

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: Каширин Кирилл Дмитриевич
Группа: М8О-208Б-20
Вариант: 8
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/lumses/OS>

Постановка задачи

Программа должна осуществлять работу с процессами и взаимодействие между ними в операционной системе Linux. В результате работы программа (основной процесс) производит деление первого числа команда, на последующие числа в команде. Взаимодействие между процессами осуществляется через каналы (pipe). Какая архитектура у приложения?

Общие сведения о программе

Parent.cpp – работа родительского процесса

Child.cpp – работа дочернего процесса

В своей программе я использую библиотеку fstream для работы с файлом, библиотекуunistd.h для работы с процессами и системными вызовами.

Основные команды в моей программе:

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

pipe(fd) - создание неименованного канала для передачи данных между процессами, т.е. конвейер, с помощью которого выход одной команды подается на вход другой

int id = fork () - создание дочернего процесса, в переменной id будет лежать “специальный код” процесса (-1 - ошибка fork, 0 - дочерний процесс, >0 - родительский). Чтобы посмотреть, какую ошибку выводит fork, можем воспользоваться errno.

int exece(const char *filename, char *const argv[], char *const envp[]) (и другие вариации exec) - замена образа памяти процесса

int dup2(int oldfd, int newfd) - переназначение файлового дескриптора

int putchar(char c) - функция записывает символ в текущей позиции в стандартный поток вывода (stdout) и перемещает внутренний указатель положения файла в следующую позицию.

close(...) – закрытие файла, файловых дескрипторов

Сборка проекта происходит при помощи make-файла

```
g++ child.cpp -o child
```

```
g++ parent.cpp
```

Общий метод и алгоритм решения

Имеется два процесса: родительский и дочерний. Родительский процесс считывает названия файла из стандартного ввода, из которого дочерний процесс осуществляет чтение. Так же родительский процесс выводит на экран все, что ему придет через pipe (взаимодействие между процессами осуществляется с помощью него). Производится перенаправление потока ввода на файл, потока вывода на pipe, происходит деление чисел, введенных в файл и выводится в стандартный поток вывода. Работа между процессами обеспечивается с помощью таких команд, как dup2, close.

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 with tests:";
    cin >> s;
    const char * buf = s.c_str();
    int fd[2];
    pipe(fd);
    if(access(buf, R_OK) == -1){
        cout << "File not exist" << endl;
        exit(EXIT_FAILURE);
    }

    pid_t pid = fork();
    if(pid == -1){
        perror("fork");
        return -1;
    } else if(pid == 0){
        int file = open(buf, O_RDONLY);
        if (file == -1) {
            perror("open");
            exit(EXIT_FAILURE);
        }
        if (dup2(fd[1], 1) == -1) {
            exit(1);
        }
        if (dup2(file, 0) == -1) {
            exit(1);
        }
        close(fd[0]);
        close(fd[1]);
        close(file);
        execl("child", "child", NULL);
    }
```

Child.cpp

```
#include <iostream>
#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) {
        if (k == 0){
            res = a;
            k=1;
        } else {
            if (a!=0){
                res/=a;
            } else {
                cout << "You can't divide it by zero" << endl;
                break;
            }
        }
        if(c == '\n') {
            cout << res << endl;
            k=0;
            fflush(stdout);
            res = 0.;
            continue;
        }
    }
    // cout << res << endl;
    close(STDOUT_FILENO);
    return 0;
}
```

Демонстрация работы программы

```
kirill@LAPTOP-F153AKTP:~$ cd os_prog
kirill@LAPTOP-F153AKTP:~/os_prog$ make
g++ child.cpp -o child
g++ parent.cpp -o main
./main
Enter a filename with tests:test.txt
10
25
kirill@LAPTOP-F153AKTP:~/os_prog$ make
g++ child.cpp -o child
g++ parent.cpp -o main
./main
Enter a filename with tests:test1.txt
You can't divide it by zero
kirill@LAPTOP-F153AKTP:~/os_prog$ make
g++ child.cpp -o child
g++ parent.cpp -o main
./main
Enter a filename with tests:wrewfw
File not exist
open: No such file or directory
kirill@LAPTOP-F153AKTP:~/os_prog$ □
```

Выводы

Эта лабораторная работа познакомила и научила меня работать с межпроцессорным взаимодействием между, с переопределениями потоков ввода/вывода. Также я научился работать с системными вызовами и файловыми дескрипторами.