Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

**Межпроцессорное взаимодействие через memory-mapped files**

Студент: Белоусов Егор Леонидович

Группа: М8О–208Б–21

Вариант: 14

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2022.

**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.

## Вариант 14

Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их процессу child1. Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода. Child1 переводит строки в нижний регистр. Child2 убирает все задвоенные пробелы.

**Общие сведения о программе**

Программа представляет из себя один файл main.c, компилирующийся с помощью CMake.

**Общий метод и алгоритм решения**.

С помощью вызова fork создаются два дочерних процесса, в родительском процессе считываются строки из стандартного ввода (stdin), которые затем передаются процессу child1. Child1 переводит строки в нижний регистр, после чего передает их child2. Он, в свою очередь, убирает задвоенные пробелы, и передает родительскому процессу финальные строки. Родительский процесс выводит результат в стандартный вывод (stdout). Передача строк между процессами осуществляется с помощью маппинга (mmap).

**Основные файлы программы**

**main.c:**

#include "unistd.h"

#include "string.h"

#include "stdio.h"

#include "ctype.h"

#include "sys/mman.h"

#include "sys/wait.h"

int main() {

char str[1000] = {};

char \*msg = "Enter the string (use ~ as end of input):\n";

write(1, msg, sizeof(char) \* strlen(msg));

for (int i = 0; i < 1000; ++i) {

read(0, &str[i], sizeof(char));

if (str[i] == '~') {

char c;

read(0, &c, sizeof(char));

break;

}

}

int len = strlen(str);

char\* ptr = mmap(NULL, sizeof(char) \* len, PROT\_READ | PROT\_WRITE,

MAP\_SHARED | MAP\_ANON, 0, 0);

if (ptr == MAP\_FAILED) {

perror("Mapping error\n");

return -1;

}

for (int i = 0; i < len; ++i)

ptr[i] = str[i];

int ch1 = fork();

if (ch1 == -1) {

perror("Child1 error\n");

return -1;

}

else if (ch1 == 0) {

for (int i = 0; i < strlen(ptr); ++i)

ptr[i] = tolower(ptr[i]);

}

else {

waitpid(ch1, NULL, 0);

int ch2 = fork();

if (ch2 == -1) {

perror("Child2 error\n");

return -1;

}

else if (ch2 == 0) {

int i = 1;

while (i < strlen(ptr)) {

if (ptr[i - 1] == 32 && ptr[i] == 32) {

memmove(&ptr[i], &ptr[i + 1], strlen(ptr) - i);

--i;

}

++i;

}

}

else {

waitpid(ch2, NULL, 0);

char \*msg = "Result string:\n";

write(1, msg, sizeof(char) \* strlen(msg));

int i = 0;

while (ptr[i] != '~') {

write(1, &ptr[i], sizeof(char));

++i;

}

write(1, "\n", sizeof(char));

}

}

int err = munmap(ptr, strlen(ptr) \* sizeof(char));

if (err != 0) {

perror("Unmapping error\n");

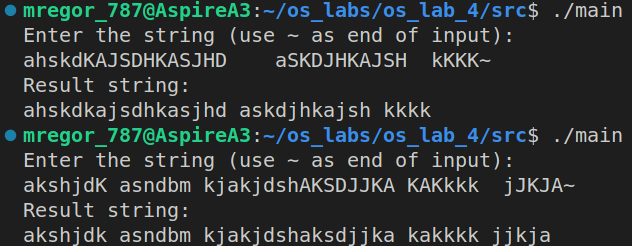
return -1;

}

return 0;

}

**Пример работы**

****

**Вывод**

Проделав лабораторную работу, я узнал, что есть альтернатива pipes для передачи данных между процессами, называемая файл-маппингом, и научился им пользоваться.