Московский Авиационный Интститут
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Операционные системы»

Файловые системы и “File mapping”

Студент: Артемьев Дмитрий Иванович
Группа: М8О-206Б-18
Вариант: 2
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Условие

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.

23. Родительский процесс считывает две координаты передает их через канал дочернему процессу. Дочерний процесс определяет к какой четверти относится точка, а далее передает результат родительскому процессу.

Описание программы

Код программы находится в файле main.c.

Ход выполнения программы

1. Создание временного файла
2. Создание общей области с помощью mmap
3. Создание двух семафоров
4. Создание дочернего процесса
5. Ожидание дочерним процессом первого семафора
6. Считывание из стандартного потока входных данных родительским процессом, запись в общую область памяти
7. Вызов первого семафора (дочернего процесса), ожидание второго семафора родительским процессом
8. Обработка данных из общей области памяти дочерним процессом, запись в неё же результата
9. Вызов второго семафора, завершение работы дочернего процесса
10. Считывание результата из общей области памяти родительским процессом, вывод результата на экран
11. Завершение работы родительского процесса

Недочёты

Выводы

Я научился взаимодействию процессов посредством маппинга и семафоров.

Исходный код

main.c

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <fcntl.h>

int create_tmp_file() {
    char filename[32] = {"./tmp_file-XXXXXX"};
    int fd = mkstemp(filename);
    unlink(filename);
    if (fd < 1) {
        perror("Creating tmp file failed\n");
        exit(1);
    }
    write(fd, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 20);
    return fd;
}

int main() {
    int fd = create_tmp_file();
    unsigned char* shmem = (unsigned char*)mmap(NULL, 100, PROT_WRITE | PROT_READ, MAP_S
    if (shmem == MAP_FAILED) {
        perror("Mapping failed\n");
        exit(1);
    }

    sem_t* sem1 = sem_open("/sem1", O_CREAT, 777, 0);
    sem_t* sem2 = sem_open("/sem2", O_CREAT, 777, 0);
    if (sem1 == SEM_FAILED || sem2 == SEM_FAILED) {
        perror("Semaphore opening falied\n");
        exit(1);
    }
    sem_unlink("/sem1");
    sem_unlink("/sem2");

    pid_t proc = fork();
    if (proc < 0) {
```

```
    printf("Error fork\n");
    exit(1);
}

if (proc == 0) {
    int x, y;
    int res;
    sem_wait(sem1);
    memcpy(&x, shmem, sizeof(x));
    memcpy(&y, shmem + sizeof(x), sizeof(y));

    if (x > 0 && y > 0)
        res = 1;
    if (x < 0 && y > 0)
        res = 2;
    if (x < 0 && y < 0)
        res = 3;
    if (x > 0 && y < 0)
        res = 4;

    memcpy(shmem + sizeof(x) + sizeof(y), &res, sizeof(res));
    sem_post(sem2);
    sem_close(sem1);
    sem_close(sem2);
    munmap(shmem, 20);
    close(fd);
    exit(EXIT_SUCCESS);
}
else if (proc > 0) {
    int x, y;
    int res;
    scanf("%d %d", &x, &y);

    memcpy(shmem, &x, sizeof(x));
    memcpy(shmem + sizeof(y), &y, sizeof(y));

    sem_post(sem1);
    sem_wait(sem2);

    memcpy(&res, shmem + sizeof(x) + sizeof(y), sizeof(res));
    printf("%d\n", res);
}
```

```
    sem_close(sem1);
    sem_close(sem2);
    munmap(shmem, 20);
    close(fd);
    exit(EXIT_SUCCESS);
}
```