Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

Студент: Примаченко Александр Александрович

Группа: М8О-208Б-20

Вариант: 3

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Исходный код
5. Выводы

**Репозиторий**

https://github.com/SashaPaladin/OS/tree/main/4\_lab

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в:

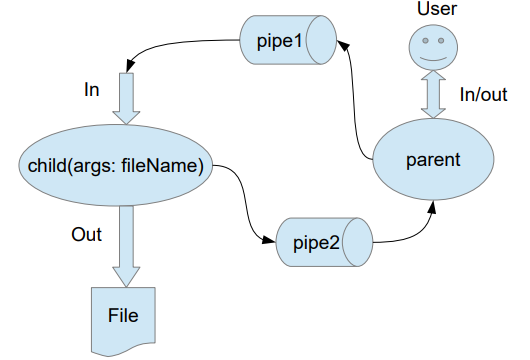
· Освоение принципов работы с файловыми системами

· Обеспечение обмена данных между процессами посредством технологии «File mapping»

**Задание**

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.



**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: unistd.h, stdio.h , stdlib.h, fcntl.h, errno.h, sys/mman.h, sys/stat.h, string.h, stdbool.h, ctype.h, sys/wait.h, semaphore.h. В программе используются следующие системные вызовы:

1. shm\_open - создаёт/открывает объекты общей памяти POSIX.
2. sem\_open - инициализирует и открывает именованный семафор.
3. ftruncate - обрезает файл до заданного размера.
4. mmap, munmap - отображает файлы или устройства в памяти, или удаляет их отображение.
5. memset - заполнение памяти значением определённого байта.
6. sem\_getvalue - возвращает значение семафора.
7. close - закрывает файловый дескриптор.
8. sem\_close - закрывает именованный семафор.
9. execl - запуск файла на исполнение.
10. sem\_getvalue - возвращает значение семафора.
11. sem\_wait - блокирует семафор.
12. sem\_post - разблокирует семафор.

**Исходный код**

**Main.cpp**

#include <fcntl.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <unistd.h>

#include <string>

#include <iostream>

#include "shrmem.h"

using namespace std;

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

int map\_fd = shm\_open(BackingFile, O\_RDWR, AccessPerms);

if (map\_fd < 0) {

perror("SHM\_OPEN");

exit(EXIT\_FAILURE);

}

struct stat statbuf;

fstat(map\_fd, &statbuf);

const size\_t map\_size = statbuf.st\_size;

caddr\_t memptr = static\_cast<char\*>(mmap(

NULL,

map\_size,

PROT\_READ | PROT\_WRITE,

MAP\_SHARED,

map\_fd,

0));

if (memptr == MAP\_FAILED) {

perror("MMAP");

exit(EXIT\_FAILURE);

}

sem\_t \*semptr = sem\_open(SemaphoreName, O\_CREAT, AccessPerms, 2);

if (semptr == SEM\_FAILED) {

perror("SEM\_OPEN");

exit(EXIT\_FAILURE);

}

if (sem\_wait(semptr) != 0) {

perror("SEM\_WAIT");

exit(EXIT\_FAILURE);

}

char \*out = (char \*)calloc(1, sizeof(char));

size\_t m\_size = 0;

int flag = 0;

string first;

string second;

FILE \*filename = fopen(argv[1], "w");

if (filename == NULL) cout << "fopen error";

for (int i = 0; i + 1 < map\_size; ++i) {// преобразование

if (flag == 0) {

first.push\_back(memptr[i]);

}

else if (flag == 1) {

second.push\_back(memptr[i]);

}

if (memptr[i] == ' ' && flag == 0) {

flag = 1;

}

else if ((memptr[i] == ' ' || memptr[i] == '\n') && flag == 1) {

if (atoi(second.c\_str()) == 0) {

perror("division by zero error\n");

break;

}

first = to\_string(atoi(first.c\_str())/atoi(second.c\_str()));

second = "";

if (memptr[i] == '\n') {

fprintf(filename, "%s\n", first.c\_str());

flag = 0;

//cout << first << endl;

first = "";

second = "";

}

}

//out[m\_size] = memptr[i];

//out = (char \*)realloc(out, (++m\_size + 1) \* sizeof(char));

}

fclose(filename);

out[m\_size++] = '\0';

ftruncate(map\_fd, (off\_t)m\_size);

memset(memptr, '\0', m\_size);

sprintf(memptr, "%s", out);

free(out);

close(map\_fd);

usleep(00150000);

sem\_post(semptr);

sem\_close(semptr);

return EXIT\_SUCCESS;

}

**Child.cpp**

#include <fcntl.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <unistd.h>

#include <string>

#include <iostream>

#include "shrmem.h"

using namespace std;

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

int map\_fd = shm\_open(BackingFile, O\_RDWR, AccessPerms);

if (map\_fd < 0) {

perror("SHM\_OPEN");

exit(EXIT\_FAILURE);

}

struct stat statbuf;

fstat(map\_fd, &statbuf);

const size\_t map\_size = statbuf.st\_size;

caddr\_t memptr = static\_cast<char\*>(mmap(

NULL,

map\_size,

PROT\_READ | PROT\_WRITE,

MAP\_SHARED,

map\_fd,

0));

if (memptr == MAP\_FAILED) {

perror("MMAP");

exit(EXIT\_FAILURE);

}

sem\_t \*semptr = sem\_open(SemaphoreName, O\_CREAT, AccessPerms, 2);

if (semptr == SEM\_FAILED) {

perror("SEM\_OPEN");

exit(EXIT\_FAILURE);

}

if (sem\_wait(semptr) != 0) {

perror("SEM\_WAIT");

exit(EXIT\_FAILURE);

}

char \*out = (char \*)calloc(1, sizeof(char));

size\_t m\_size = 0;

int flag = 0;

string first;

string second;

FILE \*filename = fopen(argv[1], "w");

if (filename == NULL) cout << "fopen error";

for (int i = 0; i + 1 < map\_size; ++i) {// преобразование

if (flag == 0) {

first.push\_back(memptr[i]);

}

else if (flag == 1) {

second.push\_back(memptr[i]);

}

if (memptr[i] == ' ' && flag == 0) {

flag = 1;

}

else if ((memptr[i] == ' ' || memptr[i] == '\n') && flag == 1) {

if (atoi(second.c\_str()) == 0) {

perror("division by zero error\n");

break;

}

first = to\_string(atoi(first.c\_str())/atoi(second.c\_str()));

second = "";

if (memptr[i] == '\n') {

fprintf(filename, "%s\n", first.c\_str());

flag = 0;

//cout << first << endl;

first = "";

second = "";

}

}

//out[m\_size] = memptr[i];

//out = (char \*)realloc(out, (++m\_size + 1) \* sizeof(char));

}

fclose(filename);

out[m\_size++] = '\0';

ftruncate(map\_fd, (off\_t)m\_size);

memset(memptr, '\0', m\_size);

sprintf(memptr, "%s", out);

free(out);

close(map\_fd);

usleep(00150000);

sem\_post(semptr);

sem\_close(semptr);

return EXIT\_SUCCESS;

}

**Shrmem.h**

#ifndef SRC\_\_SHRMEM\_H\_

#define SRC\_\_SHRMEM\_H\_

#include <fcntl.h>

const char \*BackingFile = "os\_lab4.back";

const char \*SemaphoreName = "os\_lab4.semaphore";

unsigned AccessPerms = S\_IWUSR | S\_IRUSR | S\_IRGRP | S\_IROTH;

#endif//SRC\_\_SHRMEM\_H\_

**Выводы**

В Си помимо механизма общения между процессами через pipe, также существуют и другие способы взаимодействия, например отображение файла в память, такой подход работает быстрее, за счет отсутствия постоянных вызовов read, write и тратит меньше памяти под кэш. После отображения возвращается void\*, который можно привести к своему указателю на тип и обрабатывать данные как массив, где возвращенный указатель – указатель на первый элемент.