Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин
Дисциплина: Операционные системы и системное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему
«Анализатор и редактор файловой системы»

БГУИР КР 6-05-0611-05 124 ПЗ

Студент: гр. 350501 Слепица О.Н.

Руководитель: старший преподаватель
каф. ЭВМ Поденок Л.П.

Минск 2025

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»
Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____Б. В. Никульшин
«___» _____2025 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту Слепице Ольге Николаевне
1. Тема проекта: *“Анализатор и редактор файловой системы”.*
2. Срок сдачи студентом законченного проекта *17 мая 2025 г.*
3. Исходные данные к проекту *Язык программирования – C*
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)
1. Введение.
2. Обзор литературы.
3. Постановка задачи.
4. Системное проектирование.
5. Описание программы для программиста.
6. Функциональное проектирование.
7. Разработка программных модулей.
8. Руководство пользователя.
9. Результаты работы.
10. Заключение.
11. Литература.
5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)
1. Схема алгоритма функции `editor_move_cursor();`
2. Схема алгоритма функции `format_value();`
3. Ведомость документов.
6. Консультант по проекту *Поденок Л. П.*
7. Дата выдачи задания *24 февраля 2025 г.*
8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):
разделы 1, 2, 3 к 01.03 – 15%;

разделы 4, 5, 6, 7 к 01.04 – 50%;
разделы 8, 9, 10, 11 к 01.05 – %80;
оформление пояснительной записки и графического материала к 15.05 – 100%
Защита курсового проекта с 29.05 по 09.06.25г.

РУКОВОДИТЕЛЬ _____Л. П. Поденок

(подпись)

Задание принял к исполнению_____О. Н. Слепица

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Анализатор и редактор файловой системы представляет собой специализированное приложение для низкоуровневого взаимодействия с файловыми системами `ext2/ext3/ext4` в среде `Linux`. Программа обеспечивает прямой доступ к структурам данных на блочном устройстве, позволяя не только анализировать метаданные, но и вносить контролируемые изменения. Реализованная на базе библиотеки `ncurses`, она предлагает интерактивный интерфейс для навигации по суперблоку, групповым дескрипторам, битовым картам и индексным дескрипторам с возможностью их модификации в шестнадцатеричном режиме.

Актуальность разработки обусловлена возрастающей потребностью в инструментах для диагностики и восстановления файловых систем в условиях роста объемов данных и ужесточения требований к их сохранности. В отличие от стандартных утилит, работающих на уровне файлов, данный анализатор предоставляет доступ к низкоуровневым структурам, что критически важно при восстановлении поврежденных разделов, анализе инцидентов безопасности или исследовании работы файловых систем.

Техническая реализация основана на прямом взаимодействии с устройством хранения через файловый дескриптор, что обеспечивает максимальную гибкость при работе с метаданными. Программа автоматически определяет параметры файловой системы, включая размер блока, количество групп и особенности конкретной версии `ext`. Модульная архитектура с четким разделением анализатора, редактора и интерфейса позволяет легко расширять функциональность.

Графическая часть реализует удобную навигацию с цветовой дифференциацией типов данных, подсветкой структурных полей и контекстной помощью. Особое внимание уделено безопасности операций - все критические изменения требуют подтверждения, а просмотр данных возможен без риска случайной модификации. Приложение находит применение в образовательном процессе и разработке файловых систем, заполняя нишу между стандартными утилитами и профессиональными инструментами для работы с дисками.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор аналогов

Существует несколько инструментов, предназначенных для анализа и редактирования файловых систем, однако большинство из них либо ориентировано на определенные узкоспециализированные задачи, либо не обладает удобным интерфейсом для комплексной работы с данными. Рассмотрим наиболее релевантные аналоги, используемые в среде Linux.

debugfs — низкоуровневый инструмент для работы с файловыми системами формата ext2, ext3 и ext4. Он предоставляет возможность изучать структуры данных, редактировать метаданные и выполнять диагностику файловой системы. Однако интерфейс debugfs основан на текстовых командах, что требует глубокого понимания принципов работы ext-файловых систем и затрудняет его использование для менее опытных пользователей.

e2fsprogs — набор утилит для работы с файловыми системами ext. Включает инструменты для проверки целостности данных (e2fsck), управления файловыми системами (tune2fs, resize2fs) и восстановления поврежденных данных (debugfs). Несмотря на мощный функционал, e2fsprogs не предоставляет единого интегрированного интерфейса для взаимодействия с файловой системой, что усложняет выполнение комплексного анализа.

bvi (Binary VI) — мощный бинарный редактор, ориентированный на редактирование низкоуровневых данных. Он позволяет вносить изменения в файлы и дисковые образы на уровне байтов. Однако, несмотря на широкие возможности редактирования, bvi не предоставляет специфических инструментов для анализа структуры ext-файловых систем, что делает его менее удобным для решения поставленных задач.

1.2 Постановка задачи

Целью проекта является разработка программного инструмента для детального анализа и редактирования структуры файловой системы. Программа должна обеспечивать возможность взаимодействия с ключевыми элементами файловой системы, включая суперблок, группы блоков, индексные дескрипторы и выполнение модификации содержимого, включая изменение битовых карт распределения блоков и индексных дескрипторов, редактирование данных в блоках. Интерфейс, реализованный на основе библиотеки ncurses, позволит работать в консольной среде с интерактивным управлением. Программное обеспечение должно быть разработано для операционной системы Linux, с соблюдением стандарта C11.

2 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА

Разработанная программа представляет собой инструмент для анализа и редактирования файловых систем [1] формата `ext2`, `ext3` и `ext4` с интерфейсом на основе `ncurses`. Она объединяет функциональность анализа структур файловой системы, управления блоками и индексными дескрипторами, а также модификации данных на уровне низкоуровневых операций. В отличие от традиционных файловых менеджеров, программа ориентирована на работу с метаданными файловой системы, предоставляя пользователям возможности просмотра, анализа и внесения изменений в структуру хранения данных.

2.1 Основные возможности

Программа включает широкий набор функций для анализа и модификации файловых систем. Одной из ключевых возможностей является просмотр суперблока, содержащего основные параметры файловой системы, такие как размер блоков, количество индексных дескрипторов и блоков, индекс первой свободной структуры. Пользователь может получать детализированную информацию об этих параметрах, а также изменять их, если это требуется для тестирования или восстановления данных.

Также реализован просмотр групп блоков, включающий таблицы индексных дескрипторов, битовые карты распределения блоков и индексных дескрипторов. Эти структуры позволяют анализировать, какие блоки и индексные дескрипторы используются в файловой системе, а также управлять их выделением или освобождением. Битовые карты могут быть изменены пользователем, что позволяет вручную корректировать распределение файловых данных.

Работа с индексными дескрипторами включает возможность просмотра их атрибутов и редактирования их параметров.

Важным компонентом программы является механизм работы с данными на уровне блоков. Предусмотрена возможность чтения и записи данных блоков в бинарном формате, а также редактирование их содержимого с точностью до байта. Это позволяет анализировать низкоуровневую структуру хранения информации и вносить изменения непосредственно в файловую систему.

Программа поддерживает навигацию по файловой системе с возможностью перехода между блоками, индексными дескрипторами и битовыми картами по их номерам. Визуализация данных организована таким образом, чтобы пользователь мог легко ориентироваться в структуре хранилища. Включена цветовая подсветка элементов, упрощающая анализ состояния файловой системы.

Редактирование данных производится в специальном бинарном редакторе, позволяющем изменять содержимое файловых структур. Поддерживаются операции выделения и модификации отдельных байтов, что удобно для ручного редактирования метаданных файловой системы.

Пользовательский интерфейс обеспечивает поддержку горячих клавиш, включая комбинации для быстрой навигации, редактирования данных и анализа структуры файловой системы. Также предусмотрена статусная строка, отображающая текущие параметры работы и активный режим взаимодействия с программой.

2.2 Перспективы развития программы

Дальнейшее развитие программы направлено на расширение её функциональности и улучшение удобства взаимодействия с данными. Важным направлением является интеграция механизмов анализа целостности файловой системы, включая проверку структур на предмет ошибок и несоответствий. Также планируется добавление поддержки файловых систем, отличных от `ext`, что сделает программу универсальным инструментом для анализа хранения данных.

Оптимизация работы с большими объемами данных позволит реализовать эффективную загрузку структур без полной загрузки в память, что улучшит производительность при работе с крупными файловыми системами. Расширение редакторских возможностей может включать поддержку расширенного просмотра данных, интеграцию с внешними инструментами и внедрение автоматизированных сценариев редактирования.

Развитие интерфейса предусматривает улучшение удобства взаимодействия с пользователем, включая возможность гибкой настройки отображаемых элементов, расширенные инструменты навигации и поддержку макросов для упрощения часто выполняемых действий.

В перспективе программа может быть дополнена функциями отображения изменений в журнал, что позволит отслеживать внесённые модификации и анализировать их влияние на файловую систему. Также рассматривается возможность интеграции с инструментами восстановления данных, что обеспечит пользователям средства для диагностики и исправления ошибок в файловых системах.

3 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

После определения требований к функционалу разработанной программы её архитектура была спроектирована таким образом, чтобы обеспечивать гибкость, масштабируемость и удобство сопровождения. В основе проектирования лежит модульный подход, который позволяет разделить систему на независимые компоненты, каждый из которых отвечает за выполнение конкретных задач. Такая структура способствует упрощению разработки, отладки и дальнейшего расширения функционала.

Программа представляет собой инструмент для анализа и редактирования файловых систем формата `ext2`, `ext3` и `ext4` с текстовым интерфейсом на основе `ncurses`. Её архитектура предусматривает отдельные модули для работы с файловой системой, редактирования данных, взаимодействия с пользователем, вспомогательных операций и управления запуском.

3.1 Модуль работы с файловой системой

Модуль предназначен для анализа структуры `ext2/ext3/ext4` и управления её элементами. Он отвечает за инициализацию файловой системы, получение информации о суперблоке, таблицах групп блоков, индексных дескрипторах и битовых картах. Реализованы функции чтения и записи блоков, проверка выделенных блоков и индексных дескрипторов, получение битовых карт распределения данных. Модуль также обеспечивает основные операции с файловыми структурами, такие как просмотр атрибутов индексных дескрипторов, навигация по группам блоков и доступ к системным метаданным.

3.2 Модуль редактирования файловой системы

Данный модуль предоставляет инструменты для редактирования структур файловой системы, включая модификацию битовых карт, изменение атрибутов индексных дескрипторов и редактирование содержимого блоков. Он позволяет пользователю вносить изменения в данные, используя бинарный редактор, а также выполнять операции записи и проверки целостности файловых структур. Поддерживается управление связями между индексными дескрипторами и блоками, что делает модуль полезным для диагностики и восстановления данных.

3.3 Модуль пользовательского интерфейса

Модуль интерфейса обеспечивает взаимодействие пользователя с программой, реализуя консольный `UI` на основе `ncurses`. Он управляет отображением данных, включая навигацию по файлам, блокам и индексным

дескрипторам, выделение активных элементов и поддержку горячих клавиш. Для повышения удобства работы предусмотрена поддержка горячих клавиш, упрощающая выполнение базовых операций. Цветовая подсветка элементов интерфейса способствует визуальному анализу структуры данных, обеспечивая наглядное представление содержимого. Статусная строка служит информационным элементом, предоставляя сведения о текущем состоянии системы, активном режиме работы и выполняемых операциях. Также реализованы диалоговые окна, обеспечивающие ввод команд и выполнение подтверждаемых действий.

3.4 Модуль вспомогательных утилит

Этот модуль включает функции, которые помогают другим частям программы выполнять обработку данных, поддерживая эффективное управление структурой файловой системы. В его задачи входит форматирование числовых значений, работа со строками [2], управление кодировками, а также анализ и модификация битовых карт. Реализованы методы конвертации значений, проверки типов файлов, анализа их атрибутов и представления данных в удобочитаемом формате. Модуль играет ключевую роль в поддержке работы других компонентов, обеспечивая стандартные операции и повышая точность обработки информации.

3.5 Модуль инициализации и управления

Основной модуль отвечает за запуск программы, настройку среды `ncurses` и управление основными процессами. Он выполняет инициализацию всех функциональных блоков, включая загрузку файловой системы, создание интерфейса и запуск основного цикла обработки событий. Управляет корректным завершением работы, включая освобождение выделенной памяти и закрытие дескрипторов [3]. Этот модуль связывает все компоненты программы в единую систему, обеспечивая координацию работы всех остальных элементов.

4 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются структуры программы.

4.1 Структура `fs_info_t`

Данная структура представляет собой контейнер для хранения информации о файловой системе. Она содержит следующие поля:

1) `int fd` — файловый дескриптор устройства, с которым работа;
2) `char *device_path` — путь к файловому устройству;
3) `struct ext2_super_block sb` — данные суперблока, содержащие ключевые параметры файловой системы;
4) `uint32_t block_size` — размер блока в байтах;
5) `uint32_t inodes_per_group` — количество индексных дескрипторов в группе;
6) `uint32_t blocks_per_group` — количество блоков в группе;
7) `uint32_t groups_count` — группы блоков в файловой системе;
8) `structext2_group_desc *group_desc` — массив групп блоков;
9) `bool is_ext4` — флаг, того, что файловая система формата `ext4`.

4.2 Структура `editor_context_t`

Данная структура хранит информацию о состоянии бинарного редактора. Поля структуры:

1) `fs_info_t *fs_info` — указатель на данные о файловой системе;
2) `uint64_t current_offset` — позиция в файле или устройстве;
3) `uint8_t *buffer` — буфер для отображения данных;
4) `size_t buffer_size` — размер буфера;
5) `uint32_t cursor_x` — текущая горизонтальная позиция курсора;
6) `uint32_t cursor_y` — текущая вертикальная позиция курсора;
7) `uint32_t bytes_per_row` — число отображаемых байтов в строке;
8) `uint32_t view_rows` — количество строк в области просмотра;
9) `bool editing_mode` — флаг режима редактирования данных;
10) `bool field_highlight` — флаг подсветки элементов;
11) `structure_type_t current_structure` — тип структуры;
12) `uint32_t current_id` — номер редактируемого элемента;
13) `bool should_exit` — флаг завершения работы редактора.

4.3 Структура `ui_context_t`

Данная структура управляет состоянием пользовательского интерфейса. Структура включает в себя следующие поля:

1) `WINDOW *main_win` — главное окно интерфейса;

2) `WINDOW *status_win` — окно статусной строки;
3) `WINDOW *help_win` — окно справки;
4) `ui_mode_t current_mode` — текущий режим работы интерфейса;
5) `fs_info_t *fs_info` — указатель на файловую систему;
6) `editor_context_t *editor_ctx` — контекст редактора;
7) `int current_block` — номер текущего блока в файловой системе;
8) `int current_inode` — номер текущего индексного дескриптора;
9) `int current_group` — номер текущей группы блоков.

4.4 Структура `ext2_inode`

Данная структура представляет собой описание индексного дескриптора файловой системы `ext`, хранящего информацию о файле или каталоге. Структура включает в себя следующие поля:

1) `uint16_t i_mode` — битовое поле прав доступа и типа файла;
2) `uint16_t i_uid` — идентификатор пользователя-владельца;
3) `uint32_t i_size` — размер файла в байтах;
4) `uint32_t i_atime` — время последнего доступа;
5) `uint32_t i_ctime` — время создания;
6) `uint32_t i_mtime` — время последнего изменения;
7) `uint32_t i_dtime` — время удаления файла;
8) `uint16_t i_gid` — идентификатор группы владельца;
9) `uint16_t i_links_count` — количество жёстких ссылок на файл;
10) `uint32_t i_flags` — флаги состояния индексного дескриптора;
11) `uint32_t i_block[15]` — массив указателей на блоки данных.

4.5 Структура `ext2_super_block`

Данная структура хранит параметры файловой системы, определяющие её общую организацию. Поля структуры:

1) `uint32_t s_inodes_count` — число индексных дескрипторов;
2) `uint32_t s_blocks_count` — общее количество блоков;
3) `uint32_t s_r_blocks_count` — зарезервированные блоки;
4) `uint32_t s_free_blocks_count` — число свободных блоков;
5) `uint32_t s_free_inodes_count` — свободные дескрипторы;
6) `uint32_t s_log_block_size` — логический размер блока;
7) `uint32_t s_blocks_per_group` — число блоков в группе;
8) `uint32_t s_inodes_per_group` — число дескрипторов в группе;
9) `uint32_t s_mtime` — время последнего монтирования;
10) `uint32_t s_wtime` — время последней записи;
11) `uint16_t s_magic` — сигнатура файловой системы;
12) `uint16_t s_state` — текущее состояние файловой системы;

5 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данном разделе рассмотрены алгоритмы работы четырёх функций: функции чтения индексного дескриптора `read_inode()`, функции инициализации редактора `editor_init()`, функции перемещения курсора `editor_move_cursor()` и функции форматирования `format_value()`.

5.1 Разработка схем алгоритмов

Схема алгоритма функции, которая записывает блок, `editor_move_cursor()` приведена в приложении А.

Схема алгоритма функции, которая считывает данные блока, `format_value()` приведена в приложении Б.

5.2 Разработка алгоритмов

Функция `read_inode()` выполняет чтение структуры `inode` из файловой системы `ext2` по заданному номеру.

Передаваемые параметры:

`fs_info_t *fs_info` — указатель на структуру файловой системы;

`uint32_t inode_num` — номер индексного дескриптора для чтения;

`struct ext2_inode *inode` — указатель на структуру для хранения данных индексного дескриптора;

1) начало;

2) проверка передаваемых параметров. Если `inode_num` больше максимального количества индексных дескрипторов, выводится ошибка и переход к шагу 10, иначе переход к шагу 3;

3) определение группы индексного дескриптора. Вычисление номера группы `group`;

4) определение из дескриптора группы номера блока, в котором хранится таблица индексных дескрипторов группы `inode_table_block`;

5) определение индекса дескриптора внутри его группы `index`;

6) определение фактического смещения индексного дескриптора в файловой системе `off_t offset`;

7) чтение структуры `ext2_inode` из смещённого адреса в файловой системе `bytes_read`;

8) проверка успешности чтения `sizeof(struct ext2_inode)` байт. Если ошибка, то переход к шагу 10;

9) успешное завершение. Если ошибок нет, то `return 0`;

10) конец.

Функция `editor_init()` инициализирует контекст редактора, выделяя память и настраивая его параметры.

Передаваемые параметры:

`fs_info_t *fs_info` — указатель на структуру с информацией о файловой системе.

1) начало;

2) выделение памяти под структуру `editor_context_t`. Вызов `calloc()` для выделения и обнуления памяти;

3) проверка успешности выделения памяти. Если `calloc()` вернул `NULL`, то переход к шагу 14 (ошибка), иначе переход к шагу 4;

4) инициализация указателя на файловую систему. Присвоение переменной `ctx->fs_info = fs_info`;

5) установка смещения текущей позиции редактора. Присвоение переменной `ctx->current_offset = 0`;

6) вычисление и установка размера буфера `ctx->buffer_size = fs_info->block_size * 2`;

7) выделение памяти для буфера, в котором будет храниться содержимое редактируемых данных. Вызов `malloc()` для `ctx→buffer`;

8) проверка успешности выделения буфера. Если `malloc()` вернул `NULL`, освободить ранее выделенную память и переход к шагу 14, иначе переход к шагу 9;

9) настройка ширины области отображения в байтах. Присвоение переменной `ctx->bytes_per_row = 16`;

10) определение количества строк в видимой области просмотра редактора `ctx->view_rows = 16`;

11) установка координаты курсора по `X` `ctx->cursor_x = 0`;

12) установка координаты курсора по `Y` `ctx->cursor_y = 0`;

13) установка режима `ctx->editing_mode = false`;

14) успешное завершение `return ctx`;

15) конец.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

6.1 Назначение программы

Разработанная программа представляет собой инструмент для анализа и редактирования файловых систем, работающий с низкоуровневыми структурами хранения данных. Она позволяет пользователю изучать параметры суперблока, управлять блоками и индексными дескрипторами, просматривать битовые карты распределения памяти и редактировать данные на уровне файловой системы.

Основное предназначение программы — диагностика, исследование и модификация файловых систем формата `ext2`, `ext3` и `ext4`. Она будет полезна системным администраторам, разработчикам программного обеспечения, изучающим файловые системы, а также пользователям, занимающимся восстановлением данных.

6.2 Запуск программы

Для запуска программы необходимо использовать терминал `Linux` или `Windows` (с поддержкой `ncurses`).

Команда запуска выглядит так:

```
$sudo ./fs_analyzer /dev/sda1
```

Важно: перед началом работы рекомендуется создать резервную копию устройства или образа файловой системы, чтобы избежать потери данных в случае внесения некорректных изменений.

6.3 Основные команды программы

После запуска программа переходит в главное меню. Управление осуществляется с помощью клавиш на клавиатуре.

Выбор режима работы осуществляется по клавишам:

1) «`1`» – переход в режим анализатора файловой системы;
2) «`2`» – переход в режим просмотра блоков;
3) «`3`» – переход в режим просмотра индексных дескрипторов;
4) «`4`» – редактирование суперблока;

Навигация осуществляется по клавишам:

1) «`G`» – переход к указанной группе;
2) «`ESC`» – возврат в главное меню;
3) «`E`» – редактирование текущего блока;
4) «`TAB`» – переключение режимов просмотра и редактирования;
5) «`S`» – сохранение изменений;
6) «`F1`» – вызов справки.

Навигация по группам блоков, индексных дескрипторов осуществляется с помощью клавиш-стрелок.

6.4 Рекомендации по пользованию

Перед началом работы с программой необходимо понимать, что любые изменения файловой системы могут привести к потере данных. Поэтому перед внесением правок рекомендуется делать резервное копирование или создать и смонтировать образ диска самостоятельно.

Пример создания и монтирования образа диска:

1) `$fallocate -l 100M disk.img` – создание образа;
2) `$mkfs.ext4 disk.img` – форматирование в ext4;
3) `$sudo mkdir -p /mnt/test_fs` – создание точки монтирования;
4) `$sudo mount -o loop disk.img /mnt/test_fs` – монтирование;
5) `$mount | grep /mnt/test_fs` – проверка монтирования.

При возникновении ошибок создания образа рекомендуется удалить его и повторить алгоритм создания.

Запуск программы для работы с файловой системой необходимо осуществлять с правами суперпользователя. Это необходимо для получения прав доступа и возможности редактирования.

Навигация между структурами должна выполняться осознанно, особенно при переходе к указанным блокам или индексным дескрипторам. Важно следить за изменениями, вносимыми в файловую систему, чтобы избежать некорректного распределения данных.

При завершении работы рекомендуется использовать штатную команду `Q`, чтобы корректно закрыть дескрипторы файлов, освободить память и вернуть терминал в нормальное состояние. В случае работы со смонтированным образом, следует выполнить размонтирование.

Использование программы требует осознанного подхода и знаний основ работы с файловыми системами. Соблюдение данных рекомендаций поможет минимизировать риск повреждения данных и сделать процесс анализа и редактирования безопасным.

7 РЕЗУЛЬТАТЫ РАБОТЫ

Основной анализатор файловой системы. Пример работы программы:

```
Filesystem Information:
======================
Filesystem Type: ext4
Filesystem Size: 100.00 MB
Block Size: 1024 bytes
Inode Size: 256 bytes
Blocks Count: 102400
Free Blocks: 90284
Inodes Count: 25584
Free Inodes: 25568
Block Groups: 13
Blocks Per Group: 8192
Inodes Per Group: 1968
Block Group #0 Information:
============================
Block Bitmap: 259
Inode Bitmap: 272
Inode Table: 285
Free Blocks Count: 1465
Free Inodes Count: 1953
Used Directories Count: 2
```

Пример вывода информации об одном индексном дескрипторе:

```
Inode Browser - Inode 8 of 25583
==============================
Inode Status: Allocated
Mode: 0100600
File Type: Regular File
Size: 4.00 MB
Links: 1
UID: 0
GID: 0
Access Time: 2025-05-17 13:47:57
Modify Time: 2025-05-17 13:47:57
Change Time: 2025-05-17 13:47:57
Direct Blocks:
  [0]: 127754
  [1]: 4
  [2]: 0
  [3]: 0
  [4]: 4096
  [5]: 49153
  [6]: 0
  [7]: 0
Indirect Blocks:
  Single: 0
  Double: 0
  Triple: 0
```

Пример редактирования индексного дескриптора:

```
0000: 80 81 00 00 00 00 40 00 5d 69 28 68 5d 69 28 68 | ......@.]i(h]i(h
0010: 5d 69 28 68 00 00 00 00 00 00 01 00 00 20 00 00 | ]i(h......... ..
0020: 00 00 08 00 00 00 00 00 0a f3 01 00 04 00 00 00 | ................
0030: 00 00 00 00 00 00 00 00 00 10 00 00 01 c0 00 00 | ................
0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0070: 00 00 00 00 00 00 00 00 00 00 00 00 ea 27 00 00 | .............'..
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
00a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
00d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
00e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
00f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..............…
```

Пример редактирования блока:

```
Block Browser - Block 2 of 102399
==============================
Block Status: Allocated
Block Type: Group Descriptor
Block Size: 1024 bytes
Block Group: 0
Block in Group: 2
Block Data (first 256 bytes):
0000: 03 01 00 00 10 01 00 00 1D 01 00 00 B9 05 A1 07  |................
0010: 02 00 04 00 00 00 00 00 35 7E 1A 3B A1 07 9A 8B  |........5~.;....
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................
0030: 00 00 00 00 00 00 00 00 B9 87 68 7D 00 00 00 00  |..........h}....
0040: 04 01 00 00 11 01 00 00 09 03 00 00 FC 1E AF 07  |................
0050: 01 00 04 00 00 00 00 00 C6 45 77 E9 AF 07 6C B1  |.........Ew...l.
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................
0070: 00 00 00 00 00 00 00 00 D8 73 A3 5A 00 00 00 00  |.........s.Z....
0080: 05 01 00 00 12 01 00 00 F5 04 00 00 00 20 B0 07  |............. ..
0090: 00 00 07 00 00 00 00 00 00 00 00 00 B0 07 D7 2F  |............../
```

При запуске программы инструмент valgrind обнаруживает утечки памяти в трёх блоках:

```
==5614== HEAP SUMMARY:
==5614==     in use at exit: 125 bytes in 3 blocks
==5614==   total heap usage: 524 allocs, 521 frees, 932,417
bytes allocated
==5614==
==5614== 6 bytes in 1 blocks are still reachable in loss record
1 of 3
==5614==    at 0x4841866: malloc (vg_replace_malloc.c:446)
==5614==    by 0x4A8C51E: strdup (in /usr/lib64/libc.so.6)
==5614==    by 0x48BE211: ??? (in /usr/lib64/libtinfo.so.6.5)
```

```
==5614==    by 0x48C1124: _nc_first_db (libtinfo.so.6.5)
==5614==    by 0x48CF652: _nc_read_entry2 (libtinfo.so.6.5)
==5614==    by 0x48CB627: _nc_setup_tinfo (libtinfo.so.6.5)
==5614==    by 0x48CB7D5: _nc_setupterm (libtinfo.so.6.5)
==5614==    by 0x4898863: newterm_sp (libncursesw.so.6.5)
==5614==    by 0x4898D14: newterm (libncursesw.so.6.5)
==5614==    by 0x4038E5: main (main.c:57)
==5614==
==5614== 40 bytes in 1 blocks are still reachable in loss
record 2 of 3
==5614==    at 0x4849133: calloc (vg_replace_malloc.c:1675)
==5614==    by 0x48C0CC3: _nc_first_db (libtinfo.so.6.5)
==5614==    by 0x48CF652: _nc_read_entry2 (libtinfo.so.6.5)
==5614==    by 0x48CB627: _nc_setup_tinfo (libtinfo.so.6.5)
==5614==    by 0x48CB7D5: _nc_setupterm (libtinfo.so.6.5)
==5614==    by 0x4898863: newterm_sp (libncursesw.so.6.5)
==5614==    by 0x4898D14: newterm (libncursesw.so.6.5)
==5614==    by 0x4038E5: main (main.c:57)
==5614==
==5614== 79 bytes in 1 blocks are still reachable in loss
record 3 of 3
==5614==    at 0x4841866: malloc (vg_replace_malloc.c:446)
==5614==    by 0x48C0C08: _nc_first_db (libtinfo.so.6.5)
==5614==    by 0x48CF652: _nc_read_entry2 (libtinfo.so.6.5)
==5614==    by 0x48CB627: _nc_setup_tinfo (libtinfo.so.6.5)
==5614==    by 0x48CB7D5: _nc_setupterm (libtinfo.so.6.5)
==5614==    by 0x4898863: newterm_sp (libncursesw.so.6.5)
==5614==    by 0x4898D14: newterm (libncursesw.so.6.5)
==5614==    by 0x4038E5: main (main.c:57)
==5614==
==5614== LEAK SUMMARY:
==5614==    definitely lost: 0 bytes in 0 blocks
==5614==    indirectly lost: 0 bytes in 0 blocks
==5614==      possibly lost: 0 bytes in 0 blocks
==5614==    still reachable: 125 bytes in 3 blocks
==5614==         suppressed: 0 bytes in 0 blocks
==5614==
==5614== For lists of detected and suppressed errors, rerun
with: -s
==5614== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0
from 0)
```

Обнаруженные утечки означают, что память не была освобождена, но указатели на нее все еще доступны при завершении программы. Все утечки связаны с инициализацией ncurses, что указывает на внутренние особенности данной библиотеки и не требуют исправления в пользовательском коде.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были углублены знания о принципах работы файловых систем форматов `ext2`, `ext3` и `ext4`, организации хранения данных и метаданных, а также методах их анализа и модификации. В рамках проекта освоены технологии работы с бинарными структурами, обработка битовых карт распределения блоков и индексных дескрипторов, а также механизмы низкоуровневого редактирования файловых систем. Использование библиотеки `ncurses` позволило разработать удобный консольный интерфейс для взаимодействия с файловыми системами в терминальной среде.

Результатом работы стала программа, обеспечивающая анализ и редактирование структур файловых систем, включая просмотр суперблока, управление группами блоков, анализ и модификацию индексных дескрипторов, а также работу с бинарными данными на уровне блоков.

Несмотря на завершённость проекта, программа обладает значительным потенциалом для дальнейшего развития. В перспективе возможно расширение функциональности за счёт интеграции механизмов анализа целостности файловых систем, добавления поддержки других форматов хранения данных, а также оптимизации работы с крупными файловыми системами. Дополнительным направлением развития может стать расширение возможностей редактирования и автоматизация анализа метаданных.

Работа над проектом позволила углубить понимание структуры файловых систем, оценить важность архитектурных решений и принципов модульного программирования, а также усовершенствовать навыки низкоуровневой обработки данных. Полученные знания и практический опыт могут быть полезны для дальнейшего изучения системного программирования и разработки инструментов диагностики, восстановления и управления файловыми системами.

Разработанный инструмент демонстрирует возможность эффективной работы с файловыми системами в рамках консольного интерфейса, предоставляя пользователям детализированный анализ и гибкие инструменты редактирования. Дальнейшее развитие проекта может привести к созданию мощного специализированного редактора метаданных, востребованного среди системных администраторов, исследователей и разработчиков, работающих с внутренними структурами файловых систем.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] Лав Р. Linux. Системное программирование. 2-е изд. — СПб.: Питер, 2014.

[2] Робачевский А. Программирование на языке Си в UNIX. Издательство: БХВ-Петербург, 2015.

[3] Стивенс Р., Раго С. UNIX. Профессиональное программирование, 2-е издание. — СПб.: Символ-Плюс, 2007.

ПРИЛОЖЕНИЕ А

(обязательное)

Схема алгоритма функции `editor_move_cursor()`

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема алгоритма функции `format_value()`

ПРИЛОЖЕНИЕ В

(обязательное)

Ведомость документов