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: Шапошник Никита Сергеевич

Группа: М8О-208Б-21

Вариант: 15

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

[**https://github.com/Nikitashap/OS/tree/main/lab4**](https://github.com/Nikitashap/OS/tree/main/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.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

**Вариант 15)** Правило проверки: строка должна начинаться с заглавной буквы

**Общие сведения о программе**

Программа представляет из себя один файл lab4.cpp.

**Общий метод и алгоритм решения**

Для передачи информации между процессами создаем два файла.

Далее при помощи системного вызова fork(), создаем дочерний процесс.

Родительский процесс получает на вход строки и передает их дочернему процессу, при помощи одного из созданных файлов. Дочерний процесс проверяет строки на соблюдение условий, и ,при их выполнении, записывает их в отдельный файл.

**Исходный код**

#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>

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, 777, 1);

sem\_t\* output\_semaphore = sem\_open(out\_sem\_name, O\_CREAT, 777, 0);

string filename;

string s;

int pipe1 = open("pipe1.txt", O\_RDWR | O\_CREAT, 0777);

int pipe2 = open("pipe2.txt", O\_RDWR | O\_CREAT, 0777);

if(pipe1 == -1){

cout << "Error during creating file1 for file mapping\n";

exit(1);

}

if(pipe2 == -1){

cout << "Error during creating file2 for file mapping\n";

exit(1);

}

char\* pipe\_1 = (char\*)mmap(nullptr, getpagesize(), PROT\_READ | PROT\_WRITE, MAP\_SHARED, pipe1, 0);

char\* pipe\_2 = (char\*)mmap(nullptr, getpagesize(), PROT\_READ | PROT\_WRITE, MAP\_SHARED, pipe2, 0);

if (pipe\_1 == MAP\_FAILED){

cout << "Error in creating file1 mapping\n";

exit(1);

}

if (pipe\_2 == MAP\_FAILED){

cout << "Error in creating file2 mapping\n";

exit(1);

}

cout << "Print input name of file: ";

cin >> filename;

int f = open(filename.c\_str(), O\_RDWR | O\_CREAT | O\_APPEND, 0777);

if(f == -1){

cout << "Error during creating file\n";

exit(1);

}

int id = fork();

switch(id){

case -1:

cout << "Error during creating fork\n";

exit(1);

break;

case 0: {

sem\_wait(output\_semaphore);

sem\_wait(input\_semaphore);

sem\_post(output\_semaphore);

if(dup2(f, 1) == -1){

return 6;

}

s = "";

struct stat st;

if(fstat(pipe1, &st)){

cout << "Error during fstat\n";

exit(1);

}

int ind = 0, idx = 0;

int length\_1 = 0, length\_2 = 0;

while(ind <= st.st\_size){

if(pipe\_1[ind] != '\n'){

s += pipe\_1[ind++];

} else {

if('A' <= s[0] and s[0] <= 'Z'){

s = s + "\n";

length\_1 += s.length() \* sizeof(char);

// if(ftruncate(f, length\_1)){

// cout << "Error during ftruncate\n";

// exit(1);

// }

cout << s;

} else {

s = "Error in string " + s;

s = s + "\n";

length\_2 += s.length() \* sizeof(char);

if(ftruncate(pipe2, length\_2)){

cout << "Error during ftruncate\n";

exit(1);

}

for(int i = 0; i < s.length(); i++){

pipe\_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(cin >> s && s != "EOF"){

s = s + "\n";

length += s.length() \* sizeof(char);

if(ftruncate(pipe1, length)){

cout << "Error during ftrancate\n";

exit(1);

}

for(int i = 0; i < s.length(); i++){

pipe\_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(pipe2, &st)){

cout << "Error during fstat\n";

exit(1);

}

while(ind <= st.st\_size){

if(pipe\_2[ind] != '\n'){

s += pipe\_2[ind++];

} else {

s += "\n";

cout << s;

++ind;

s = "";

}

}

close(pipe1);

close(pipe2);

close(f);

remove("pipe1.txt");

remove("pipe2.txt");

sem\_destroy(input\_semaphore);

sem\_destroy(output\_semaphore);

return 0;

}

}

}

**Демонстрация работы программы**

Ввод в консоль:

iMac-Apple:lab4 appleimac$ ./a.out

Print input name of file: File.txt

Truth

Peace

got

row

^C

iMac-Apple:lab4 appleimac$ cat File.txt

Truth

Peace

**Выводы**

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