**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Операционные системы" №1**

Студент: Федоров М. А.

Преподаватель: Миронов Е.С.

Группа: М8О-207Б-22

Дата: 15.09.2022

Оценка:

Подпись:

Оглавление

[Цель работы 3](#__RefHeading__738_2004454639)

[Постановка задачи 3](#__RefHeading__740_2004454639)

[Общие сведения о программе 4](#__RefHeading__742_2004454639)

[Общий алгоритм решения 5](#__RefHeading__744_2004454639)

[Реализация 5](#__RefHeading__746_2004454639)

[Пример работы 8](#__RefHeading__748_2004454639)

[Вывод 9](#__RefHeading__750_2004454639)

# **Цель работы**

Приобретение практических навыков в:

* Управлении процессами в ОС
* Обеспечении обмена данных между процессами посредством каналов

# **Постановка задачи**

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 решения 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.

*Группа вариантов 4:*

*Изображение выглядит как текст, диаграмма, круг, снимок экрана

Автоматически созданное описание*

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 представлена двумя файлами **–** l1\_child.c и l1\_parent.c.

В программе используются следующие системные вызовы:

**pipe() –** создаёт однонаправленный канал данных, который можно использовать для взаимодействия между процессами(конвейер)

**fork() –** создание дочернего процесса, в переменной id будет лежать «специальный код» процесса(-1 -ошибка, 0- дочерний процесс, >0- родительский)

**read()** **–** чтение из канала pipe()

**write()** **–** запись в канал pipe()

**dup2() –** перенаправление дескриптора

**execl() –** создание процесса с другой программой

**close() –** закрытие файлового дескриптора, который после этого не ссылается ни на один и файл и может быть использован повторно.

# **Общий алгоритм решения**

# 

# **Реализация**

**L1\_child.c**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

#include<ctype.h>

    const int STDIN\_STREAM\_NUMBER = 0;

    const int STDOUT\_STREAM\_NUMBER = 1;

int main(int argc, char\* argv[]){

    int number\_of\_parameters = strtol(argv[1], NULL, 10);

    int pipe\_buffer\_from\_child\_to\_stdout[2];

    int pipe\_buffer\_from\_father\_to\_child[2];

    int pipe\_buffer\_from\_child\_to\_father[2];

    char\* pipe\_input = malloc(100);

    char\* filename = argv[2];

    int i = 0;

    FILE \* filestream = fopen(filename, "w");

    pipe\_buffer\_from\_child\_to\_father[1] = number\_of\_parameters%10;

    number\_of\_parameters /= 10;

    pipe\_buffer\_from\_child\_to\_father[0] = number\_of\_parameters%10;

    number\_of\_parameters /= 10;

    pipe\_buffer\_from\_father\_to\_child[1] = number\_of\_parameters%10;

    number\_of\_parameters /= 10;

    pipe\_buffer\_from\_father\_to\_child[0] = number\_of\_parameters%10;

    pipe(pipe\_buffer\_from\_child\_to\_stdout);

    close(pipe\_buffer\_from\_father\_to\_child[1]);

    close(pipe\_buffer\_from\_child\_to\_father[0]);

    close(pipe\_buffer\_from\_child\_to\_stdout[0]);

    dup2(pipe\_buffer\_from\_father\_to\_child[0], STDIN\_STREAM\_NUMBER);

    dup2(pipe\_buffer\_from\_child\_to\_stdout[1], STDOUT\_STREAM\_NUMBER);

    do{

        read(pipe\_buffer\_from\_father\_to\_child[0], pipe\_input, 100);

        if (isupper(pipe\_input[0])){

            fprintf(filestream, "%s", pipe\_input);

        } else if (\*pipe\_input != '\0') {

            write(pipe\_buffer\_from\_child\_to\_father[1], "1", 1);

            close(pipe\_buffer\_from\_father\_to\_child[0]);

            close(pipe\_buffer\_from\_child\_to\_father[1]);

            close(pipe\_buffer\_from\_child\_to\_stdout[1]);

            fclose(filestream);

            return -1;

        }

        write(pipe\_buffer\_from\_child\_to\_father[1], "0", 1);

    } while(\*pipe\_input != '\0');

    close(pipe\_buffer\_from\_father\_to\_child[0]);

    close(pipe\_buffer\_from\_child\_to\_father[1]);

    close(pipe\_buffer\_from\_child\_to\_stdout[1]);

    fclose(filestream);

    printf("child process succesfully end work\n");

    return 0;

}

**-----------------------------------------------------------**

**L1\_parent.c**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

#include<sys/wait.h>

int main(){

    printf("max size of input is 100 symbols. If input has more symbols, programm may work incorrect. Every input must start with hight letter\n");

    char\* filename = malloc(100);

    char\* std\_input = malloc(100);

    char child\_status[1];

    char child\_parameters[4];

    int pipe\_buffer\_from\_father\_to\_child[2];

    int pipe\_buffer\_from\_child\_to\_father[2];

    printf("input filename to write:");

    scanf("%s", filename);

    if (pipe(pipe\_buffer\_from\_father\_to\_child) == -1){

        perror("error when creationg pipe");

        return -1;

    } else if (pipe(pipe\_buffer\_from\_child\_to\_father) == -1){

        perror("error when creationg pipe");

        return -1;

    }

    pid\_t pid = fork();

    if (pid == -1){

        perror("fork");

        return -1;

    } else if (pid == 0) {

        child\_parameters[0] = (pipe\_buffer\_from\_father\_to\_child[0]+'0');

        child\_parameters[1] = (pipe\_buffer\_from\_father\_to\_child[1]+'0');

        child\_parameters[2] = (pipe\_buffer\_from\_child\_to\_father[0]+'0');

        child\_parameters[3] = (pipe\_buffer\_from\_child\_to\_father[1]+'0');

        execl("./child", "./child", child\_parameters, filename, NULL);

    } else {

        close(pipe\_buffer\_from\_father\_to\_child[0]);

        close(pipe\_buffer\_from\_child\_to\_father[1]);

        getc(stdin);

        printf("scanning:");

        while (fgets(std\_input, 100, stdin)!=NULL && \*std\_input != '\n') {

            write(pipe\_buffer\_from\_father\_to\_child[1], std\_input, 100);

            read(pipe\_buffer\_from\_child\_to\_father[0], child\_status, 1);

            if (child\_status[0] == '1'){

                wait(NULL);

                close(pipe\_buffer\_from\_father\_to\_child[1]);

                close(pipe\_buffer\_from\_child\_to\_father[0]);

                printf("error: wrong string format\n");

                return -1;

            }

            printf("scanning:");

        }

        printf("\n");

        write(pipe\_buffer\_from\_father\_to\_child[1], "\0", 1);

    }

    printf("waiting for end of child process...\n");

    wait(NULL);

    printf("parent process succesfully end work\n");

    return 0;

}

**-----------------------------------------------------------**

# **Пример работы**

**Test 1**

|  |  |
| --- | --- |
| Input | Output |
| 1  Asad  Fasd |  |

# **Вывод**