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

НИЗКОУРОВНЕВЫЙ РЕДАКТОР БЛОЧНЫХ УТСРОЙСТВ УРОВНЯ СЕКТОРОВ(NCURSES)

БГУИР КП 1-40 02 01 111 ПЗ

Студент: гр.250501 Гулевич В.А.
Руководитель: Игнатович А.О.

Минск 2024

РЕФЕРАТ

Курсовой проект предоставлен следующим образом. Чертежный материал: X листа формата А4. Пояснительная записка: XX страниц, XX рисунка, X литературных источника, X 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 так же анализ аналогов приложения.

2.1 Анализ существующих аналогов

Тема курсового проекта была выбрана в первую очередь для углубления знаний по языку C 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.

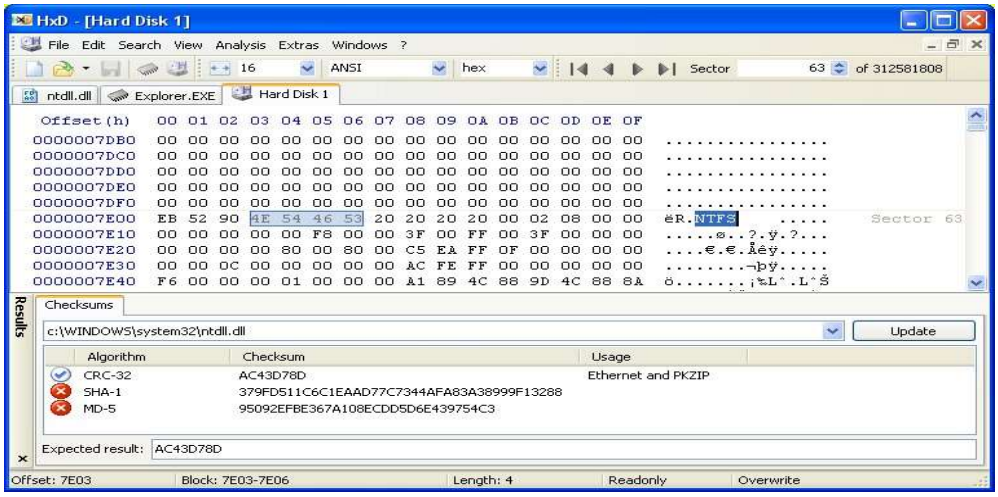

Рисунок 2.1 — Пользовательский интерфейс программы HxD

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.

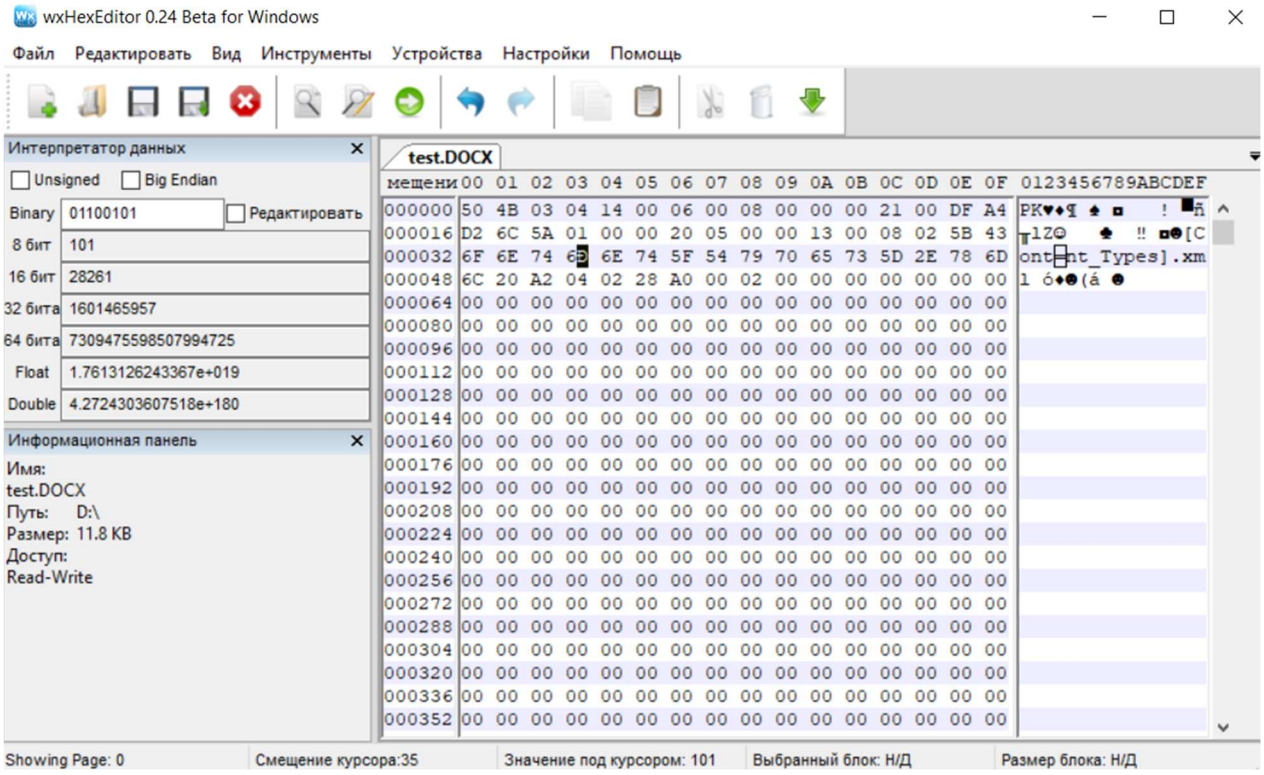

Рисунок 2.2 — Пользовательский интерфейс программы wxHexEditor

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.

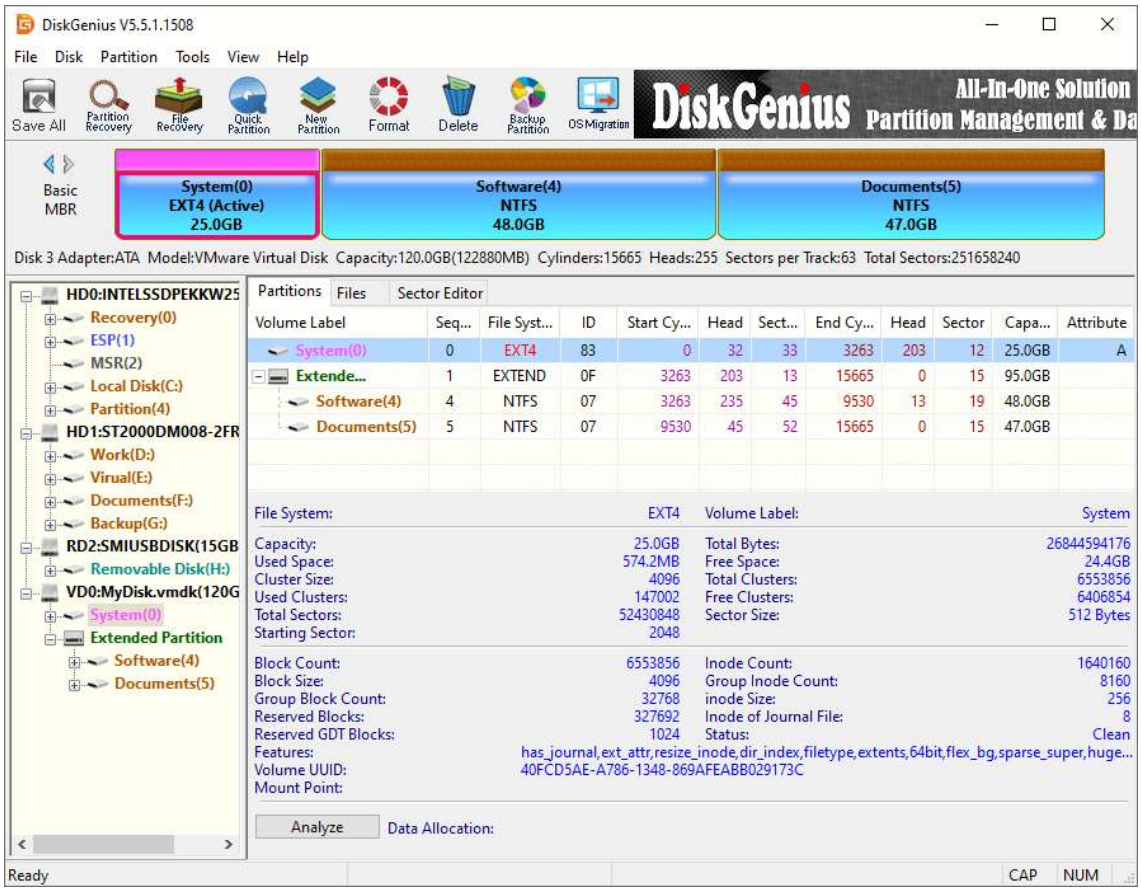

Рисунок 2.3 — Пользовательский интерфейс программы DiskGenius

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 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.

2.3 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.

2.4 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.

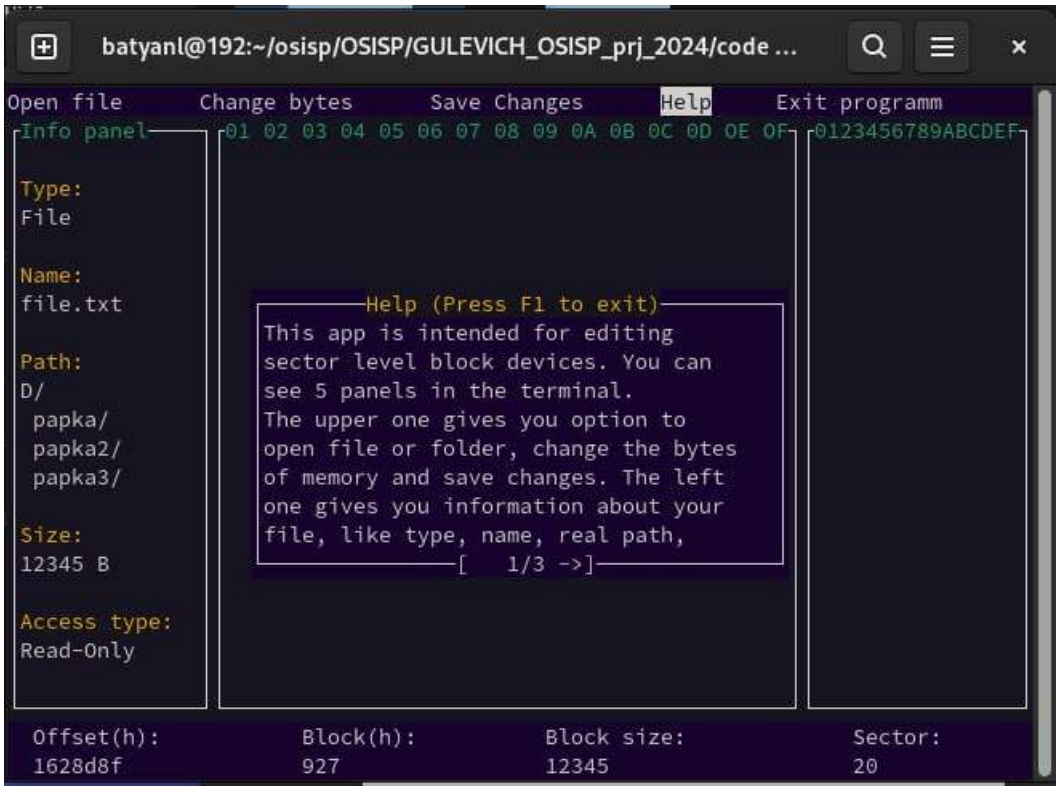

Рисунок 2.4 – создание окон с помощью ncurses

На рисунке 2.4 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.

2.5 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.

2.6 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.

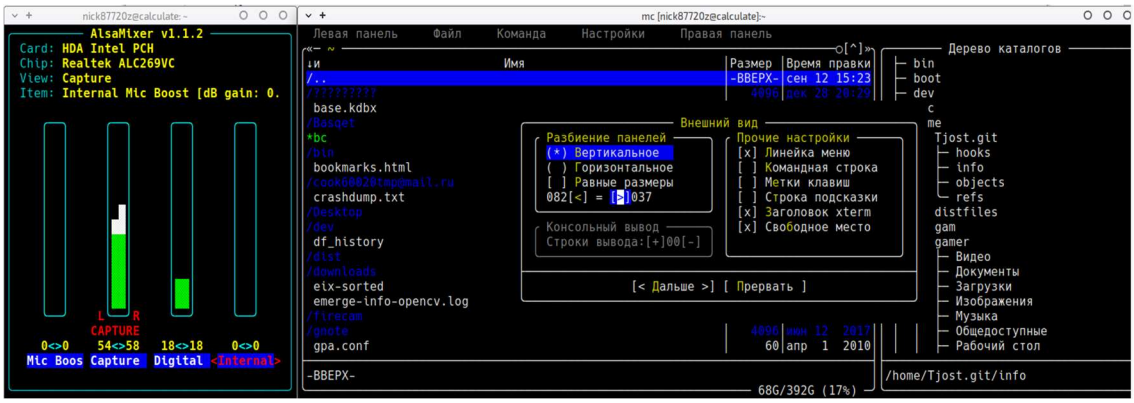

Рисунок 2.5 – приложение с перекрывающимися окнами

2.7 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 2.6:

Рисунок 2.6 – пример реализации меню

3 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ

4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА

5 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

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

Полный код программы