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 курсовой работе
на тему
НИЗКОУРОВНЕВЫЙ РЕДАКТОР БЛОЧНЫХ УТСРОЙСТВ УРОВНЯ СЕКТОРОВ(NCURSES)

БГУИР КП 1-40 02 01 108 ПЗ

Студент: Гулевич В.А.
Руководитель: Игнатович А.О.

Минск 2024

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет: КСиС. Кафедра: ЭВМ.
Специальность: 40 02 01 «Вычислительные машины, системы и сети».
Специализация: 400201-01 «Проектирование и применение локальных компьютерных сетей».

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Б.В. Никульшин
«____» _____ 2024 г.

ЗАДАНИЕ
по курсовому проекту студента
Гулевича Владислава Александровича

1 Тема проекта: «Низкоуровневый редактор блочных устройств уровня секторов(ncurses)»
2 Срок сдачи студентом законченного проекта: 10 мая 2024 г.
3 Исходные данные к проекту: нет.
4 Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Титульный лист.
Реферат.
Введение.
1. Постановка задачи.
2. Обзор методов и алгоритмов решения поставленной задачи.
3. Обоснование выбранных методов и алгоритмов.
4. Описание программы для программиста.
5. Описание алгоритмов решения задачи.
6. Руководство пользователя.
Заключение.
Список использованных источников.
Приложения.

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов курсового проекта	Объем этапа, %	Срок выполнения этапа	Примечания
Выбор темы курсового проекта	5	17.02 – 01.03	
Начальный этап ПЗ	30	01.03 – 01.04	
Основная часть кода	50	01.04 – 01.05	
Оформление пояснительной записки и графического материала	15	01.05 – 10.05	с выполнением чертежа
Защита курсового проекта		28.05 – 10.06	

Дата выдачи задания: 20.02.2024 г.

Руководитель А.О. Игнатович

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ _____

РЕФЕРАТ

Курсовой проект предоставлен следующим образом. Чертежный материал: 2 листа формата А4. Пояснительная записка: 53 страницы, 17 рисунков, 4 литературных источника, 3 приложения.

Ключевые слова: блок, сектор, байт, память, смещение, `ncurses`, окно, панель, файл.

Объектом разработки является программа, которая позволяет открыть блок памяти и отредактировать байты в нем.

Целью разработки данной программы является редактирование памяти напрямую без использования файловой системы.

Для выполнения цели была использована библиотека `ncurses.h` для реализации интерфейса и функции работы с памятью, чтобы получать корректное смещение, размер блоков и секторов.

В результате были получены блоки и секторы памяти и отредактированы напрямую, не используя файловую систему. Так же был реализован простой и понятный пользовательский интерфейс.

Данное приложение можно использовать для редактирования содержимого файлов или для редактирования памяти блочных устройств.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данного курсового проекта является разработка низкоуровневого редактора блочного устройства уровня секторов с использованием библиотеки `ncurses`. В современных операционных системах(ОС) блочные устройства играют важную роль в работе с данными, так как они предоставляют доступ к физическим секторам на диске. Редактор блочного устройства предоставляет возможность прямого доступа и редактирования данных на физическом уровне блочных устройств, основываясь на работе с отдельными секторами данных.

Такой редактор позволяет осуществлять чтение и запись данных на уровне секторов, минуя файловую систему и обрабатывая данные непосредственно на физическом носителе, таком как жесткий диск или твердотельный накопитель. Это означает, что данные могут быть прочитаны или записаны без учета структуры файловой системы или файловых атрибутов.

Данный тип редактора обычно используется системными администраторами, разработчиками операционных систем и специалистами по восстановлению данных для различных целей, включая работу с поврежденными файловыми системами, исследование и анализ данных и разработку и отладку драйверов устройств.

К очевидным минусам можно отнести сложность работы с устройством и потенциальные риски, так как неправильное редактирование или действия с данными на физическом уровне могут привести к потере данных или повреждению устройств. Так же из-за того, что редактор ориентирован на работу с секторами данных напрямую, он может быть ограничен в функциональности, поскольку не предоставляет полного набора возможностей, доступных в более высокоуровневых инструментах, таких как файловые менеджеры или инструменты анализа данных.

Для реализации редактора блочного устройства мы выбрали библиотеку `ncurses`. `Ncurses` предоставляет набор функций и макросов, которые позволяют программистам управлять выводом текста, обрабатывать клавиатурные и мышиные события, а также создавать интерактивные элементы интерфейса, такие как кнопки, текстовые поля, меню, окна и прогресс-индикаторы. Библиотека `ncurses` обеспечивает переносимость кода между различными операционными системами и терминалами, что позволяет создавать кросс–платформенные текстовые интерфейсы. Одной из

ключевых особенностей `ncurses` является возможность работы с текстом в режиме символов, а не только в режиме строк, что позволяет создавать более гибкие и интерактивные интерфейсы. Библиотека также обладает возможностями цветового оформления и управления курсором, что позволяет создавать более привлекательные и информативные пользовательские интерфейсы.

Разработка низкоуровневого редактора блочного устройства требует понимания структуры и организации данных на физическом носителе, а также знания основных операций чтения, записи и модификации данных на уровне секторов. Будут реализованы основные функции редактирования, такие как чтение и запись секторов, перемещение по диску, поиск и замена данных.

В процессе выполнения курсового проекта мы надеемся углубить наши знания в области низкоуровневого программирования, а также приобрести практические навыки разработки текстового пользовательского интерфейса с использованием библиотеки `ncurses`. Мы также рассчитываем на получение полезного опыта работы с блочными устройствами и анализа данных на низком уровне.

1 ПОСТАНОВКА ЗАДАЧИ

В качестве языка был выбран язык программирования C. Основные достоинства этого языка для реализации этого проекта – это близость к аппаратуре, так как C является низкоуровневым языком программирования, что обеспечивает близкое соответствие кода программы аппаратуре компьютера. Другим достоинством языка является эффективность и производительность, что важно для работы с большими объемами данных и выполнения операций непосредственно на уровне секторов. Также C дает доступ к системным ресурсам и позволяет работать с памятью.

В проекте нужно реализовать удобный и понятный пользовательский интерфейс с необходимыми кнопками выбора действия, основные функции низкоуровневого редактора блочных устройств уровня секторов, такие как чтение секторов и запись в них, редактирование данных, поиск, копирование и перемещение, изменение размеров секторов и создание, и удаление разделов.

Для реализации графического пользовательского интерфейса будет использована библиотека `ncurses`. Она предоставляет набор функций для создания текстового пользовательского интерфейса в терминале. `Ncurses` позволяет управлять экраном, обрабатывать ввод с клавиатуры и события мыши, создавать окна и панели, что позволяет организовать различные части интерфейса и управлять их расположением.

Данный список средств позволяет реализовать все задачи, поставленные для курсового проекта.

2 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

В данном разделе приводится описание используемых структур, алгоритмов и подходов к проектированию, а также анализ аналогов приложения.

2.1 Анализ существующих аналогов

Тема курсового проекта была выбрана в первую очередь для углубления знаний по языку С а также получения знаний в взаимодействии с данными, поэтому моей целью не является разработать конкурентоспособный продукт. Тем не менее, чтобы создать корректно работающее приложение, нужно иметь представление о существующих аналогах, об их недостатках, преимуществах и реализованных внутри функциях.

2.1.1 HxD

HxD – это редактор шестнадцатеричных данных для ОС Windows. Он предоставляет возможность просмотра и редактирования данных на низком уровне, включая работу с блочными устройствами уровня секторов. В программе реализованы алгоритмы чтения и записи, поиска, сравнения файлов, шифрования и дешифрования и алгоритм отображения данных. Для разработки был использован WinAPI для взаимодействия с ОС Windows, Microsoft Foundation Classes для разработки пользовательского интерфейса и иные библиотеки. Интерфейс HxD изображен на рисунке 1.1.

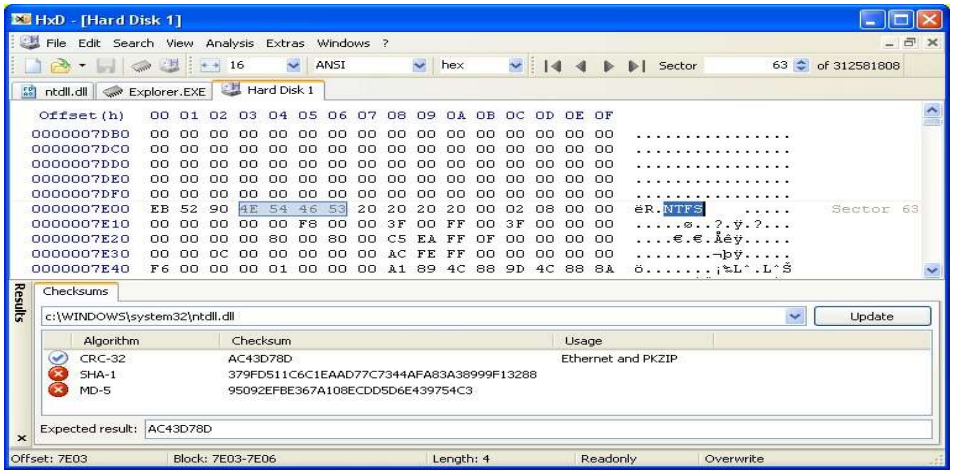

Рисунок 2.1 — Пользовательский интерфейс программы HxD

Из очевидных плюсов приложения можно отметить простой и понятный интерфейс, содержащий исчерпывающую информацию о блоках, секторах, смещении. Так же имеется 2 окна, в которых выводятся байты в 16-ричной системе и в виде ASCII. Более того имеются полезные вкладки с помощью пользователю.

HxD разработан Майелем Герцем в 2003 году. Приложение написано на языке C++ для ОС Windows XP, 2003, Vista, 7, 8 или 10. Последняя версия была выпущена в 2021 году.

2.1.2 wxHexEditor

wxHexEditor – это редактор шестнадцатеричных данных с графическим пользовательским интерфейсом. wxHexEditor использует 64–битный файловый дескриптор, что позволяет ему поддерживать файлы размером до 2^64 байт. Так же он хорошо оптимизирован для эффективной обработки больших объемов данных. Хорошей оптимизации помогли добиться технологии ленивой загрузки данных, буферизации данных и кэширование отображения. Интерфейс wxHexEditor изображен на рисунке 1.2.

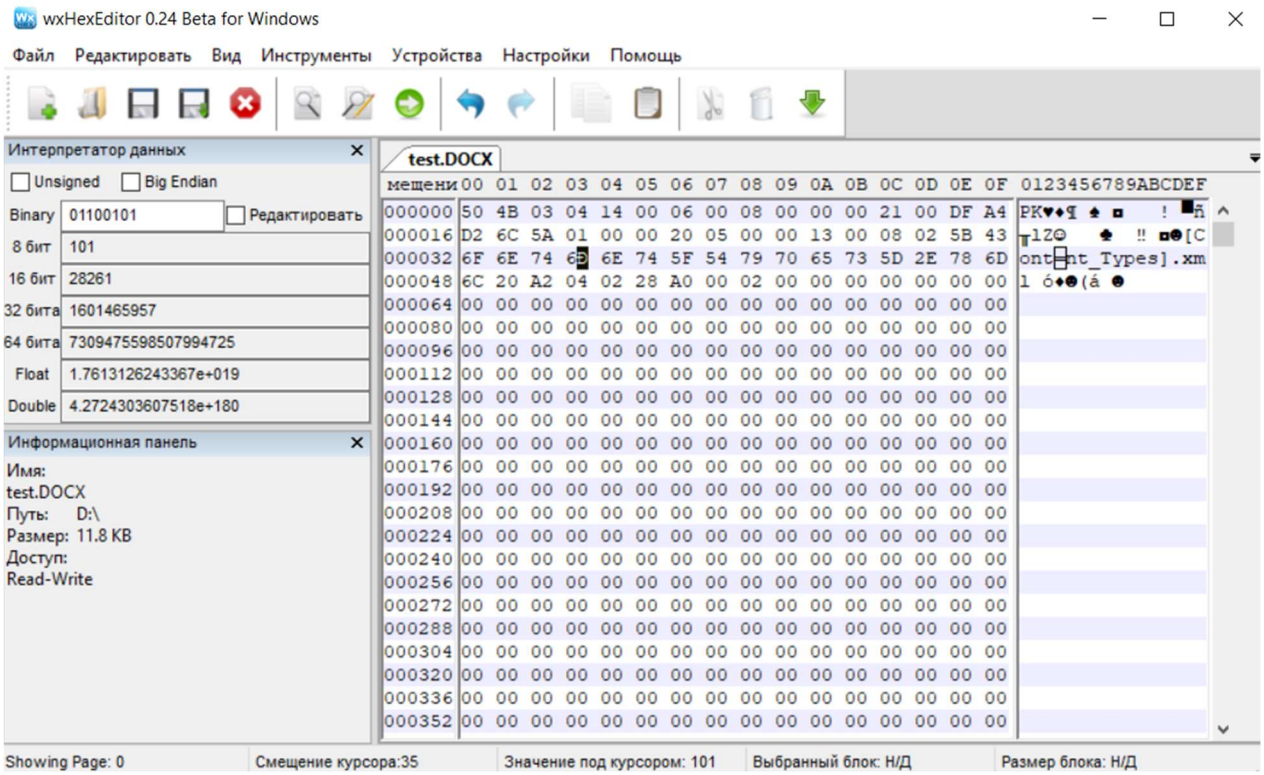

Рисунок 2.2 — Пользовательский интерфейс программы wxHexEditor

В этом приложении можно отметить полезную информационную панель с именем файла, его путем размером и типом доступа. Так же технология ленивой загрузки может значительно оптимизировать приложение в консоли.

wxHexEditor был разработан Андреасом Шретером в 2006 году. Приложение написано на C++ и поддерживает ОС Windows, Linux и MacOS. Последняя версия была выпущена в 2012 году, и сейчас приложение не поддерживается.

2.1.3 DiskGenius

DiskGenius отличается от двух предыдущих аналогов. Он является мощным программным обеспечением для управления дисками и восстановления данных. Он предоставляет широкий спектр функций для работы с жесткими дисками, разделами и файловыми системами. DiskGenius включает в себя шестнадцатеричный редактор, имеет низкоуровневый доступ к блочным устройствам и имеет алгоритмы сканирования и восстановления данных. Это приложение не позволяет редактировать блоки, однако оно предоставляет информацию о типе файловой системы, общий размер, количество блоков и секторов и их размер. Интерфейс DiskGenius изображен на рисунке 1.3.

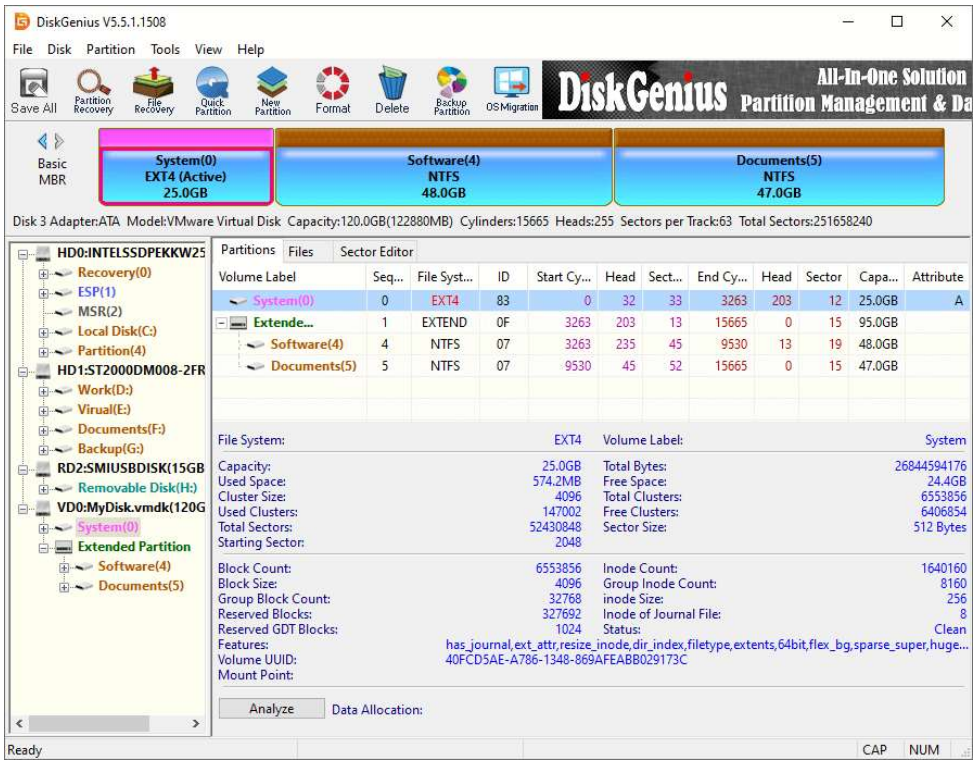

Рисунок 2.3 — Пользовательский интерфейс программы DiskGenius

DiskGenius был разработан Eassos Ltd. Он был выпущен в 2005 году и с тех пор продолжает обновляться. Приложение написано на C++ и поддерживает ОС Windows.

2.2 Текстовый пользовательский интерфейс

Для создания пользовательского интерфейса был выбран текстовый пользовательский интерфейс. Текстовый пользовательский интерфейс (TUI) – это форма пользовательского интерфейса, которая использует только текстовый ввод и вывод для взаимодействия с пользователем. В отличие от графического пользовательского интерфейса, который использует графические элементы для отображения интерфейса, текстовый использует исключительно набор буквенно-цифровых символов для представления информации. Поэтому текстовый пользовательский интерфейс требует значительно меньше ресурсов, чем графический.

Текстовый пользовательский интерфейс обычно работает в командной строке или терминале, то есть его можно реализовать на любом устройстве. Более того, консоль предоставляет наибольшую гибкость и управляемость. На рисунке 2.4 предоставлена реализация текстового пользовательского интерфейса двухпанельного файлового менеджера в консоли.

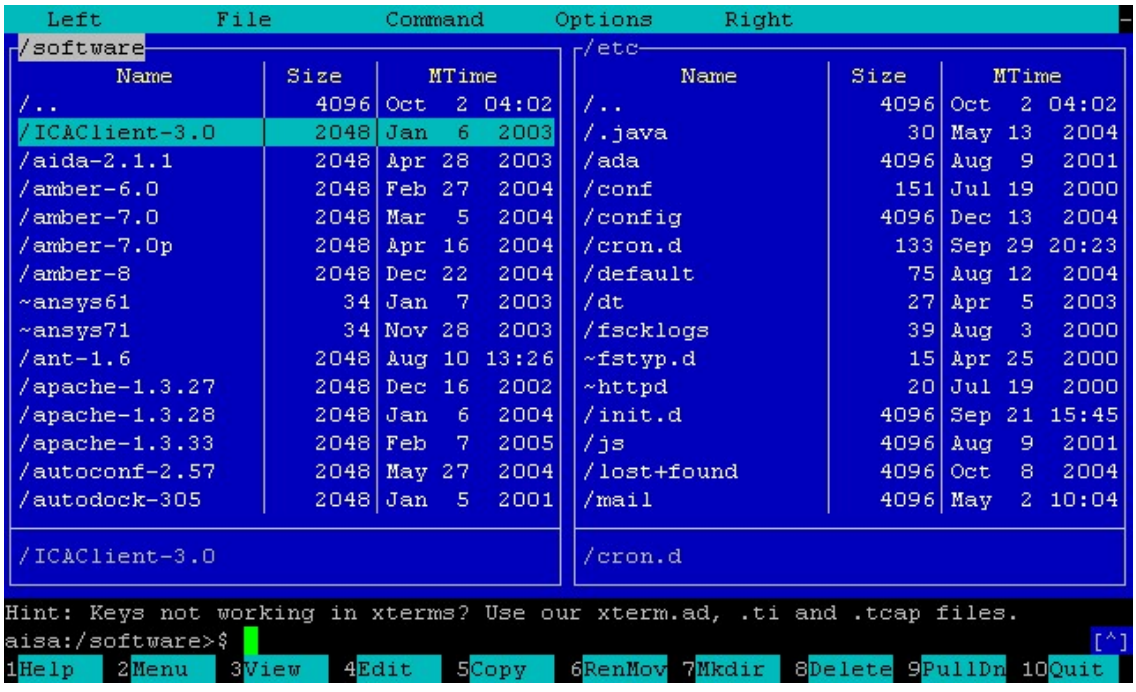

Рисунок 2.4 – интерфейс двухпанельного файлового менеджера

2.3 Использование библиотеки mmap (memory map)

Для работы с памятью используется библиотека mmap. Она необходима для выполнения основной задачи проекта, то есть чтение и редактирования блоков и секторов. Библиотека является частью стандартной библиотеки операционной системы и представляется ядром операционной системы Linux.

Библиотека предоставляет возможность отображения файлов непосредственно в память компьютера. Это означает, что файл может быть доступен как массив байтов, которые можно читать и записывать напрямую в память, минуя операции чтения и записи ввода-вывода. Библиотека mmap предоставляет функции для создания и управления отображенными областями памяти.

Преимущество использования mmap заключается в том, что это может быть очень эффективным способом работать с большими файлами или файлами, к которым требуется частый доступ. Вместо того, чтобы загружать весь файл в память, mmap позволяет работать с файлом кусочно, загружая только необходимые части в память по мере необходимости.

Основными преимуществами библиотеки mmap можно отметить отображение файлов в память, разделяемая память для обмена данными между несколькими процессами, контроль доступа, который позволяет задать различные режимы доступа к отображенной области памяти, и управление памятью.

2.4 Использование алгоритма ленивой загрузка

Алгоритм ленивой загрузки – это стратегия загрузки данных или ресурсов, при которой они загружаются только по мере необходимости, а не заранее. Этот подход позволяет уменьшить время загрузки и уменьшить потребление ресурсов, так как загружаются только те данные, которые действительно используются.

Основная идея ленивой загрузки состоит в том, чтобы отложить загрузку данных до момента, когда они действительно понадобятся. Вместо загрузки всех данных заранее, приложение загружает только небольшую часть данных или только те данные, которые необходимы для начала работы. При необходимости загружаются дополнительные данные по мере продвижения выполнения программы или по запросу пользователя.

Алгоритм ленивой загрузки имеет такие преимущества, как улучшение

производительности, экономия ресурсов и улучшение масштабируемости, так как объем данных велик и неизвестен заранее. Однако, у этого алгоритма имеются и свои недостатки, такие как задержка загрузки. При использовании ленивой загрузки данные загружаются по мере необходимости, что может привести к небольшой задержке при первом доступе к данным. Если данные требуются немедленно, это может привести к ощутимому времени ожидания. Другой недостаток это управление состоянием: в случае ленивой загрузки требуется управление состоянием, чтобы определить, какие данные уже были загружены и какие еще нужно загрузить, что может потребовать дополнительной логики и сложности в программе.

В проекте происходит работа с огромным количеством памяти, а пользователь может увидеть только отображение 336 байт, поэтому такая технология просто необходима для оптимизации приложения.

2.5 Работа с библиотекой `ncurses`

Для разработки приложения используется библиотека `ncurses`. Она необходима для создания пользовательского интерфейса с помощью текста в консоли. Библиотека написана на языках C и Ада. Она предоставляет разработчикам широкий спектр функций и инструментов для создания интерактивных консольных приложений, которые могут быть запущены в терминале.

Основное взаимодействие библиотеки с терминалом заключается в использовании структуры данных `WINDOW`, которая нужна для создания и отображения окна в терминале. Окно хранит информацию о его размере, координатах, текстовом наполнении и атрибутах, которые могут быть использованы для изменения цвета фона или текста. С помощью указателей на структуру `WINDOW` и функций библиотеки можно создавать окна или управлять ими. Для начала работы с окнами нужно инициализировать главное окно с помощью функции `stdscr()`, указатель на которое будет храниться в глобальной переменной `stdscr`.

Окно представляет из себя буфер, в котором хранятся данные, отображаемые в окне. Также `ncurses` предоставляет возможность создания иерархии окон, когда окна используют одну и ту же память, что позволяет оптимизировать приложение. Имеется возможность изменения размера окна и его местоположения, что позволяет предусмотреть реакцию программы на изменение размеров терминала и переопределить пользовательский

интерфейс под новые размеры. Текущие размеры терминала отслеживаются с помощью глобальных переменных `COLS` и `LINE`, которые хранят размер терминала. Система буферизации позволяет не обновлять весь экран при каждом малейшем изменении, а с помощью специальных функций отображать изменения на экране в нужный момент.

`Ncurses` поддерживает форматированный вывод данных, который включает в себя изменение цвета фона и текста, изменение атрибутов символов: толщина, мерцание, курсив и т.д. Для хранения цвета и атрибутов символа используется специальный тип данных предоставляемый `ncurses chtype`, который представляет собой 8-битную маску, содержащую кодировку символа и его атрибутов и цветов.

Так же библиотека включает в себя функции для обработки нажатий мыши, клавиатуры или сенсорной панели. Более того, имеется возможность чтения специальных клавиш клавиатуры, таких как F1 или стрелок, и возвращать код нажатой клавиши, что удобно при реализации меню или при реализации приложения, не используя обработку нажатий мыши. Эти функции могут быть использованы для реализации пользовательского ввода в приложении. Более того, `ncurses` предоставляет механизмы для обработки ошибок ввода, что обеспечивает корректную реакцию на неправильный ввод пользователя.

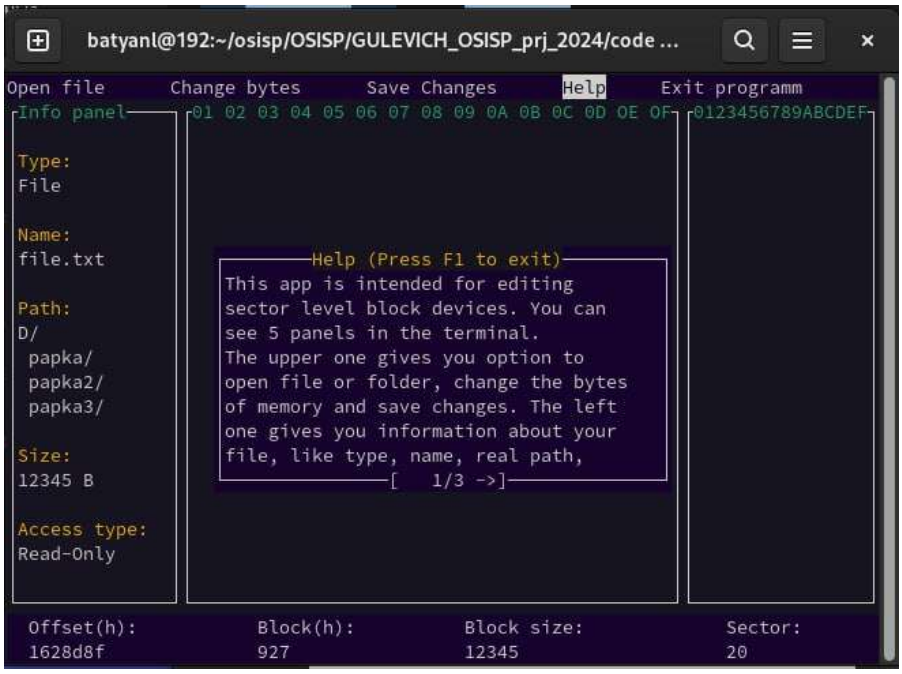

Рисунок 2.5 – создание окон с помощью `ncurses`

На рисунке 2.5 предоставлен интерфейс, созданный с помощью библиотеки ncurses, в котором было реализовано наложение окон друг на друга, использование атрибутов и создание нескольких окон в разных частях экрана.

2.6 Работа с `form.h`

Form является дополнительной библиотекой предназначенной для создания и управления формами в текстовом пользовательском интерфейсе. Она предоставляет набор функций и структур данных для создания, отображения и обработки форм в терминале. Основной элемент библиотеки form это структура FORM, которая представляет собой форму в терминале. Каждая форма содержит набор полей представленными структурой FIELD, которые определяют вводимые пользователем данные. Поля формы представляют собой элементы, с которыми пользователь может взаимодействовать, вводя данные. Форма может любое число полей, а также эти поля могут состоять из нескольких страниц, что позволяет обеспечить пользователю ввод данных в любом формате. Структура FORM:

```
Typedef struct formnode
{
   unsigned short status;
   short rows;
   short cols;
   int currow;
   int curcol;
   int toprow;
   int begincol;
   short maxfield;
   short maxpage;
   short curpage;
   Form_Options opts;
   Window* win;
   Window* sub;
   Window* w;
   FIELD** field;
   FIELD* current;
```

```
    _PAGE* page;
    void* usrptr;
    void (*forminit)(struct formnode*);
    void (*formterm)(struct formnode*);
    void (*fieldinit)(struct formnode*);
    void (*fieldterm)(struct formnode*);
}FORM;
```

Поддерживается позиционирование по полям ввода с помощью мыши или клавиатуры. Библиотека поддерживает удобное взаимодействие ввода в различных окнах, позволяя установить родительское окно, от состояния которого будет зависеть форма, и дочернее окно, в котором будут выводится поля для ввода. `Form.h` предоставляет инструменты для указания точно местоположения полей для ввода.

Также библиотека предоставляет расширенные функции для ввода символов в кодировке `UTF16`, что делает возможным ввод русских и специальных символов. `Form.h` поддерживает различные методы валидации ввода, имеется возможность устанавливать ограничения на вводимые пользователем данные. Например, вы можете указать минимальное и максимальное допустимое значение для числового поля, ограничить ввод различных символов, установить маску для текстового поля или создать поле для ввода с собственными правилами ввода, также имеется поддержка регулярных выражений для проверки ввода строк. Если введенные данные не соответствуют установленным ограничениям или не проходят проверку, библиотека `form.h` позволяет обрабатывать ошибки ввода. Это может включать в себя вывод сообщений об ошибках или блокировку продолжения ввода до тех пор, пока данные не будут исправлены.

Библиотека `form.h` также предоставляет механизм управления фокусом на поля формы. Это позволяет определять порядок ввода данных и управлять тем, какие поля активны для ввода в данный момент.

2.7 Работа с `panel.h`

Библиотека `panel.h` является частью библиотеки `ncurses` и предоставляет возможности для работы с панелями в текстовых пользовательских интерфейсах в терминале. Панели представляют собой специальные объекты, которые позволяют разработчикам управлять порядком

и видимостью окон внутри окна терминала. Использование панелей позволяет создавать сложные многокомпонентные интерфейсы, включающие в себя несколько окон, которые могут быть управляемыми и перемещаемыми независимо друг от друга.

В библиотеке panel.h предоставляются функции для создания, управления и манипулирования панелями, что делает ее мощным инструментом для разработки интерактивных и многокомпонентных пользовательских интерфейсов в терминале. Панели могут быть использованы для различных целей, таких как создание сложных макетов окон, реализация вкладок или панелей инструментов, а также для управления видимостью и перекрыванием окон в интерфейсе.

В этом контексте библиотека panel.h представляет собой важный инструмент для создания более функциональных, гибких и удобных пользовательских интерфейсов в текстовом режиме, что делает ее полезной для различных приложений, включая консольные игры, системные утилиты, текстовые редакторы и другие программы, работающие в терминале. PANEL – основная структура предоставляемая библиотекой для оптимизации работы многооконного приложения. Структура PANEL:

```
Typedef struct panel{
   WINDOW *win;
   struct panel *below;
   struct panel *above;
}PANEL;
```

Структуру PANEL можно рассмотреть, как стек, состоящий из всех других объектов PANEL. Пользователь видит только верхнюю панель, и может взаимодействовать только с ней. основная идея состоит в том, чтобы создать стопку перекрывающихся панелей и использовать библиотеку панелей для их корректного отображения. Существует функция, которая при вызове отображает панели в правильном порядке. Предусмотрены функции для скрытия или отображения панелей, перемещения панелей, изменения их размера и т.д. Проблема перекрытия решается библиотекой panels во время всех вызовов этих функций.

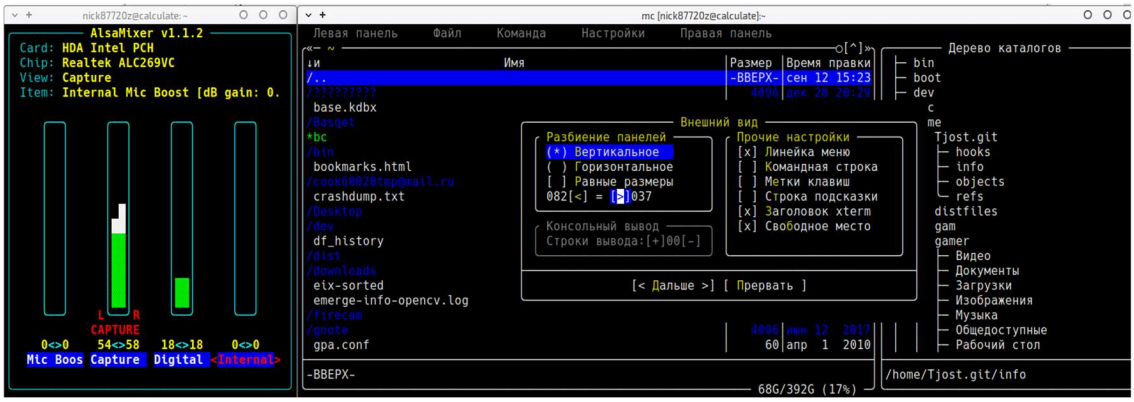

Рисунок 2.6 – приложение с перекрывающимися окнами

2.8 Работа с `menu.h`

Библиотека `menu.h` является дополнительной библиотекой для работы с `ncurses`. `menu.h` позволяет разработчикам легко создавать различные виды меню, включая вертикальные, горизонтальные, вкладочные и контекстные меню, которые могут содержать разнообразные элементы, такие как текст, радиокнопки, флажки, подсказки и другие. Меню также могут быть связаны с определенными действиями или функциями, что делает их полезными инструментами для реализации интерактивных функций в приложениях.

В библиотеке `menu.h` предоставляются функции для создания, управления и манипулирования меню, а также для обработки событий выбора опций пользователем. Это позволяет разработчикам легко интегрировать меню в свои приложения и обеспечить удобную и интуитивно понятную навигацию для пользователей.

В основе разработанной библиотеки лежит структура `MENU`, которая содержит различные настройки, такие как количество столбцов, строк, обработчик событий, формат вывода и цвет элементов. Структура поддерживает несколько режимов работы, выбор элементов с помощью мышки или сенсорной панели и клавиатуры. Предусмотрены функции для изменения размера меню, при изменении размера терминала, и обработчики для сокращенного вывода его элементов, если места в терминале слишком мало. Структура содержит указатели на родительское и дочернее окно, которые позволяют выводить данные в определенном месте экрана. Структура `MENU`:

```
typedef struct {
   ITEM **items;
   int item_count;
   WINDOW *win;
   int rows, cols;
   int curitem;
   int toprow;
   bool mark;
   bool posted;
   bool fore;
   bool pad;
   bool user_win;
   bool user_frame;
   bool user_grey;
   int marklen;
   int deswidth;
   int scale;
   int spc_desc;
   int ext_fld;
   struct ldat *ldat;
   void (*free_userptr)(void *);
   void (*pattern)(const char *);
   bool (*hook)(const struct ldat *, const char *);
   void *userptr;
} MENU;
```

Меню состоит из массива структур `ITEM`, которые содержат размер элемента и координаты в окне, а также обработчик событий при выборе пункта меню и дополнительные данные для форматированного вывода элемента, такие как цвет в обычном состоянии, цвет при выборе данного пункта, указательный символ при выборе пункта меню. Структура `ITEM`:

```
typedef struct {
   char *name;
   char *description;
   int index;
   bool visible;
```

```
    bool enabled;
    bool selected;
    bool value;
    void *userptr;
    void(*init)(struct _win_st*, struct _menu_item*);
    void (*term)(struct _win_st *);
    void (*hook)(struct _menu_item *);
} ITEM;
```

В библиотеке предусмотрены функции для создания столбцов разного размера, а также функции для определения названия столбцов. Размер столбцов можно указывать в виде определенного значения или в процентном соотношении. При изменении размера окна значения размеров таблиц пересчитываются.

Также предусмотрены функции для сокращенного вывода данных, которые хранятся с помощью указателя на функцию ABREVIATED, что позволяет указать собственную логику сокращения для каждого столбца. Для более удобной настройки и создания меню используется битовая маска, где каждый пункт настройки представляет отдельный бит. Структура битовой маски _SETTINGS_MENU:

```
typedef enum _SETTINGS_MENU {
    NONE_SPRT = 1,
    SPRT_ALL = 2,
    SPRT_INTERMEDIATE = 4;
    NON_DESIG_ITEMS = 8,
    DESIG_ITEMS = 16,
    STNDRT_COL_SIZE = 32,
    USER_COL_SIZE = 64,
    NON_COL_SIZE = 128,
    USE_COL_NAME = 256,
    NON_COL_NAME = 512,
    ALLIGMENT_CENTER = 1024,
    ALLIGMENT_LEFT = 2048
} SETTINGS_MENU;
```

Благодаря этому можно удобно определять определенные настройки для

каждого меню. Пример реализации меню с использованием библиотеки menu.h предоставлен на рисунке 2.7:

Рисунок 2.7 – пример реализации меню

3 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ

3.1 Текстовый пользовательский интерфейс

Текстовый пользовательский интерфейс (TUI) имеет свои преимущества относительно графического пользовательского интерфейса (GUI). TUI является более эффективным в использовании ресурсов системы, так как текстовый ввод и вывод требуют меньше вычислительной мощности и памяти, по сравнению с графическими элементами. Это важно для приложений, которые будут обрабатывать большие объемы информации. Также TUI имеет более простую структуру и основывается на текстовых командах.

TUI может быть быстрее для выполнения некоторых задач. Он может быть легко автоматизирован с помощью скриптов и командных файлов, что позволяет создавать мощные и гибкие рабочие процессы и автоматически выполнять задачи. Это делает его предпочтительным для системного администрирования и разработки программного обеспечения. Более того, ему не нужно тратить время на рендеринг графических элементов, в отличие от GUI.

С другой стороны, TUI ограничен текстовыми символами и не может передать информацию так наглядно, как может это сделать GUI. Также GUI обычно поддерживает мультизадачность, позволяя пользователям легко работать с несколькими приложениями или окнами одновременно.

3.2 Использование библиотеки `mmap (memory map)`

Для чтения информации из файлов или из памяти напрямую используется библиотека `mmap`. Она предоставляет интерфейс для отображения файлов непосредственно в память процессора, что позволяет работать с файлами, кка если бы они были массивами в памяти, обеспечивая быстрый доступ к данным и упрощая обмен информации между процессами.

Основным достоинством данной библиотеки является производительность. `Mmap` позволяет обращаться к данным в файле непосредственно из памяти, минимизируя операции чтения и записи. Это полезно в случае работы с большими файлами. Более того, работа с данной библиотекой помогает сэкономить память. Из-за того, что файлы отображаются в виртуальную память процесса, физическая память выделяется только для тех областей файла, которые реально были прочитаны или

записаны. Это позволяет работать с файлами, объем которых превышает допустимый размер оперативной памяти. Данные в отображаемом файле могут быть доступны как массив в памяти, то есть это обеспечивает простой доступ к данным, используя указатели и индексацию массивов.

Минусом данной библиотеки можно отметить ограниченную работу с текстовыми данными. `Mmap` предоставляет низкоуровневый доступ к данным в памяти. Поэтому его не рекомендуется использовать для работы с текстовыми данными, в связи с необходимостью в интерпретации байтов.

3.3 Использование алгоритма ленивой загрузки

Для чтения информации из файлов или из памяти использовался алгоритм ленивой загрузки. Основная идея состоит в том, чтобы загружать дополнительную информацию по мере необходимости. В приложении происходит работа с памятью и огромными файлами, а на экране отображается только 336 байт информации. Поэтому использование данного алгоритма необходимо для возможности работы с памятью и для оптимизации работы с файлами.

Из минусов можно отметить только возможную задержку при первом доступе, так как требуется время на загрузку и инициализацию.

3.4 Использование библиотеки `NCURSES`

Библиотека `ncurses` является инструментом для создания текстового пользовательского интерфейса в окне терминала. Основным достоинством библиотеки `ncurses` является возможность создания множества окон, на которые можно разбить интерфейс. При работе в определенном окне, данные будут перезаписываться только в пределах этого окна, а не в пределах всего терминала. Это позволяет создать довольно быстрый и оптимизированный пользовательский интерфейс.

Другим плюсом является портативность приложений, интерфейс которых написан на `ncurses`. Однако, интерфейс, написанный на `ncurses`, имеет привязку к размеру терминала. Поэтому, для переноса приложения на другие операционные системы нужно предусмотреть возможность адаптации интерфейса к изменениям размера окна терминала.

Также библиотека позволяет управлять пользовательским вводом, включая обработку нажатий клавиш клавиатуры или кнопок мыши.

Еще одним плюсом является поддержка использования цветов для текста и фона. Из очевидных минусов можно отметить ограничение визуального интерфейса. Она не может предоставить сложные графические элементы, такие как изображения или анимации.

`Ncurses` работает в терминале и полагается на его функциональность. Разные терминалы могут иметь различные возможности и поддержку, что может привести к нарушению функциональности приложения.

3.5 Использование библиотеки `form.h`

Библиотека form.h нужна для создания форма и работы с ними в текстовом пользовательском интерфейсе. Формы позволяют создавать интерактивные элементы ввода данных, такие как текстовые поля, переключатели или списки выбора. Основным плюсом данной библиотеки можно отметить удобство в организации пользовательского ввода. С ее помощью можно реализовать позиционирование по вводимым данным, проверку правильности ввода или ограничение количества вводимой информации. Также библиотека предоставляет средства для адаптации интерфейса к различным размерам окна терминала.

Минусом библиотеки является более сложная организация структуры для создания формы. Для использования структуры необходимо задать родительское и дочернее окно, что требует использования большего количества ресурсов.

3.6 Использование библиотеки `menu.h`

Библиотека menu.h предоставляет набор функция для создания и управления меню в терминале. Библиотека позволяет устанавливать обработчики событий для пунктов меню, настраивать формат вывода. Из плюсов можно отметить простоту и понятность реализации меню с помощью данной библиотеки. Она автоматически обрабатывает навигацию по пунктам меню с помощью стрелок и клавиш ENTER и ESC.

Из минусов можно отметить ограниченные возможности библиотеки в плане дизайна и расширения. Библиотека не предоставляет механизмов для расширения функциональности меню или для создания сложных пользовательских элементов.

3.7 Использование библиотеки `panel.h`

Библиотека `panel.h` предоставляет возможности для создания многоуровневых панелей, которые накладываются друг на друга. Библиотека позволяет управлять панелями и устанавливать их положение в соответствии с заданным порядком слоев. Это значительно упрощает решение проблемы с наложением нескольких окон на одно место в терминале. Также библиотека минимизирует количество обновлений экрана, что позволяет улучшить производительность программы.

4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА

В данном разделе приводится описание взаимодействия и работы сосновными компонентами приложения.

4.1 Библиотека ncurses

Для создания и работы с текстовым интерфейсом используется библиотека `ncurses`. Чтобы начать работу нужно инициализировать главное окно с помощью функции `initstr()`. Она инициализирует глобальную переменную `stdscr`, которая является указателем на объект структуры `WINDOW`. Относительно нее будут располагаться все нужные окна. Более того нужно установить дополнительные параметры программы. Используется `noecho()` для того, чтобы на экране не отображались вводимые символы. Данный параметр нужен для корректного отображения меню, и каких–либо выборов, основанных на введении определенного символа. Функцией `echo()` можно включить отображение.

Следующий параметр устанавливается функцией `start_colors()`. Данный параметр включает отображение цветов. После инициализации цветом любому окну можно установить цветовую пару, то есть цвет заднего фона и цвет текста. Данная функция была расширена до функции `ext_start_colors()`, в которой происходит инициализация цветов с помощью `start_colors()`, а так же проверка поддерживаемости цветов терминалом с помощью функции `has_colors()`. Более того в расширенной функции были инициализированы дополнительные цвета и пары цветов с помощью функций:

```
Int init_color(short color, short red, short green, short blue);
Int init_pair(short pair, short foreground, short background);
```

Параметры `init_color()` отвечают за код цвета для переопределения(0-7 для стандартных цветов, 8 и выше для пользовательских). Остальные 3 параметра содержат значение компоненты цвета в диапазоне от 0 до 1000 для красного, зеленого, и синего цветов

соответственно. Параметры init_pair() отвечают за код цветовой пары для переопределения и коды цветов для установки текста и фона.

Функция cbreak() устанавливает режим ввода терминала на неканонический(по умолчанию – канонический). В неканоническом режиме ввода символы передаются программе немедленно после нажатия, без ожидания нажатия клавиши ENTER. В каноническом режиме символы обрабатываются построчно, то есть накапливаются в буфере, пока не нажата клавиша ENTER, после чего они будут отправлены программе для обработки. Для включения канонического режима используется функция nocbreak().

Функцией curs_set(false) мы отключаем отображение моргающего курсора. Изменением параметра с false на true мы включим его обратно.

Для работы с меню и для обработки специальных клавиш используется функция keypad(). Параметр WINDOW* win отвечает за окно, в котором будут обрабатываться нажатия специальных клавиш. Bool key отвечает за включение или выключение нажатий. Ncurses преобразует последовательности символов, представляющие специальные клавиши, в значения, которые могут быть обработаны приложением. Без включения этого режима некоторые специальные клавиши могут быть представлены как последовательности символов, и обработка их может быть нетривиальной.

Далее нам нужно инициализировать различные окна программы. Для этого подойдет функция newwin(), в которую передаются указатель на окно и параметры высоты и ширины окна, а также начальных координат по y и x, которые указываются относительно главного окна stdscr.

Далее после инициализации всех нужных окон нужно обновить главное окно stdscr с помощью функции refresh().

Стоит отметить функцию move(y, x), которая переставляет курсор на координаты y и x относительно главного окна stdscr. Данная функция входит во все расширенные функции ввода-вывода.

Для установки дополнительных атрибутов окна необходимо использовать функции wattron() и wattrof(), которые устанавливают различные атрибуты окна, например, фон и цвет. Для установки атрибутов всему окну используется функция wbkgd().

Для вывода информации на экран используется функция printw(), которая является аналогом функции printf(). Расширенными функциями являются wprintw(), в которую передается указатель на нужное окно, в

пределах которого нужно вывести информацию. Более расширенной функцией является функция mvwprintw(), которая является комбинацией функций wprintw() и move(), то есть вывод происходит в заданном окне по заданным координатам.

```
printw(format, ...);
mvprintw(y, x, format, ...);
mvwprintw(win, y, x, format, ...);
```

Также для вывода строк и символов есть addstr() и addch() и их расширенные версии waddstr(), mvaddstr(), mvwaddstr() и waddch(), mvaddch(), mvwaddch().Так же используется функция box() для графического обозначения границ окна. Для обновления выведенной информации и окна в целом используется функция wrefresh() с указателем на нужное окно.

Для вввода информации с клавиатуры используется функция scanw(), которая является аналогом функции printf(). Расширенными функциями являются wscanw(), в которую передается указатель на нужное окно, в пределах которого нужно вывести информацию. Более расширенной функцией является функция mvwscanw(), которая является комбинацией функций wscanw() и move(), то есть ввод происходит в заданном окне по заданным координатам.

```
scanw(format, ...);
mvscanw(y, x, format, ...);
mvwscanw(*win, y, x, format, ...);
```

Также для ввода строк и символов есть getstr() и getch() и их расширенные версии wgetstr(), mvgetstr(), mvwgetstr() и wgetch(), mvgetch(), mvwgetch().

Для получения длины и ширины окна используются макросы getmaxyx(), getmaxx(), getmaxy(). Более того, параметры длины и ширины главного окна stdscr сохранены в глобальных переменных COLS и ROWS.

```
getmaxyx(*win, maxy, maxx);
getmaxy(*win, maxy);
getmaxx(*win, maxx);
```

Для очистки содержимого окна используется функция `werase()`. Чтобы очистить память, выделенную под это окно используется функция `delwin()`.

Пример с реализацией окна с помощью ncurses с выводом информации, установкой атрибутов и показом границы окна представлен на рисунке 4.1.

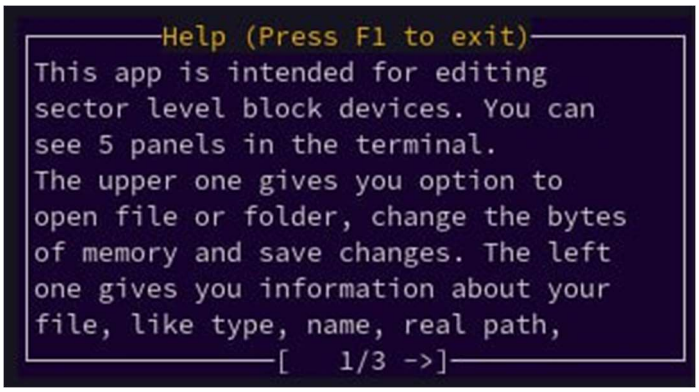

Рисунок 4.1 – пример реализации окна с помощью ncurses

4.2 Библиотека menu.h

Для поддержки выбора различных действий в контексте одного списка была использована библиотека menu.h. Чтобы начать работу с меню, нужно создать пункты меню и определить их с помощью функции `new_item()`. Таким образом мы получим указатель на структуру `ITEM`. Структура содержит имя элемента, его описание, индекс, опции и указатель на пользовательские данные.

```
Typedef struct item {
   Const char* name;
   Const char* description;
   Int index;
   Int opt;
   Void *userptr;
} ITEM;
```

Дальше мы должны создать само меню из его элементов используя `new_menu()`. В него передается массив указателей на ITEM, то есть

несколько элементов меню. Чтобы создать новое меню нужно иметь указатель на структуру MENU. Структура включает массив элементов меню, окно для отображения меню, подокно с элементами меню, количество строк и столбцов подокна, индекс текущего выбранного элемента, индекс верхней строки подокна, индекс строки с маркером, максимальное количество одновременно отображаемых элементов, формат отображения элементов, общее количество элементов, статус меню, количество строк на странице, указатель на пользовательские данные, флаг, указывающий на наличие описаний элемента и флаг, указывающий на наличие маркера.

```
Typedef struct menu{
   ITEM **items;
   WINDOW *win;
   WINDOW *sub;
   Int rows, cols;
   Int curitem;
   Int toprow;
   Int markrow;
   Int maxshow;
   Int format;
   Int nitems;
   Int status;
   Int rowsperpage;
   Void *userptr;
   Bool desclist;
   Bool mark;
} MENU;
```

Для настройки окна меню используются функции set_menu_win(), которая устанавливает указанное окно win в качестве окна, в котором будет отображаться меню menu. Функция set_menu_sub() устанавливает указанное подокно в качестве подокна, содержащего элементы меню menu. Далее этим окнам можно установить свои атрибуты, тем самым изменить внешний вид меню.

```
Int set_menu_sub(MENU* menu, WINDOW* subwin);
Int set_menu_win(MENU* menu, WINDOW* win);
```

Чтобы отобразить меню на экране используется функция `menu_post(MENU* menu)`. Она принимает указатель на меню menu и отображает его на экране в заданной позиции курсора. Она также делает его видимым и готовым с взаимодействием с пользователем. Стоит отметить, что перед вызовом функции `menu_post()` стоит обновить главное окно `stdscr` с помощью функции `refresh()`, чтобы избежать ошибок сегментации.

Для обработки пользовательского ввода используется функция `menu_driver(MENU* menu, int c)`. Она принимает указатель на меню `menu` и символ c, который представляет пользовательское действие или команду для обработки.

Так же используется цикл для обработки нажатий с помощью функции `wgetch()`, которая передает программе скан-код нажатой клавиши. После завершения работы нужно очистить память, которая использовалась под меню и ее элементы с помощью функций `del_menu()` и `del_item()`.

Рисунок 4.2 – реализация меню с обработкой нажатий клавиш влево и вправо

4.3 Библиотека panel.h

Для создания нескольких окон с разным приоритетом отображения была использована библиотека panel.h.

Для начала работы нужно создать несколько панелей с помощью функции `PANEL* new_panel(WINDOW* win)`, которая создает объект структуры `PANEL` и связывает его с указанным окном. Структура `PANEL` состоит из указателя на окно, связанное с этой панелью, указатель на панель, расположенную ниже и выше текущей и координаты по y и x.

```
Typedef struct panel{
   WINDOW *win;
   Struct panel *below;
   Struct panel *above;
   Int startx;
```

```
    Int starty;
} PANEL;
```

Далее нужно указать начальный порядок панелей. Для этого подойдут функции top_panel() и bottom_panel(). Первая перемещает указанную панель на верхний слой, делая ее видимой. Это означает, что она получит высший приоритет отображения и будет отображаться поверх остальных панелей. Вторая функция перемещает указанную панель на самый нижний слой. Таким образом, она получит наименьший приоритет отображения.

```
Void top_panel(PANEL* panel);
Void bottom_panel(PANEL* panel);
```

Чтобы изменить порядок или положение панели можно использовать функции move_panel(), panel_above() и panel_below(). Первая функция позволяет переместить указанную панель на заданные координаты, при этом не изменяя ее относительного порядка или видимости. С помощью второй и третьей функции можно получить указатель на панели, которые на находятся выше или ниже указанной соответственно. Более того, эти функции вернут NULL, если панель находится на верхнем или нижнем уровнях.

```
Void move_panel(PANEL *panel, int startx, int
starty);
PANEL* panel_above(const PANEL* panel);
PANEL* panel_below(const PANEL* panel);
```

Для отображения и скрытия панелей были использованы функции show_panel() и hide_panel(). С помощью функции отображения панелей можно сделать указанную панель видимой, то есть она будет отображаться поверх всех панелей, которые находятся под ней. Если панель уже отображена, то вызов функции не будет иметь эффекта. Функция скрытия панели делает указанную панель невидимой без изменения ее порядка.

```
Void show_panel(PANEL* panel);
Void hide_panel(PANEL* panel);
```

Чтобы отобразить функции в соответствии с их текущим порядком вызывается функция update_panels(). После вызова этой функции панели, перемещенные на верхний слой будут отображаться поверх остальных панелей. Эта функция вызывается после обновления порядка панелей или изменения их видимости. Функция doupdate() отображает все визуальные изменения на экране. Она работает аналогично с refresh(), только дополнительно обновляет видимость панелей в соответствии с их порядком. Для удаления панелей и очистки памяти используется функция del_panel().

4.4 Библиотека form.h

Библиотека form.h была использована для создания форм в терминале, которые позволяют создать интерактивный ввод с использованием полей ввода, кнопок и других элементов управления.

Для начала работы нужно создать форму с помощью функции new_form(), которая принимает массив указателей на структуру FIELD, а на выходе получает указатель на структуру FORM. Указатель на FIELD получается путем создания поля с помощью функции new_field().

```
Typedef strucr {
   Int rows;
   Int cols;
   Int frow;
   Int fcol;
   Int max;
   Int offscreen;
   Int dynamic_buf;
   Char* buf;
} FIELD;
```

Данная структура представляет отдельное поле ввода или элемент управления в форме. Она содержит информацию о размерах поля, его позиции на экране, максимальной длине вводимого текста, буфере для хранения введенного цвета, а так же 2 флага, которые отмечают использование динамического буфера для ввода, и нахождение поля за пределами экрана.

```
Typedef struct form {
   Int rows;
   Int cols;
   Int currow;
   Int curcol;
   WINDOW* win;
   FIELD** field;
} FORM;
```

Структура FORM представляет информацию о размерах формы, текущей позиции курсора, связанном окне и массиве полей формы. Также нужно установить родительское и дочернее окно с помощью функций form_win() и form_sub(). Дочернее окно определяет в каком месте будут вводится поля для ввода данных.

Библиотека содержит функцию set_field_opts() для настройки полей ввода. Так же нам нужно связать форму с определенным окном с помощью функции set_form_win(). Для размещения формы в окне и отображения на экране используется функция post_form();

Функция form_driver() является ключевой в библиотеке form.h. Она используется для обработки пользовательского ввода. С помощью функции можно перемещать курсор между полями формы, вводить текст в активное поле формы или удалять введенные символы. Принимает указатель на FORM и код символа, представляющий пользовательский ввод.

```
Int form_driver(FORM* form, int ch);
```

Для очистки занятой формы и памяти, выделенной под форму и поля, используются функции unpost_form(), free_form(), free_field().

4.5 Работа с памятью

Для работы с памятью и файлами на низком уровне использовались библиотеки stat.h и mman.h, fcntl.h. Библиотека stat.h предоставляет структуру и функции для получения информации о файле. Структура stat содержит информацию о режиме доступа к файлу и его

размеру, а также время последнего доступа и модификации. Основная цель структуры `stat` – получить размер файла и его тип доступа. Сокращенная структура stat с основными полями:

```
Typedef struct stat{
   Dev_t st_dev;
   Mode_t st_mode;
   Off_t st_size;
   Time_t st_atime;
   Time_t st_mtime;
}
```

Чтобы заполнить структуру соответствующей информацией о файле используется функция `stat()`.

Библиотека `fcntl.h` используется для работы с файловыми дескрипторами, а так же содержит основные функции для работы с файлами. Для открытия файла используется функция `open()`, которая позволяет создать файл или открыть существующий с различными типами доступа. Закрыть файл можно с помощью функции `close()`.

Библиотека `mman.h` является основной логической библиотекой в данном проекте. Она обеспечивает возможность работы с отображенными в память файлами и для управления памятью.

Для отображения файла в память процессора используется функция `mmap()`. Она позволяет работать с файлом, как если бы он находился в памяти, и предоставляет доступ к нему через указатель.

Параметрами являются указатель на желаемое начальное отображение в памяти, размер отображаемой области в байтах, режимы защиты памяти для отображения, флаги с дополнительными свойствами отображения, файловый дескриптор файла и смещение от начала файла, с которого начинается отображения.

С помощью `mmap()` также было реализовано редактирование содержимого файла. Для удаления отображенной памяти используется функция `munmap()`.

```
Void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
```

Особенностью этой функции является то, что система отображает файл на основе страницы, поэтому смещение должно быть кратно размеру страницы. Размер страницы можно найти с помощью функции `sysconf(_SC_PAGESIZE)`.

Для получения доступа к памяти устройства использовались системные файлы `/dev/mem` и `/proc/kcore`. `/dev/mem` представляет собой символьное устройство, которое позволяет получить прямой доступ к памяти компьютера, включая операционную систему и другие процессы. `/proc/kcore` отличается тем, что представляет виртуальное отображение всей доступной физической памяти, а также виртуальное отображение ядра.

5 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ

В данном разделе приведены алгоритмы по шагам самых важных функций программы.

5.1 Алгоритм функции `print_normal()`

Функция предназначена для вывода на экран окна, содержащего представление прочитанных байт в ASCII символах.

1. Начало.
2. Входные данные
   `WINDOW *normal win` – указатель на созданное окно.
   `unsigned char *bytes` – массив символов для вывода.
   `int highlight_number` – индекс элемента для подсветки.
   `short MENU_GREEN` – цветовая пара для вывода заголовка окна.
   `short MENU_SLCTD_ITEM` – цветовая пара для обозначения выбранного элемента.
3. Очистить содержимое окна `normal_win`.
4. Вывести на экран границы окна `normal_win`.
5. Включить атрибут `MENU_GREEN` для окна `normal_win`.
6. Вывести заголовок окна в окне `normal_win` по координатам 0, 1.
7. Выключить атрибут `MENU_GREEN` для окна `normal_win`.
8. Если массив символов для вывода не пуст, перейти на шаг 9, иначе 18
9. Цикл по `i`: от 0 до 19.
10. Цикл по `j`: от 0 до 16.
11. Если `highlight_number` равен `16*i + j + 1`, то включить атрибут `MENU_SLCTD_ITEM` для окна `normal_win`.
12. Если `(16*i+j)`-ый символ массива `bytes` больше 0, но меньше 32 или больше 126, но меньше 160, то вывести значение `(16*i+j)`-ого символа + `0x2654` в окне `normal_win` по координатам `i+1, j+1`, иначе перейти на шаг 13.
13. Если `(16*i+j)`-ый символ массива `bytes` равен 0, то вывести символ ‘.’ в окне `normal_win` по координатам `i+1, j+1`, иначе перейти на шаг 14.

14. Вывести значение (16*i+j)-ого символа в окне normal_win по координатам i+1, j+1.
15. Если highlight_number равен 16*i + j + 1, то выключить атрибут MENU_SLCTD_ITEM для окна normal_win.
16. Конец цикла по i.
17. Конец цикла по j.
18. Обновить окно normal_win.
19. Конец.

5.2 Алгоритм функции readOffset()

Функция предназначена для прочтения физической памяти устройства по заданному смещению.

1. Начало
2. Входные данные:
   unsigned char** page – массив для чтения байтов из памяти.
   off_t offset – смещение, по которому будет проводиться чтение.
   «/dev/mem» – путь к файлу, через который осуществляется доступ к физической памяти устройства.
   Промежуточные данные:
   Int fd – дескриптор файла «/dev/mem».
   Struct stat fileStat – объект структуры с информацией о файле.
   Long page_size – размер читаемой страницы.
   Off_t true_offset – настоящее смещение, по которому будет проводиться чтение.
3. Открыть файл «/dev/mem» с типом доступа только чтение.
4. Если открытие не удалось, то вернуть 2.
5. Заполнить структуру fileStat по дескриптору fd функцией fstat().
6. Получить размер страницы с помощью функции sysconf(_SC_PAGESIZE) и записать его в page_size.
7. Записать в true_offset значение, полученное путем вычитания из offset его остатка от деления на page_size.

8. Если `true_offset` больше размера файла, то вернуть 4.
9. Если `page_size + true_offset` больше размера файла, то присвоить `page_size` значение, равное размеру файла – `true_offset`.
10. Отобразить массив mem в память с длиной page_size, режимом доступа `PROT_READ`, флагом `MAP_SHARED`, дескриптором `fd`, и смещением `true_offset`.
11. Если отображение в память не удалось, то вернуть 3.
12. Скопировать память в page из mem с длиной `page_size`.
13. Удалить отображенную память.
14. Закрыть файл по дескриптору `fd`.
15. Вернуть 1.
16. Конец.

5.3 Алгоритм функции `helpPanel()`

Функция реализует вывод окна с помощью и переключение страниц помощи с помощью обработок нажатий клавиатуры.

1. Начало.
2. Входные дынные:
   `WINDOW* win_help` – указатель на окно помощи.
   `Short TOP_PANEL_COLOR` – цветовая пара для фона окна помощи.
   Промежуточные данные:
   `Int maxx, maxy` – максимальные координаты терминала.
   `Int c` – переменная для сохранения скан-кодов нажатых клавиш.
3. Получить `maxx` и `maxy` функцией `getmaxyx()`.
4. Создать окно `win_help` с шириной 10, длиной 42, по координатам `(maxy - 10)/2` и `(max - 42)/2`.
5. Установить атрибут цветовой пары `TOP_PANEL_COLOR` окну `win_help`.
6. Включить обработку нажатий.
   Вывести первую страницу окна помощи функцией `printHelp()`.
7. Выполнять бесконечный цикл.
8. Обработать нажатие и записать его в c.

9. Если нажата F1, выйти из цикла.
10. Если нажата левая стрелка, то выполнить пункт 12.
11. Если страница уже 1, то перейти на шаг 9, иначе 13.
12. Уменьшить страницу на 1.
13. Вывести окно помощи с нужной страницей.
14. Если нажата стрелка вправо, то выполнить шаг 16.
15. Если страница уже 3, то перейти на шаг 9, иначе 17.
16. Увеличить страницу на 1.
17. Вывести окно помощи с нужной страницей.
18. Конец цикла.
19. Освободить память от окна `win_help`.
20. Конец.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

В данном разделе приведено руководство по использованию клиентской программы.

После запуска приложения загружается основной интерфейс с двумя информационными панелями, которые расположены слева и снизу, управляющей панелью сверху и двумя рабочими панелями справа и в центре (см. рисунок 6.1).

Рисунок 6.1 – основной интерфейс приложения

На левой информационной панели с обозначением «Info panel» показывается информация о файле, который открыт в данный момент. Показывается его тип, имя, прямой путь, размер и тип доступа к нему. На нижней информационной панели показывается информация о памяти, то есть смещение курсора, номер блока и сектора в нем, а также их размер. На верхней управляющей панели есть 5 вкладок, которые можно переключать с помощью

стрелок влево и вправо. Выбранная вкладка подсвечивается голубым цветом. Четвертая вкладка открывает руководство пользователя в самой программе (см. рисунок 6.2).

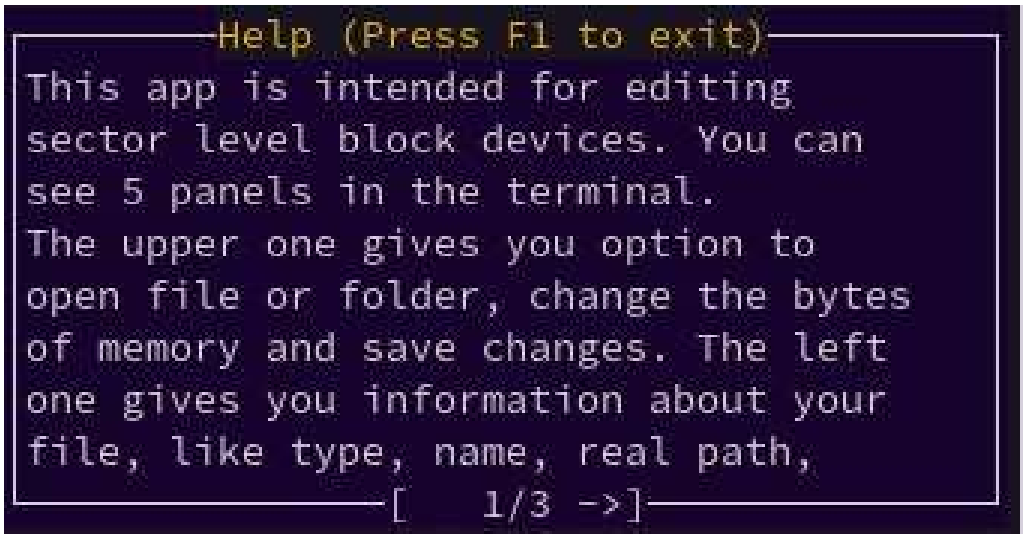

Рисунок 6.2 – руководство пользователя в приложении

Страницы переключаются с помощью нажатия клавиш стрелок влево и вправо, а выход из вкладки помощи в окно основного интерфейса осуществляется с помощью нажатия клавиши F1.

Первая вкладка отвечает за открытие файла. При ее нажатии пользователь увидит окно, в котором он должен ввести путь к файлу (см. рисунок 6.3).

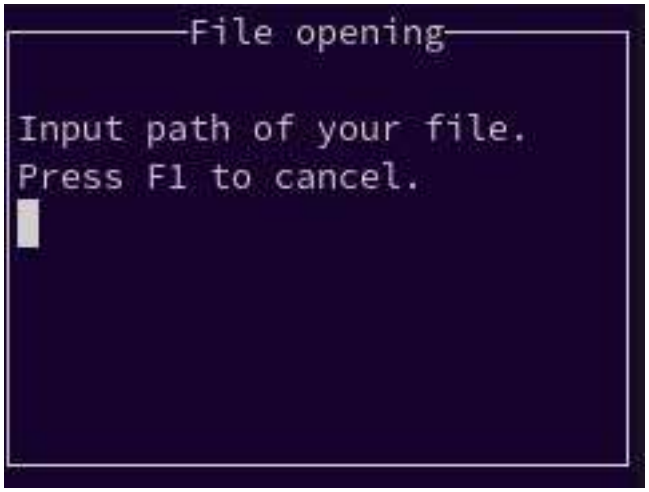

Рисунок 6.3 – окно ввода пути к файлу

При неправильном вводе пути к файлу или при ошибке открытия или чтения файла пользователь получит окно с ошибкой, похожую на окно с ошибкой о прочтении памяти(см. рисунок 6.9) и возможными путями решения возникшей ошибки. После открытия файла на рабочих панелях будет отображено содержание файла в двух видах. На центральной панели информация будет представлена в виде 16-ричных значений (см. рисунок 6.5), а на правой панели информация будет предоставлена в читаемом виде (см рисунок 6.6).

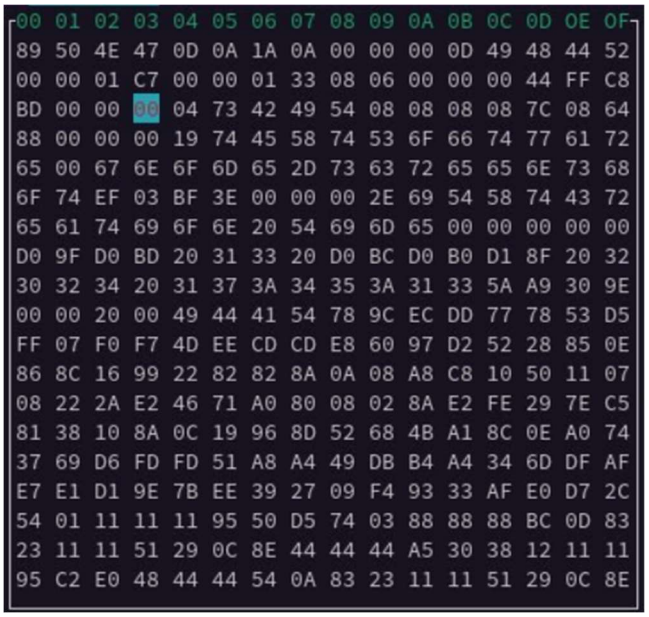

Рисунок 6.5 – 16-ричный вид информации

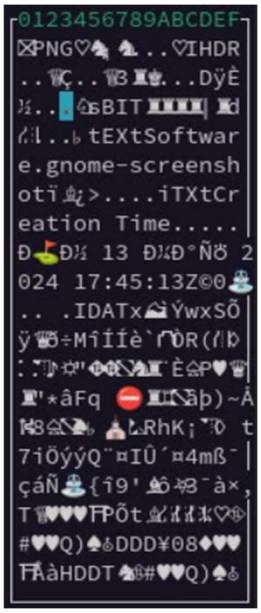

Рисунок 6.6 – информация в читаемом виде

Более того, изменится содержание информационных панелей. Левая панель будет содержать информацию об открытом файле, а нижняя – о блоках и смещении, на котором находится курсор, который помечен на рабочих панелях голубым цветом (см. рисунок 6.7). Нажатием клавиши F2 можно изменить окно, которым управляет пользователь, то есть переключать активное окно между управляющей панелью и рабочими панелями.

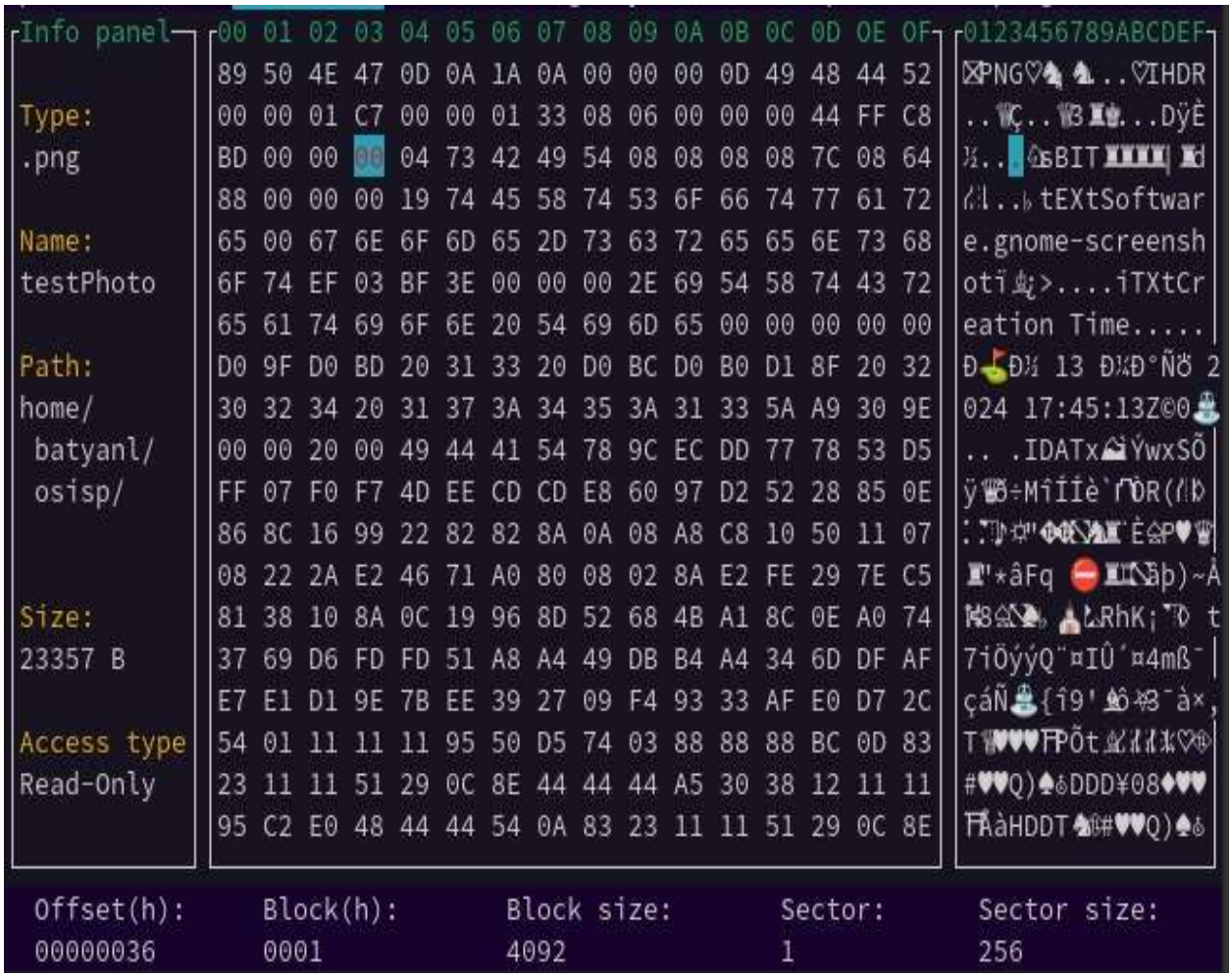

Рисунок 6.7 – общий вид интерфейса при открытом файле

С помощью третьей вкладки в управляющей панели можно включить режим редактирования информации. В таком случае тип доступа изменится на Read-Write, и пользователь сможет редактировать байты. Для этого нужно нажать F2 для переключения курсора на рабочую панель. После этого пользователь сможет выбрать нужный байт для редактирования, нажимая стрелки вправо, влево, вниз и вверх, а потом нажать клавишу ENTER для редактирования байта. При введении неправильной информации, пользователь получит ошибку похожую на ошибку, продемонстрированную

на рисунке 6.9.

С помощью второй вкладки пользователь сможет открыть память компьютера для чтения и редактирования. Для этого нужно ввести смещение, с которого пользователь хочет прочитать информацию (см. рисунок 6.8).

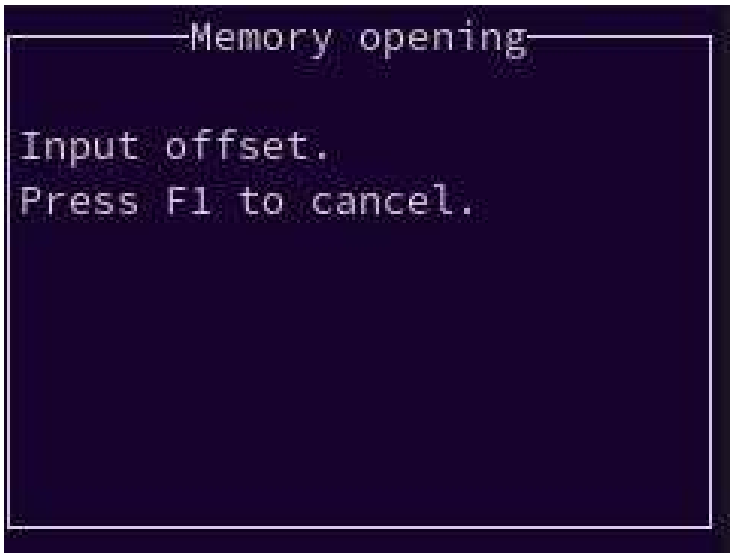

Рисунок 6.8 – окно с введением смещения

Для открытия памяти программу нужно запускать с правами администратора. Если пользователь ввел неверное смещение, или программа не была запущена с правами администратора, то пользователь получит окно с ошибкой (см. рисунок 6.9).

Рисунок 6.9 – окно с ошибкой

Последующие шаги аналогичны с чтением и редактированием файлов. Следует отметить, что не рекомендуется редактировать байты при работе с памятью, так ка это может привести к непредвиденным последствиям, таким как поломка операционной системы, утеря данных или потеря доступа к компонентам компьютера.

ЗАКЛЮЧЕНИЕ

В результате работы над данным курсовым проектом было разработано работоспособное приложение со своим набором функций и графическим интерфейсом. Данный курсовой проект был разработан в соответствии с поставленными задачами, весь функционал был реализован в полном объеме.

Для создания курсового проекта была подробна исследована технология работы с блочными устройствами. В ходе разработки были углублены знания языка программирования C. Также были успешно использованы такие преимущества языка C, как близость к аппаратуре, эффективность и доступ к системным ресурсам. Как и ожидалось, библиотека `ncurses` значительно упростила разработку текстового пользовательского интерфейса для редактора блочных устройств.

Работа была разделена на такие этапы, как анализ существующих аналогов, литературных источников, постановка требований к проектируемому программному продукту, системное и функциональное проектирование, конструирование программного продукта, разработка программных модулей и тестирование проекта. После последовательного выполнения вышеперечисленных этапов разработки было получено исправно работающее приложение.

В дальнейшем планируется усовершенствование текущего функционала приложения, путем улучшения графического интерфейса, добавления новых функций и модулей, а также добавления возможности работы под разными системами.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Вычислительные машины, системы и сети: дипломное проектирование (методическое пособие) [Электронный ресурс]. – Минск БГУИР 2019. – Режим доступа: https://www.bsuir.by/m/12_100229_1_136308.pdf

Habr [Электронный ресурс]. – Руководство по ncurses. – Режим доступа: https://habr.com/ru/articles/778040/ – Дата доступа: 29.03.2024

NCURSES Programming HOWTO [Электронный ресурс]. – Руководство по ncurses. – Режим доступа: https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/windows.html – Дата доступа: 29.03.2024

Жешке Рекс. Толковый словарь стандарта языка Си / Пер. с англ.– М.: Мир, 1992.–687с.

ПРИЛОЖЕНИЕ А

Ведомость документов

ПРИЛОЖЕНИЕ Б

Схема функции printError()

ПРИЛОЖЕНИЕ В

Полный код программы