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: Кочкожаров Иван Вячеславович

Группа: М8О-208Б-22

Преподаватель: Миронов Евгений Сергеевич

Вариант: 20

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2023

Содержание

1 Репозиторий

https://github.com/kochkozharov/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

lab1.c

```
#include "lab1.h"

#include <errno.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>

#include "error_handling.h"
#include "utils.h"

int ParentRoutine(const char* pathToChild, FILE* stream) {
    int fds[2] = {-1, -1};
    int pipes[2][2] = {{-1, -1}, {-1, -1}};

    char* line = NULL;
    size_t len = 0;
    char* extraLine = NULL;
    size_t extraLen = 0;

    errno = 0;
    ssize_t nread = getline(&line, &len, stream);
    GOTO_IF(errno == ENOMEM, "getline" , err);
    if (nread == -1) {
        fprintf(stderr, "Input 2 filenames\n");
        goto err;
    }
    line[nread - 1] = '\0';

    errno = 0;
    nread = getline(&extraLine, &extraLen, stream);
    GOTO_IF(errno == ENOMEM, "getline" , err);
    if (nread == -1) {
        fprintf(stderr, "Input 2 filenames\n");
        goto err;
    }
    extraLine[nread - 1] = '\0';

    fds[0] = open(line, O_CREAT | O_WRONLY | O_TRUNC, MODE);
    GOTO_IF(fds[0] == -1, "open", err);
    fds[1] = open(extraLine, O_CREAT | O_WRONLY | O_TRUNC,
MODE);
    GOTO_IF(fds[1] == -1, "open", err);
    free(extraLine);
    extraLine = NULL;

    GOTO_IF(pipe(pipes[0]), "pipe", err);
    GOTO_IF(pipe(pipes[1]), "pipe", err);

    pid_t pids[2];
    pids[0] = fork();
    GOTO_IF(pids[0] == -1, "fork", err);
    if (pids[0]) {
        pids[1] = fork();
        GOTO_IF(pids[1] == -1, "fork", err);
    }
```

```
56
57          if (pids[0] == 0) {  // child1
58              ABORT_IF(close(fds[1]), "close");
59              fds[1] = -1;
60              ABORT_IF(close(pipes[1][READ_END]), "close");
61              pipes[1][READ_END] = -1;
62              ABORT_IF(close(pipes[1][WRITE_END]), "close");
63              pipes[1][WRITE_END] = -1;
64              ABORT_IF(close(pipes[0][WRITE_END]), "close");
65              pipes[0][WRITE_END] = -1;
66
67              GOTO_IF(dup2(pipes[0][READ_END], STDIN_FILENO) == -1,
    "dup2", err);
68              ABORT_IF(close(pipes[0][READ_END]), "close");
69              pipes[0][READ_END] = -1;
70
71              GOTO_IF(dup2(fds[0], STDOUT_FILENO) == -1, "dup2", err
    );
72              ABORT_IF(close(fds[0]), "close");
73              fds[0] = -1;
74
75              GOTO_IF(execl(pathToChild, "child", NULL), "execl",
    err);
76          } else if (pids[1] == 0) {  // child2
77              ABORT_IF(close(fds[0]), "close");
78              fds[0] = -1;
79              ABORT_IF(close(pipes[0][READ_END]), "close");
80              pipes[0][READ_END] = -1;
81              ABORT_IF(close(pipes[0][WRITE_END]), "close");
82              pipes[0][WRITE_END] = -1;
83              ABORT_IF(close(pipes[1][WRITE_END]), "close");
84              pipes[1][WRITE_END] = -1;
85
86              GOTO_IF(dup2(pipes[1][READ_END], STDIN_FILENO) == -1,
    "dup2", err);
87              ABORT_IF(close(pipes[1][READ_END]), "close");
88              pipes[1][READ_END] = -1;
89
90              GOTO_IF(dup2(fds[1], STDOUT_FILENO) == -1, "dup2", err
    );
91              ABORT_IF(close(fds[1]), "close");
92              fds[1] = -1;
93
94              GOTO_IF(execl(pathToChild, "child", NULL), "execl",
    err);
95          } else {  // parent
96              ABORT_IF(close(pipes[0][READ_END]), "close");
97              pipes[0][READ_END] = -1;
98              ABORT_IF(close(pipes[1][READ_END]), "close");
99              pipes[1][READ_END] = -1;
100             ABORT_IF(close(fds[0]), "close");
101             fds[0] = -1;
102             ABORT_IF(close(fds[1]), "close");
103             fds[1] = -1;
104
105             GOTO_IF(waitpid(-1, NULL, WNOHANG), "waitpid", err);
106             ssize_t nread;
107             while ((nread = getline(&line, &len, stream)) != -1) {
```

```
                GOTO_IF(waitpid(-1, NULL, WNOHANG), "waitpid", err
    );
                if (nread <= FILTER_LEN) {
                    GOTO_IF(write(pipes[0][WRITE_END], line, nread
    ) == -1, "write", err);
                } else {
                    GOTO_IF(write(pipes[1][WRITE_END], line, nread
    ) == -1, "write", err);
                }
            }
            GOTO_IF(errno == ENOMEM, "getline", err);
            ABORT_IF(close(pipes[0][WRITE_END]), "close");
            pipes[0][WRITE_END] = -1;
            ABORT_IF(close(pipes[1][WRITE_END]), "close");
            pipes[1][WRITE_END] = -1;
            for (int i = 0; i < 2; ++i) {
                int status;
                int waitPid = wait(&status);
                GOTO_IF(status || !(waitPid == pids[0] || waitPid
    == pids[1]), "wait", err);
            }
        }

        free(line);
        return 0;

    err:
        free(line);
        free(extraLine);
        errno = 0;
        close(fds[0]);
        close(fds[1]);
        close(pipes[0][READ_END]);
        close(pipes[0][WRITE_END]);
        close(pipes[1][READ_END]);
        close(pipes[1][WRITE_END]);
        if (errno == EIO) {
            abort();
        }
        return -1;
    }
```

utils.c

```
#include "utils.h"

#include <stdio.h>
#include <stdlib.h>

void ReverseString(char* string, size_t length) {
    for (size_t i = 0; i < length >> 1; ++i) {
        char temp = string[i];
        string[i] = string[length - i - 1];
        string[length - i - 1] = temp;
    }
}
```

child.c

```
#include <errno.h>
#include <stdlib.h>
```

```c
#include <unistd.h>
#include <sys/wait.h>

#include "error_handling.h"
#include "utils.h"

int main(void) {
    ssize_t nread;
    char *line = NULL;
    size_t len;

    while ((nread = getline(&line, &len, stdin)) != -1) {
        ReverseString(line, nread-1);
        GOTO_IF(printf("%s", line) < 0, "printf", err);
    }
    GOTO_IF(errno == ENOMEM, "getline", err);

    free(line);
    return 0;

err:
    free(line);
    return -1;
}
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>

#include "lab1.h"

int main(void) {
    const char *childPath = getenv("PATH_TO_CHILD");
    if (!childPath) {
        fprintf(stderr, "Set environment variable PATH_TO_CHILD\n");
        exit(EXIT_FAILURE);
    }
    return ParentRoutine(childPath , stdin);
}
```

6 Тесты

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

extern "C" {
#include "lab1.h"
}

#include <array>
#include <filesystem>
#include <fstream>
#include <memory>

namespace fs = std::filesystem;

TEST(FirstLabTests, SimpleTest) {
    const char *childPath = getenv("PATH_TO_CHILD");
    ASSERT_TRUE(childPath);

    const std::string fileWithInput = "input.txt";
    const std::string fileWithOutput1 = "output1.txt";
    const std::string fileWithOutput2 = "output2.txt";

    constexpr int outputSize1 = 5;
    constexpr int outputSize2 = 3;
    constexpr int inputSize = outputSize1 + outputSize2;

    const std::array<std::string, inputSize> input = {
        "rev",
        "revvvvv",
        "162te16782r18223r2",
        "",
        "r",
        "pqwrpqlwfpqwoglqwpglqpgwpqw.,g;q.wg;q.wg;w.qg;.w",
        "12345678900",
        "1234567890",
    };

    const std::array<std::string, outputSize1> expectedOutput1 = {
        "ver",
        "vvvvver",
        "",
        "r",
        "0987654321",
    };

    const std::array<std::string, outputSize2> expectedOutput2 = {
        "2r32281r28761et261",
        "w.;gq.w;gw.q;gw.q;g,.wqpwgpqlgpwqlgowqpfwlqprwqp",
        "00987654321",
    };

    {
        auto inFile = std::ofstream(fileWithInput);
        inFile << fileWithOutput1 << '\n'
               << fileWithOutput2 << '\n';
        for (const auto &line : input) {
            inFile << line << '\n';
        }
```

```
58      }
59
60      auto deleter = [](FILE *file) {
61          fclose(file);
62      };
63
64      std::unique_ptr<FILE, decltype(deleter)> inFile(fopen(
    fileWithInput.c_str(), "r"), deleter);
65
66      ASSERT_EQ(ParentRoutine(childPath, inFile.get()), 0);
67
68      auto outFile1 = std::ifstream(fileWithOutput1);
69      auto outFile2 = std::ifstream(fileWithOutput2);
70      ASSERT_TRUE(outFile1.good() && outFile2.good());
71
72      std::string result;
73
74      for (const std::string &line : expectedOutput1) {
75          std::getline(outFile1, result);
76          EXPECT_EQ(result, line);
77      }
78
79      for (const std::string &line : expectedOutput2) {
80          std::getline(outFile2, result);
81          EXPECT_EQ(result, line);
82      }
83
84      auto removeIfExists = [](const std::string &path) {
85          if (fs::exists(path)) {
86              fs::remove(path);
87          }
88      };
89
90      removeIfExists(fileWithInput);
91      removeIfExists(fileWithOutput1);
92      removeIfExists(fileWithOutput2);
93  }
94
95  TEST(FirstLabTests, ZeroOutputFileTest) {
96      const char *childPath = getenv("PATH_TO_CHILD");
97      ASSERT_TRUE(childPath);
98      const std::string fileWithInput = "input.txt";
99
100     {
101         auto inFile = std::ofstream(fileWithInput);
102     }
103     auto deleter = [](FILE *file) {
104         fclose(file);
105     };
106
107     std::unique_ptr<FILE, decltype(deleter)> inFile(fopen(
    fileWithInput.c_str(), "r"), deleter);
108
109     ASSERT_EQ(ParentRoutine(childPath, inFile.get()), -1);
110 }
111
112 TEST(FirstLabTests, OneOutputFileTest) {
113     const char *childPath = getenv("PATH_TO_CHILD");
114     ASSERT_TRUE(childPath);
```

```
115     const std::string fileWithInput = "input.txt";
116     const std::string fileWithOutput = "output.txt";
117 
118     {
119         auto inFile = std::ofstream(fileWithInput);
120         inFile << fileWithOutput << '\n';
121     }
122     auto deleter = [](FILE *file) {
123         fclose(file);
124     };
125 
126     std::unique_ptr<FILE, decltype(deleter)> inFile(fopen(
    fileWithInput.c_str(), "r"), deleter);
127 
128     ASSERT_EQ(ParentRoutine(childPath, inFile.get()), -1);
129     ASSERT_FALSE(fs::exists(fileWithOutput));
130 }
131 
132 TEST(FirstLabTests, EmptyInputTest) {
133     const char *childPath = getenv("PATH_TO_CHILD");
134     ASSERT_TRUE(childPath);
135 
136     const std::string fileWithInput = "input.txt";
137     const std::string fileWithOutput1 = "output1.txt";
138     const std::string fileWithOutput2 = "output2.txt";
139 
140     {
141         auto inFile = std::ofstream(fileWithInput);
142         inFile << fileWithOutput1 << '\n'
143                << fileWithOutput2 << '\n';
144     }
145 
146     auto deleter = [](FILE *file) {
147         fclose(file);
148     };
149 
150     std::unique_ptr<FILE, decltype(deleter)> inFile(fopen(
    fileWithInput.c_str(), "r"), deleter);
151 
152     ASSERT_EQ(ParentRoutine(childPath, inFile.get()), 0);
153 
154     auto outFile1 = std::ifstream(fileWithOutput1);
155     auto outFile2 = std::ifstream(fileWithOutput2);
156     ASSERT_TRUE(outFile1.good() && outFile2.good());
157     ASSERT_TRUE(fs::is_empty(fileWithOutput1) && fs::is_empty(
    fileWithOutput2));
158 
159     auto removeIfExists = [](const std::string &path) {
160         if (fs::exists(path)) {
161             fs::remove(path);
162         }
163     };
164 
165     removeIfExists(fileWithInput);
166     removeIfExists(fileWithOutput1);
167     removeIfExists(fileWithOutput2);
168 }
```

7 Демонстрация работы программы

```
ivan@asus-vivobook ~/c/o/b/lab1 (reports)> PATH_TO_CHILD=child lab1
1
2
3l,34lgt,3,;34,;h534;h,43;lh543
ergerg,ererg
fd

d
gfd
gewegelh,;
sgl,r,ge;
fsdf
ivan@asus-vivobook ~/c/o/b/lab1 (reports)> ls
1  2  CMakeFiles/  CTestTestfile.cmake  Makefile  child*  cmake_install.cmake  lab
ivan@asus-vivobook ~/c/o/b/lab1 (reports)> cat 1
df

d
dfg
;,hlegeweg
;eg,r,lgs
fdsf
ivan@asus-vivobook ~/c/o/b/lab1 (reports)> cat 2
345hl;34,h;435h;,43;,3,tgl43,l3
grere,gregre
```

8 Запуск тестов

```
ivan@asus-vivobook ~/c/o/build (reports) [1]> PATH_TO_CHILD=lab1/child tests/lab1_t
Running main() from /var/tmp/portage/dev-cpp/gtest-1.13.0/work/googletest-1.13.0/go
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from FirstLabTests
[ RUN      ] FirstLabTests.SimpleTest
[       OK ] FirstLabTests.SimpleTest (6 ms)
[ RUN      ] FirstLabTests.ZeroOutputFileTest
Input 2 filenames
[       OK ] FirstLabTests.ZeroOutputFileTest (0 ms)
[ RUN      ] FirstLabTests.OneOutputFileTest
Input 2 filenames
[       OK ] FirstLabTests.OneOutputFileTest (0 ms)
[ RUN      ] FirstLabTests.EmptyInputTest
[       OK ] FirstLabTests.EmptyInputTest (4 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.
```

9 Выводы

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