**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ** **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** **«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ** **“СИНЕРГИЯ”»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  |  |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  |  |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  |  |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Отчет по лабораторной работе №2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **на тему** |  | Файловые операции средствами системных вызовов | | |
|  |  | (наименование темы) | | |
|  |  |  | | |
| **по дисциплине** | | |  | Системное программирование |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | Волков Никита Валерьевич |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | ДКИП – 205 прог |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев И.В. |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2025 г.**

**Лабораторная работа №2. «**Файловые операции средствами системных вызовов**»**

**Цель работы:**

получить навыки разработки приложений, реализующих операции с файлами средствами системных вызовов Linux API на языке C в операционных системах семейства Linux.

2. Краткие теоретические сведения: см. главу 3, стр. 25. Пример кода: листинг 3.1 (см. стр. 27) и файл file.c в разделе «Примеры программ», «Файлы» сайта по дисциплине [6].

3. Методические указания.

3.1. Проект может быть реализован в виде консольного приложения в среде ОС Ubuntu, Fedora, CentOS или других средствами компилятора gcc версии не ниже 4.

3.2. Проект должен быть реализован с применением системных вызовов для открытия, чтения, записи и закрытия файла. Ввод исходных данных производится исключительно через аргументы командной строки. Вывод сообщений об ошибках и результатах работы программы может производиться средствами стандартной библиотеки языка С.

3.3. Проект должен предусматривать обработку исключительных ситуаций (отсутствие или неверное количество входных параметров, ошибки открытия входного и/или выходного файла, ошибки чтения и записи).

3.4. Проект может содержать системные вызовы для блокировки файлов (входного – на чтение, выходного – на запись). Наличие функций блокировки файлов соответствует заданию повышенной сложности.

4. Порядок выполнения работы.

4.1. Написать и отладить программу, получающую в аргументах командной строки имя существующего текстового файла, имя выходного файла, который будет переписан при его наличии и в который будет помещен результат работы программы и символ, слово или число, используемый (ое) для обработки файла.

4.2. Результатом работы программы является выходной текстовый файл, содержащий текст, обработанный согласно вариантам, и возвращаемое значение – количество выполненных операций или –1 в случае ошибки.

4.3. При обработке исключительных ситуаций сообщения об ошибках выводятся на терминал.

4.4. Результатом выполнения лабораторной работы считается демонстрация работы программы и обработки исключительных ситуаций преподавателю.

5. Варианты заданий.

Варианты заданий

Вариант 1

Задание

Параметры командной строки

Удалить из текста заданный символ

1. Имя входного файла

2. Заданный символ

6. Содержание отчета.

6.1. Цель работы.

В современном мире программирования навыки разработки приложений, которые эффективно работают с файлами, являются крайне важными. Особенно это касается операционных систем семейства Linux, где системные вызовы предоставляют мощные инструменты для управления файлами и взаимодействия с файловой системой. Для того чтобы стать квалифицированным разработчиком в этой области, необходимо освоить основные концепции работы с системными вызовами в Linux. Это включает в себя изучение таких операций, как создание, открытие, чтение, запись и закрытие файлов. В языке C для этого используются различные функции, такие как open(), read(), write(), close() и многие другие, которые позволяют напрямую взаимодействовать с файловой системой. Кроме того, важно понимать, как работают дескрипторы файлов и как их использовать для управления доступом к ресурсам. Знание о том, как обрабатывать ошибки при выполнении системных вызовов и как правильно управлять памятью, также играет ключевую роль в разработке надежных приложений. Изучение API Linux на языке C предоставляет разработчикам возможность создавать высокопроизводительные и эффективные приложения, которые могут выполнять сложные задачи, связанные с обработкой данных. Это может включать в себя создание утилит для обработки текстовых файлов, работы с бинарными данными или даже разработки серверных приложений, которые требуют интенсивного ввода-вывода. Для достижения этих целей рекомендуется изучать документацию по системным вызовам, а также практиковаться в написании кода, который использует эти вызовы. Создание собственных проектов и участие в open-source инициативах также могут значительно ускорить процесс обучения и углубить понимание работы с файлами в Linux. Таким образом, получение навыков разработки приложений, реализующих операции с файлами средствами системных вызовов Linux API на языке C, является важной частью профессионального роста для любого программиста, стремящегося к успеху в области системного программирования и разработки программного обеспечения.

#include <stdio.h>  
#include <stdlib.h>  
#include <fcntl.h>  
#include <unistd.h>  
  
void remove\_char(const char \*filename, char target) {  
    int input\_fd = open(filename, O\_RDONLY);  
    if (input\_fd == -1) {  
        perror("Ошибка открытия файла");  
        return;  
    }  
      
    char temp\_filename[] = "temp\_XXXXXX";  
    int output\_fd = mkstemp(temp\_filename);  
    if (output\_fd == -1) {  
        perror("Ошибка создания временного файла");  
        close(input\_fd);  
        return;  
    }  
      
    char buffer;  
    ssize\_t bytes\_read;  
    while ((bytes\_read = read(input\_fd, &buffer, 1)) > 0) {  
        if (buffer != target) {  
            write(output\_fd, &buffer, 1);  
        }  
    }  
      
    if (bytes\_read == -1) {  
        perror("Ошибка чтения файла");  
    }  
      
    close(input\_fd);  
    close(output\_fd);  
      
    if (rename(temp\_filename, filename) == -1) {  
        perror("Ошибка замены файла");  
        remove(temp\_filename);  
    }  
}  
  
int main(int argc, char \*argv[]) {  
    if (argc != 3) {  
        printf("Использование: %s <файл> <символ>\n", argv[0]);  
        return 1;  
    }  
      
    remove\_char(argv[1], argv[2][0]);  
    printf("Обработка завершена.\n");  
    return 0;  
}

