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

Лабораторная работа №4 по курсу
«Операционные системы»

Тема работы

Студент: Попов Матвей Романович
Группа: М8О-208Б-20
Вариант: 1
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/.../os_lab4

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 через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

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

Вариант 1: Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. Количество чисел может быть произвольным.

Общие сведения о программе

Программа представляет из себя один файл main.cpp.

Общий метод и алгоритм решения

Создаём структуру bebra, в которой будем хранить полученное от пользователя число и операцию дочернего процесса. Создадим для структуры bebra отображённую память, доступную для обоих процессов, а для регулировки доступа процессов к памяти используем семафор.

Исходный код

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <semaphore.h>

using namespace std;
```

```
int human_get(sem_t *semaphore)
{
    int s;
    sem_getvalue(semaphore, &s);
    return s;
}

void human_set(sem_t *semaphore, int n)
{
    while (human_get(semaphore) < n)
    {
        sem_post(semaphore);
    }
    while (human_get(semaphore) > n)
    {
        sem_wait(semaphore);
    }
}

struct bebra
{
    int num;
    int st;
};

int main()
{
    int sum = 0;
    bebra* mapped = (bebra*)mmap(0, sizeof(bebra), PROT_READ|PROT_WRITE,
MAP_SHARED|MAP_ANONYMOUS, 0, 0);
    if (mapped == MAP_FAILED)
    {
        cout << "mmap error\n";
        return -1;
    }
    sem_unlink("_sem");
    sem_t *sem = sem_open("_sem", O_CREAT, 0, 2);
    string filename;
    int n;
    ofstream out;
    cout << "Enter name of the file:\n";
    getline(cin, filename);
    cout << "Enter some numbers:\n";
    int id = fork();
    if (id < 0)
    {
        cout << "fork error\n";
        return -1;
    }
```

```
    if (id == 0)
    {
        out.open(filename);
        while(1)
        {
            while(human_get(sem) == 2)
            {
                continue;
            }
            if (mapped->st == 1)
            {
                sum += mapped->num;
                out << sum << endl;
                sum = 0;
                human_set(sem, 2);
            }
            else if (mapped->st == 2)
            {
                sum += mapped->num;
                out << sum << endl;
                out.close();
                human_set(sem, 0);
                exit(0);
            }
            else if (mapped->st == 0)
            {
                sum += mapped->num;
                human_set(sem, 2);
            }
        }
    }
    else if (id > 0)
    {
        while(human_get(sem) != 0)
        {
            char c;
            scanf("%d%c", &n, &c);
            mapped->num = n;
            if (c == ' ')
            {
                mapped->st = 0;
            }
            if (c == '\n')
            {
                mapped->st = 1;
            }
            if (c == '\0')
            {
                mapped->st = 2;
            }
```

```
            human_set(sem, 1);
            while(human_get(sem) == 1)
            {
                continue;
            }
        }
    }
    munmap(mapped, sizeof(bebra));
    sem_close(sem);
    sem_destroy(sem);
    return 0;
}
```

Демонстрация работы программы

Ввод в консоль:

```
papey@PAPEY:~/Ubuntu/OS/os_lab4/src$ ./main
Enter name of the file:
bebra.txt
Enter some numbers:
1 2 3 4
1 2 -1
2
5 6 7
^Z[1]   Done                    g++ -pthread main.cpp -o main

[2]+  Stopped                 ./main _
```

Содержимое файла bebra.txt:

```
10
2
2
18
```

Выводы

Проделав лабораторную работу, я приобрёл практические навыки, необходимые для работы с отображаемой памятью и семафорами.