Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Управление процессами в ОС. Обеспечение обмена данных между процессами посредством каналов.**

Студент: Федоров А.Д

Группа: М80-206Б-20

Вариант: 8

Преподаватель: Соколов Андрей Алексеевич

Дата: 18.12.2021

Оценка: 4

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**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.

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.

8 вариант. В файле записаны команды вида:«число число число». 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 и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int. Количество чисел может быть произвольным.

**Листинг программы**

**child.c**

#include <stdio.h>

#include <stdbool.h>

#include "unistd.h"

typedef enum{

reading\_suc,

reading\_eol,

reading\_wrong\_value,

reading\_eof,

} read\_rvl\_stat;

read\_rvl\_stat reading\_int(int fd, int\* cur){

char c;

bool negative = false;

\*cur = 0;

int k = read(fd, &c, sizeof(char));

if(c == '-'){

negative = true;

k = read(fd, &c, sizeof(char));

}

while(k > 0){

if(c == '\n') {

if(negative) \*cur = -(\*cur);

return reading\_eol;

}

if(c == ' ') break;

if(((c < '0') || (c > '9'))){

return reading\_wrong\_value;

}

\*cur = \*cur \* 10 + c - '0';

k = read(fd, &c, sizeof(char));

}

if(negative) \*cur = -(\*cur);

if(k == 0) return reading\_eof;

return reading\_suc;

}

int main(){

int cur = 0;

float first = 0.0f;

bool fst = true;

read\_rvl\_stat status = reading\_int(0, &cur);

char c = '\n';

while(status != reading\_wrong\_value){

if(fst){

first = cur;

fst = false;

} else{

if (cur == 0) return -3;

else {

first = first / cur;

}

}

if(status == reading\_eof){

fprintf(stderr, "Test should end with <endline>\n");

return -1;

} else if(status == reading\_eol){

break;

}

cur = 0;

status = reading\_int(0, &cur);

}

write(1, &first, sizeof(float));

if (status == reading\_wrong\_value){

fprintf(stderr, "Wrong value in test\_file \n");

return -2;

}

return 0;

}

**LR2.c**

#include <stdio.h>

#include <stdbool.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/wait.h>

int main() {

int count;

char num;

while (read(0, &num, sizeof(char)) > 0) {

if (num == ' ') break;

count = count \* 10 + (num - '0');

}

char name[count + 1];

read (0, &name, count \* sizeof(char));

name[count] = '\0';

int fd[2];

if (pipe(fd) < 0) {

return -1;

}

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

if (file < 0){

return -3;

}

int processID = fork();

if (processID == 0){

dup2(file, 0);

dup2(fd[1], 1);

execl("child", "", NULL);

} else {

int status;

wait(&status);

if (WEXITSTATUS(status)){

return 2;

}

float cur;

read(fd[0], &cur, sizeof(float)) > 0;

printf("%f\n", cur);

}

return 0;

}

**Примеры работы**

mediocrity@LAPTOP-EEF7UMGS:~$ ./LR2

4 test

12.0

**Вывод**

Данная работа была для меня сложной из-за того, что я в первый раз работал с процессами. Но в то же время было крайне интересно передавать строку между тремя процессами, контролируя последовательность выполнения программы. Эта работа научила меня правильно создавать pipes на Си, создавать новые процессы, переназначать ввод и вывод программы в pipe с помощью dup2. Думаю, что эти знания мне пригодятся в будущем.