Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Тема работы**

**“Изучение взаимодействий между процессами”**

Студент: Ханнанов Руслан Маратович

Группа: М8О-208Б-20

Вариант: 14

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**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/Naksen/OS>

**Постановка задачи**

Задача: Родительский процесс создает два дочерних процесса. Через pipe fd\_1 идет передача данных из родительского процесса в первый дочерний. Первый дочерний процесс переводит строки в нижний регистр, оправляет уже изменные данные через pipe fd в во второй дочерний процесс. Второй дочерний процесс убирает все задвоенные пробелы и отправляет данные назад в родительский процесс через pipe fd\_2. Родительский процесс производит вывод данных в консоль.

**Общие сведения о программе**

Реализация программы была бы невозможна без специальной библиотеки “unistd.h” для операционной системы Linux, которая позволяет работать с процессами и системными вызовами. По мере реализации задания используются такие строки(команды), как:

int fd[2] - создание массива из 2 дескрипторов, 0 - чтение (read), 1 - передача (write):

pipe(fd) - конвейер, с помощью которого выход одной команды подается на вход другой (оно же “труба”);

int pid\_1 = fork () - создание дочернего процесса, в переменной pid\_1 будет лежать “специальный код” процесса (-1 - ошибка fork, 0 - дочерний процесс, >0 - родительский);

read(...) - команда, предназначенная для чтения данных, посланных из

другого процесса, принимающая на вход три параметра: элемент массива дескрипторов с индексом 0, значение получаемого объекта (переменной, массива и т.д.), размер получаемого объекта (например, в случае переменной int - sizeof(int), в случае массива из 10 переменных типа int - sizeof(int) \* 10);

write(...) - команда, принимающая на вход три параметра: элемент массива дескрипторов с индексом 1, значение посылаемого объекта (переменной, массива и т.д.), размер посылаемого объекта (например, в случае переменной int - sizeof(int), в случае массива из 10 переменных типа int - sizeof(int) \* 10);

close(...) - команда, использующаяся, когда нам больше не нужно передавать, либо считывать что-либо из другого процесса.

**Общий метод и алгоритм решения**

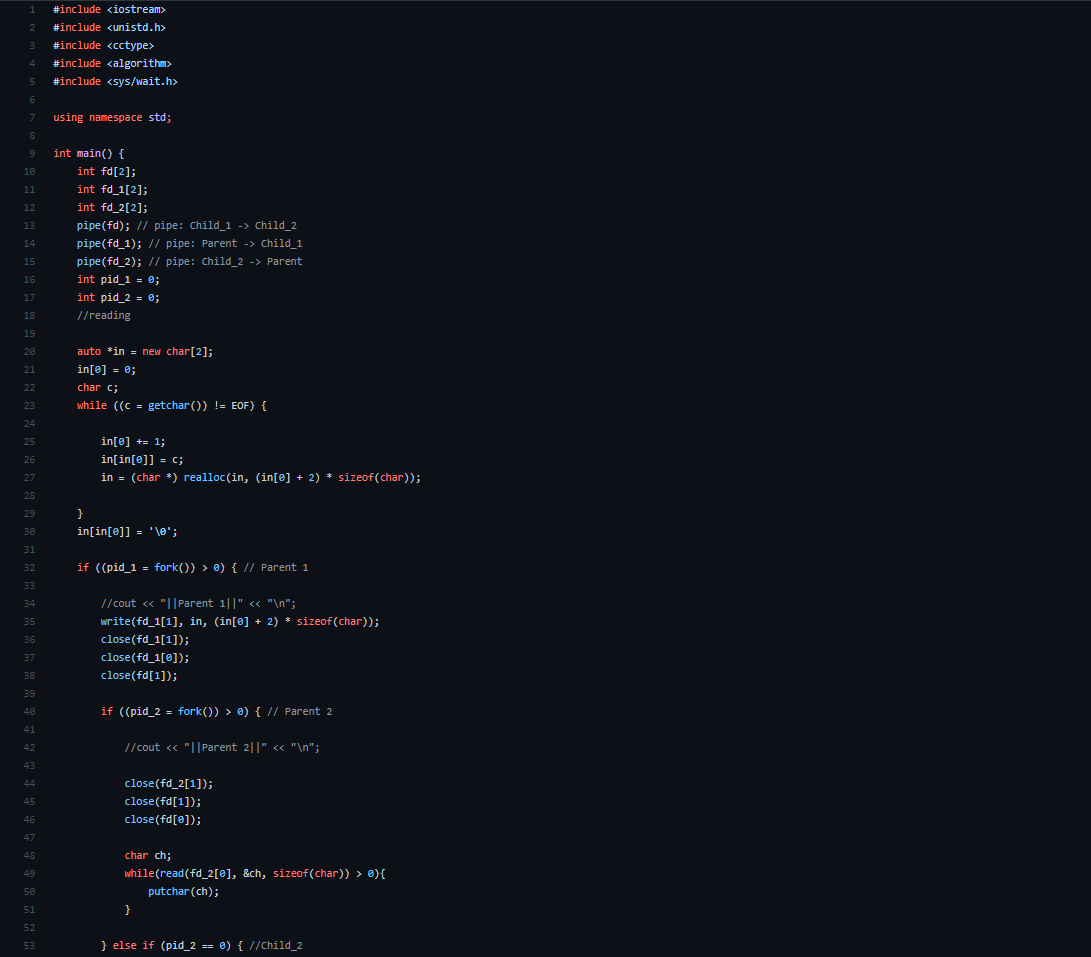
Программа получает на вход символьные данные. После чего данные считываются в “буфферный” массив. В первом дочернем процессе изменяется стандартный поток ввода stdin на дескриптор чтения fd\_1[0]. А стандартный поток вывода stdout на дескриптор записи fd[1]. Во втором дочернем процессе тоже происходит замена стандартных поток ввода и вывода. Каждый символ строки переходит в нижний регистр использованием функции tolower. А поиск на задвоенные пробелы проходит с помощью поиска двух символов пробела стоящих в строке подряд. Как следствие, первый и второй дочерние процессы производят обработку данных и возвращают их в родительский процесс. Программа завершена.

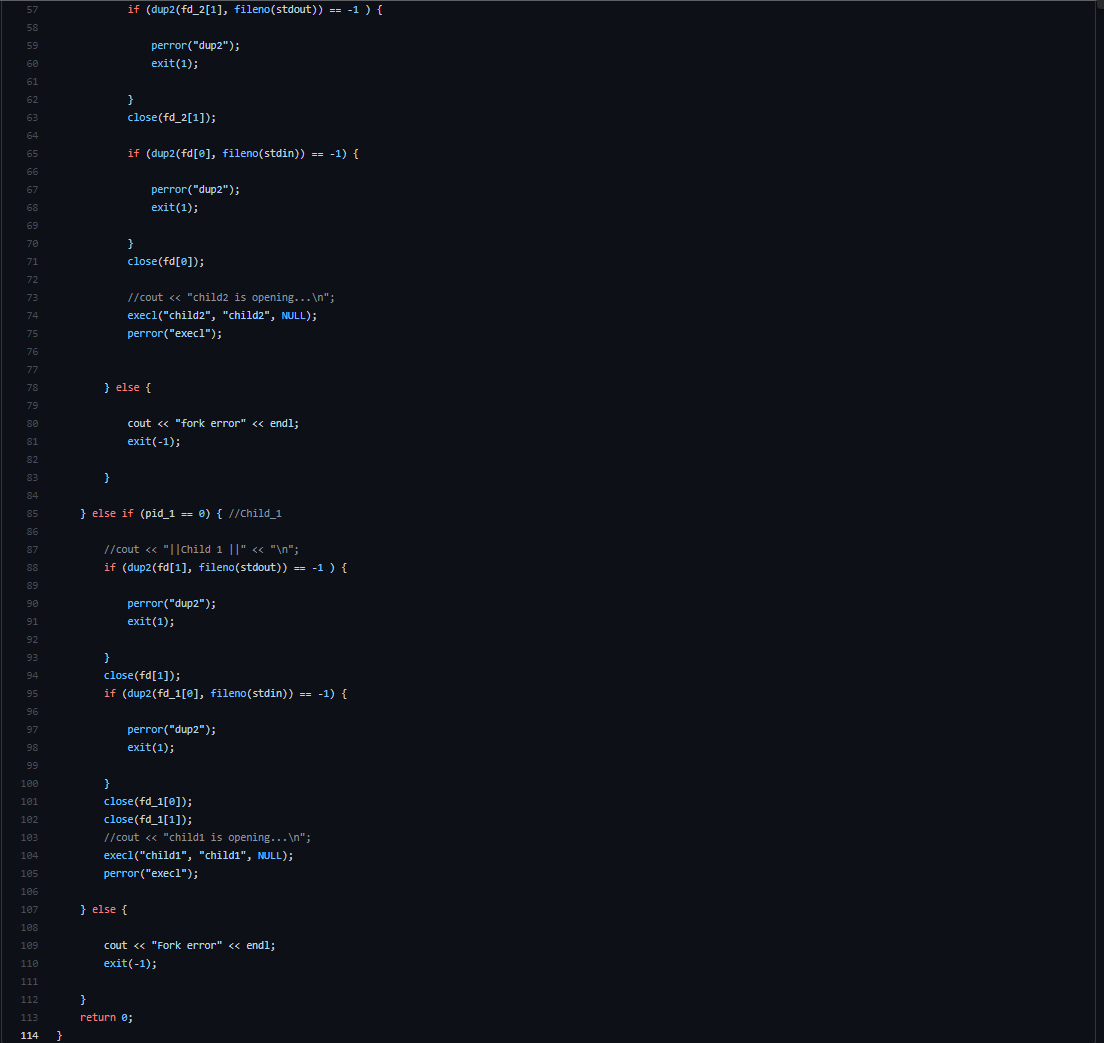
Лабораторная работа была выполнена в среде Visual Studio code, название файла – main.cpp.

Собирается программа при помощи команды Makefile, находящегося в папке src.

**Исходный код**

**main.c:**

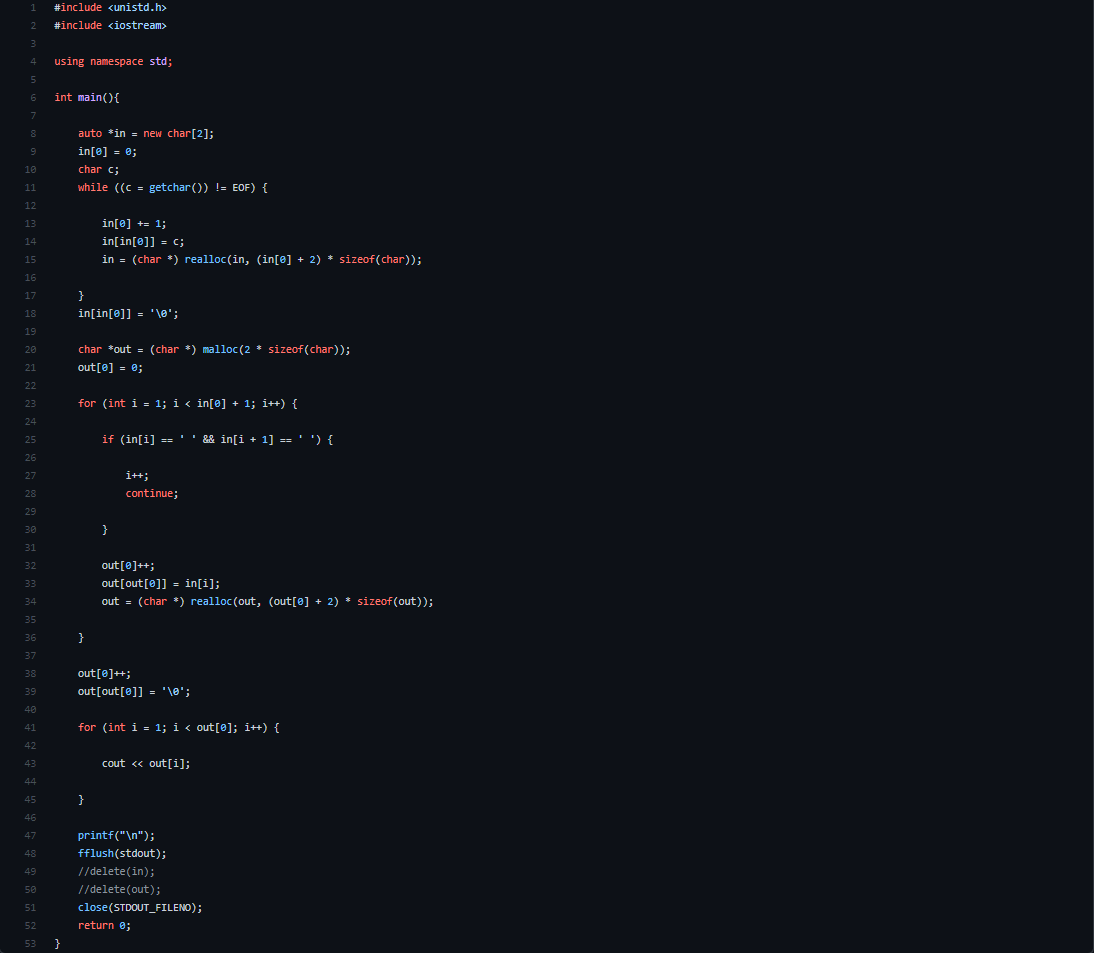




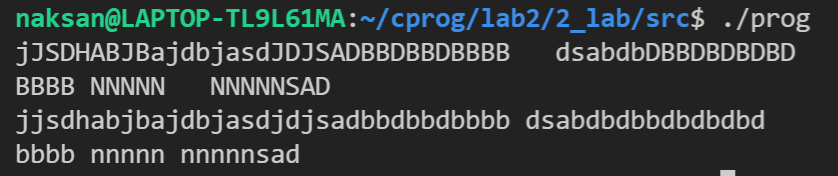
**child1.cpp:**



**child2.cpp:**



**Демонстрация работы программы**



**Выводы**

После выполнения данной лабораторной работы я с уверенностью могу сказать, что хорошо ознакомился с темой создания процессов в Linux. Я на примере собственного задания осознал принципы работы вышеперечисленных команд pipe, fork, write, read, close, dup2, научился ими пользоваться и даже познал некоторые тонкости (например, все, что было создано в родительском процессе, есть и в дочернем). Уверен, что полученные навыки помогут мне дальше осваивать курс по операционным системам.