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: Савинова Екатерина
Группа: М8О-207Б-21
Вариант: 15
Преподаватель: Черемисинов Максим
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/savinova-kati/operating-systems

Постановка задачи

Цель работы

Приобретение практических навыков в:

- Управление процессами в ОС
- Обеспечение обмена данных между процессами посредством каналов

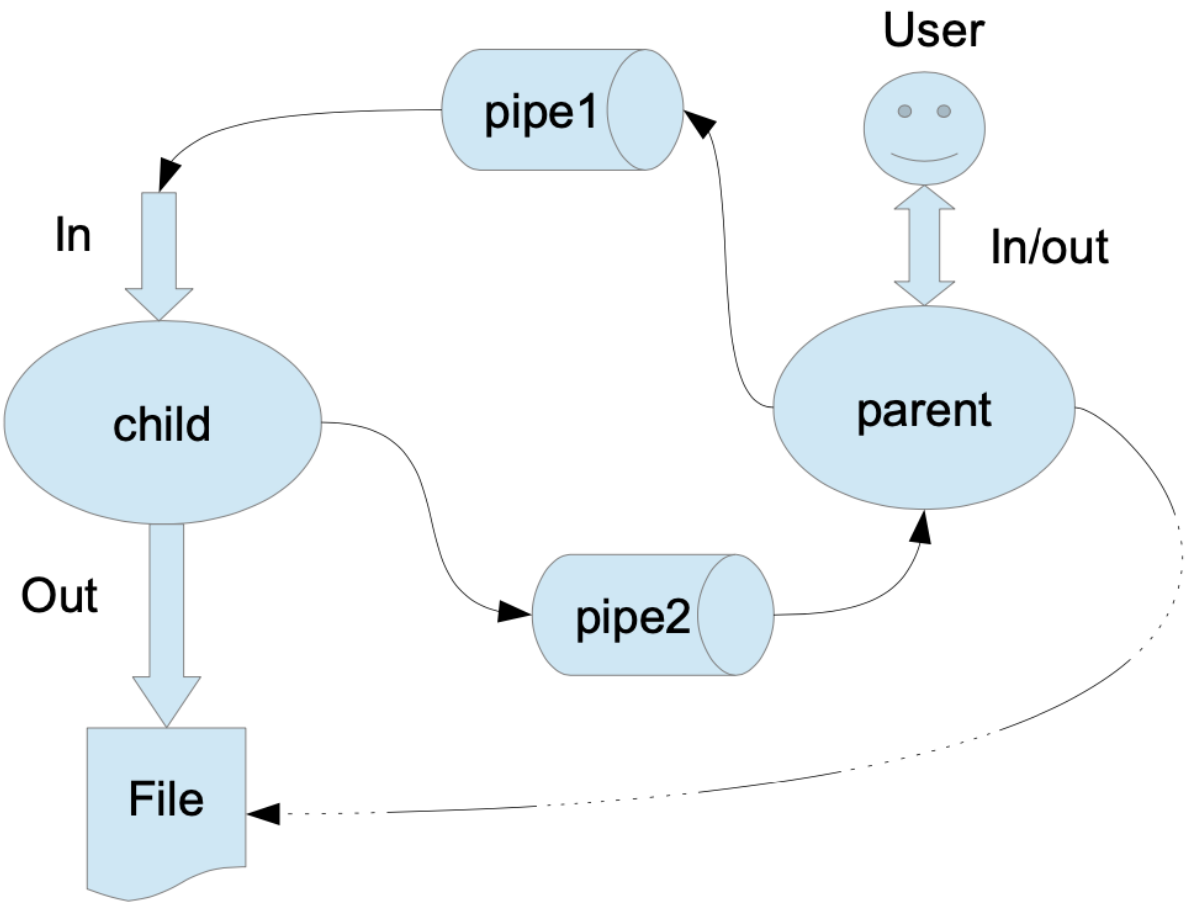
Задание

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.

Группа вариантов 4:

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 компилируется из файла main.cpp с помощью cmake. Дочерний процесс представлен в child_.cpp

Системные вызовы:

- *pid_t fork() – создание дочернего процесса*
- *int execl(const char *filename, char *const argv[], char *const) – замена образа памяти процесса*
- *int pipe(int pipefd[2]) – создание неименованного канала для передачи данных между процессами*
- *int open(const char *pathname, int flags, mode_t mode) – открытие\создание файла*

- int close(int fd) – закрыть файл

Общий метод и алгоритм решения

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки. Если строка начинается с заглавной буквы, 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.

Исходный код

main.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <sys/types.h>
#include <unistd.h>
#include <fstream>
#include <errno.h>
#include <signal.h>
#include <sys/wait.h>


using namespace std;

int main(){

    fstream file_1;
```

```
string name;
cout<<"Enter filename: "<<endl;

cin >> name;


int truba[2];
int truba_2[2];
int pipe_1[2];
int pipe_2[2];

if (pipe(pipe_1) == -1){
        perror("pipe");
        exit(EXIT_FAILURE);
}

if (pipe(pipe_2) == -1){
        perror("pipe");
        exit(EXIT_FAILURE);
}


string string_r;

pid_t id = fork();

if (id == -1){
        perror("fork");
```

```
            cout << "1";
            exit(EXIT_FAILURE);
        }

        else if (id == 0) {
            truba[0] = pipe_1[0];
            truba[1] = pipe_1[1];
            truba_2[0] = pipe_2[0];
            truba_2[1] = pipe_2[1];



            execl("./child_", to_string(truba[0]).c_str(), to_string(truba[1]).c_str(),
to_string(truba_2[0]).c_str(), to_string(truba_2[1]).c_str(), name.c_str(), NULL);


        }


        else {
            cout<<"Enter amount of strings: ";
            int amount;
            cin >> amount;
            cout << endl;
            for (int i = 0; i < amount; ++i) {
                cin >> string_r;
                int s_size = string_r.size();
                char str_array[s_size];
                for (int k = 0; k < s_size; ++k) {
                    str_array[k] = string_r[k];
                }
```

```
                write(pipe_1[1], &s_size, sizeof(int));
                write(pipe_1[1], str_array, sizeof(char)*s_size);

                int flag_;

                read(pipe_2[0], &flag_, sizeof(int));

                //cout << truba_2[0] << endl;
                if (flag_ == 1) {
                        cout << "The string does not fit the rule" << endl;
                }
        }
    }

    close(pipe_1[0]);
    close(pipe_1[1]);
    close(pipe_2[0]);
    close(pipe_2[1]);

    return 0;

}
```

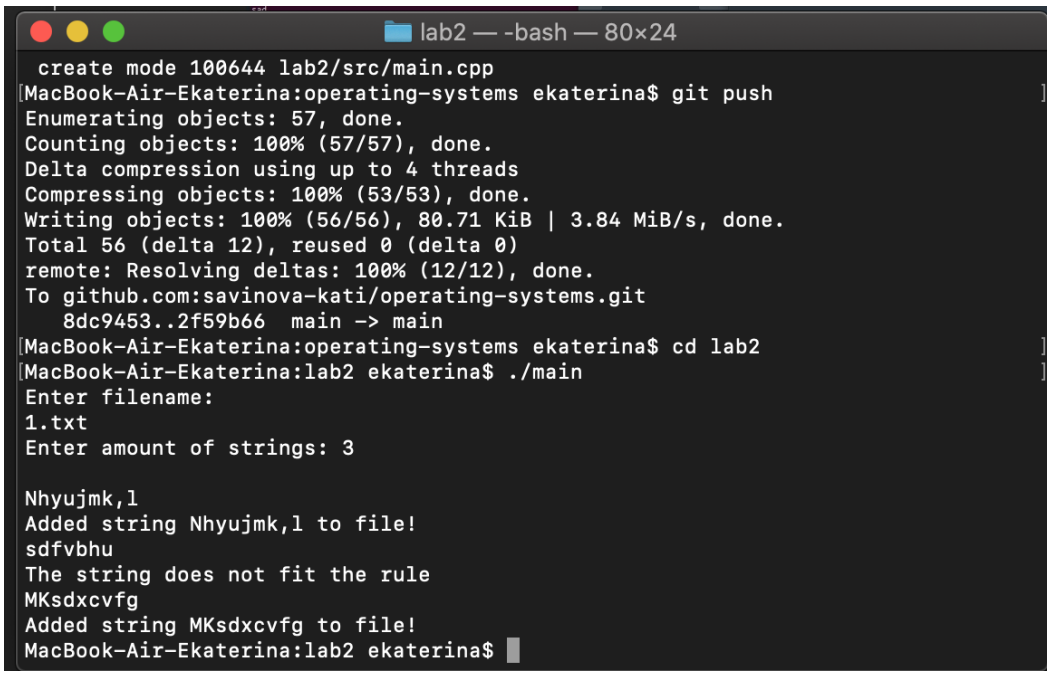
child_.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <sys/types.h>
#include <unistd.h>
#include <fstream>
```

```
#include <errno.h>
#include <signal.h>
#include <sys/wait.h>


using namespace std;


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

	string filename = argv[4];
	int truba[2];
	int truba_2[2];
	int flag1 = 1;
	int flag2 = 22;

	truba[0] = stoi(argv[0]);
	truba[1] = stoi(argv[1]);

	truba_2[0] = stoi(argv[2]);
	truba_2[1] = stoi(argv[3]);


	fstream file_1;
	file_1.open(filename, fstream::in | fstream::out | fstream::app);

	while(true) {
		int stroka_size;

		read(truba[0], &stroka_size, sizeof(int));
		char stroka[stroka_size];
		read(truba[0], &stroka, sizeof(char)*stroka_size);
```

```
                string result;
                for (int i = 0; i < stroka_size; i++) {
                        result.push_back(stroka[i]);
                }
                if (stroka[0] >= 65 && stroka[0] <= 90) {
                        file_1 << result << endl;
                        cout << "Added string " << result << " to file!" << endl;

                        write(truba_2[1], &flag2, sizeof(int));

                        //cout << truba_2[1] << endl;
                } else {

                        write(truba_2[1], &flag1, sizeof(int));

                        //cout << truba_2[1] << endl;
                }

        }

        return 0;
}
```

Демонстрация работы программы



Выводы

В заключении хотелось бы сказать, что благодаря данной лабораторной работы я познакомилась с некоторыми системными вызовами в Unix, разобралась с понятием процесс. Я научилась создавать pipe и работать с файловым дескриптором. В целом, теперь у меня есть понимание как управлять процессами и информацией, передающейся между ними.