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: Останина Анна Андреевна
Группа: М8О-208Б-22
Вариант: 9
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Содержание

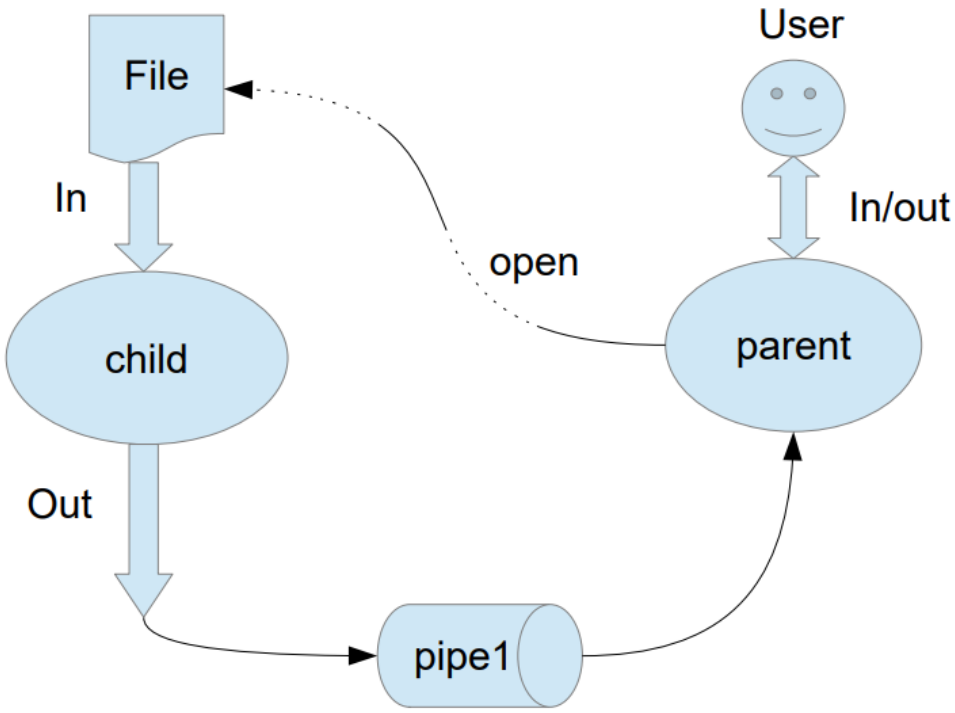
Репозиторий

https://github.com/Imariiii/os_labs

Цель работы

Изучить управление процессами в ОС и обеспечение обмена данных между процессами посредством каналов

Задание



9 вариант) В файле записаны команды вида:«число число число». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат выводит в стандартный поток вывода. 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.

Описание работы программы

Программа компилируется из файлов lab1.cpp с основным процессом, child.cpp с дочерним процессом. Также имеется заголовочный файл lab1.h и файл с тестами lab1_test.cpp. В программе работы были использованы следующие системные вызовы:

- fork() - создание нового процесса,
- execlp() - замена текущего образа процесса новым,
- pipe() - создание однонапрвленного канала для передачи строк родительским процессом дочернему,
- read() - чтение из pipe,
- write() - запись в pipe,
- close() - закрытие файлового дискриптора,
- dup2() - перенаправление одного файлового дескриптора на другой.

Исходный код

lab1.h
```
#pragma once
#include <iostream>

enum PipeEnd {
    READ_END,
    WRITE_END
};

void ParentRoutine(const char* pathToChild, std::istream& streamIn,
std::ostream& streamOut);
```

lab1.cpp
```
#include "lab1.h"

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

#include <cerrno>
#include <cmath>
#include <cstdlib>
#include <iostream>
#include <string>

void ParentRoutine(const char* pathToChild, std::istream& streamIn,
std::ostream& streamOut) {
    std::string filename;
    streamIn >> filename;

    int file = open(filename.c_str(), O_RDONLY);
    if (file == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    int fds[2];
    if (pipe(fds)) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }
```

```
    pid_t pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid > 0) {  // parent
        close(file);
        close(fds[WRITE_END]);
        double res;
        while (read(fds[READ_END], &res, sizeof(double)) > 0) {
            if (std::isinf(res)) {
                streamOut << "Division by zero\n";
                exit(EXIT_FAILURE);
            }
            streamOut << res << '\n';
        }
        close(fds[READ_END]);
    } else {  // child
        close(fds[READ_END]);
        dup2(file, STDIN_FILENO);
        close(file);
        dup2(fds[WRITE_END], STDOUT_FILENO);
        close(file);
        execl(pathToChild, "child", NULL);
        perror("exec");
        exit(EXIT_FAILURE);
    }
}
```

child.cpp

```
#include <iostream>
#include <sstream>
#include <string>
#include <unistd.h>

int main() {
    std::string line;
    while (std::getline(std::cin, line)) {
        std::istringstream iss(line);
        double dividend;
        if(!(iss >> dividend)) {
            dividend = 0;
        }
        double divisor;
        while (iss >> divisor) {
            dividend /= divisor;
            if (divisor == 0) {
                write(STDOUT_FILENO, &dividend, sizeof(double));
                return -1;
            }
        }
        write(STDOUT_FILENO, &dividend, sizeof(double));
    }
    return 0;
}
```

main.cpp

```
#include <cstdio>
#include <iostream>

#include "lab1.h"
```

```
int main() {
    const char *childPath = getenv("PATH_TO_CHILD");
    if (!childPath) {
        fprintf(stderr, "Set PATH_TO_CHILD\n");
        return -1;
    }
    ParentRoutine(childPath, std::cin, std::cout);
}
```

lab1_test.cpp

```
#include "lab1.h"

#include <gtest/gtest.h>

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

namespace fs = std::filesystem;

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

    const std::string filenameInput = "in.txt";
    const std::string filenameTest = "test.txt";
    const std::string filenameOutput = "out.txt";

    constexpr int testSize = 5;
    constexpr int outputSize = testSize;

    const std::array<std::string, outputSize> testLines = {
        "10 1",
        "100 10",
        "",
        "1 9 9",
        "100 3 4 6",
    };

    const std::array<std::string, outputSize> expectedOutputLines = {
        "10",
        "10",
        "0",
        "0.0123457",
        "1.38889",
    };

    {
        auto input = std::ofstream(filenameInput);
        input << filenameTest << '\n';
        auto test = std::ofstream(filenameTest);
        for (const auto &line : testLines) {
            test << line << '\n';
        }
    }
    {
        auto input = std::ifstream(filenameInput);
        auto output = std::ofstream(filenameOutput);

        ParentRoutine(childPath, input, output);
    }
```

```
    auto output = std::ifstream(filenameOutput);

    std::string result;
    for (const std::string &line : expectedOutputLines) {
        std::getline(output, result);
        EXPECT_EQ(result, line);
    }

    auto removeIfExists = [](const std::string &path) {
        if (fs::exists(path)) {
            fs::remove(path);
        }
    };

    removeIfExists(filenameInput);
    removeIfExists(filenameTest);
    removeIfExists(filenameOutput);
}
```

Консоль

```
anna@anna-virtual-machine:~/labs_3sem/os_labs/build/lab1$
PATH_TO_CHILD=../lab1/child ./lab1
lab1.txt
2
11
1
3
Division by zero
```

Запуск тестов

```
anna@anna-virtual-machine:~/labs_3sem/os_labs/build/tests$
PATH_TO_CHILD=../lab1/child ./lab1_test
Running main() from /home/anna/labs_3sem/os_labs/build/_deps/googletest-
src/googletest/src/gtest_main.cc
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from FirstLabTests
[ RUN      ] FirstLabTests.SimpleTest
[       OK ] FirstLabTests.SimpleTest (8 ms)
[----------] 1 test from FirstLabTests (9 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (9 ms total)
[  PASSED  ] 1 test.
```

Выводы

В ходе выполнения данной лабораторной работы я изучила работу процессов, реализацию обмена информацией между дочерним и родительским процессами с использованием каналов в ОС Linux. В процессе работы я познакомилась с системными вызовами, которые представляют собой интерфейс для взаимодействия прикладных программ с операционной системой.

Системные вызовы отличаются от обычных функций тем, что они выполняются на уровне ядра операционной системы и предоставляют доступ к ресурсам, защищенным от прямого доступа программ. Эти вызовы позволяют программам выполнять такие операции, как создание процессов (fork), создание каналов для межпроцессного взаимодействия (pipe), замещение текущего процесса новой программой (execlp) и дублирование файловых дескрипторов (dup2), что широко используется для реализации различных аспектов многозадачности и взаимодействия процессов в операционных системах.

Также я изучила устройство памяти процесса, которое делится на несколько секций, каждая из которых отвечает за определенные части работы программы.