

Московский Авиационный Институт  
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики  
Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу  
«Операционные системы»**

**ФАЙЛЫ, ОТОБРАЖАЕМЫЕ В ПАМЯТЬ**

Студент: Филиппов Владимир Михайлович

Группа: М8О–210Б–22

Вариант: 15

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_

Дата: \_\_\_\_\_

Подпись: \_\_\_\_\_

Москва, 2023.

## Постановка задачи

### Цель работы

Целью является приобретение практических навыков в:

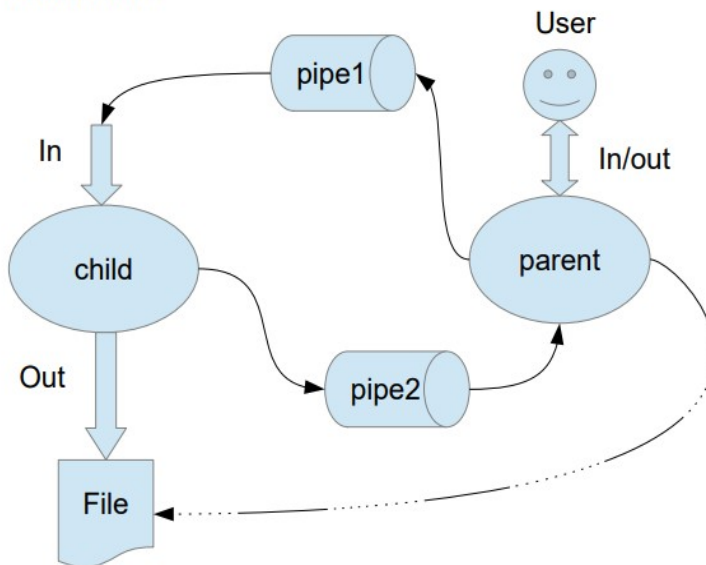
- Освоении принципов работы с файловыми системами.
- Обеспечение обмена данными между процессами посредством технологии «File mapping»

### 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 15: строка должна начинаться с заглавной буквы

Группа вариантов 4



### Общие сведения о программе

В программе используются следующие системные вызовы:

- 1. mmap** – отражает length байтов, начиная со смещения offset файла (или другого объекта), определенного файловым дескриптором fd, в память, начиная с адреса start. Последний параметр (адрес) необязателен, и обычно бывает равен 0. Настоящее местоположение отраженных

данных возвращается самой функцией mmap, и никогда не бывает равным 0.

**2. munmap** – снимает отражения файла в памяти.

**3. shm\_unlink** – удаляет файл, отраженный в память.

**4. fork** — создает новый процесс.

### **Общий метод и алгоритм решения.**

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы файлов, отображаемых в память.
2. Написать программу дочернего и родительского процессов.
3. Организовать простейший командный интерфейс в файлах parent.cpp и son.cpp

### **Основные файлы программы**

**main.cpp:**

```
#include <iostream>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <string>
#include <cstring>
#include <vector>
#define SHARED_MEMORY_SIZE 1024
void checkStr(const char *ptr, char *shared_error, FILE *file, int size) {
    std::string res;
    for (int i = 1; i < size - 1; ++i) {
        if (isupper(ptr[i])) {
            int j = i;
            std::string tmp = "";
            while (ptr[j] != '\0' && ptr[j] != '\n') {
                tmp.push_back(ptr[j]);
                j++;
            }
            fprintf(file, "%s\n", tmp.c_str());
            res += "String start with uppercase\n";
        }
    }
}
```

```

        i = j;
    } else {
        res += "String does not start with uppercase\n";
        int j = i;
        while (ptr[j] != '\0' && ptr[j] != '\n') {
            j++;
        }
        i = j;
    }
}
strcpy(shared_error, res.c_str());
}
char* getInput(int& size) {
    char symbol;
    char* in = (char*)malloc(sizeof(char));
    if (in == NULL) {
        perror("malloc");
    }
    size_t i = 0;
    size = 1;
    std::cout << "Enter something strings. If you want to stop, press Ctrl+D" <<
std::endl;
    while ((symbol = getchar()) != EOF) {
        in[i++] = symbol;
        if (i == size) {
            size *= 2;
            in = (char*)realloc(in, size * sizeof(char));
            if (in == NULL) {
                perror("realloc");
            }
        }
    }
    size = i + 1;
    in = (char*)realloc(in, size * sizeof(char));
    if (in == NULL) {
        perror("realloc");
    }
    in[size - 1] = '\0';
    return in;
}
int main() {
    std::string filePath;

```

```

std::cout << "write path to file\n";
std::cin >> filePath;
FILE *file = fopen(filePath.c_str(), "a");
int fd_input = shm_open("input", O_CREAT | O_RDWR, 0666);
if (fd_input == -1) {
    shm_unlink("input");
    perror("shm_open");
}
if (ftruncate(fd_input, SHARED_MEMORY_SIZE) == -1) {
    shm_unlink("input");
    perror("ftruncate");
}
char * shared_input = (char *) mmap(NULL, SHARED_MEMORY_SIZE,
PROT_READ | PROT_WRITE, MAP_SHARED, fd_input, 0);
if (shared_input == MAP_FAILED) {
    munmap(shared_input, SHARED_MEMORY_SIZE);
    shm_unlink("input");
    perror("mmap");
}
close(fd_input);
int fd_error = shm_open("error", O_CREAT | O_RDWR, 0666);

if (fd_error == -1) {
    shm_unlink("error");
    munmap(shared_input, SHARED_MEMORY_SIZE);
    shm_unlink("input");
    perror("shm_open");
}
if (ftruncate(fd_error, SHARED_MEMORY_SIZE) == -1) {
    shm_unlink("error");
    munmap(shared_input, SHARED_MEMORY_SIZE);
    shm_unlink("input");
    perror("ftruncate");
}
char * shared_error = (char *) mmap(NULL, SHARED_MEMORY_SIZE,
PROT_READ | PROT_WRITE, MAP_SHARED, fd_error, 0);
if (shared_error == MAP_FAILED) {
    munmap(shared_error, SHARED_MEMORY_SIZE);
    shm_unlink("error");
    munmap(shared_input, SHARED_MEMORY_SIZE);
    shm_unlink("input");
    perror("mmap");
}

```

```

    }
    close(fd_error);
    char * input;
    int size;
    input = getInput(size);
    pid_t pid = fork();
    if (pid == 0) {
        checkStr(input, shared_error, file, size);
    } else if (pid > 0) {
        wait(NULL);
        int i = 0;
        for (int i = 0; i < strlen(shared_error); i++) {
            std::cout << shared_error[i];
        }
    } else {
        munmap(shared_error, SHARED_MEMORY_SIZE);
        shm_unlink("error");
        munmap(shared_input, SHARED_MEMORY_SIZE);
        shm_unlink("input");
        perror("fork");
    }
    munmap(shared_error, SHARED_MEMORY_SIZE);
    shm_unlink("error");
    munmap(shared_input, SHARED_MEMORY_SIZE);
    shm_unlink("input");
}

```

### Пример работы

```

[main][src]$ ./a.out
write path to file
abc.txt
Enter something strings. If you want to stop, press Ctrl+D
Abcd
Bafsjdkl
2134
sdfn
String start with uppercase
String start with uppercase
String does not start with uppercase
String does not start with uppercase

```

## **Вывод**

В Си и С++ кроме механизма общения между процессами через каналы, также существуют и другие способы IPC. Например, отображение файла в память. Такой подход работает быстрее, за счет отсутствия постоянных вызовов, а сам файл находится в оперативной памяти.

При выполнении Л.Р. я научился работать с такими файлами, написал учебную программу для IPC. Думаю, что файлы отображаемые в память – удобный и главное быстрый механизм, который можно использовать во многих сферах программирования (обычный файл можно, например, отобразить в память и читать с него быстрее).