Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу

«Операционные системы»

ВЗАИМОДЕЙСТВИЕ МЕЖДУ ПРОЦЕССАМИ ПРИ ПОМОЩИ MEMORY-MAPPED FILES

Студент: Забелкин Андрей Алексеевич
Группа: М8О–210Б–22
Вариант: 6
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023.

Постановка задачи

Цель работы

Целью является приобретение практических навыков в:

- Обеспечении межпроцессорного взаимодействия посредствам технологии file mapped
- Освоение принципов работы с файловыми системами

Задание

Составить программу на языке Си, осуществляющую работу с процессами и их взаимодействие в ОС на базе UNIX.

Родительский процесс должен открыть файл из которого дочерний процесс читает все числа типа int и передает родительскому процессу их сумму.

Общие сведения о программе

Программа компилируется из с помощью Makefile, сгенерированным cmake.

Также используется заголовочные файлы:lab3.c, test_lab3.c.

1. **void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);** - этот системный вызов используется для отображения файла в виртуальную память процесса.
2. **int munmap(void *addr, size_t length);** - этот системный вызов используется для отмены отображения ранее созданной области памяти, возвращая выделенные ресурсы ядру операционной системы.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы mmap.
2. Написать и отладить работу основной функции.
3. Придумать тесты и ответы к этим тестам.
4. Написать bash-скрипт, который запускает и проверяет программу на тестах.

Основные файлы программы

lab3.c

```c
#include <stdio.h>
#include <stdlib.h>
```

```c
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char *argv[]) {
   if (argc != 2) {
       fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
       return 1}
   int fd = open(argv[1], O_RDWR, S_IRUSR | S_IWUSR);
   if (fd == -1) {
       perror("open");
       return 1}
   // Получение размера файла
   struct stat file_stat;
   if (fstat(fd, &file_stat) == -1) {
       perror("fstat");
       close(fd);
       return 1}
   // Создание отображения файла в памяти для родительского процесса
   char* shared_memory = (mmap(NULL, file_stat.st_size, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0));
   if (shared_memory == MAP_FAILED) {
       perror("mmap error");
       close(fd);
       return 1}
   if (lseek(fd, 0, SEEK_SET) == -1) {
       perror("lseek");
       munmap(shared_memory, file_stat.st_size);
       close(fd);
       return 1}
   // Создание отображения для новой переменной типа int
   int* sum_shared_memory = (mmap(NULL, file_stat.st_size + 2*sizeof(int), PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0));
   if (sum_shared_memory == MAP_FAILED) {
       perror("mmap error");
       munmap(shared_memory, file_stat.st_size);
       close(fd);
       return 1}
   // Создание дочернего процесса
   pid_t child_pid = fork();
   if (child_pid == -1) {
       perror("fork");
       munmap(shared_memory, file_stat.st_size);
       munmap(sum_shared_memory, file_stat.st_size + 2*sizeof(int));
       close(fd);
       return 1;
   }
   if (child_pid == 0) {
       // Дочерний процесс
       int sum = 0;
       char current_number[20];
       int current_number_index = 0;
       int is_negative = 0;
```

```
        for (int i = 0; i < file_stat.st_size; ++i) {
            if (shared_memory[i] == '-' && current_number_index == 0) {
                is_negative = 1;
            } else if ((shared_memory[i] >= '0' && shared_memory[i] <= '9') ||
(shared_memory[i] == '.')) {
                current_number[current_number_index] = shared_memory[i];
                current_number_index++;
            } else if (shared_memory[i] == ' ' || shared_memory[i] == '\n' ||
shared_memory[i] == '\t') {
                if (current_number_index > 0) {
                    current_number[current_number_index] = '\0';
                    int current_value = my_atoi(current_number);
                    if (is_negative) {
                        current_value *= -1;
                        is_negative = 0;
                    }
                    sum += current_value;
                    current_number_index = 0;
                }}}
        sum_shared_memory[file_stat.st_size /sizeof(int) + 1] = sum;
        _exit(0);
    } else {
        int status;
        if (waitpid(child_pid, &status, 0) == -1) {
            perror("waitpid");
        }
        int result = sum_shared_memory[file_stat.st_size /sizeof(int) + 1];
        printf("%d", result);
        munmap(shared_memory, file_stat.st_size);
        munmap(sum_shared_memory, file_stat.st_size + 2*sizeof(int));
        close(fd);
    }
    return 0;
}
```

Пример работы

./lab3 ../1.txt

15

Вывод

Эта лабораторная фактически повторяет первую с одним отличием: вместо pipe используется mmap. Я постарался сделать её максимально просто без использования семафоров, у меня это получилось, но это выглядит небезопасно, т. к. там фактически могут потерятся какие то данные, т.к. запись получается в конце существующего файла.