Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 2 по курсу**

**«Операционные системы»**

Студент: Пирогов М.Д.

Группа: М8О-207Б-21

Вариант: 12

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/pirogovmark/OS-Labs/

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в управлении процессов в С++ и 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) 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.

12 вариант) Child1 переводит строки в верхний регистр. Child2 убирает все задвоенные пробелы.

**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: unistd.h, ... В программе используются следующие системные вызовы:

1. pipe() - создает канал для чтения и записи
2. fork() - создает новый процесс, копируя вызывающий процесс
3. read() & write() - соответственно чтение и запись из файлового дескриптора в буфер
4. close() - закрывает файловый дескриптор

**Общий метод и алгоритм решения**

C помощью системного вызова fork создается Child 1. В Parent снова используем вызов fork(), создавая тем самым Child 2. Pipe вызывается 3 раза для связи Parent - Child 1, Child 1 - Child 2 и Child 2 - Parent. Через каждый вызов pipe будем передавать данные в соответствии со схемой задания варианта.

**Исходный код**

#include <iostream>

#include <unistd.h>

#include <string>

std::string spaces(std::string s) {

std::string out = "";

char prev = 'a';

for (int i = 0; i < s.size(); ++i) {

if (!(prev == ' ' && s[i] == ' ')) {

out += s[i];

prev = s[i];

}

}

return out;

}

std::string to\_upper(std::string s) {

for (int i = 0; i < s.size(); ++i) {

if (s[i] >= 'a' && s[i] <= 'z') {

s[i] = toupper(s[i]);

}

}

return s;

}

int main() {

int pipefd\_p1[2]; // Pipe for communicating Parent and Child1

pipe(pipefd\_p1);

int pipefd\_12[2]; // Pipe for communicating Child1 and Child2

pipe(pipefd\_12);

int pipefd\_2p[2]; // Pipe for communicating Child2 and Parent

pipe(pipefd\_2p);

int id\_1 = fork();

if (id\_1 == -1) {

return -1;

} else if (id\_1 == 0) { // Child 1

std::string s\_1;

read(pipefd\_p1[0], &s\_1, sizeof(std::string));

std::cout << "Child 1 in: " << s\_1 << "\n";

close(pipefd\_p1[0]);

close(pipefd\_p1[1]);

s\_1 = to\_upper(s\_1);

std::cout << "Child 1 out: " << s\_1 << "\n";

write(pipefd\_12[1], &s\_1, sizeof(std::string));

close(pipefd\_12[0]);

close(pipefd\_12[1]);

std::cout << '\n';

} else {

int id\_2 = fork();

if (id\_2 == -1) {

return -1;

} else if (id\_2 == 0) { // Child 2

std::string s\_2;

read(pipefd\_12[0], &s\_2, sizeof(std::string));

std::cout << "Child 2 in: " << s\_2 << "\n";

close(pipefd\_12[0]);

close(pipefd\_12[1]);

s\_2 = spaces(s\_2);

std::cout << "Child 2 out: " << s\_2 << "\n";

write(pipefd\_2p[1], &s\_2, sizeof(std::string));

close(pipefd\_2p[0]);

close(pipefd\_2p[1]);

std::cout << '\n';

} else { // Parent

std::cin >> s;

std::cout << "Parent in: " << s << "\n\n";

write(pipefd\_p1[1], &s, sizeof(std::string));

close(pipefd\_p1[0]);

close(pipefd\_p1[1]);

read(pipefd\_2p[0], &s, sizeof(std::string));

std::cout << "Parent out: " << s << "\n";

close(pipefd\_2p[0]);

close(pipefd\_2p[1]);

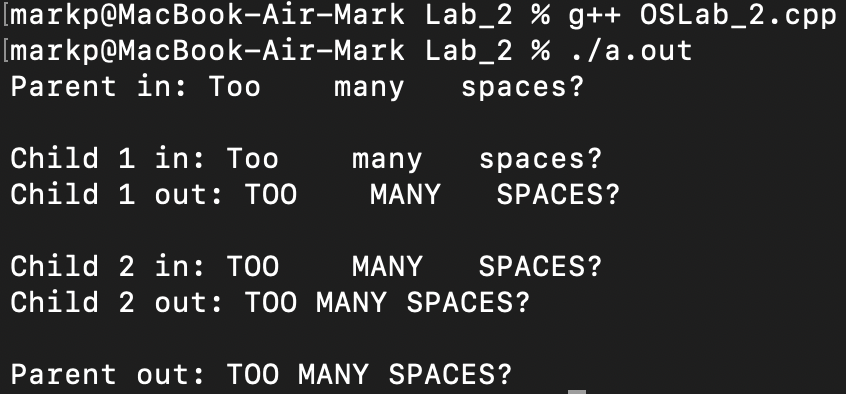
}

}

return 0;

}

**Демонстрация работы программы**



**Выводы**

Я приобрел навыки управления процессами в С++ и обеспечении обмена данных между процессами через каналы.