**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные Системы»

**Лабораторная работа № 2**

Тема: Управление процессами в ОС и обеспечение обмена данных между процессами посредством каналов

Студент: Семин Александр Витальевич

Группа: М8О-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Вариант 9.

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.

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 и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип float.

1. **Набор тестов**

*Первый тестовый набор:*

file.txt:

100.12 2.3456 1.1111

2.3456 2.3233 5.34455

Результат:

Child's process was created. It's id is 161798

line 1: res1 = 42.684170, res2 = 90.108887

line 2: res1 = 1.009598, res2 = 0.438877

*Второй тестовый набор:*

file.txt

100.12 2.3456 0

2.3456 2.3233 5.34455

Результат:

Child's process was created. It's id is 161863

Error: division on 0 is forbidden!

*Третий тестовый набор:*

file.txt:

100.12 2.3456 123.45 2.3456 2.3233 5.34455

Результат:

Child's process was created. It's id is 161863

Error: division on 0 is forbidden!

1. **Листинг программы**

**parent.c**

#include <unistd.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <stdlib.h>

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

int lines;

printf("Enter number of lines in your text file:\n");

scanf("%d", &lines);

int file = open(argv[1], 0);

if (file == -1) {

printf("Can't open file %s\n", argv[1]);

return 2;

}

int fd[2];

pipe(fd);

pid\_t pid = fork();

if (pid == -1) {

perror("Fork error");

return -1;

}

if (pid != 0) printf("Child's process was created. It's id is %d\n", pid);

int child\_res = 0;

wait(&child\_res);

/\* if (child\_res == 0) {

printf("Incorrect execution of a child process\n");

printf("%d\n", child\_res);

return -1;

}\*/

if (pid == 0) { // child process

// printf("It's part of child's process\n");

close(fd[0]);

dup2(file, STDIN\_FILENO);

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

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

// printf("%d\n", STDIN\_FILENO);

} else { // parent process

int line\_in\_file = 0;

float res1 = 0, res2 = 0;

// printf("I'm parent\n");

close(fd[1]);

while (lines > 0) {

read(fd[0], &line\_in\_file, sizeof(int));

read(fd[0], &res1, sizeof(float));

read(fd[0], &res2, sizeof(float));

printf("line %d: res1 = %f, res2 = %f\n", line\_in\_file, res1, res2);

lines--;

}

}

close(file);

return 0;

}

**child.c**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdbool.h>

#include <stdlib.h>

typedef enum{

read\_suc,

read\_eol,

read\_wrong\_value,

read\_eof,

} read\_num\_stat;

read\_num\_stat read\_float(int fd, float\* cur){

bool dot\_fnd = false;

char c;

\*cur = 0;

double i = 0.1;

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

while(res > 0){

if(c == '\n') return read\_eol;

if(c == ' ')

break;

if(((c < '0') || (c > '9')) && c != '.'){

return read\_wrong\_value;

}

if (!dot\_fnd) {

if(c == '.')

dot\_fnd = true;

else {

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

}

} else {

if(c == '.')

return read\_wrong\_value;

\*cur = \*cur + i \* (c - '0');

i /= 10;

}

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

}

if(res == 0)

return read\_eof;

return read\_suc;

}

int main() {

float cur = 0, sec = 0.0, third = 0.0;

int line\_in\_file = 0;

read\_num\_stat status = read\_float(STDIN\_FILENO, &cur);

while (status == read\_eol || status == read\_suc) {

line\_in\_file++;

status = read\_float(STDIN\_FILENO, &sec);

// fprintf(stderr,"ya v while %f\n", sec);

if (status == read\_wrong\_value)

return -1;

if (status == read\_eof) {

fprintf(stderr, "Wrong commands! Line should looks like <number number number<endline>>\n");

return -2;

}

if (status == read\_eol){

fprintf(stderr, "Incorrect type of commands in file\n");

return -3;

}

status = read\_float(STDIN\_FILENO, &third);

if (status == read\_wrong\_value)

return -1;

if (sec == 0 || third == 0) {

fprintf(stderr, "Error: division on 0 is forbidden!\n");

return -4;

}

if (status == read\_suc) {

fprintf(stderr, "Wrong commadns! Line should looks like <number number number<endline>>\n");

return -5;

}

float res1 = cur / sec;

float res2 = cur / third;

write(STDOUT\_FILENO, &line\_in\_file, sizeof(int));

write(STDOUT\_FILENO, &res1, sizeof(float));

write(STDOUT\_FILENO, &res2, sizeof(float));

status = read\_float(STDIN\_FILENO, &cur);

}

if (status == read\_wrong\_value || status == read\_eol) {

return -1;

}

return 0;

}

1. **Выводы**

Данная лабораторная работа знакомит с простейшим межпроцессным взаимодействием с помощью специальных каналов связи – пайпов. Выполняя эту работу, я приобрел знания по работе с низкоуровневым вводом и выводом на языке Си, научился создавать дочерний процесс и осуществлять его взаимодействие с родительским. Уверен, эти знания несомненно пригодятся мне в будущем.