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**

Студент: Шапошник Никита Сергеевич

Группа: М8О-208Б-21

Вариант: 15

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**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/Nikitashap/OS/tree/main/lab2**](https://github.com/Nikitashap/OS/tree/main/lab2)

**Постановка задачи**

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. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть 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.

Вариант 15) Правило проверки: строка должна начинаться с заглавной буквы

**Общие сведения о программе**

Программа представляет из себя один файл lab2.c .

**Общий метод и алгоритм решения**

С помощью вызова fork создаются родительский и дочерний процессы, родительский процесс считывает название будущего файла и некоторое количество строк, которые передаются в дочерний процесс. Дочерний процесс проверяет, соответствует ли строка заданному условию. Если да, то записывает строку в файл, если нет, то передает информацию об ошибке.

**Исходный код**

**#include <unistd.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <fcntl.h>**

**#include <stdbool.h>**

**char\* get\_string(int \*len) {**

**\*len = 0;**

**int capacity = 1;**

**char \*s = (char\*) malloc(sizeof(char));**

**char c = getchar();**

**if(c == '\n') c = getchar();**

**while (c != '\n') {**

**s[(\*len)++] = c;**

**if (\*len >= capacity) {**

**capacity \*= 2;**

**s = (char\*) realloc(s, capacity \* sizeof(char));**

**}**

**c = getchar();**

**}**

**s[\*len] = '\0';**

**return s;**

**}**

**bool check\_conditions(char\* str){**

**if (str[0] >= 'A' && str[0] <= 'Z'){**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**int main(void)**

**{**

**char \*filename;**

**int fd1[2];**

**int fd2[2];**

**int len;**

**filename = get\_string(&len);**

**int file = open(filename, O\_RDWR | O\_CREAT | O\_APPEND , 0777);**

**if (file < 0){**

**printf("File error");**

**exit(1);**

**}**

**if(pipe(fd1) == -1){**

**printf("Pipe failed\n");**

**exit(2);**

**}**

**if(pipe(fd2) == -1){**

**printf("Pipe failed\n");**

**exit(3);**

**}**

**pid\_t child = fork();**

**if(child == -1){**

**printf("Fork failed\n");**

**exit(4);**

**}**

**if(child == 0){**

**while(1) {**

**int size;**

**char string[size];**

**read(fd1[0], &size, sizeof(int));**

**read(fd1[0], string, sizeof(char) \* size);**

**if(check\_conditions(string)){;**

**write(file, string, sizeof(char) \* size);**

**write(file, "\n", sizeof(char));**

**write(fd2[1],"1",sizeof(char));**

**}else{**

**write(fd2[1],"0",sizeof(char));**

**}**

**}**

**} else {**

**while(1) {**

**int length;**

**char \*string = get\_string(&length);**

**write(fd1[1], &length, sizeof(int));**

**write(fd1[1], string, sizeof(char) \* length);**

**char err;**

**read(fd2[0], &err, sizeof(char));**

**if (err == '0'){**

**write(1,"Error", sizeof(char)\*5);**

**write(1,"\n", sizeof(char));**

**}**

**}**

**close(fd1[0]); close(fd2[0]);**

**close(fd1[1]); close(fd2[1]);**

**close(file);**

**}**

**return 0;**

**}**

**Демонстрация работы программы**

Ввод в консоль:

iMac-Apple:lab2 appleimac$ ./a.out

file

Truth

TTr

error

Error

^C

iMac-Apple:lab2 appleimac$ cat file

Truth

TTr

**Выводы**

Проделав лабораторную работу, я приобрёл практические навыки в управлении процессами в ОС и обеспечил обмен данными между процессами посредством каналов.