**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные Системы»

**Лабораторная работа № 4**

Тема: Освоение принципов работы с файловыми системами и обеспечение обмена данных между процессами посредством технологии «File mapping»

Студент: Семин Александр Витальевич

Группа: М8О-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата:

Оценка:

Москва, 2021

1. **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 через отображаемые файлы (memory-mapped files).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 9

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». 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:*

…:~/Desktop/OS\_labs/l4$ ./a.out file2.txt

Enter number of lines in your text file:

3

Child's process was created. Id is 5247

In child

In parent

line 1: res1 = 7.7311, res2 = 3.3836

line 2: res1 = 3.9178, res2 = 5.5895

line 3: res1 = 8.9991, res2 = 2.9834

*file2.txt:*

3

110.123 14.244 32.546

321.348 82.021 57.491

31.452 3.495 10.542

*Второй тестовый набор: (программа завершается при ошибочном задании данных в исходном файле в процессе выполнения)*

…:~/Desktop/OS\_labs/l4$ ./a.out file2.txt

Enter number of lines in your text file:

2

Child's process was created. Id is 5057

In child

*file2.txt:*

2

110.123 14.244 32.546

321.348 82.021

1. **Листинг программы**

**child.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>

#include <sys/mman.h>

#include <stdbool.h>

#include "iout.h"

void print\_num\_in\_data(char\* data, int a, int DATA\_COUNT) {

if (a > 0 && a < 10) {

data[DATA\_COUNT] = '0' + a;

}

}

void write\_in\_data(char\* data,int\* DATA\_COUNT, float val) {

int counter = \*DATA\_COUNT;

if (val < 0) {

data[counter] = '-';

counter++;

val \*= -1;

}

float left;

float right = modff(val, &left);

//вывод целой части

int l = left;

int arr[100] = {'\0'};

int arr\_size = reverse(l, arr);

if (l == 0) {

data[counter] = '0';

counter++;

} else {

for (int i = arr\_size -1; i >= 0; i--) {

print\_num\_in\_data(data, arr[i], counter);

counter++;

}

}

data[counter] = '.';

counter++;

//вывод дробной часи

int i = 0;

while (right > 0) { //0,456

float new = right\*10; //4,56

right = right\*10;

while (right > 1) right--; // по итогу right = 0,56

float for\_print = new - right; // 4,56 - 0,56

print\_num\_in\_data(data,for\_print,counter); // вывод цифры 4

counter++;

//повтор цикла для 0,56 и тд

i++;

if (i == 4) {

break;

}

}

\*DATA\_COUNT = counter;

}

int main() {

int N;

my\_read\_int(STDIN\_FILENO, &N);

int DATA\_COUNT = 0;

// my\_read\_int(STDIN\_FILENO, &N);

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

int status\_trunc = ftruncate(STDOUT\_FILENO, sizeof(char)\*100\*N);

if (status\_trunc < 0) {

// my\_print("FTRUNCATE ERROR\n");

return -1;

}

char\* data = mmap(0, sizeof(char)\*100\*N, PROT\_WRITE, MAP\_SHARED, STDOUT\_FILENO, 0);

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

while (status == 0 && N > 0) {

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

if (status == -1) //неверное значение

return -1;

if (status == -3) { //перенос строки после второго числа

// my\_print("Incorrect type of commands in file\n");

return -3;

}

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

if (status == -1) // неверное значение

return -1;

if (sec == 0 || third == 0) { // проверка деления на 0

// my\_print("Error: division by 0 is forbidden!\n");

return -4;

}

if (status == 0) { //отсутствует перенос строки после 3-го числа

// my\_print("Wrong commadns! Line should looks like <number number number<endline>>\n");

return -5;

}

float res1 = cur / sec;

float res2 = cur / third;

write\_in\_data(data, &DATA\_COUNT, res1);

data[DATA\_COUNT] = ' ';

DATA\_COUNT++;

write\_in\_data(data, &DATA\_COUNT, res2);

data[DATA\_COUNT] = '\n';

DATA\_COUNT++;

N--;

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

}

msync(data, sizeof(char)\*100\*N, MS\_SYNC);

int err = munmap(data, sizeof(char)\*100\*N);

/\* if (err < 0) {

my\_print("MUNMAP ERROR\n");

return -1;

}\*/

return 0;

**parent.c**

#include <unistd.h>

#include <stdio.h>

#include <stdbool.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <stdlib.h>

#include "iout.h"

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

int lines;

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

int stat = my\_read\_int(0, &lines);

if (stat < 0) {

return -1;

}

if (argc != 2) {

my\_print("USAGE: ./a.out <filename>\n");

return -1;

}

int N = lines;

//открытие входного файла

int input = open(argv[1], O\_RDWR);

if (input == -1) {

my\_print("Can't open file\n");

return -2;

}

//временный файл

char tmp\_name[] = "tmpXXXXXX";

if (mkstemp(tmp\_name) < 0) {

my\_print("Can't create temporary file\n");

return -3;

}

int fd\_tmp = open(tmp\_name, O\_RDWR);

if (fd\_tmp < 0) {

my\_print("Can't open temporary file\n");

return -2;

}

//разделение процессов

pid\_t pid = fork();

if (pid == -1) {

perror("Fork error");

return -1;

}

if (pid != 0) {

my\_print("Child's process was created. Id is ");

print\_int(pid);

my\_print("\n");

}

//ожидание выполнения дочернего процесса

int status = 0;

wait(&status);

if (WEXITSTATUS(status)) {

return -5;

}

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

my\_print("In child\n");

dup2(input, STDIN\_FILENO);

dup2(fd\_tmp, STDOUT\_FILENO);

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

} else { // parent process

int line\_in\_file = 1;

my\_print("In parent\n");

char\* data = mmap(0, sizeof(char)\*100\*N,PROT\_READ, MAP\_SHARED, fd\_tmp, 0);

int DATA\_COUNT = 0;

while (lines > 0) {

my\_print("line "); print\_int(line\_in\_file);

//write int numb of line

my\_print(": res1 = ");

if (data[DATA\_COUNT] == '\n') DATA\_COUNT++;

for (DATA\_COUNT; data[DATA\_COUNT] != ' '; DATA\_COUNT++) {

write(1, &data[DATA\_COUNT], sizeof(char));

}

//write float res1

my\_print(", res2 = ");

for (DATA\_COUNT; data[DATA\_COUNT] != '\n' && data[DATA\_COUNT] != '\0'; DATA\_COUNT++) {

write(1, &data[DATA\_COUNT], sizeof(char));

}

//write float res2

my\_print("\n");

line\_in\_file++;

lines--;

}

int err = munmap(data, sizeof(char)\*100\*N);

if (err < 0) {

my\_print("MUNMAP ERROR\n");

}

}

close(input);

unlink(tmp\_name);

return 0;

}

**iout.h**

#ifndef IOUT\_H

#define IOUT\_H

#include <math.h>

#include <string.h>

#include <unistd.h>

#include <stdio.h>

#include <stdbool.h>

void my\_print(char str[1024]) {

write(1, str, strlen(str));

}

int my\_read\_int(int fd, int\* a) {

int res = 0;

char c;

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

while(c != '\n' && c != EOF && c != ' ') {

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

// my\_print("Incorrect input\n");

return -1;

}

res = res\*10 + (c -'0');

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

}

\*a = res;

return 0;

}

int read\_2\_floats(char\* data,int\* i, int data\_size, float\* fst, float\* sec) {

bool read\_fst = true, read\_sec = false;

int count = \*i;

while(read\_fst || read\_sec) {

bool dot\_fnd = false;

bool mines = false;

char c;

float cur = 0;

double i = 0.1;

c = data[count];

while(data[count] != ' ' && data[count] != '\n' && data[count] != EOF) {

if (c == '-') {

mines = true;

count++;

c = data[count];

continue;

}

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

return -1;

}

if (!dot\_fnd) {

if(c == '.')

dot\_fnd = true;

else {

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

}

} else {

if(c == '.')

return -1;

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

i /= 10;

}

if (count == data\_size - 1) {

printf("konec data\n");

}

count++;

c = data[count];

}

count++;

if (mines) {

cur \*= -1;

mines = false;

}

if (read\_fst) {

\*fst = cur;

read\_fst = false;

read\_sec = true;

continue;

}

if (read\_sec) {

\*sec = cur;

read\_sec = false;

continue;

}

}

\*i = count;

return 0;

}

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

bool dot\_fnd = false;

bool mines = false;

char c;

\*cur = 0;

double i = 0.1;

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

while(res > 0){

if (c == '-') {

mines = true;

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

continue;

}

if(c == '\n') return -3;

if(c == ' ')

break;

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

return -1;

}

if (!dot\_fnd) {

if(c == '.')

dot\_fnd = true;

else {

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

}

} else {

if(c == '.')

return -1;

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

i /= 10;

}

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

}

if (mines) {

\*cur \*= -1;

}

if(res == 0)

return 0;

return 0;

}

int reverse(int x, int\* l) {

int i = 0;

while (x > 0) {

int a = x % 10;

l[i] = a;

i++;

x /= 10;

}

return i;

}

void print\_num(int a) {

char\* num;

if (a == 0) num = "0";

if (a == 1) num = "1";

if (a == 2) num = "2";

if (a == 3) num = "3";

if (a == 4) num = "4";

if (a == 5) num = "5";

if (a == 6) num = "6";

if (a == 7) num = "7";

if (a == 8) num = "8";

if (a == 9) num = "9";

write(1, num, sizeof(char));

}

void print\_int(int x) {

int data\_int[100];

int data\_sz = reverse(x, data\_int);

if (x == 0) {

print\_num(x);

}

for (int i = 0; i < data\_sz; i++) {

print\_num(data\_int[i]);

}

}

void print\_float(float x) {

if (x < 0) {

write(1, "-", sizeof(char));

x \*= -1;

}

float left;

float right = modff(x, &left);

//вывод целой части

int q = left;

print\_int(q);

write(1, ".", sizeof(char));

//вывод дробной часи

int i = 0;

while (right > 0) { //0,456

float new = right\*10; //4,56

right = right\*10;

while (right > 1) right--; // по итогу right = 0,56

float for\_print = new - right; // 4,56 - 0,56

print\_num(for\_print); // вывод цифры 4

//повтор цикла для 0,56 и тд

i++;

if (i == 5) {

break;

}

}

}

#endif

1. **Выводы**

Данная лабораторная работа знакомит с механизмом межпроцессорного взаимодействия при помощи отображаемых файлов (File Mapping). Данный механизм позволяет отобразить необходимое количество информации из файла в оперативную память, чтобы несколько процессов могли иметь доступ к ней.

По сравнению с пайпами, данный метод более быстрых за счет отсутствия запросов на чтение и запись, а также благодаря тому, что доступ к отображенным данным происходит за О(1). Как недостаток можно отметить, что дочерние процессы должны знать имя отображаемого файла (или дескриптор), что вызывает некоторые неудобства, а также этот метод требует больше памяти, чем пайпы.