Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работ №4 по курсу

«Операционные системы»

ОТОБРАЖЕНИЕ ФАЙЛОВ В ПАМЯТЬ

Студент: Косогоров Владислав Валерьевич
Группа: М8О–206Б–18
Вариант: 5
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019.

Содержание

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 решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется с помощью отображения файла на память.

Вариант 11: написать целочисленный калькулятор с операциями +, -. В дочернем процессе должны происходить вычисления выражений. В родительском процессе должны происходить ввод и вывод.

Общие сведения о программе

Программа компилируется из одного файла main.c. В данном файле используются заголовочные файлы stdio.h, stdlib.h, sys/mman.h, fcntl.h, unistd.h, ctype.h, sys/wait.h. В программе используются следующие системные вызовы:

shm_open – создание файла в разделяемой памяти.

ftruncate – задание размера созданного файла.

mmap – отображение файла в адресное пространство программы.

fork – создание дочернего процесса.

execvp – выполнение заданной команды интерпретатора команд.

munmap – удаление отображения файла.

Программа принимает на вход команду интерпретатора команд.

Общий метод и алгоритм решения.

Функцией shm_open() создается файл «memfile» в разделяемой памяти, после чего с помощью ftruncate() устанавливается его размер. После этого вызовом mmap() мы получаем адрес начала строки, в которой будет храниться результат работы execvp(), который выполняется в дочернем процессе. При этом нужно перенаправить стандартный поток вывода в созданный файл с помощью dup2(). После завершения работы дочернего процесса мы с помощью функций islower() и toupper() преобразуем строку в разделяемой памяти и выводим её. В конце с помощью системного вызова munmap() мы удаляем эту строку из адресного пространства программы.

Основные файлы программы.

Файл main.c

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


const int ARG_MAX = 2097152 + 1;


int main(int argc, char** argv) {
```

```
    pid_t pid;
    int rv, fd;

    char* filename = "memfile";

    if ((fd = shm_open(filename, O_RDWR | O_CREAT | O_TRUNC, S_IRUSR |
S_IWUSR)) == -1) {
        perror("shm_open error");
        exit(-1);
    }

    if (ftruncate(fd, ARG_MAX * 2) == -1) {
        perror("failed to truncate");
        exit(-1);
    }

    char* memory = mmap(NULL, ARG_MAX * 2, PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0);

    if (memory == MAP_FAILED) {
        perror("mapping error");
        fprintf(stderr, "%p", memory);
        exit(-1);
    }

    if ((pid = fork()) < 0) {
        perror("fork error");
        exit(-1);
```

```
} else if (pid == 0) {

    dup2(fd, STDOUT_FILENO);
    rv = execvp(argv[1], argv + 1);
    if (rv) {
        perror("exec error");
    }
    exit(rv);

} else if (pid > 0) {
    waitpid(pid, &rv, 0);

    for (int i = 0; memory[i] != '\0'; ++i) {
        if (islower(memory[i])) {
            putchar(toupper(memory[i]));
        } else {
            putchar(memory[i]);
        }
    }
    exit(WEXITSTATUS(rv));
}

if (munmap(memory, ARG_MAX * 2)) {
    perror("munmap error");
    exit(-1);
}

return 0;
```

```
}
```

Демонстрация работы программы.

```
…/prog_3_sem/os/lab4 ż ./a.out cat main.c | head -4
#INCLUDE <STDIO.H>
#INCLUDE <SYS/MMAN.H>
#INCLUDE <SYS/STAT.H>
#INCLUDE <FCNTL.H>

…/prog_3_sem/os/lab4 ż ./a.out man exec | tail -15 | head -5
    INTERNALLY  AND  WERE  CONSEQUENTLY  NOT ASYNC-SIGNAL-
SAFE, IN
    VIOLATION OF THE REQUIREMENTS OF POSIX.1.  THIS WAS  FIXED
IN
    GLIBC 2.24.

SEE ALSO

…/prog_3_sem/os/lab4 ż ./a.out echo test123
TEST123
```

Вывод

Отображение файлов дает удобство при работе с файлами, так как позволяет работать с областью файла как с обычным участком памяти. Другими словами, мы имеем доступ к каждому байту области памяти, которую мы отобразили и для этого не надо использовать lseek, также количество системных вызовов по чтению и записи сводится к нулю, так как мы работаем с оперативной памятью.