

Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"
Кафедра "Вычислительная математика и программирование"

**Лабораторная работа №1 по курсу
“Операционные системы”**

Студент: Былькова Кристина Алексеевна

Группа: М8О-208Б-22

Преподаватель: Миронов Евгений Сергеевич

Вариант: 16

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2023

Содержание

1	Репозиторий	3
2	Цель работы	3
3	Задание	3
4	Описание работы программы	3
5	Исходный код	4
6	Тесты	7
7	Демонстрация работы программы	10
8	Запуск тестов	10
9	Выводы	11

1 Репозиторий

https://github.com/kr1st1na0/OS_labs

2 Цель работы

Приобретение практических навыков в:

- Управлении процессами в ОС
- Обеспечении обмена данных между процессами посредством каналов

3 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) 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.

4 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 использовала 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

5 Исходный код

utils.hpp

```
1 #pragma once
2
3 #include <iostream>
4 #include <string>
5 #include <sstream>
6 #include <stdlib.h>
7 #include <unistd.h>
8 #include <sys/types.h>
9 #include <ext/stdio_filebuf.h>
10
11 void createPipe(int fd[2]);
12 pid_t createChildProcess();
13 std::stringstream readFromPipe (int fd);
14 bool checkString(const std::string &str);
```

parent.hpp

```
1 #pragma once
2
3 #include "utils.hpp"
4
5 void parentProcess(const char *pathToChild);
```

utils.cpp

```
1 #include "utils.hpp"
2
3 void createPipe(int fd[2]) {
4     if (pipe(fd) == -1) {
5         perror("Couldn't create pipe");
6         exit(EXIT_FAILURE);
7     }
8 }
9
10 pid_t createChildProcess() {
11     pid_t pid = fork();
12     if (pid == -1) {
13         perror("Couldn't create child process");
14         exit(EXIT_FAILURE);
15     }
16     return pid;
17 }
18
19 std::stringstream readFromPipe (int fd) {
20     constexpr int BUFFER_SIZE = 256;
21     char buffer[BUFFER_SIZE] = "";
22     // char c;
23     std::stringstream stream;
24     while(true) {
25         int t = read(fd, &buffer, BUFFER_SIZE);
26         // int t = read(fd, &c, sizeof(char));
27         if (t == -1) {
28             perror("Couldn't read from pipe");
29             exit(EXIT_FAILURE);
30         } else if (t > 0) {
31             stream << buffer;
32             // stream << c;
```

```

33         } else {
34             return stream;
35         }
36     }
37 }
38
39 bool checkString(const std::string &str) {
40     if (str[str.size() - 1] == '.' || str[str.size() - 1] == ';')
41     {
42         return true;
43     }
44     return false;
45 }

```

child.cpp

```

1 #include "utils.hpp"
2
3 int main(int argc, char *argv[]) {
4     if (argc != 2) {
5         perror("Not enough arguments");
6         exit(EXIT_FAILURE);
7     }
8
9     const char *fileName = argv[1];
10    std::ofstream fout(fileName);
11    if (!fout.is_open()) {
12        perror("Couldn't open the file");
13        exit(EXIT_FAILURE);
14    }
15
16    std::string str;
17    while (std::getline(std::cin, str)) {
18        if (checkString(str)) {
19            fout << str << '\n';
20        } else {
21            std::string error = "ERROR with string: " + str;
22            std::cout << error << std::endl;
23        }
24    }
25
26    fout.close();
27    exit(EXIT_SUCCESS);
28 }

```

parent.cpp

```

1 #include "parent.hpp"
2
3 void parentProcess(const char *pathToChild) {
4     std::string fileName;
5     getline(std::cin, fileName);
6
7     int fd1[2], fd2[2];
8     createPipe(fd1);
9     createPipe(fd2);
10
11    int pid = createChildProcess();
12    if (pid != 0) { // Parent process
13        close(fd1[0]);
14        close(fd2[1]);
15    }
16 }

```

```

15         std::string str;
16         while (getline(std::cin, str)) {
17             str += "\n";
18             write(fd1[1], str.c_str(), str.length()); // from str
19         to fd1[1]
20         }
21         close(fd1[1]);
22
23         std::stringstream output = readFromPipe(fd2[0]);
24         while (std::getline(output, str)) {
25             std::cout << str << std::endl;
26         }
27         close(fd2[0]);
28     } else { // Child process
29         close(fd1[1]);
30         close(fd2[0]);
31
32         if (dup2(fd1[0], STDIN_FILENO) == -1 || dup2(fd2[1],
33             STDOUT_FILENO) == -1) {
34             perror("Error with dup2");
35             exit(EXIT_FAILURE);
36         }
37
38         if (execlp(pathToChild, pathToChild, fileName.c_str(),
39             nullptr) == -1) { // to child.cpp
40             perror("Error with execlp");
41             exit(EXIT_FAILURE);
42         }
43     }
44 }

```

main.cpp

```

1 #include "parent.hpp"
2
3 int main() {
4     parentProcess(getenv("PATH_TO_CHILD"));
5     // bash: export PATH_TO_CHILD="/home/kristinab/ubuntu_main/
6     OS_labs/build/lab1/child"
7     exit(EXIT_SUCCESS);
8 }

```

6 Тесты

```
1 #include <gtest/gtest.h>
2
3 #include <filesystem>
4 #include <memory>
5 #include <vector>
6
7 #include <parent.hpp>
8
9 namespace fs = std::filesystem;
10
11 void testingProgram(const std::vector<std::string> &input, const
    std::vector<std::string> &expectedOutput, const std::vector<std
    ::string> &expectedFile) {
12     const char *fileName = "file.txt";
13
14     std::stringstream inFile;
15     inFile << fileName << std::endl;
16     for (std::string line : input) {
17         inFile << line << std::endl;
18     }
19
20     std::streambuf* oldInBuf = std::cin.rdbuf(inFile.rdbuf()); //
21
22     ASSERT_TRUE(fs::exists(getenv("PATH_TO_CHILD")));
23
24     testing::internal::CaptureStdout();
25
26     parentProcess(getenv("PATH_TO_CHILD"));
27     std::cin.rdbuf(oldInBuf); //          cin
28
29     std::stringstream errorOut(testing::internal::
    GetCapturedStdout());
30     for(const std::string &expectation : expectedOutput) {
31         std::string result;
32         getline(errorOut, result);
33         EXPECT_EQ(result, expectation);
34     }
35
36     std::ifstream fin(fileName);
37     if (!fin.is_open()) {
38         perror("Couldn't open the file");
39         exit(EXIT_FAILURE);
40     }
41     for (const std::string &expectation : expectedFile) {
42         std::string result;
43         getline(fin, result);
44         EXPECT_EQ(result, expectation);
45     }
46     fin.close();
47 }
48
49 TEST(firstLabTests, emptyTest) {
50     std::vector<std::string> input = {};
51
52     std::vector<std::string> expectedOutput = {};
```

```

53     std::vector<std::string> expectedFile = {};
54
55     testingProgram(input, expectedOutput, expectedFile);
56 }
57
58 TEST(firstLabTests, simpleTest) {
59     std::vector<std::string> input = {
60         "No,",
61         "you'll never be alone.",
62         "When darkness comes;",
63         "I'll light the night with stars",
64         "Hear my whispers in the dark!"
65     };
66
67     std::vector<std::string> expectedOutput = {
68         "ERROR with string: No,",
69         "ERROR with string: I'll light the night with stars",
70         "ERROR with string: Hear my whispers in the dark!"
71     };
72
73     std::vector<std::string> expectedFile = {
74         "you'll never be alone.",
75         "When darkness comes;"
76     };
77
78     testingProgram(input, expectedOutput, expectedFile);
79 }
80
81 TEST(firstLabTests, aQuedaTest) {
82     std::vector<std::string> input = {
83         "A QUEDA:",
84         "E venha ver os deslizes que eu vou cometer;",
85         "E venha ver os amigos que eu vou perder;",
86         "N o t cobrando entrada, vem ver o show na faixa.",
87         "Hoje tem open bar pra ver minha desgra a."
88     };
89
90     std::vector<std::string> expectedOutput = {
91         "ERROR with string: A QUEDA:"
92     };
93
94     std::vector<std::string> expectedFile = {
95         "E venha ver os deslizes que eu vou cometer;",
96         "E venha ver os amigos que eu vou perder;",
97         "N o t cobrando entrada, vem ver o show na faixa.",
98         "Hoje tem open bar pra ver minha desgra a."
99     };
100
101     testingProgram(input, expectedOutput, expectedFile);
102 }
103
104 TEST(firstLabTests, anotherTest) {
105     std::vector<std::string> input = {
106         "But I set fire to the rain.",
107         "Watched it pour as- I touched your- face-",
108         "Well, it burned while I cried!!!!!!!!!!",
109         "Cause I heard it screamin' out your name;",
110         "Your name."
111     };

```



```

112     };
113
114     std::vector<std::string> expectedOutput = {
115         "ERROR with string: Watched it pour as- I touched your-
face-",
116         "ERROR with string: Well, it burned while I cried!!!!!!!!!!"
117     };
118
119     std::vector<std::string> expectedFile = {
120         "But I set fire to the rain.",
121         "Cause I heard it screamin' out your name;",
122         "Your name."
123     };
124
125     testingProgram(input, expectedOutput, expectedFile);
126 }
127
128 int main(int argc, char *argv[]) {
129     std::cout << getenv("PATH_TO_CHILD") << std::endl;
130
131     testing::InitGoogleTest(&argc, argv);
132     return RUN_ALL_TESTS();
133 }

```

7 Демонстрация работы программы

```
kristinab@LAPTOP-SFU9B1F4:~/ubuntu_main/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!
kristinab@LAPTOP-SFU9B1F4:~/ubuntu_main/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;
Não tô cobrando entrada, vem ver o show na faixa.
Hoje tem open bar pra ver minha desgraça.
ERROR with string: A QUEDA:
kristinab@LAPTOP-SFU9B1F4:~/ubuntu_main/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!!!!!!!
```

8 Запуск тестов

```
kristinab@LAPTOP-SFU9B1F4:~/ubuntu_main/OS_labs/build/tests$ ./lab1_test
/home/kristinab/ubuntu_main/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 (7 ms)
[ RUN      ] firstLabTests.simpleTest
[      OK  ] firstLabTests.simpleTest (1 ms)
[ RUN      ] firstLabTests.aQuedaTest
[      OK  ] firstLabTests.aQuedaTest (0 ms)
[ RUN      ] firstLabTests.anotherTest
[      OK  ] firstLabTests.anotherTest (1 ms)
[-----] 4 tests from firstLabTests (11 ms total)

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

9 Выводы

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