Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

**ВЗАИМОДЕЙСТВИЕ МЕЖДУ ПРОЦЕССАМИ**

Студент: Мамонтов Егор Олегович

Группа: М8О–212Б–22

Вариант: 9

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

* Управление процессами в ОС
* Обеспечение обмена данными между процессами посредством труб

## 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. Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода.

Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами.

**9 вариант)** В файле записаны команды вида:«число число число<endline>». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат выводит в стандартный поток вывода. Если происходит деление на 0, то тогда дочерний родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип float. Количество чисел может быть произвольным.

**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: stdio.h, stdlib.h, unistd.h, sys/wait.h, sys/fcntl.h. Программы собраны с помощью Cmake. В программе используются следующие системные вызовы:

1. read — считывает с консоли
2. write — записывает в консоль
3. exit — завершает программу
4. close — закрывает файл
5. dup2 — переназначение файлового дескриптора.
6. execvp — запускает исполняемый файл изнутри другой программы.
7. open — открывает файл.
8. pipe — создаёт одностороннюю трубу для передачи данных между процессами.
9. fork — создаёт дочерний процесс, который повторяет программу родительского процесса.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы функции **fork()**, **pipe()**.
2. Создать **child.c** и **main.c**.
3. Реализовать в **main**:

3.1. Создание двух процессов и общение между программами **child** и **main** с помощью двух односторонних труб.

3.2. Открытие файла, считывание чисел из файла и отправка чисел в **child**.

3.3. Прием ответа из **child** и вывод ответа.

1. Реализовать в **child**:

4.1. Считывание строки чисел с помощью системного вызова **read**.

4.2. Функцию, которая будет решать задачу и проверять деление на ноль.

4.3. Вывод значения с помощью системного вызова **write**.

**Основные файлы программы**

**main.c:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/wait.h>

#include <sys/fcntl.h>

int create\_process() {

pid\_t pid = fork();

if (-1 == pid) {

write(STDERR\_FILENO, "fork", 5);

exit(-1);

}

return pid;

}

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

int fd[2]; // from parent to child

if (pipe(fd) == -1) {

write(STDERR\_FILENO, "Pipe wasn\'t created", 20);

}

int fd2[2]; // from child to parent

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

write(STDERR\_FILENO, "Pipe wasn\'t created", 20);

}

pid\_t pid = create\_process();

// Child

if (0 == pid) {

close(fd[1]);

dup2(fd[0], STDIN\_FILENO);

close(fd2[0]);

dup2(fd2[1], STDOUT\_FILENO);

execvp("./child", argv);

}

// Parent

else {

// Enter file path

char file\_name[128];

read(STDIN\_FILENO,file\_name, 128);

int pos = strlen(file\_name) - 1;

if (file\_name[pos] == '\n') {

file\_name[pos] = '\0';

}

// Open the file

int file = open(file\_name, O\_RDONLY);

char numbers[128];

int flag\_answer = 1;

if (file != -1) {

read(file, numbers, 128);

write(STDERR\_FILENO, "The file successfully opened\n", 30);

}

else {

write(STDERR\_FILENO, "The file did\'t open\n", 21);

flag\_answer = 0;

}

// Send 'numbers' to the child

close(fd[0]);

write(fd[1], numbers, 128);

close(fd[1]);

waitpid(pid, NULL, WUNTRACED);

// Read 'answer' from the child

close(fd2[1]);

char answer[128] = " ";

read(fd2[0], answer, 128);

if (answer[0] == ' ') {

exit(1);

}

if (flag\_answer == 1) {

write(STDOUT\_FILENO, answer, 128);

write(STDOUT\_FILENO, "\n", 2);

}

else {

exit(1);

}

close(fd2[0]);

}

return 0;

}

**child.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

// Function for solving a task

float foo (char \* num) {

// First number

long num1\_l = strtol(num, NULL, 10);

float answer = (float)num1\_l;

// Another numbers

float num2 = 0;

int flag = 0;

for (int i = 0; i < strlen(num)+1; ++i) {

if ((flag == 0) && ((num[i] == ' ') | (num[i] == '\0'))) {

flag = 1;

}

else if ((flag == 1) && (num[i] != ' ') && (num[i] != '\0')) { // 100 26

num2 = num2\*10 + ((int)num[i] - 48);

}

else if ((flag == 1) && ((num[i] == ' ') | (num[i] == '\0'))) {

if (num2 == 0) {

exit(1);

}

answer = answer / num2;

num2 = 0;

}

}

return answer;

}

int main () {

char child\_numbers[128];

float child\_answer;

read(STDIN\_FILENO, child\_numbers, 128);

int pos = strlen(child\_numbers)- 1;

if ( child\_numbers[pos] == '\n') {

child\_numbers[pos]='\0';

}

child\_answer = foo(child\_numbers); // Solving a task

char answer\_c[128];

gcvt(child\_answer, 6, answer\_c);

write(STDOUT\_FILENO, answer\_c, 128);

return 0;

}

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.10)

project(OS/lab\_01)

set(CMAKE\_C\_STANDARD 11)

add\_executable(child child.c)

set(REQUIRED\_FILES child)

add\_executable(main main.c)

**Пример работы**

**1.txt** (Одно число)

1

***Вывод программы:***

chelik@TeoBlow:~/Desktop/study/OS/os\_lab\_1/src/build$ /home/chelik/Desktop/study/OS/os\_lab\_1/src/build/main

/home/chelik/Desktop/study/OS/os\_lab\_1/test/1.txt

The file successfully opened

1

**2.txt** (Деление одного числа на ноль)

1 0

***Вывод программы:***

chelik@TeoBlow:~/Desktop/study/OS/os\_lab\_1/src/build$ /home/chelik/Desktop/study/OS/os\_lab\_1/src/build/main

/home/chelik/Desktop/study/OS/os\_lab\_1/test/2.txt

The file successfully opened

**3.txt** (Деление чисел с float ответом)

1 2 3 4 5

***Вывод программы:***

chelik@TeoBlow:~/Desktop/study/OS/os\_lab\_1/src/build$ /home/chelik/Desktop/study/OS/os\_lab\_1/src/build/main

/home/chelik/Desktop/study/OS/os\_lab\_1/test/3.txt

The file successfully opened

0.00833333

**4.txt** (Деление чисел с целым ответом)

100 2 5

***Вывод программы:***

chelik@TeoBlow:~/Desktop/study/OS/os\_lab\_1/src/build$ /home/chelik/Desktop/study/OS/os\_lab\_1/src/build/main

/home/chelik/Desktop/study/OS/os\_lab\_1/test/4.txt

The file successfully opened

10

**5.txt** (Деление на несколько чисел среди которых 0)

100 5 2 4 0

***Вывод программы:***

chelik@TeoBlow:~/Desktop/study/OS/os\_lab\_1/src/build$ /home/chelik/Desktop/study/OS/os\_lab\_1/src/build/main

/home/chelik/Desktop/study/OS/os\_lab\_1/test/5.txt

The file successfully opened

(Ошибка ввода названия файла)

***Вывод программы:***

chelik@TeoBlow:~/Desktop/study/OS/os\_lab\_1/src/build$ /home/chelik/Desktop/study/OS/os\_lab\_1/src/build/main

/home/chelik/Desktop/study/OS/os\_lab\_1/test/6.txt

The file did't open

**Вывод**

Я изучил основные системные вызовы для работы с трубами и процессами. Научился запускать несколько процессов и обмениваться между ними с помощью pipe(). Также, для решения проблемы с отправлением информации в child программу, пришлось научиться пользоваться dup2.