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

“Межпроцессорное взаимодействие через memory-mapped files”

Студент: Каширин Кирилл Дмитриевич
Группа: М8О-208Б-20
Вариант: 8
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/lumses/OS

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 решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через отображаемые файлы (memory-mapped files).

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

Общие сведения о программе

Программа написана на языке C++ в UNIX-подобной операционной системе.

Для компиляции требуется указать ключ –pthread и -lrt.

Сборка проекта происходит при помощи make-файла
g++ -g -Wall -std=c++17 -pthread lab4.cpp -lrt -fsanitize=address

Общий метод и алгоритм решения

Программа на вход требует названия файла. Если такого файла не существует программа сразу завершается. Создаём два семафора, которые будут регулировать взаимодействие между дочерним и родительским процессором. Также создаем два файловых дескриптора, с помощью которых будет делать отображение на память вызовом mmap. Считываем построчно информацию из файла и передаем от родительского процессора через memptr1 дочернему. Он обрабатывает строку, полученную из memptr1 и результат кладёт в memptr2, который передаёт информацию из дочернего процесса в родительский. После завершения снимаем отображение файлов на память с помощью munmap и удаляем семафор функцией sem_destroy.

Исходный код

os_lab4.cpp

```
#include <iostream>
#include <unistd.h>
#include <sstream>
#include <signal.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pthread.h>
#include <stdio.h>
#include <semaphore.h>
#include <fstream>
using namespace std;
int main(){
    const char* in_sem_name = "input_semaphor";
    const char* out_sem_name = "output_semaphor";
    sem_unlink(in_sem_name);
    sem_unlink(out_sem_name);
    sem_t* input_semaphore = sem_open(in_sem_name, O_CREAT,
S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH, 1);
    sem_t* output_semaphore = sem_open(out_sem_name, O_CREAT,
S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH, 0);
    std::string filename;
    std::string s;
    int map_fd1 = shm_open("map_fd1.txt", O_RDWR | O_CREAT,
S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH);
    int map_fd2 = shm_open("map_fd2.txt", O_RDWR | O_CREAT,
S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH);
    if(map_fd1 == -1){
        std::cout << "Error during creating/open file1 for file
mapping" << endl;
        exit(1);
    }
    if(map_fd2 == -1){
        std::cout << "Error during creating/open file2 for file
mapping" << endl;
        exit(1);
    }
    char* memptr_1 = (char*)mmap(nullptr, getpagesize(),
PROT_READ | PROT_WRITE,  MAP_SHARED, map_fd1, 0);
```

```
char* memptr_2 = (char*)mmap(nullptr, getpagesize(), PROT_READ
| PROT_WRITE,  MAP_SHARED, map_fd2, 0);
    if (memptr_1 == MAP_FAILED){
        cout << "Error in creating file1 mapping << endl";
        exit(1);
    }
    if (memptr_2 == MAP_FAILED){
        cout << "Error in creating file2 mapping" << endl;
        exit(1);
    }
    cout << "Print name of file: ";
    cin >> filename;
    ifstream file1;
    file1.open(filename, ios_base::in);
    if (!file1) {
        cout << "File not exist!" << endl;
        exit(1);
    }

    int id = fork();
    switch(id){
        case -1:
            cout << "Error during creating fork" << endl;
            exit(1);
            break;
        case 0: {
            sem_wait(output_semaphore);
            sem_wait(input_semaphore);
            sem_post(output_semaphore);
            s = "";
            struct stat st;
            if(fstat(map_fd1, &st)){
                cout << "Error during fstat" << endl;
                exit(1);
            }
            int ind = 0, idx = 0;
            int key = 0, length_1 = 0;
            while(ind <= st.st_size){
                if(memptr_1[ind] != '\n'){
                    s += memptr_1[ind++];
```

```
 } else {
                    long double res = 0;
                    string r;int a;
                    int first = 1;
                    s = s + " ";
                    for (unsigned i =0;i<s.length();i++){
                        if (s[i] != ' '){
                            r+=s[i];
                        } else {
                            if (first == 1) {
                                res = stoi(r);
                                first = 0;
                            } else {
                                a = stoi(r);
                                if (a == 0) {
                                    key = 1;
                                    break;
                                } else {
                                    res/=a;
                                }
                            }
                            r ="";
                        }
                    }
                    if (key == 0) {
                        s =  to_string(res);
                    } else {
                        s = "You can not div by zero!";
                        key = 0;
                    }
                    s = s + "\n";
                    length_1 += s.length() * sizeof(char);
                    if(ftruncate(map_fd2, length_1)){
                        cout << "Error during ftruncate" <<
endl;
                        exit(1);
                    }
                    for(unsigned i = 0; i < s.length(); i++){
                        memptr_2[idx++] = s[i];
                    }
```

```
  ind++;
                    s = "";
                }
            }
            sem_post(input_semaphore);
            break;
        }
        default: {
            sem_wait(input_semaphore);
            int idx = 0;
            int length = 0;
            sem_post(output_semaphore);
            while(!file1.eof()){
                getline(file1,s);
                if (s!="") {
                    s = s + "\n";
                    length += s.length() * sizeof(char);
                    if(ftruncate(map_fd1, length)){
                        cout << "Error during ftruncate" <<
endl;
                        exit(1);
                    }
                    for(unsigned i = 0; i < s.length(); i++){
                        memptr_1[idx++] = s[i];
                    }
                }
            }
            sem_post(input_semaphore);
            s = "";
            int ind = 0;
            sem_wait(output_semaphore);
            sem_wait(input_semaphore);
            struct stat st;
            if(fstat(map_fd2, &st)) {
                cout << "Error during fstat" << endl;
                exit(1);
            }
            while(ind <= st.st_size) {
                if(memptr_2[ind] != '\n') {
                    s += memptr_2[ind++];
```

```
} else {
                    s += "\n";
                    cout << s;
                    ++ind;
                    s = "";
                }
            }
            close(map_fd1);
            close(map_fd2);
            shm_unlink("map_fd1.txt");
            shm_unlink("map_fd2.txt");
            remove("map_fd1.txt");
            remove("map_fd2.txt");
            sem_destroy(input_semaphore);
            sem_destroy(output_semaphore);
            munmap(memptr_1,getpagesize());
            munmap(memptr_2,getpagesize());
            return 0;
        }
    }
}
```

Демонстрация работы программы

```
kirill@LAPTOP-F153AKTP:~$ cd OS/os_lab4/src
kirill@LAPTOP-F153AKTP:~/OS/os_lab4/src$ make
g++ -g -Wall -std=c++17 -pthread os_lab4.cpp -lrt -fsanitize=address
kirill@LAPTOP-F153AKTP:~/OS/os_lab4/src$ ./a.out
Print name of file: test.txt
1.000000
0.000000
0.000002
5.000000
kirill@LAPTOP-F153AKTP:~/OS/os_lab4/src$ ./a.out
Print name of file: etwtwfsa
File not exist!
kirill@LAPTOP-F153AKTP:~/OS/os_lab4/src$ ./a.out
Print name of file: test.txt
1.000000
You can not div by zero!
0.000002
5.000000
```

Выводы

Эта лабораторная работа ознакомила и научила меня работать с расширяемой памятью. Научился синхронизировать работу процессов и потоков с помощью семафоров. В отличие от лабораторной работы №2, где мы вызывали read и write, взаимодействие между процессами через mmaped – files происходит эффективнее и требует меньше памяти.