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: Зубко Дмитрий Валерьевич

Группа: М8О-208Б-20

Вариант: 7

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

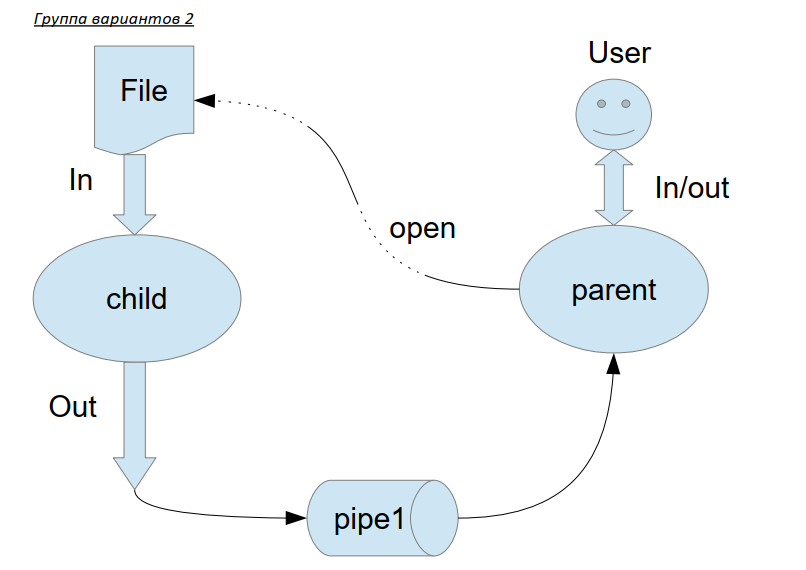
1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

<https://github.com/usernameMAI/OS/tree/main/os_lab2>

**Постановка задачи**

**Цель работы**  
Приобретение практических навыков в:  
 1). Управление процессами в ОС;  
 2). Обеспечение 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.



7 вариант) В файле записаны команды вида: «число число число<endline>». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип float. Количество чисел может быть произвольным.

**Общие сведения о программе**

Программа представляет из себя два файла – parent.cpp и child.cpp. Используются заголовочные файлы: stdio.h, unistd.h, iostream, stdlib.h, string.h, fstream, fcntl.h. В программе используются следующие системные вызовы:

1. **fork –** создает копию текущего процесса, который является дочерним процессом для текущего процесса;
2. **pipe –** создает однонаправленный канал данных, который можно использовать для взаимодействия между процессами;
3. **open –** открывает файловый дескриптор;
4. **close –** закрывает файловый дескриптор;
5. **dup2 –** переназначение файлового дескриптора.

**Общий метод и алгоритм решения**

Опишите принцип работы программы, с какими ключами её можно запустить (при наличии).

**Исходный код**

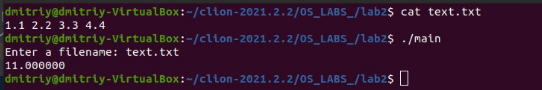
parent.cpp

#include <iostream>  
#include <unistd.h>  
#include <stdlib.h>  
#include <string.h>  
#include <fstream>  
#include <fcntl.h>  
using namespace std;  
  
int main() {  
 string s;  
 cout << "Enter a filename: ";  
 cin >> s;  
 //Преобразуем в const char\*, для работы access  
 const char\* buf = s.c\_str();  
  
 //Проверяем, есть ли файл с именем buf  
 if(access(buf, R\_OK) == -1){  
 cout << "File not exist" << endl;  
 return -1;  
 }  
  
 //Создаём канал данных для межпроцессорного взаимодействия   
 int fd[2];  
 pipe(fd);  
  
 //Создаем процесс-потомок  
 int pid = fork();  
 if(pid == -1){  
 perror("fork");  
 return -1;  
 }  
 else if(pid == 0){  
 int file = open(buf, O\_RDONLY);  
 if (file == -1) {  
 perror("no open");  
 return -1;  
 }  
 //перенаправляет   
 if (dup2(fd[1], 1) == -1) {  
 exit(1);  
 }  
 if (dup2(file, 0) == -1) {  
 exit(1);  
 }  
 close(fd[0]);  
 close(fd[1]);  
 close(file);  
 //открывается программа child  
 execl("child", "child", NULL);  
 }  
 close(fd[1]);  
 char c;  
 //трубка в родительском, вывод в стдаут  
 while(read(fd[0], &c, sizeof(char)) > 0){  
 putchar(c);  
 }  
 close(fd[0]);  
 return 0;  
}

child.cpp

#include <stdio.h>  
#include <unistd.h>  
using namespace std;  
int main(){  
 double a;  
 char c;  
 double res = 0;  
 int k = 0;  
 while(scanf("%lf%c", &a, &c) != EOF) {  
 res += a;  
 if(c == '\n') {  
 printf("%lf\n",res);  
 //Сбрасывает буферы потока  
 fflush(stdout);  
 res = 0.;  
 continue;  
 }  
 }  
 return 0;  
}

**Демонстрация работы программы**



**Выводы**

Я приобрел практические навыки в управлении процессами в ОС Unix и обеспечении данных между процессами при помощи каналов. Также я научился работать с системными вызовами и файловыми дескрипторами.