("Couldn't read from pipe");
            exit(EXIT_FAILURE);
        } else if (t > 0) {
            stream << buffer;
            // stream << c;
        } else {
            return stream;
        }
    }
}

bool checkString(const std::string &str) {
    if (str[str.size() - 1] == '.' || str[str.size() - 1] == ';') {
        return true;
    }
    return false;
}


========================== child.cpp ========================

#include "utils.hpp"

int main(int argc, char *argv[]) {
    if (argc != 2) {
        perror("Not enough arguments");
        exit(EXIT_FAILURE);
    }
```

```
    const char *fileName = argv[1];
    std::ofstream fout(fileName);
    if (!fout.is_open()) {
        perror("Couldn't open the file");
        exit(EXIT_FAILURE);
    }

    std::string str;
    while (std::getline(std::cin, str)) {
        if (checkString(str)) {
            fout << str << '\n';
        } else {
            std::string error = "ERROR with string: " + str;
            std::cout << error << std::endl;
        }
    }

    fout.close();
    exit(EXIT_SUCCESS);
}
```

```
========================= child.cpp ======================
```

```
#include "parent.hpp"

int main() {
    parentProcess(getenv("PATH_TO_CHILD"));
    // bash: export PATH_TO_CHILD="/home/qwz/OS_LABS/build/lab1/child"
    exit(EXIT_SUCCESS);
}
```

## Тесты

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

```
#include <filesystem>
#include <memory>
#include <vector>

#include <parent.hpp>

namespace fs = std::filesystem;

void testingProgram(const std::vector<std::string> &input, const
std::vector<std::string> &expectedOutput, const
std::vector<std::string> &expectedFile) {
    const char *fileName = "file.txt";

    std::stringstream inFile;
    inFile << fileName << std::endl;
    for (std::string line : input) {
        inFile << line << std::endl;
    }

    std::streambuf* oldInBuf = std::cin.rdbuf(inFile.rdbuf()); //
выдает старый буфер

    ASSERT_TRUE(fs::exists(getenv("PATH_TO_CHILD")));

    testing::internal::CaptureStdout();

    parentProcess(getenv("PATH_TO_CHILD"));
    std::cin.rdbuf(oldInBuf); // чтобы cin вернулся обратно

    std::stringstream
errorOut(testing::internal::GetCapturedStdout());
    for(const std::string &expectation : expectedOutput) {
        std::string result;
```

```
        getline(errorOut, result);
        EXPECT_EQ(result, expectation);
    }

    std::ifstream fin(fileName);
    if (!fin.is_open()) {
        perror("Couldn't open the file");
        exit(EXIT_FAILURE);
    }
    for (const std::string &expectation : expectedFile) {
        std::string result;
        getline(fin, result);
        EXPECT_EQ(result, expectation);
    }
    fin.close();
}

TEST(firstLabTests, emptyTest) {
    std::vector<std::string> input = {};

    std::vector<std::string> expectedOutput = {};

    std::vector<std::string> expectedFile = {};

    testingProgram(input, expectedOutput, expectedFile);
}

TEST(firstLabTests, simpleTest) {
    std::vector<std::string> input = {
        "No,",
        "you'll never be alone.",
        "When darkness comes;",
        "I'll light the night with stars",
        "Hear my whispers in the dark!"
```

```
    };

    std::vector<std::string> expectedOutput = {
        "ERROR with string: No,",
        "ERROR with string: I'll light the night with stars",
        "ERROR with string: Hear my whispers in the dark!"
    };

    std::vector<std::string> expectedFile = {
        "you'll never be alone.",
        "When darkness comes;"
    };

    testingProgram(input, expectedOutput, expectedFile);
}

TEST(firstLabTests, aQuedaTest) {
    std::vector<std::string> input = {
        "A QUEDA:",
        "E venha ver os deslizes que eu vou cometer;",
        "E venha ver os amigos que eu vou perder;",
        "Não tô cobrando entrada, vem ver o show na faixa.",
        "Hoje tem open bar pra ver minha desgraça."
    };

    std::vector<std::string> expectedOutput = {
        "ERROR with string: A QUEDA:"
    };

    std::vector<std::string> expectedFile = {
        "E venha ver os deslizes que eu vou cometer;",
        "E venha ver os amigos que eu vou perder;",
        "Não tô cobrando entrada, vem ver o show na faixa.",
        "Hoje tem open bar pra ver minha desgraça."
```

```
    };

    testingProgram(input, expectedOutput, expectedFile);
}

TEST(firstLabTests, anotherTest) {
    std::vector<std::string> input = {
        "But I set fire to the rain.",
        "Watched it pour as- I touched your- face-",
        "Well, it burned while I cried!!!!!!!!",
        "Cause I heard it screamin' out your name;",
        "Your name."
    };

    std::vector<std::string> expectedOutput = {
        "ERROR with string: Watched it pour as- I touched your- face-
",
        "ERROR with string: Well, it burned while I cried!!!!!!!!"
    };

    std::vector<std::string> expectedFile = {
        "But I set fire to the rain.",
        "Cause I heard it screamin' out your name;",
        "Your name."
    };

    testingProgram(input, expectedOutput, expectedFile);
}

int main(int argc, char *argv[]) {
    std::cout << getenv("PATH_TO_CHILD") << std::endl;

    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
```

```
}
```

## Демонстрация работы программы

```
qwz@qwz-VirtualBox:~/OS_LABS/build/lab1$  ./child file.txt
No,
you’ll never be alone.
When darkness comes;
I’ll light the night with stars
Hear my whispers in the dark!
ERROR with string: I’ll light the night with stars
ERROR with string: No,
ERROR with string: Hear my whispers in the dark!
qwz@qwz-VirtualBox:~/OS_LABS/build/lab1$  ./child file.txt
A QUEDA:
E venha ver os deslizes que eu vou cometer;
E venha ver os amigos que eu vou perder;
N˜ao tˆo cobrando entrada, vem ver o show na faixa.
Hoje tem open bar pra ver minha desgraca.
ERROR with string: A QUEDA:
qwz@qwz-VirtualBox:~/OS_LABS/build/lab1$  ./child file.txt
But I set fire to the rain.
Watched it pour as- I touched your- face
Well, it burned while I cried!!!!!!!!
Cause I heard it screamin’ out your name;
Your name.
ERROR with string: Watched it pour as- I touched your- face
ERROR with string: Well, it burned while I cried!!!!!!!!
```

## Запуск тестов

```
/home/qwz/OS_LABS/build/lab1/child
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from firstLabTests
[ RUN      ] firstLabTests.emptyTest
```

```
[       OK ] firstLabTests.emptyTest (2 ms)
[ RUN      ] firstLabTests.simpleTest
[       OK ] firstLabTests.simpleTest (2 ms)
[ RUN      ] firstLabTests.aQuedaTest
[       OK ] firstLabTests.aQuedaTest (2 ms)
[ RUN      ] firstLabTests.anotherTest
[       OK ] firstLabTests.anotherTest (3 ms)
[----------] 4 tests from firstLabTests (10 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (11 ms total)
[  PASSED  ] 4 tests.
```

## Выводы

В результате выполнения данной лабораторной работы была написана программа на языке C++, осуществляющая работу с процессами и взаимодействие между ними. Я приобрел практические навыки в управлении процессами в ОС и обеспечении обмена данных между процессами посредством каналов.