Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Операционные системы»

Лабораторная работа № 4

Студент: Марочкин И.А.
Группа: М8О-206Б-19
Преподаватель: Соколов А.А.
Дата: 24.04.2021
Оценка:

Москва, 2020

Цель работы

Приобретение практических навыков в:

- Освоение принципов работы с файловыми системами
- Обеспечение обмена данных между процессами посредством технологии «File mapping»

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 через отображаемые файлы (memory-mapped files). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант:

В файле записаны команды вида: “число число число<endline>”. Дочерний процесс производит деление первого числа в команде на последующие числа в команде, а результат выводит в стандартный поток вывода. Если происходит деление на 0, то дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. числа имеют тип float.

Общие сведения о программе

Программа компилируется из файла laba.c. Для реализации поставленной задачи в программе используются следующие системные вызовы:

mmap - создает отображение файла в память.

fork - создает копию текущего процесса, который является дочерним процессом для текущего процесса.

open - открывает файл. Аргументом можно задать, открыть файл на запись или чтение.

close - закрывает файл.

mkstemp - создает временный файл с уникальным именем.

unlink - удаляет файл, определенный по pathname.

Листинг программы

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <math.h>
#include <stdint.h>
#include <sys/mman.h>

typedef struct {
    uint8_t check;
    double result;
} Data;

// создание временного файла:
int get_tmpFile(void){
    char* tmpFileName = strdup("/tmp/OS_laba4_tmpFile.XXXXXX");
    if (tmpFileName == NULL){
        perror("strdup failed");
        exit(-7);
    }
    int fd = mkstemp(tmpFileName);
    if (fd == -1){
        perror("mfstemp failed");
        exit(-8);
    }
    if (unlink(tmpFileName) == -1){
        perror("unlink failed");
        exit(-9);
    }
    free(tmpFileName);
    write(fd, "\0\0\0\0\0\0\0\0\0\0", 10);
    return fd;
}

// запись:
void write_in_mmapFile(Data* data, void* mmapFile){
    uint8_t* check = (uint8_t*)mmapFile;
    double* result = (double*)(mmapFile + sizeof(uint8_t));

    *check = data->check;
    *result = data->result;
}
// чтение:
void read_from_mmapFile(Data* data, void* mmapFile){
    uint8_t* check = (uint8_t*)mmapFile;
    double* result = (double*)(mmapFile + sizeof(uint8_t));

    data->check = *check;
    data->result = *result;
}

int main(){

    printf("Enter file name: ");

    // считываем имя файла:
    char file_name[256];
    gets(file_name);

    // открываем файл с введенным именем:
```

```
    int commands = open(file_name, O_RDONLY);
    if (commands == -1){
        perror("Can't open file");
        exit(-2);
    }
    // создаем и мапим временный файл:
    int fd_tmpFile = get_tmpFile();
    void* mmapFile = mmap(/*start*/ NULL,/*length*/ 10,/*prot*/ PROT_READ |
PROT_WRITE,/*flags*/ MAP_SHARED,/*fd*/ fd_tmpFile,/*offset*/ 0);
    if (mmapFile == MAP_FAILED){
        perror("mmap failed");
        exit(-6);
    }
    // создаем дочерний процесс:
    int id = fork();

    if (id == -1){
        perror("fork error");
        exit(-3);
    }
    else if (id == 0){

        double num[50];
        Data data;
        data.check = 1;

        dup2(commands, 0); // перенаправляем стандартный поток ввода дочернего процесса на
открытый файл
        // считываем 3 числа типа double из файла:
        int i = 0;
        while(scanf("%lf", &num[i]) != EOF){
            ++i;
        }
        printf("Count of numbers = %d\n", i);
        // делаем проверку деления на 0:
        for (int j = 1; j < i; ++j){
            if (num[j] == 0){
                perror("Can't divide");
                data.check = 0;
                write_in_mmapFile(&data, mmapFile); // если возможно деление на 0,
отправляем данные родительскому процессу и завершаем работу
                exit(-5);
            }
        }

        // считаем и отправляем данные родительскому процессу:
        data.result = num[0];
        for (int j = 1; j < i; ++j){
            data.result /= num[j];
        }
        write_in_mmapFile(&data, mmapFile);
    }
    else {
        Data data;
        wait(&id); // ждем завершения работы дочернего процесса
        // считываем данные от дочернего процесса:
        read_from_mmapFile(&data, mmapFile);
        if (data.check == 0)
            exit(-5); // завершаем работу, если в дочернем процессе было деление на 0

        printf("Result from child = %lf\n", data.result);
    }

    close(fd_tmpFile);
    close(commands);
    return 0;
}
```

Пример работы

test1.text: 933.546 2.45 43.995
test2.text: 10000000000 1034341 13456 3 121 674 235
test3.text: 256 2 2 2 2
test4.text: 2 0
test5.text:

```
Vanya:Src ivan$ ./main
warning: this program uses gets(), which is unsafe.
Enter file name: ./Tests/test1.txt
Count of numbers = 3
Result from child = 8.660966
Vanya:Src ivan$ ./main
warning: this program uses gets(), which is unsafe.
Enter file name: ./Tests/test2.txt
Count of numbers = 7
Result from child = 0.000000
Vanya:Src ivan$ ./main
warning: this program uses gets(), which is unsafe.
Enter file name: ./Tests/test3.txt
Count of numbers = 5
Result from child = 16.000000
Vanya:Src ivan$ ./main
warning: this program uses gets(), which is unsafe.
Enter file name: ./Tests/test4.txt
Count of numbers = 2
Can't divide: Invalid argument
Vanya:Src ivan$ ./main
warning: this program uses gets(), which is unsafe.
Enter file name: ./Tests/test5.txt
Count of numbers = 0
Result from child = 0.000000
Vanya:Src ivan$
```

Вывод

В СИ помимо механизма общения между процессами через pipe, также существуют и другие способы взаимодействия, например, отображение файла в память. Такой подход работает быстрее, за счет отсутствия постоянных вызовов read, write и тратит меньше памяти под кэш. После отображения возвращается void*, который можно привести к своему указателю на тип и обрабатывать данные как массив, где возвращенный указатель – указатель на первый элемент.