

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ
по лабораторной работе № 2.1
по дисциплине «Операционные системы»
Тема: «Управление файловой системой»

Студент гр. 3311 Пасечный Л.В._____

Преподаватель Тимофеев А. В._____

Санкт-Петербург

2025

Введение

Цель работы:

Исследовать механизмы управления виртуальной памятью

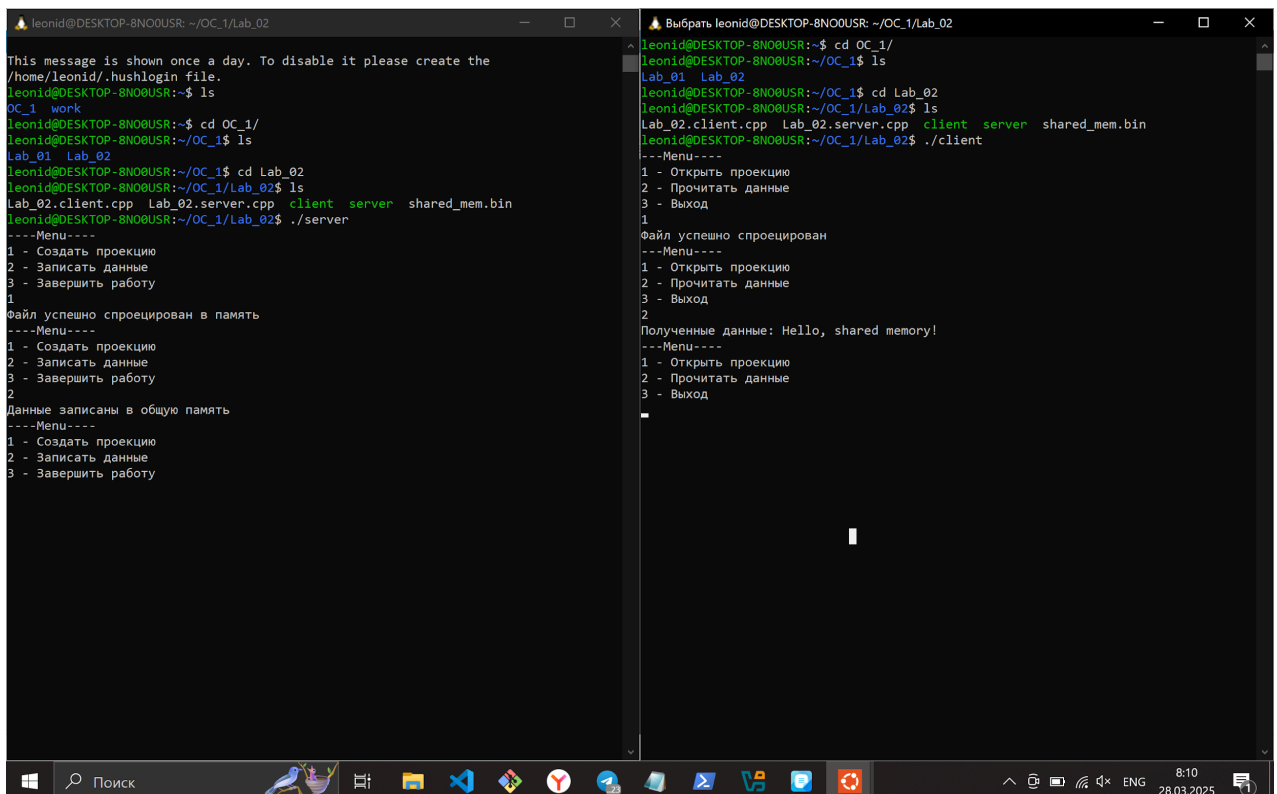
Постановка задачи:

Сервер создает файл на диске и проецирует его в память, далее записывает информацию и ждет, пока клиент не прочтет, затем отменяет проецирование и удаляет файл. Клиент открывает файл, проецирует и ждет доступности чтения, затем читает и выводит результат, в завершении работы отменяет проецирование.

Сделать меню в каждой программе:

- 1) пункты меню Сервера – «выполнить проецирование», «записать данные», «завершить работу»;
- 2) пункты меню Клиента – «выполнить проецирование», «прочитать данные», «завершить работу».

Результаты:



```
leonid@DESKTOP-8N00USR: ~/OC_1/Lab_02
This message is shown once a day. To disable it please create the
/home/leonid/.hushlogin file.
leonid@DESKTOP-8N00USR:~$ ls
OC_1  work
leonid@DESKTOP-8N00USR:~$ cd OC_1/
leonid@DESKTOP-8N00USR:~/OC_1$ ls
Lab_01  Lab_02
leonid@DESKTOP-8N00USR:~/OC_1$ cd Lab_02
leonid@DESKTOP-8N00USR:~/OC_1/Lab_02$ ls
Lab_02.client.cpp  Lab_02.server.cpp  client  server  shared_mem.bin
leonid@DESKTOP-8N00USR:~/OC_1/Lab_02$ ./server
---Menu---
1 - Создать проекцию
2 - Записать данные
3 - Завершить работу
1
Файл успешно спроецирован в память
---Menu---
1 - Создать проекцию
2 - Записать данные
3 - Завершить работу
2
Данные записаны в общую память
---Menu---
1 - Создать проекцию
2 - Записать данные
3 - Завершить работу
3

leonid@DESKTOP-8N00USR:~$ cd OC_1/
leonid@DESKTOP-8N00USR:~/OC_1$ ls
Lab_01  Lab_02
leonid@DESKTOP-8N00USR:~/OC_1$ cd Lab_02
leonid@DESKTOP-8N00USR:~/OC_1/Lab_02$ ls
Lab_02.client.cpp  Lab_02.server.cpp  client  server  shared_mem.bin
leonid@DESKTOP-8N00USR:~/OC_1/Lab_02$ ./client
---Menu---
1 - Открыть проекцию
2 - Прочитать данные
3 - Выход
1
Файл успешно спроецирован
---Menu---
1 - Открыть проекцию
2 - Прочитать данные
3 - Выход
2
Полученные данные: Hello, shared memory!
---Menu---
1 - Открыть проекцию
2 - Прочитать данные
3 - Выход
3
```

Заключение

Мы научились проецировать данные в память компьютера.

Код программы

Server:

```
#include <iostream>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <string.h>

#define FILENAME "shared_mem.bin"
#define FILESIZE 1024

int fd = -1;
char *ptr = NULL;

void create_mapping(){
    fd = open(FILENAME, O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    if (fd == -1){
        perror("Ошибка при создании файла");
        exit(EXIT_FAILURE);
    }

    // Установка размеров файла
    if (ftruncate(fd, FILESIZE) == -1){
        perror("Ошибка установки размеров файла");
        exit(EXIT_FAILURE);
    }

    // Проецирование файла в память
    ptr = (char*)mmap(NULL, FILESIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if (ptr == MAP_FAILED){
        perror("Ошибка проецирования файла");
        close(fd);
        exit(EXIT_FAILURE);
    }
    printf("Файл успешно спроецирован в память\n");
}

void write_data(){
    if (!ptr){
        printf("Сначала выполните проецирование!");
        return;
    }

    //Запись данных
    sprintf(ptr, "Hello, shared memory!");
    printf("Данные записаны в общую память\n");
}

void cleanup(){
    if (ptr != NULL){
        munmap(ptr, FILESIZE);
        ptr = NULL;
    }
    if (fd == -1){
        close(fd);
        unlink(FILENAME);
        fd = -1;
    }
    printf("Ресурсы освобождены. Файл удалён.\n");
}
```

```

}
void menu(){
    printf("----Menu----\n");
    printf("1 - Создать проекцию\n");
    printf("2 - Записать данные\n");
    printf("3 - Завершить работу\n");
}
int main(){
    int choice = 0;

    do{
        menu();
        scanf("%d", &choice);
        switch(choice){
            case 1:
                create_mapping();
                break;
            case 2:
                write_data();
                break;
            case 3:
                cleanup();
                break;
            default:
                printf("Попробуйте ещё раз\n");
                break;
        }
    } while (choice != 3);

    return 0;
}

```

Client:

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/select.h>

#define FILENAME "shared_mem.bin"
#define FILESIZE 1024

int fd = -1;
char *ptr = NULL;

void open_mapping(){
    fd = open(FILENAME, O_RDONLY);
    if (fd == -1){
        perror("Ошибка открытия файла");
        exit(EXIT_FAILURE);
    }

    // Проецирование файла
    ptr = (char*)mmap(NULL, FILESIZE, PROT_READ, MAP_SHARED, fd, 0);
    if (ptr == MAP_FAILED){
        perror("Ошибка проецирования файла");
        close(fd);
        exit(EXIT_FAILURE);
    }
    printf("Файл успешно спроецирован\n");
}

```

```

}

void read_data(){
    if (!ptr){
        printf("Сначала выполните проецирование!\n");
        return;
    }

    //Ожидание доступности данных
    fd_set read_fds;
    FD_ZERO(&read_fds);
    FD_SET(fd, &read_fds);

    struct timeval timeout = {5, 0}; // Таймаут 5 секунд

    int ready = select(fd + 1, &read_fds, NULL, NULL, &timeout);
    if (ready == -1){
        perror("Ошибка select");
        exit(EXIT_FAILURE);
    } else if (ready == 0){
        printf("Данные недоступны (таймаут)\n");
        return;
    }

    //Чтение данных
    printf("Полученные данные: %s\n", ptr);
}

void cleanup(){
    if (ptr != NULL){
        munmap(ptr, FILESIZE);
        ptr = NULL;
    }
    if (fd == -1){
        close(fd);
        fd = -1;
    }
    printf("Ресурсы клиента освобождены\n");
}

void menu(){
    printf("---Menu----\n");
    printf("1 - Открыть проекцию\n");
    printf("2 - Прочитать данные\n");
    printf("3 - Выход\n");
}

int main(){
    int choice = 0;

    do{
        menu();
        scanf("%d", &choice);
        switch(choice){
            case 1:
                open_mapping();
                break;
            case 2:
                read_data();
                break;
            case 3:

```

```
        cleanup();
        break;
    default:
        printf("Попробуйте ещё раз!");
        break;
    }
} while (choice != 3);

return 0;
}
```