

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

ПРОГРАММА ДЛЯ КОНТРОЛЯ ИЗМЕНЕНИЙ И ЦЕЛОСТНОСТИ
ГРУППЫ ФАЙЛОВ

БГУИР КП 1–40 02 01 323 ПЗ

Студент: гр. 250503 Патюпин М.С.

Руководитель: ассистент Калютчик А.А.

МИНСК 2024

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой

(подпись)

2024 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту Патюпину Михаилу Сергеевичу

1. Тема проекта Программа для контроля изменений и целостности группы файлов
2. Срок сдачи студентом законченного проекта 10 мая 2024 г.
3. Исходные данные к проекту: Программа предназначена для контроля групп файлов от несанкционированных изменений . В случае несоответствия с эталоном восстанавливает предыдущую сохранённую версию.
4. Содержание расчетно–пояснительной записки (перечень вопросов, которые подлежат разработке)

Лист задания.

Введение.

1. Обзор источников.

1.1. Обоснование разработки.

1.2. Обзор аналогов.

1.3. Обзор методов и алгоритмов решения поставленной задачи.

2. Функциональное проектирование.

2.1. Структура входных и выходных данных.

2.2. Разработка диаграммы классов.

2.3. Описание классов.

3. Разработка программных модулей.

3.1. Разработка схем алгоритмов.

3.2. Разработка алгоритмов.

4. Результаты работы.

5. Литература

Заключение

Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков):

1. Диаграмма классов.

2. Схема алгоритма .

3. Схема алгоритма .

6. Консультант по проекту (с обозначением разделов проекта) А.А. Калютчик

7. Дата выдачи задания 20 февраля 2024

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

Выбор задания. Разработка содержания пояснительной записки.

Перечень графического материала – 15 %;

разделы 2, 3 – 10 %;

разделы 4 к – 20 %;

разделы 5 к – 35 %;

раздел 6,7,8 – 10 %;

оформление пояснительной записки и графического материала к 10.05.24 – 10 %

Защита курсового проекта с 28.05 по 10.06.24г.

РУКОВОДИТЕЛЬ

А.А. Калютчик

(подпись)

Задание принял к исполнению

_____ М.С. Патюпин
(дата и подпись студента)

СОДЕРЖАНИЕ

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ	5
ВВЕДЕНИЕ.....	6
1 ОБЗОР ИСТОЧНИКОВ	7
1.1 Обоснование разработки	7
1.2 Обзор аналогов	7
1.3 Обзор методов и алгоритмов решения поставленной задачи	10
2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ	11
2.1 Структура пакета программ	11
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ	14
3.1 Структура входных и выходных данных.....	14
3.2 Описание констант и переменных.....	14
3.3 Описание подпрограмм	17
3.4 Описание файлов.....	21
4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ	24
4.1 Разработка схем алгоритмов работы приложения.....	24
4.2 Описание подпрограммы <code>to_encrypt_file</code>	24
4.3 Описание подпрограммы <code>main_proc</code>	24
5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	26
5.1 Требования к программному обеспечению.....	26
5.2 Описание процесса инсталляции.....	31
5.3 Описание работы с программой	33
ЗАКЛЮЧЕНИЕ	49
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	50
ПРИЛОЖЕНИЕ А.	51
ПРИЛОЖЕНИЕ Б.....	53
ПРИЛОЖЕНИЕ В	55
ПРИЛОЖЕНИЕ Г	57
ПРИЛОЖЕНИЕ Д	98

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Целью данного курсового проекта является разработка утилиты, предназначенной для обеспечения сохранности и целостности группы файлов путем предотвращения несанкционированных изменений, случайных изменений или удаления файлов. В случае обнаружения изменений, утилита должна восстанавливать измененные файлы в их последнее сохраненное состояние.

Основной функционал, который должен быть реализован в утилите, включает:

- предоставление возможности обновления защищаемых файлов;
- создание резервной копии, которая может быть использована для восстановления данных в случае необходимости;
- периодическая проверка файлов на наличие изменений используя хэш–суммы вычисленные несколькими алгоритмами;
- в случае нарушения целостности файлов восстановление их в последнюю сохраненную версию;
- обеспечение уведомления о несанкционированном изменении файлов.

ВВЕДЕНИЕ

В современном информационном обществе, где цифровые данные играют центральную роль, обеспечение безопасности и сохранение целостности файлов становится все более неотъемлемой задачей. Программа, разработанная в ходе данного курсового проекта, представляет собой важный инструмент, который имеет множество полезных применений и значительное значение для пользователей и организаций.

Программа для контроля изменений и целостности группы файлов позволяет пользователям обеспечивать сохранность и неприкосновенность своих данных путем предотвращения несанкционированных изменений, случайных изменений, удаления файлов. Ее полезность проявляется в ряде аспектов:

1 **Защита от потери данных.** Предотвращение потери важной информации, обеспечивая резервное копирование и восстановление файлов. Это важно в случаях, когда файлы были повреждены, и удалены из-за ошибок пользователя, атак вредоносных программ или аппаратных сбоев.

2 **Обеспечение целостности данных.** Контролирование изменений в файлах и предотвращение несанкционированных изменений или вмешательств. Программа обнаруживает любые изменения в защищаемых файлах и может восстановить их в последнюю сохраненную версию, восстанавливая целостность данных.

3 **Обеспечение безопасности информации.** Программа выступает дополнительным уровнем безопасности для файлов и данных защищая файлы от несанкционированного доступа и предотвращая их изменение без соответствующих разрешений.

4 **Возможность контролировать и управлять любыми файлами в группе, возможность пользователям легко обновлять файлы, автоматически создавать резервные копии, удобно восстанавливать данные.**

В итоге, программа для контроля изменений и целостности группы файлов является неотъемлемым инструментом для пользователей и организаций, которые ценят безопасность, надежность и целостность своих данных. Она обеспечивает защиту информации, предотвращает потерю данных и обеспечивает контроль изменений, способствуя более эффективному управлению файлами и повышению общей безопасности информационных систем.

1. ОБЗОР ИСТОЧНИКОВ

1.1 Обоснование разработки

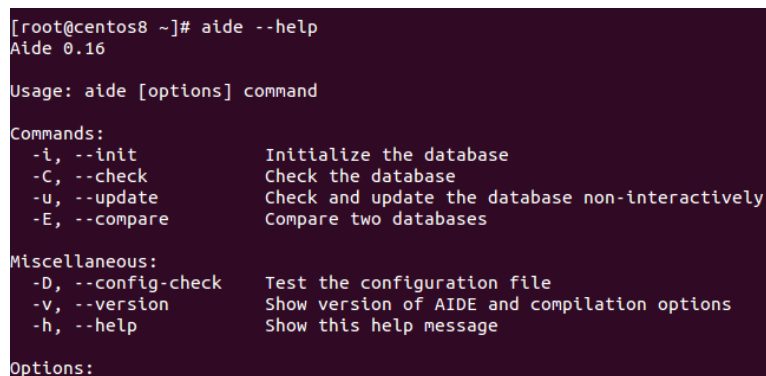
Для того, чтобы приступить к разработке, было необходимо ознакомиться с основными подходами к проверке файлов на изменения, изучить основные подходы и функционал уже существующих утилит, для создания собственного аналога, ничем не уступающего вышеупомянутым. Требовалось изучить подходы к мониторингу изменений, варианты защиты данных, продумать взаимодействие с пользователем. Необходимо иметь четкое представление о принципах работы файловой структуры в Linux.

Также было необходимо хорошо владеть языком программирования C++. Для написания грамотного кода требовалось ознакомиться с так называемыми «соглашениями» по написанию кода на C++, чтобы возможные расширения программы или ее рефакторинг не вызывали затруднений при реализации.

1.2 Обзор аналогов

В сфере мониторинга целостности файлов существует множество приложений, имеющих свои уникальные особенности, которые делают их привлекательными для различных потребностей и сценариев использования: Advanced Intrusion Detection Environment, Log360, Kaspersky Security, tripwire и многие другие. Оценка возможностей существующих решений, позволяет определиться с уже зарекомендовавшими себя подходами к решению поставленных задач.

Advanced Intrusion Detection Environment (AIDE) – рисунок 1.1, и Tripwire, фокусируются на создании базы данных с хэш–суммами файлов и метаданными для сравнения с текущим состоянием файловой системы. Они обеспечивают обнаружение изменений в файловой системе, а также предоставляют гибкие настройки и конфигурируемые правила для обнаружения изменений и нарушений целостности.



```
[root@centos8 ~]# aide --help
Aide 0.16

Usage: aide [options] command

Commands:
-i, --init           Initialize the database
-C, --check          Check the database
-u, --update         Check and update the database non-interactively
-E, --compare        Compare two databases

Miscellaneous:
-D, --config-check   Test the configuration file
-V, --version         Show version of AIDE and compilation options
-h, --help           Show this help message

Options:
```

Рисунок 1.1 – Окно справки AIDE

Log360 – рисунок 1.2, специализируется на сборе, анализе и мониторинге журналов событий, что позволяет обнаруживать аномалии, атаки и угрозы безопасности на основе анализа журналов и событий. Он предоставляет функции мониторинга доступа, обнаружения инцидентов, управления угрозами и анализа поведения пользователей.

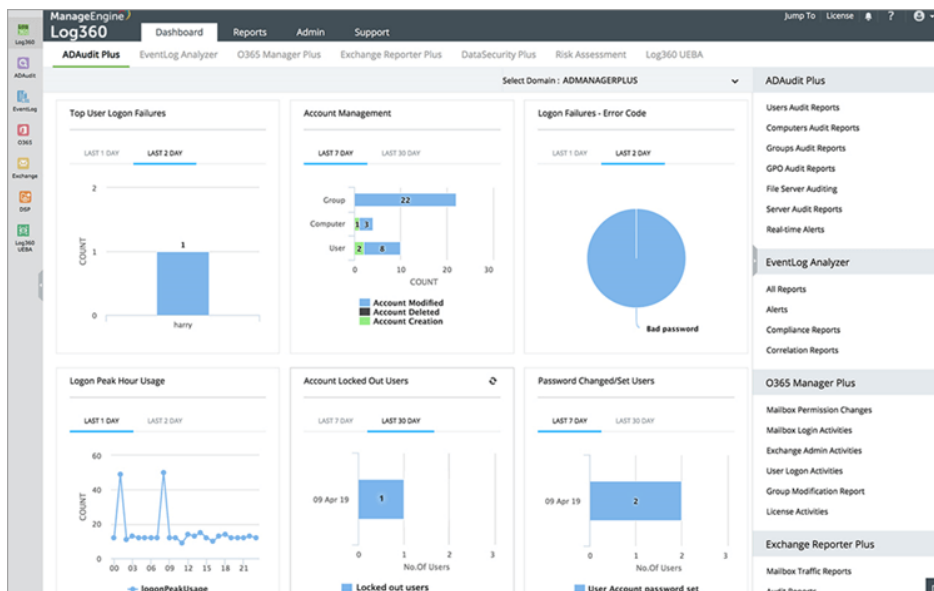


Рисунок 1.2 – Главное окно приложения Log360

Kaspersky Security – рисунок 1.3, представляет собой комплексное решение для защиты информации и борьбы с киберугрозами. Оно включает в себя решения для обнаружения и предотвращения вторжений, а также функции контроля и защиты файлов и шифрование данных.



Рисунок 1.3 – Главное окно приложения Kaspersky Security

Крайне сложно перечислить все существующие на данный момент приложения: число их растет постоянно, а отличаются они тем, что стараются обогнать конкурентов, углубляясь в более узкие отрасли работы, или наоборот создавая целый комплекс решений. Но все они, при этом, имеют и общий функционал, от которого и будет отталкиваться данный проект. Сравнение аналогов представлено в таблице 1.1.

Таблица 1.1 – Сравнение аналогов

Приложение	AIDE	Tripwire	Log360	Kaspersky Security
Основной функционал	Создание базы данных с хэш–суммами файлов и метаданным и для сравнения с текущим состоянием файловой системы.	Создание базы данных с хэш–суммами файлов и метаданным и для сравнения с текущим состоянием файловой системы.	Сбор, анализ и мониторинг журналов событий для обнаружения аномалий, атак и угроз безопасности.	Комплексное решение для защиты информации и борьбы с угрозами.
Обнаружение изменений в файловой системе	Да	Да	Нет	Нет
Обнаружение нарушений целостности файлов	Да	Да	Нет	Да
Гибкие настройки и конфигурируемые правила	Да	Да	Да	Да
Мониторинг доступа	Нет	Нет	Да	Да

1.3 Обзор методов и алгоритмов решения поставленной задачи

Для написания курсового проекта был выбран язык программирования C++. Это компилируемый, мультипарадигмальный язык общего назначения с поддержкой процедурного программирования (возможность реализации на чистом C), обобщенного программирования (шаблоны) и объектно-ориентированного (классы и их объекты). C++ широко используется для разработки программного обеспечения, являясь при этом одним из самых популярных языков программирования в мире. На этом языке можно разработать как простейшие консольные приложения, так и целые операционные системы и драйверы.

Разработка курсового проекта строилась на основе парадигмы процедурного программирования, которая подразумевает представление всей программы в виде совокупности взаимодействующих между собой функций или процедур.

Широко использовались возможности языка C++, такие как работа с указателями и динамической памятью, хранение объектов в стандартных контейнерах библиотеки STL, использование пользовательских библиотек.

Создание проекта выполнялось в среде разработки CLion версии 2023.3. Данная интегрированная среда нацелена на работу с C/C++, позволяет упростить разработку с использованием git, отладчиком, дополнительных плагинов. Для сборки проекта был использован компилятор G++.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются структура пакета программ, приводится назначение основных блоков, связь между ними.

2.1 Структура пакета программ

Результатом курсового проектирования являются две программы написанных на языке высокого уровня – C++.

2.1.1 Описание программы **Front_sfc**

Программа **Front_sfc** – основной задачей является взаимодействие с пользователем, посредством передачи аргументов командной строки, или при их отсутствии – используя классический графический интерфейс. После обработки информации от пользователя формируется, создается или обновляется конфигурационный файл с указаниями для второй программы. Опциями при вызове **Front_sfc** являются следующие ключи:

- 1 **enable**. Включение защиты, с этого момента начнется наблюдение за файлами, будет создана резервная копия.
- 2 **disable**. Выключение защиты, снятие контроля над файлами, автоматическое восстановление измененных, удаленных файлов.
- 3 **file-info**. Окно справки, в нем содержится информация о пути к защищаемой директории, состояние основной программы, интервал между проверками хэш-сумм, тип восстановления с резервной копии, предпочтительный алгоритм хеширования, настройки уведомлений. Снимок экрана окна справки приведен на рисунке 2.1.
- 4 **force-restore**. Принудительное восстановление файлов, согласно выбранного типа. Восстановление происходит только с включенной защитой.
- 5 **help**. Открытие справки, включающей в себя описание ключей и дополнительные требования по вводу.
- 6 **exit**. Выход из программы, сохранение конфигурационного файла.
- 7 **path <string>**. Установка пути к директории, требующей защиты.
- 8 **interval <int>**. Установка временного интервала в минутах проверки хэш-сумм файлов.
- 9 **backup-type <string>**. Установка типа восстановления после резервного копирования.
- 10 **hash-algorithm <string>**. Установка приоритетного алгоритма для вычисления, и сверки хеш-сумм.
- 11 **notification-channel <string>**. Установка канала уведомлений, или их отключение.

При отсутствии дополнительных ключей при вызове пользователю предстает графический интерфейс, включающий в себя весь

вышеперечисленный функционал. Вид графического интерфейса представлен на рисунках 2.2, 2.3.

```
*****
* File information:                               *
* Path: /path/to/file's
* Protection: Enabled
* Interval: 30
* Backup Type: full
* Hash Algorithm: MD5
* Notification Channel: system
*****
```

Рисунок 2.1 – Окно справки

```
=====
|| 1. Enable protection                        ||
|| 2. Disable protection                      ||
|| 3. Get information about files            ||
|| 4. Force file restoration                 ||
|| 5. Configure protection                  ||
|| 6. Show help                             ||
|| 7. Exit the program                      ||
=====
Enter your choice:
```

Рисунок 2.2 – Основное окно программы

```
=====
|| 1. Specify the file path                  ||
|| 2. Specify the file check interval       ||
|| 3. Specify the backup type               ||
|| 4. Specify the hashing algorithm         ||
|| 5. Specify the notification channel      ||
|| 6. Return to main menu                  ||
=====
Enter your choice:
```

Рисунок 2.3 – Установка параметров программы

2.1.2 Описание программы Back_sfc

Программа Back_sfc – это многопоточная программа «демон», основной функцией которой является «контроль», который осуществляется отдельным потоком над конфигурационным файлом, при первичном обнаружении которого, или изменении запускается основная функция, отвечающая за чтение конфигурационного файла, и в зависимости от полученных и текущих значений настраивается защита.

Программа отлавливает изменения файлов в охраняемой директории, удаление, или добавление ранее неизвестных.

В автоматическом режиме, проводит резервное копирование файлов в директории. При резервном копировании происходит архивация файлов, и последующие шифрование с использованием уникального ключа.

Back_sfc также содержит опцию системных уведомлений пользователя, которая предупреждает, об изменении настроек, и конфигурации защиты, и об нарушении целостности охраняемых файлов. Это обеспечивает прозрачность для пользователя, позволяя ему видеть текущие настройки и состояние файла, над которым осуществляется контроль.

2.1.3 Описание связи между Front_sfc и Back_sfc

Программы Back_sfc и front_sfc обмениваются данными между собой посредством конфигурационного файла. В этом файле хранятся различные параметры, такие как путь к файлу, настройки защиты, интервал, тип резервного копирования, используемый алгоритм хэширования и канал уведомлений. Эти параметры обновляются функцией в программе Front_sfc, и считываются процедурой в программе Back_sfc и затем используются для настройки функциональности программы.

Это обеспечивают эффективный обмен информацией между программами, обеспечивая гибкость и адаптивность в реагировании на изменения в окружающей среде. И позволяет программам работать вместе, чтобы обеспечить надежную и эффективную защиту данных.

Схема работы системы приведена в приложении А.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются входные и выходные данные программы, описание констант и переменных, а также приводится описание используемых классов и их методов.

3.1 Структура входных и выходных данных

Ниже приведено описание структуры входных и выходных данных для программы по контролю изменений и целостности группы файлов:

Входные данные:

- режим работы;
- путь к файлу, или директории;
- интервал проверки файлов;
- тип резервного копирования;
- алгоритм хеширования;
- канал уведомлений;
- запрос на восстановление файлов;
- запрос на просмотр информации о файлах.

Выходные данные:

- информация о режиме работы;
- информация о группе файлов;
- справка;
- системные уведомления.

3.2 Описание констант и переменных

Этот подраздел содержит описание основных констант, переменных, структур данных в полученном пакете программ.

3.2.1 Описание констант и переменных в программе `Front_sfc`

В программе `Front_sfc` существуют две глобальные структуры данных – это структуры `buffer_menu` и `flag_menu`. Их описание будет приведено ниже.

Структура `flag_menu` служит для записи настроек, указанных пользователем, и для указания необходимости обратить внимания на измененные поля. `flag_menu` содержит следующие поля:

- `bool protection`, булева переменная, которая указывает, включена ли защита или нет;
- `int flag_protection = 0`, флаг, который указывает, была ли установлена переменная `protection`;
- `bool force_restore = false`, булева переменная, которая указывает, следует ли принудительно восстановить файлы или нет;

- `int flag_force_restore = 0`, флаг, который указывает, была ли установлена переменная `force_restore`;
- `std::string path`, строковая переменная, которая содержит путь к файлу или директории;
- `int flag_path = 0`, флаг, который указывает, была ли установлена переменная `path`;
- `int interval`, целочисленная переменная, которая содержит интервал времени между операциями резервного копирования;
- `int flag_interval = 0`, флаг, который указывает, была ли установлена переменная `interval`;
- `std::string backup_type`, строковая переменная, которая содержит тип резервного копирования;
- `int flag_backup_type = 0`, флаг, который указывает, была ли установлена переменная `backup_type`;
- `std::string hash_algorithm`, строковая переменная, которая содержит используемый алгоритм хэширования;
- `int flag_hash_algorithm = 0`, флаг, который указывает, была ли установлена переменная `hash_algorithm`;
- `std::string notification_channel`, строковая переменная, которая содержит канал уведомлений;
- `int flag_notification_channel = 0`, флаг, который указывает, была ли установлена переменная `notification_channel`.

Структура `buffer_menu` служит для хранения настроек, уже указанных в конфигурационном файле. `buffer_menu` содержит следующие поля:

- `bool protection`, булева переменная, которая указывает, включена ли защита или нет;
- `bool force_restore = false`, булева переменная, которая указывает, следует ли принудительно восстановить файлы или нет;
- `std::string path`, строковая переменная, которая содержит путь к файлу или директории;
- `int interval`, целочисленная переменная, которая содержит интервал времени между операциями резервного копирования;
- `std::string backup_type`, строковая переменная, которая содержит тип резервного копирования;
- `std::string hash_algorithm`, строковая переменная, которая содержит используемый алгоритм хэширования;
- `std::string access_permissions`, строковая переменная, которая содержит информацию о правах доступа;
- `std::string notification_channel`, строковая переменная, которая содержит канал уведомлений.

3.2.2 Описание констант и переменных в программе Back_sfc

Структура `flag_menu` служит для хранения настроек, указанных пользователем. `flag_menu` содержит следующие поля:

- `bool protection`, булева переменная, которая указывает, включена ли защита или нет;
- `bool force_restore`, булева переменная, которая указывает, следует ли принудительно восстановить файлы или нет;
- `std::string path`, строковая переменная, которая содержит путь к файлу или директории;
- `int interval`, целочисленная переменная, которая содержит интервал времени между операциями резервного копирования;
- `std::string backup_type`, строковая переменная, которая содержит тип резервного копирования;
- `std::string hash_algorithm`, строковая переменная, которая содержит используемый алгоритм хэширования;
- `std::string notification_channel`, строковая переменная, которая содержит канал уведомлений;
- `std::string password`, строковая переменная, которая содержит пароль;
- `std::string directory`; строковая переменная, которая содержит директорию.

Структура `flag_menu_changes` служит для отслеживания изменений в настройках, указанных пользователем. `flag_menu_changes` содержит следующие поля:

- `bool protection_changed = false`, булева переменная, которая указывает, была ли изменена переменная `protection`;
- `bool force_restore_changed = false`, булева переменная, которая указывает, была ли изменена переменная `force_restore`;
- `bool path_changed = false`, булева переменная, которая указывает, был ли изменен путь;
- `bool interval_changed = false`, булева переменная, которая указывает, был ли изменен интервал;
- `bool backup_type_changed = false`, булева переменная, которая указывает, был ли изменен тип резервного копирования;
- `bool hash_algorithm_changed = false`, булева переменная, которая указывает, был ли изменен алгоритм хэширования;
- `bool notification_channel_changed = false`, булева переменная, которая указывает, был ли изменен канал уведомлений.

`std::unordered_map<std::string, std::pair<std::string, std::string>>` `file_hashes` – это ассоциативный контейнер, который хранит пары ключ–значение в неупорядоченном порядке. Ключом является строка, а значением – пара строк. Этот контейнер использован для хранения

первоначальных хешей файлов, где ключ – это имя файла, а пара строк – это хеш вычисленный алгоритмом MD5 и алгоритмом SHA–256.

`std::unordered_map<std::string, std::pair<std::string, std::string>>` `new_file_hashes`, – это аналогичный контейнер, который использован для хранения вычисленных хешей при проверке целостности файлов.

`std::atomic<bool>` `check_interval(true)` – это атомарная булева переменная, которая инициализируется значением `true`. Атомарная переменные используются в многопоточном программировании для обеспечения безопасности данных при одновременном доступе из нескольких потоков. Эта переменная использована как флаг для ежеинтервального контроля файлов.

3.3 Описание подпрограмм

В этом подразделе приведено описание разработанных в ходе курсового проектирования процедур и функций.

3.3.1 Описание подпрограмм программы `Front_sfc`

Функция `file_exists()` – проверяет существует ли файл с указанным именем. Она использует системный вызов `stat` для получения информации о файле и возвращает `true`, если файл существует, и `false` в противном случае.

Процедура `update_config_file()` – обновляет конфигурационный файл. Она создает директорию для конфигурационного файла, если она не существует. Затем функция проверяет, существует ли конфигурационный файл, и если нет, создает его. Если файл существует, функция читает его содержимое в JSON–объект. Если файл пуст, функция устанавливает значения по умолчанию. Затем функция обновляет значения в JSON–объекте в соответствии с текущими значениями глобальной переменной `flagMenu`. Наконец, функция записывает обновленный JSON–объект обратно в файл.

Процедура `read_config_file()` – читает конфигурационный файл и обновляет значения в глобальной переменной `bufferMenu`. Она открывает файл, записывает его содержимое в JSON–объект и затем обновляет значения в `bufferMenu` на основе значений в JSON–объекте. Если файл не может быть открыт или его содержимое не может быть прочитано, функция просто возвращает управление.

Процедура `enable_protection()` – включает защиту, устанавливая соответствующие флаги и выводит сообщение "Enable protection".

Процедура `disable_protection()` – отключает защиту, устанавливая соответствующие флаги и выводит сообщение "Disable protection".

Процедура `get_file_info()` – выводит информацию о защищенных файлах, вызывая функцию `file_info()`.

Процедура `force_file_restore()` – принудительно восстанавливает файл, устанавливая соответствующие флаги и выводит сообщение "Force file restoration".

Процедура `show_help_info(po::options_description& desc)` – выводит информацию о доступных опциях, принимая в качестве аргумента описание этих опций.

Процедура `exit_program()` – завершает программу, выводя сообщение "Exit the program".

Процедура `set_file_path(po::variables_map& vm)` – устанавливает путь к файлу или директории для защиты, принимая в качестве аргумента карту переменных.

Процедура `set_file_check_interval(po::variables_map& vm)` – устанавливает интервал проверки файла или директории, принимая в качестве аргумента карту переменных.

Процедура `set_backup_type(po::variables_map& vm)` – устанавливает тип резервного копирования, принимая в качестве аргумента карту переменных.

Процедура `set_hash_algorithm(po::variables_map& vm)` – устанавливает алгоритм хеширования, принимая в качестве аргумента карту переменных.

Процедура `set_notification_channel(po::variables_map& vm)` – устанавливает канал уведомлений, принимая в качестве аргумента карту переменных.

Процедура `show_help_menu()` – очищает экран и выводит справочную информацию, вызывая функцию `show_help()`.

Процедура `file_info()` – читает конфигурационный файл и выводит информацию о файлах и состояниях из структуры `flagMenu`.

Процедура `show_help()` – выводит справочную информацию о функциях программы.

Процедура `ignore_cin()` – игнорирует ввод пользователя, очищая буфер ввода.

Функция `menu_arg_main(int argc, char* argv[])` – определяет была ли запущена программа с дополнительными ключами, и исходя из этого определяет ее дальнейшее поведение.

Процедура `process_command_line_options(int argc, char* argv[])` – вызывается при запуске программы вместе с дополнительным ключем, в зависимости от которого вызывается одноименная процедура, после чего происходит обновление конфигурационного файла.

Процедура `process_menu_options()` – вызывается при запуске программы без дополнительных ключей, выводит графическое меню, и считывает ответ пользователя, после чего обновляет конфигурационный файл.

Процедура `configure_protection()` – является меню, позволяющее настроить основные аспекты необходимые для защиты информации.

3.3.2 Описание подпрограмм программы Back_sfc

Процедура `watch_config_file()` отслеживает изменения в конфигурационном файле, путь к которому передается в качестве аргумента. При обнаружении изменений, функция перезагружает конфигурационный файл. В случае ошибок при инициализации `inotify` или чтении событий, функция выводит сообщения об ошибках.

Функция `parse_config()` открывает и анализирует конфигурационный файл JSON, обновляя глобальные настройки в `flagMenu`. Если какие-либо настройки изменились, функция `send_notification()` вызывается с сообщением о перезагрузке конфигурационного файла. Функция возвращает `true`, если конфигурационный файл был успешно проанализирован, и `false` в противном случае.

Процедура `file_info()` выводит текущие настройки `flagMenu` в лог. Это включает в себя информацию о пути к файлу, настройках восстановления, защиты, интервала, типа резервного копирования, алгоритма хеширования и канала уведомлений.

Процедура `daemonize()` преобразует текущий процесс в демон, создавая дочерний процесс, изменяя маску файла, создавая новый сеанс, изменяя рабочий каталог на корневой и закрывая стандартные файловые дескрипторы.

Функция `is_process_running()` проверяет, запущен ли процесс с заданным именем. Она выполняет команду `pgrep`, передавая ей имя процесса, и проверяет, возвращает ли команда какой-либо вывод. Если команда `pgrep` возвращает какой-либо вывод, это означает, что процесс запущен, и функция возвращает `true`. В противном случае функция возвращает `false`.

Процедура `setup_logger()` настраивает логгер для записи логов в файл `logs.txt` с автоматической промывкой после каждой записи. Общие атрибуты, такие как метка времени и ID строки, добавляются ко всем логам. В конце функции записывается информационное сообщение в лог.

Процедура `send_notification()` отправляет обычное уведомление с помощью команды `notify-send`.

Процедура `send_critical_urgency_notification()` отправляет уведомление с критическим приоритетом, используя флаг `-u critical` в команде `notify-send`.

Процедура `full_backup()` – эта функция принимает исходный каталог и каталог резервного копирования в качестве аргументов. Она создает резервную копию исходного каталога в каталоге резервного копирования. Процесс резервного копирования включает архивацию исходного каталога в файл `tar`, сжатие файла `tar` в файл `gzip`, а затем шифрование файла `gzip` с помощью `gpg`. Затем удаляются исходные файлы `tar` и `gzip`, оставляя только зашифрованный файл `gpg`.

Процедура `restore_backup()` – эта функция проверяет тип резервной копии (полный или дифференциальный) и вызывает соответствующую функцию восстановления. В данном случае она вызывает `full_restore_backup()` для полного резервного копирования и `differential_restore_backup()` для дифференциального резервного копирования.

Процедура `full_restore_backup()` – эта функция принимает исходный каталог и каталог резервного копирования в качестве аргументов. Она восстанавливает полную резервную копию, расшифровывая файл `grp` в файл `gzip`, распаковывая файл `gzip` в файл `tar`, а затем извлекая файл `tar` в исходный каталог. Затем удаляются расшифрованные и извлеченные файлы, оставляя только восстановленные файлы в исходном каталоге.

Процедура `differential_restore_backup(const std::string &source_directory, const std::string &backup_directory)` – эта функция восстанавливает дифференциальный бэкап из указанной директории бэкапа в исходную директорию. Она расшифровывает зашифрованный файл бэкапа, извлекает его и затем копирует содержимое в исходную директорию. Если исходная директория не существует, она создается.

Процедура `force_restore_if_needed()` – эта функция проверяет, нужно ли принудительно восстановить бэкап. Если включена опция принудительного восстановления, она вызывает соответствующую функцию восстановления в зависимости от типа бэкапа. Если защита отключена, она отправляет уведомление о том, что восстановление файлов без включенной защиты невозможно.

Процедура `force_full_restore(const std::string &source_directory, const std::string &backup_directory)` – эта функция принудительно восстанавливает полный бэкап из указанной директории бэкапа в исходную директорию. Она расшифровывает зашифрованный файл бэкапа, извлекает его и затем копирует содержимое в исходную директорию. Если исходная директория существует, все ее содержимое удаляется перед восстановлением.

Процедура `force_differential_restore(const std::string &source_directory, const std::string &backup_directory)` – эта функция принудительно восстанавливает дифференциальный бэкап из указанной директории бэкапа в исходную директорию. Она расшифровывает зашифрованный файл бэкапа, извлекает его и затем копирует содержимое в исходную директорию. Если исходная директория не существует, она создается.

Процедура `calculate_and_store_hashes()` – обходит указанную директорию и для каждого файла вычисляет хеши MD5 и SHA256. Полученные хеши сохраняются в переданную функции мапу, где ключом является путь к файлу, а значением – пара хешей.

Функция `exec()` – выполняет переданную ей команду в оболочке и возвращает результат её выполнения.

Функция `calculate_md5_hash()` – вычисляет и возвращает хеш MD5 для указанного файла.

Функция `calculate_sha256_hash()` – вычисляет и возвращает хеш SHA256 для указанного файла.

Функция `calculate_hashes()` – вычисляет и возвращает пару хешей – MD5 и SHA256 для указанного файла.

Процедура `check_file()` – запускает новый поток, который периодически проверяет целостность файлов в указанной директории. Она вычисляет хеши файлов и сравнивает их с ранее сохраненными хешами. Если обнаруживается новый файл, или хеш файла не совпадает с сохраненным хешем, функция записывает ошибку в лог и отправляет уведомление. Если файл не найден при новом сканировании, функция также отправляет уведомление.

Процедура `stop_check_file()` – останавливает процесс проверки файлов, устанавливая флаг в `false`. Она также отправляет уведомление о том, что проверка файлов была остановлена.

Процедура `main_proc()` – обрабатывает изменения в настройках. Если было изменено принудительное восстановление, и оно включено, функция вызывает `force_restore_if_needed`. Если была изменена защита или путь, и защита включена, функция начинает вычисление хешей, резервное копирование и проверку файлов. Если защита выключена, функция восстанавливает резервную копию и останавливает проверку файлов.

Функция `main()` – это основная функция программы. Она настраивает логгер и выводит сообщение о начале работы. Затем генерируется случайный пароль и сохраняется в глобальной переменной `flagMenu`. Далее функция получает домашний каталог пользователя и устанавливает путь к директории контроля и файлу конфигурации. Затем запускается новый поток, который отслеживает изменения в файле конфигурации. После этого основной поток ожидает завершения потока–наблюдателя.

3.4 Описание файлов

В ходе разработки программного приложения, были созданы файлы нескольких разрешений:

- `.crr` – для реализаций функций и процедур, содержат логику и алгоритмы, которые выполняются при работе программы.
- `.h` – заголовочные файлы, содержат объявления функций, процедур, глобальных переменных, структур, реализация которых содержится в файлах `.crr`.
- `.txt` – конфигурационные файлы, для системы сборки CMake.

3.4.1 Описание файлов программы Front_sfc

CMakeLists.txt – файл конфигурации для системы сборки CMake. Он содержит инструкции и команды, которые используются для создания файлов сборки проекта.

Config.cpp – исходный файл, содержащий реализацию функций необходимых для чтения, проверки и обновления файла конфигурации.

Config.h – заголовочный файл, содержащий объявление функций необходимых для чтения, проверки и обновления файла конфигурации, а также подключение необходимых библиотек.

Main.cpp – основной исходный файл, который содержит функцию являющейся точкой входа в программу.

Menu_func.cpp – исходный файл, содержащий реализацию функций использующихся для определения действий при выборе пользователем соответствующих пунктов меню, или получения ключа при запуске программы.

Menu_func.h – заголовочный файл, содержащий объявление функций использующихся для определения действий при выборе пользователем соответствующих пунктов меню, или получения ключа при запуске программы, а также подключение необходимых библиотек.

Menu.cpp – исходный файл, содержащий реализацию функций меню, а также функцию определяющей использование ключей.

Menu.h – заголовочный файл, содержащий объявление функций меню, а также подключение необходимых библиотек.

3.4.2 Описание файлов программы Back_sfc

CMakeLists.txt – файл конфигурации для системы сборки CMake. Он содержит инструкции и команды, которые используются для создания файлов сборки проекта.

Main.cpp – основной исходный файл, который содержит функцию являющейся точкой входа в программу.

Chek_conf_file.cpp – исходный файл, содержащий реализацию функции создающий отдельный поток, который наблюдает за изменениями конфигурационного файла.

Chek_conf_file.h – заголовочный файл, содержащий объявление функции наблюдающей за изменением конфигурационного файла, а также подключение необходимых библиотек.

Config.cpp – исходный файл, содержащий реализацию функции чтения и представления информации из конфигурационного файла.

Config.h – заголовочный файл, содержащий объявление функций чтения, представления информации из конфигурационного файла, а также

подключение необходимых библиотек. Объявление структур `flag_menu_changes`, `flag_menu`

`Deamon_works.cpp` – исходный файл, содержащий реализацию функции необходимой для работы программы, как программы–демона.

`Deamon_works.h` – заголовочный файл, содержащий объявление функции необходимой для работы программы, как программы–демона, а также подключение необходимых библиотек.

`Logger.cpp` – исходный файл, содержащий реализацию функций для ведения журнала событий.

`Logger.h` – заголовочный файл, содержащий объявление функций для ведения журнала событий, а также подключение необходимых библиотек.

`Notification.cpp` – исходный файл, содержащий реализацию функций для отправки уведомлений.

`Notification.h` – заголовочный файл, содержащий объявление функций для отправки уведомлений, а также подключение необходимых библиотек.

`Backup.cpp` – исходный файл, содержащий реализацию функций для создания резервных копий.

`Encrypted.cpp` – исходный файл, содержащий реализацию функций для шифрования данных.

`Hash.cpp` – исходный файл, содержащий реализацию функций для создания хешей.

`Main_proc.cpp` – исходный файл, содержащий реализацию основных функций обработки.

`Main_proc.h` – заголовочный файл, содержащий объявление основных функций обработки, а также подключение необходимых библиотек. Объявление глобальных переменных `new_file_hashes`, `file_hashes`. Атомарной переменной `check_interval`.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Разработка схем алгоритмов работы приложения

Схема подпрограммы реализующей обновление конфигурационного файла с настройками пользователя – `update_config_file()` из программы `Front_sfc` приведена в приложении Б.

Схема подпрограммы реализующей первичное резервное копирование данных – `full_backup(const std::string &source_directory, const std::string &backup_directory)` из программы `Back_sfc` приведена в приложении В. Подпрограмма `full_backup` принимает ссылку на директорию с защищаемыми файлами, и ссылку на директорию в которой будет храниться резервная копия.

4.2 Описание подпрограммы `to_encrypt_file(const std::string& filename, const std::string& password)`

Шаг 1. Создание нового имени для зашифрованного файла путем добавления расширения «.enc» к имени исходного файла.

Шаг 2. Проверка существования файла с таким именем. Если такой файл существует, он удаляется.

Шаг 3. Открытие исходного файла для чтения и создание нового файла для записи зашифрованных данных.

Шаг 4. Определение типа шифра как AES–256–CBC и типа дайджеста как SHA–256.

Шаг 5. Создание ключа и вектора инициализации на основе введенного пароля с использованием функции `EVP_BytesToKey`.

Шаг 6. Инициализация контекста шифрования и установка параметров шифрования.

Шаг 7. Чтение исходного файла блоками по 4096 байт и шифрование каждого блока с записью результата в выходной файл.

Шаг 8. После того как все данные прочитаны и зашифрованы, вызов `EVP_EncryptFinal_ex` для обработки последнего блока данных, если он меньше 4096 байт.

Шаг 9. Освобождение контекста шифрования.

4.3 Описание подпрограммы `main_proc(flag_menu_changes flagMenuChanges)`

Шаг 1. Проверка, изменилось ли значение `force_restore_changed` в структуре `flagMenuChanges`.

Шаг 2. Если `force_restore_changed` изменилось, проверка, установлен ли `force_restore` в структуре `flagMenu`.

Шаг 3. Если `force_restore` установлен, вызывается функция `force_restore_if_needed()`.

Шаг 4. Проверка, изменились ли значения `protection_changed` или `path_changed` в структуре `flagMenuChanges`.

Шаг 5. Если `protection_changed` или `path_changed` изменились, проверка, установлен ли `protection` в структуре `flagMenu`.

Шаг 6. Если `protection` установлен, вызываются функции `begin_hash()`, `begin_backup()`, и `check_file()` с `path` из структуры `flagMenu` в качестве аргумента.

Шаг 7. Если `protection` не установлен, вызываются функции `restore_backup()` и `stop_check_file()`.

Шаг 8. Если `path_changed` изменилось, функции `restore_backup()` и `stop_check_file()` вызываются снова.

Шаг 9. После вызова `restore_backup()` и `stop_check_file()`, вызываются функции `begin_hash()`, `begin_backup()`, и `check_file()` с `path` из структуры `flagMenu` в качестве аргумента.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Результатом выполнения курсового проекта являются две программы для платформы Linux – Front_sfc, Back_sfc.

5.1 Требования к программному обеспечению

Операционная система Linux, с установленными следующими компонентами:

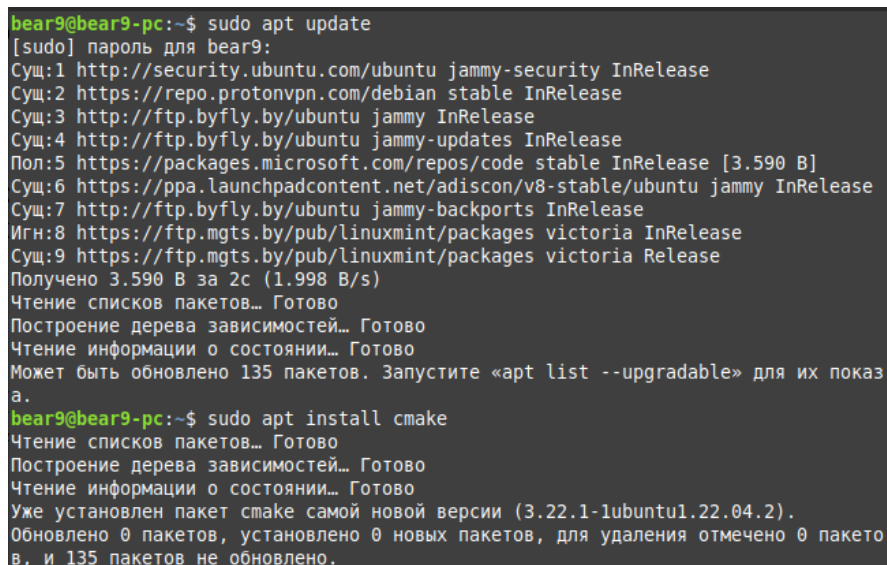
- CMake версии 3.22 или выше;
- компилятор C++, поддерживающий стандарт C++17;
- набор библиотек Boost;
- библиотека Jsoncpp;
- библиотека OpenSSL;
- утилита pkg-config.

CMake – это кросс-платформенная система автоматизации сборки, которая используется для генерации файлов сборки на основе информации, найденной в CMakeLists.txt файлах.

Для установки CMake на Ubuntu или Debian, можно использовать пакетный менеджер apt. Выполните в терминале следующие команды:

```
sudo apt update  
sudo apt install cmake
```

Пример использования приведен на рисунке 5.1. В связи с тем, что на моей системе CMake последней версии уже установлен, пакеты не обновлены.



```
bear9@bear9-pc:~$ sudo apt update  
[sudo] пароль для bear9:  
Сущ:1 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Сущ:2 https://repo.protonvpn.com/debian stable InRelease  
Сущ:3 http://ftp.byfly.by/ubuntu jammy InRelease  
Сущ:4 http://ftp.byfly.by/ubuntu jammy-updates InRelease  
Пол:5 https://packages.microsoft.com/repos/code stable InRelease [3.590 B]  
Сущ:6 https://ppa.launchpadcontent.net/adiscon/v8-stable/ubuntu jammy InRelease  
Сущ:7 http://ftp.byfly.by/ubuntu jammy-backports InRelease  
Игн:8 https://ftp.mgts.by/pub/linuxmint/packages victoria InRelease  
Сущ:9 https://ftp.mgts.by/pub/linuxmint/packages victoria Release  
Получено 3.590 В за 2с (1.998 В/с)  
Чтение списков пакетов... Готово  
Построение дерева зависимостей... Готово  
Чтение информации о состоянии... Готово  
Может быть обновлено 135 пакетов. Запустите «apt list --upgradable» для их показ  
а.  
bear9@bear9-pc:~$ sudo apt install cmake  
Чтение списков пакетов... Готово  
Построение дерева зависимостей... Готово  
Чтение информации о состоянии... Готово  
Уже установлен пакет cmake самой новой версии (3.22.1-1ubuntu1.22.04.2).  
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов,  
и 135 пакетов не обновлено.
```

Рисунок 5.1 – Установка CMake

Для установки CMake на Arch или Manjaro, можно использовать пакетный менеджер pacman. Выполните в терминале следующие команды:

```
sudo pacman -Syu
sudo pacman -S cmake
```

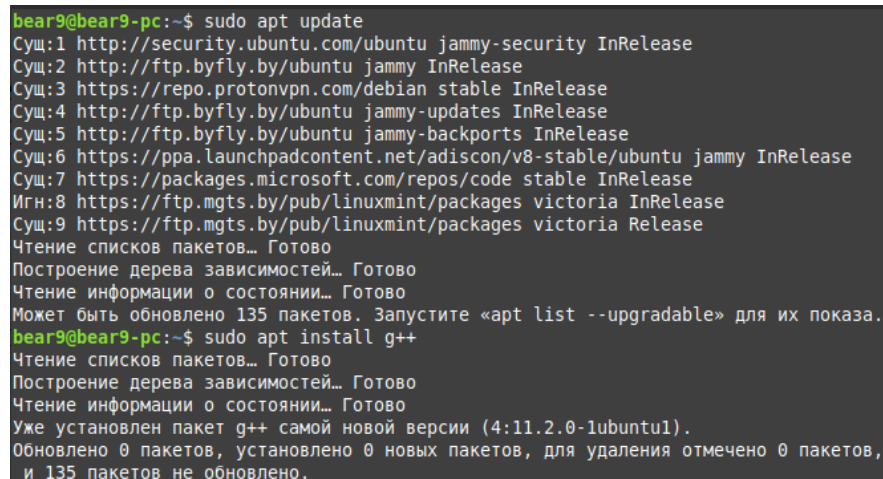
Это обновит список пакетов в вашем менеджере пакетов и установит CMake. После можно проверить версию CMake для этого можно использовать следующую команду:

```
cmake --version
```

Для установки компилятора g++, поддерживающего стандарт C++17, на Ubuntu или Debian, можно использовать пакетный менеджер apt. Выполните в терминале следующие команды:

```
sudo apt update
sudo apt install g++
```

Пример установки приведен на рисунке 5.2.



```
bear9@bear9-pc:~$ sudo apt update
Сущ:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Сущ:2 http://ftp.byfly.by/ubuntu jammy InRelease
Сущ:3 https://repo.protonvpn.com/debian stable InRelease
Сущ:4 http://ftp.byfly.by/ubuntu jammy-updates InRelease
Сущ:5 http://ftp.byfly.by/ubuntu jammy-backports InRelease
Сущ:6 https://ppa.launchpadcontent.net/adiscon/v8-stable/ubuntu jammy InRelease
Сущ:7 https://packages.microsoft.com/repos/code stable InRelease
Игн:8 https://ftp.mgts.by/pub/linuxmint/packages victoria InRelease
Сущ:9 https://ftp.mgts.by/pub/linuxmint/packages victoria Release
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 135 пакетов. Запустите «apt list --upgradable» для их показа.
bear9@bear9-pc:~$ sudo apt install g++
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет g++ самой новой версии (4:11.2.0-1ubuntu1).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов,
и 135 пакетов не обновлено.
```

Рисунок 5.2 – Установка g++

Для установки g++ на Arch или Manjaro, можно использовать пакетный менеджер pacman. Выполните в терминале следующие команды:

```
sudo pacman -Syu
sudo pacman -S gcc
```

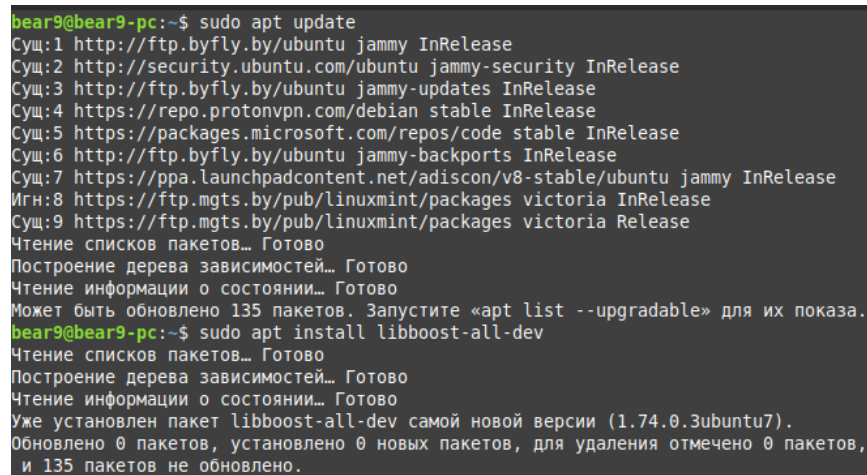
Boost. Это набор библиотек C++, которые расширяют функциональность языка. Для front_sfc требуется компонент program_options,

а для `back_sfc` требуются компоненты `log_setup`, `log`, `filesystem`, `regex`, `thread`, `date_time`, `system`, `chrono`, `atomic`.

Для установки библиотеки `boost` на `Ubuntu` или `Debian`, можно использовать пакетный менеджер `apt`. Выполните в терминале следующие команды:

```
sudo apt update
sudo apt install libboost-all-dev
```

Пример установки приведен на рисунке 5.3.



```
bear9@bear9-pc:~$ sudo apt update
Сущ:1 http://ftp.byfly.by/ubuntu jammy InRelease
Сущ:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Сущ:3 http://ftp.byfly.by/ubuntu jammy-updates InRelease
Сущ:4 https://repo.protonvpn.com/debian stable InRelease
Сущ:5 https://packages.microsoft.com/repos/code stable InRelease
Сущ:6 http://ftp.byfly.by/ubuntu jammy-backports InRelease
Сущ:7 https://ppa.launchpadcontent.net/adiscon/v8-stable/ubuntu jammy InRelease
Игн:8 https://ftp.mgts.by/pub/linuxmint/packages victoria InRelease
Сущ:9 https://ftp.mgts.by/pub/linuxmint/packages victoria Release
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 135 пакетов. Запустите «apt list --upgradable» для их показа.
bear9@bear9-pc:~$ sudo apt install libboost-all-dev
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет libboost-all-dev самой новой версии (1.74.0.3ubuntu7).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов,
и 135 пакетов не обновлено.
```

Рисунок 5.3 – Установка библиотеки Boost

Для проверки корректной установки, можно выполнить следующие команду, она проверит, и выведет версии найденных компонентов:

```
ldconfig -p | grep boost_log_setup && ldconfig -p | grep
boost_log && ldconfig -p | grep boost_filesystem && ldconfig -p |
grep boost_regex && ldconfig -p | grep boost_thread && ldconfig -
p | grep boost_date_time && ldconfig -p | grep boost_system &&
ldconfig -p | grep boost_chrono && ldconfig -p | grep boost_atomic
```

Для установки `boost` на `Arch` или `Manjaro`, можно использовать пакетный менеджер `pacman`. Выполните в терминале следующие команды:

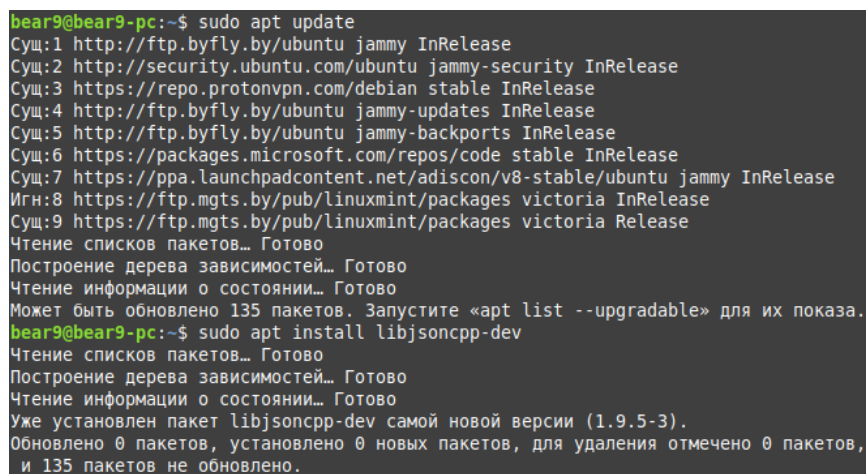
```
sudo pacman -Syu
sudo pacman -S boost
```

`Jsoncpp` – это `C++` библиотека, которая позволяет обрабатывать данные в формате `JSON`.

Для установки библиотеки `jsoncpp` на `Ubuntu` или `Debian`, можно использовать пакетный менеджер `apt`. Выполните в терминале следующие команды:

```
sudo apt update
sudo apt install libjsoncpp-dev
```

Пример установки приведен на рисунке 5.4.



```
bear9@bear9-pc:~$ sudo apt update
Сущ:1 http://ftp.byfly.by/ubuntu jammy InRelease
Сущ:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Сущ:3 https://repo.protonvpn.com/debian stable InRelease
Сущ:4 http://ftp.byfly.by/ubuntu jammy-updates InRelease
Сущ:5 http://ftp.byfly.by/ubuntu jammy-backports InRelease
Сущ:6 https://packages.microsoft.com/repos/code stable InRelease
Сущ:7 https://ppa.launchpadcontent.net/adiscon/v8-stable/ubuntu jammy InRelease
Игн:8 https://ftp.mgts.by/pub/linuxmint/packages victoria InRelease
Сущ:9 https://ftp.mgts.by/pub/linuxmint/packages victoria Release
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 135 пакетов. Запустите «apt list --upgradable» для их показа.
bear9@bear9-pc:~$ sudo apt install libjsoncpp-dev
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет libjsoncpp-dev самой новой версии (1.9.5-3).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов,
и 135 пакетов не обновлено.
```

Рисунок 5.4 – Установка библиотеки jsoncpp

Для проверки наличия библиотеки выполняется следующая команда:

```
sudo ldconfig -p | grep jsoncpp
```

Для установки jsoncpp на Arch или Manjaro, можно использовать пакетный менеджер pacman. Выполните в терминале следующие команды:

```
sudo pacman -Syu
sudo pacman -S jsoncpp
```

OpenSSL. Это набор инструментов для безопасного взаимодействия с сетью и работой с криптографией.

Для установки библиотеки openssl на Ubuntu или Debian, можно использовать пакетный менеджер apt. Выполните в терминале следующие команды:

```
sudo apt update
sudo apt install libssl-dev
```

Пример установки библиотеки приведен на рисунке 5.5.

```

bear9@bear9-pc:~$ sudo apt update
Сущ:1 http://ftp.byfly.by/ubuntu jammy InRelease
Сущ:2 https://repo.protonvpn.com/debian stable InRelease
Сущ:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Сущ:4 http://ftp.byfly.by/ubuntu jammy-updates InRelease
Сущ:5 http://ftp.byfly.by/ubuntu jammy-backports InRelease
Сущ:6 https://ppa.launchpadcontent.net/adiscon/v8-stable/ubuntu jammy InRelease
Пол:7 https://packages.microsoft.com/repos/code stable InRelease [3.590 B]
Игн:8 https://ftp.mgts.by/pub/linuxmint/packages victoria InRelease
Сущ:9 https://ftp.mgts.by/pub/linuxmint/packages victoria Release
Получено 3.590 В за 2с (2.379 В/с)
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 135 пакетов. Запустите «apt list --upgradable» для их показа.
bear9@bear9-pc:~$ sudo apt install libssl-dev
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет libssl-dev самой новой версии (3.0.2-0ubuntu1.15).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов,
и 135 пакетов не обновлено.

```

Рисунок 5.5 – Установка библиотеки OpenSSL

Для проверки наличия библиотеки выполняется следующая команда:

```
sudo ldconfig -p | grep openssl
```

Для установки openssl на Arch или Manjaro, можно использовать пакетный менеджер pacman. Выполните в терминале следующие команды:

```
sudo pacman -Syu
sudo pacman -S openssl
```

Pkg-config. Это система, которая помогает управлять и обнаруживать библиотеки и пакеты, используемые в проекте.

Для установки pkg-config на Ubuntu или Debian, можно использовать пакетный менеджер apt. Выполните в терминале следующие команды:

```
sudo apt update
sudo apt install pkg-config
```

Пример установки pkg-config приведен на рисунке 5.6.

```

bear9@bear9-pc:~$ sudo apt update
Сущ:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Сущ:2 http://ftp.byfly.by/ubuntu jammy InRelease
Сущ:3 https://repo.protonvpn.com/debian stable InRelease
Сущ:4 http://ftp.byfly.by/ubuntu jammy-updates InRelease
Сущ:5 https://ppa.launchpadcontent.net/adiscon/v8-stable/ubuntu jammy InRelease
Сущ:6 http://ftp.byfly.by/ubuntu jammy-backports InRelease
Пол:7 https://packages.microsoft.com/repos/code stable InRelease [3.590 B]
Игн:8 https://ftp.mgts.by/pub/linuxmint/packages victoria InRelease
Сущ:9 https://ftp.mgts.by/pub/linuxmint/packages victoria Release
Получено 3.590 В за 2с (1.832 В/с)
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 135 пакетов. Запустите «apt list --upgradable» для их показа.
bear9@bear9-pc:~$ sudo apt install pkg-config
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет pkg-config самой новой версии (0.29.2-1ubuntu3).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов,
и 135 пакетов не обновлено.

```

Рисунок 5.6 – Установка pkg-config

Для установки pkg-config на Arch или Manjaro, можно использовать пакетный менеджер pacman. Выполните в терминале следующие команды:

```

sudo pacman -Syu
sudo pacman -S pkg-config

```

5.2 Описание процесса инсталляции

Для инсталляции программы для контроля изменений и целостности группы файлов, вам необходимо из приложенного диска скопировать на вашу машину два каталога: front_sfc, back_sfc. Далее описан процесс установки программы front_sfc:

1 Перейдите в каталог front_sfc:

```
cd front_sfc
```

2 Создайте каталог для сборки и перейдите в него:

```
mkdir build
cd build
```

3 Запустите CMake для генерации Makefile:

```
cmake ..
```

4 Запустите make для сборки проекта:

```
make
```

Проделайте аналогичные действия для установки программы back_sfc.

Пример выполнения команд необходимых для установки программ представлен на рисунках 5.7, 5.8 для программы front_sfc и back_sfc соответственно.

```
bear9@bear9-pc:~/Рабочий стол/диск$ cd front_sfc/
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc$ mkdir build
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc$ cd build/
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ cmake ..
-- The C compiler identification is GNU 11.4.0
-- The CXX compiler identification is GNU 11.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Boost: /usr/lib/x86_64-linux-gnu/cmake/Boost-1.74.0/BoostConfig.cmake (found version "1.74.0") found components: program_options
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.2")
-- Checking for module 'jsoncpp'
-- Found jsoncpp, version 1.9.5
-- Configuring done
-- Generating done
-- Build files have been written to: /home/bear9/Рабочий стол/диск/front_sfc/build
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ make
[ 20%] Building CXX object CMakeFiles/front_sfc.dir/main.cpp.o
[ 40%] Building CXX object CMakeFiles/front_sfc.dir/menu.cpp.o
[ 60%] Building CXX object CMakeFiles/front_sfc.dir/config.cpp.o
[ 80%] Building CXX object CMakeFiles/front_sfc.dir/menu_func.cpp.o
[100%] Linking CXX executable front_sfc
[100%] Built target front_sfc
```

Рисунок 5.7 – Инсталляция программы front_sfc

```
bear9@bear9-pc:~/Рабочий стол/диск$ cd back_sfc/
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc$ mkdir build
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc$ cd build/
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc/build$ cmake ..
-- The C compiler identification is GNU 11.4.0
-- The CXX compiler identification is GNU 11.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Boost: /usr/lib/x86_64-linux-gnu/cmake/Boost-1.74.0/BoostConfig.cmake (found version "1.74.0") found components: log_setup log filesystem regex thread date_time system chrono atomic filesystem
-- Found OpenSSL: /usr/lib/x86_64-linux-gnu/libcrypto.so (found version "3.0.2")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/bear9/Рабочий стол/диск/back_sfc/build
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc/build$ make
[  9%] Building CXX object CMakeFiles/back_sfc.dir/src/main.cpp.o
[ 18%] Building CXX object CMakeFiles/back_sfc.dir/src/configuration/config.cpp.o
[ 27%] Building CXX object CMakeFiles/back_sfc.dir/src/configuration/chek_conf_file.cpp.o
[ 36%] Building CXX object CMakeFiles/back_sfc.dir/src/daemon/daemon works.cpp.o
[ 45%] Building CXX object CMakeFiles/back_sfc.dir/src/other/logger.cpp.o
[ 54%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/main_proc.cpp.o
[ 63%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/backup.cpp.o
[ 72%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/hash.cpp.o
[ 81%] Building CXX object CMakeFiles/back_sfc.dir/src/other/notification.cpp.o
[ 90%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/encrypted.cpp.o
[100%] Linking CXX executable back_sfc
[100%] Built target back_sfc
```

Рисунок 5.8 – Инсталляция программы back_sfc

5.3 Описание работы с программой

В этом подразделе приводится описание пользовательского интерфейса, режимов работы программы, последовательность действий пользователя.

5.3.1 Описание пользовательского интерфейса

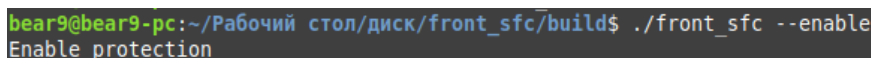
Управление и настройка основной программой `back_sfc`, осуществляется через утилиту `front_sfc`. Взаимодействие с утилитой `front_sfc` может осуществляться двумя способами – ключами, или примитивным графическим интерфейсом.

Рассмотрим взаимодействие с использованием ключей. Для этого утилита запускается, из каталога `build`, в виде:

```
./front_sfc <ключ> <параметр>
```

Треугольные скобки опускаются, параметр указывается при необходимости. Ключом являются следующие строки, приведены с описанием:

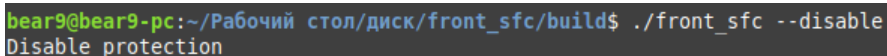
1 `enable`. Включение защиты, с этого момента начнется наблюдение за файлами, будет создана резервная копия. Результат выполнения команды представлен на рисунке 5.9.



```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --enable
Enable protection
```

Рисунок 5.9 – Включение защиты

2 `disable`. Выключение защиты, снятие контроля над файлами, автоматическое восстановление измененных, удаленных файлов. Результат выполнения команды представлен на рисунке 5.10.



```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --disable
Disable protection
```

Рисунок 5.10 – Снятие контроля над файлами

3 `file-info`. Окно справки, в нем содержится информация о пути к защищаемой директории, состояние основной программы, интервал между проверками хэш-сумм, тип восстановления с резервной копии, предпочтительный алгоритм хеширования, настройки уведомлений. Результат выполнения команды представлен на рисунке 5.11.

```

bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --file
Get information about protected files

*****
* File information:                               *
* Path: /home/bear9/testDir                       *
* Protection: Disabled                           *
* Interval: 1                                     *
* Backup Type: full                               *
* Hash Algorithm: MD5                             *
* Notification Channel: system                    *
*****

```

Рисунок 5.11 – Окно информации о настройках

4 `force-restore`. Принудительное восстановление файлов, согласно выбранного типу. Восстановление происходит только с включенной защитой. Результат выполнения команды представлен на рисунке 5.12.

```

bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --force
Force file restoration.

```

Рисунок 5.12 – Принудительное восстановление файлов

5 `help`. Открытие справки, включающей в себя описание ключей и дополнительные требования по вводу. Результат выполнения команды представлен на рисунке 5.13.

```

bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --help
Options:
  --enable           Enable protection
  --disable          Disable protection
  --file-info        Get information about protected files
  --force-restore    Force file restoration
  --help             Show help
  --exit             Exit the program
  --path arg         Specify the file path
  --interval arg     Specify the time file check interval (1, 2, ...180
                    , minute)
  --backup-type arg  Specify the backup type (full/differential
  --hash-algorithm arg Specify the hashing algorithm (MD5/SHA256)
  --notification-channel arg Specify the notification channel (no/system)

```

Рисунок 5.13 – Окно справки

6 `path <string>`. Установка пути к директории, требующей защиты. Результат выполнения команды представлен на рисунке 5.14.

```

bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --path /home/bear9
/testDir
Path: /home/bear9/testDir

```

Рисунок 5.14 – Установка пути к директории

7 interval <int>. Установка временного интервала в минутах проверки хэш–сумм файлов. Результат выполнения команды представлен на рисунке 5.15.

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --interval 5
Interval: 5 minute
```

Рисунок 5.15 – Установка временного интервала

8 backup-type <string>. Установка типа восстановления после резервного копирования. Результат выполнения команды представлен на рисунке 5.16.

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --backup full
Backup Type: full
```

Рисунок 5.16 – Установка типа восстановления

9 hash-algorithm <string>. Установка приоритетного алгоритма для вычисления, и сверки хэш–сумм. Результат выполнения команды представлен на рисунке 5.17.

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --hash SHA256
Hash Algorithm: SHA256
```

Рисунок 5.17 – Выбор хэш–алгоритма

10 notification-channel <string>. Установка канала уведомлений, или их отключение. Результат выполнения команды представлен на рисунке 5.18.

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --not no
Notification Channel: no
```

Рисунок 5.18 – Установка канала уведомлений

При введении несуществующего ключа или недоступного аргумента ключа, возникнет исключение – рисунок 5.19, 5.20 соответственно.

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --fjjj
Error: unrecognised option '--fjjj'
Use --help to see available commands.
```

Рисунок 5.19 – Исключение при введении несуществующего ключа

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --hash SHA1
Invalid input. Please enter 'MD5' or 'SHA256'.
```

Рисунок 5.20 – Исключение при введении недоступного аргумента ключа

Взаимодействие при помощи графического интерфейса осуществляется при запуске утилиты, из каталога build, без дополнительных ключей:

```
./front_sfc
```

В результате этого, предстает основное меню, рисунок 5.21.

```
=====
|| 1. Enable protection      ||
|| 2. Disable protection    ||
|| 3. Get information about  ||
|| 4. Force file restoration ||
|| 5. Configure protection  ||
|| 6. Show help             ||
|| 7. Exit the program      ||
=====
Enter your choice: █
```

Рисунок 5.21 – Основное меню

Описание пунктов меню:

1 Enable protection. Включение защиты. Выбора этого пункта означает что при выходе с утилиты, программа back_sfc в соответствии с выставленными настройками, или настройками по умолчанию начнет свою деятельность. Результат выполнения пункта представлен на рисунке 5.22.

```
=====
|| 1. Enable protection      ||
|| 2. Disable protection    ||
|| 3. Get information about  ||
|| 4. Force file restoration ||
|| 5. Configure protection  ||
|| 6. Show help             ||
|| 7. Exit the program      ||
=====
Enter your choice: 1
Enable protection
█
```

Рисунок 5.22 – Включение защиты

2 Disable protection. Снятие защиты. Охраняемый каталог будет возвращен в свое первоначальное состояние, файлы в каталоге больше не будут

подвергаться мониторингу выявляющему изменения, зашифрованная резервная копия будет удалена. Оставшиеся пользовательские настройки будут сохранены. Результат выполнения пункта представлен на рисунке 5.23.

```
=====
|| 1. Enable protection      ||
|| 2. Disable protection     ||
|| 3. Get information about files ||
|| 4. Force file restoration ||
|| 5. Configure protection   ||
|| 6. Show help              ||
|| 7. Exit the program       ||
=====
Enter your choice: 2
Disable protection
█
```

Рисунок 5.23 – Снятие защиты

3 Get information about files. Просмотр текущих настроек, таких как:

- путь к наблюдаемой директории;
- состояние защиты;
- интервал проверки хеш–сумм файлов;
- вид резервного копирования;
- приоритетный алгоритм при сравнении хеш–сумм;
- канал уведомлений.

Результат выполнения пункта представлен на рисунке 5.24.

```
=====
|| 1. Enable protection      ||
|| 2. Disable protection     ||
|| 3. Get information about files ||
|| 4. Force file restoration ||
|| 5. Configure protection   ||
|| 6. Show help              ||
|| 7. Exit the program       ||
=====
Enter your choice: 3
Get information about protected files

*****
* File information:          *
* Path: /home/bear9/testDir
* Protection: Disabled
* Interval: 5
* Backup Type: full
* Hash Algorithm: SHA256
* Notification Channel: no
*****
```

Рисунок 5.24 – Просмотр информации о настройках

4 Force file restoration. Принудительное восстановление данных с резервной копии, применяется когда необходимо обратиться к первоначальным файлам, после их изменения, при этом не прекращая защиты. Результат выполнения пункта представлен на рисунке 5.25.

```
=====
|| 1. Enable protection      ||
|| 2. Disable protection    ||
|| 3. Get information about  ||
|| 4. Force file restoration ||
|| 5. Configure protection  ||
|| 6. Show help             ||
|| 7. Exit the program      ||
=====
Enter your choice: 4
Force file restoration.
█
```

Рисунок 5.25 – Принудительное восстановление файлов

5 Configure protection. Настроить защиту. При выборе этого пункта, откроется подменю. Подменю будет описано ниже. Результат выполнения пункта представлен на рисунке 5.26.

```
=====
|| 1. Specify the file path  ||
|| 2. Specify the file check ||
|| 3. Specify the backup type||
|| 4. Specify the hashing algorithm||
|| 5. Specify the notification channel||
|| 6. Return to main menu   ||
=====
Enter your choice: █
```

Рисунок 5.26 – Открытие подменю

6 Show help. Показ справки, содержащей информацию о всех пунктах, и разделах меню. Результат выполнения пункта представлен на рисунке 5.27.

```

*****
*                               Help Information:                               *
* Enable protection: This will enable the file protection.                       *
* Disable protection: This will disable the file protection.                     *
* Get information about protected files: This will display information about the   *
*                               protected files.                               *
* Force file restoration: This will force the restoration of the protected files. *
* Configure protection:                                                       *
** Specify the file path: Specify the path to the file or directory to be protected. *
** Specify the file check interval: Specify the interval in minute (1-180) at which the *
*                               file or directory should be checked.             *
** Specify the backup type: The type of backup to be performed (full/differential). *
** Specify the hashing algorithm: Specify the hashing algorithm to be used (MD5/SHA256). *
** Specify the notification channel: Specify the channel through which notifications *
*                               should be sent (no/system).                     *
*****

```

Рисунок 5.27 – Окно справки

7 Exit the program. Выход из утилиты front_sfc. Корректный необходим чтобы сохранить настройки ваших параметров, и передачи в back_sfc. Результат выполнения пункта представлен на рисунке 5.28.

```

=====
|| 1. Enable protection ||
|| 2. Disable protection ||
|| 3. Get information about files ||
|| 4. Force file restoration ||
|| 5. Configure protection ||
|| 6. Show help ||
|| 7. Exit the program ||
=====
Enter your choice: 7
Exit the program

```

Рисунок 5.28 – Выход из программы

Описание пунктов подменю, изображённого на рисунке 5.26:

1 Specify the file path. Указать путь к охраняемой директории. После указания пути, будет произведена проверка на его существование. Значением по умолчанию является домашняя директория. Результат выполнения пункта представлен на рисунке 5.29.

```

=====
|| 1. Specify the file path ||
|| 2. Specify the file check interval ||
|| 3. Specify the backup type ||
|| 4. Specify the hashing algorithm ||
|| 5. Specify the notification channel ||
|| 6. Return to main menu ||
=====
Enter your choice: 1
Enter the file path: /ghjk/kjhbv
The file or directory does not exist.

```

Рисунок 5.29 – Установка пути

2 Specify the file check interval. Указать интервал сверки хеш–сумм файлов, в охраняемой директории, а также выявление новых/удаленных файлов, при этом выводится системное уведомление. Интервал указывается в минутах, в пределах от 1 до 180 минут. По умолчанию интервал выставлен на проверку каждые 3 минуты. Результат выполнения пункта представлен на рисунке 5.30.

```
=====
|| 1. Specify the file path      ||
|| 2. Specify the file check interval ||
|| 3. Specify the backup type   ||
|| 4. Specify the hashing algorithm ||
|| 5. Specify the notification channel ||
|| 6. Return to main menu      ||
=====
Enter your choice: 2
Enter the file check interval, 1 <= {} <= 180 minute: 1
```

Рисунок 5.30 – Указание временного интервала

3 Specify the backup type. Указать тип резервного копирования, full (полное), или differential (частичное). Отличия заключается в порядке восстановления файлов. При полном охраняемая директория будет стерта, и создана из резервной копии, при частичном дополнена резервной копией, то есть, новые, добавленные файлы в период действия защиты останутся, предупреждения о существовании таковых выводится не перестанут. По умолчанию тип указан как полное восстановление. Результат выполнения пункта представлен на рисунке 5.31.

```
=====
|| 1. Specify the file path      ||
|| 2. Specify the file check interval ||
|| 3. Specify the backup type   ||
|| 4. Specify the hashing algorithm ||
|| 5. Specify the notification channel ||
|| 6. Return to main menu      ||
=====
Enter your choice: 3
Enter the backup type (full/differential): full
```

Рисунок 5.31 – Выбор типа резервного копирования

4 Specify the hashing algorithm. Указать приоритетный алгоритм для сравнения хеш–сумм файлов, SHA256 или MD5. По умолчанию приоритет отдается алгоритму MD5. Результат выполнения пункта представлен на рисунке 5.32.


```
=====
|| 1. Specify the file path ||
|| 2. Specify the file check interval ||
|| 3. Specify the backup type ||
|| 4. Specify the hashing algorithm ||
|| 5. Specify the notification channel ||
|| 6. Return to main menu ||
=====
Enter your choice: 4
Enter the hashing algorithm (MD5/SHA256): MD5
|
```

Рисунок 5.31 – Выбор хэш-алгоритма

5 Specify the notification channel. Настроить канал уведомлений – no/system. В текущей версии программы доступна возможность использовать системные уведомления, или отключить их (за исключением критических). По умолчанию уведомление представлены в виде системных. Результат выполнения пункта представлен на рисунке 5.32.

```
=====
|| 1. Specify the file path ||
|| 2. Specify the file check interval ||
|| 3. Specify the backup type ||
|| 4. Specify the hashing algorithm ||
|| 5. Specify the notification channel ||
|| 6. Return to main menu ||
=====
Enter your choice: 5
Enter the notification channel (no/system): system
|
```

Рисунок 5.32 – Установка канала уведомлений

6 Return to main menu. Вернуться к основному меню.

5.3.2 Описание режимов работы

Программа имеет два основных режима работы – режим мониторинга и защиты, и режим ожидания.

Режим ожидания характеризуется выключенной настройкой защитой, или отсутствием файла конфигурации. Последний случай возникает при первом запуске программы на машине, в следующей последовательности: back_sfc, front_sfc. В результате чего back_sfc ожидает настроек. Со стороны back_sfc режим ожидания представляет собой ожидание изменения в конфигурационном файле состояния защиты.

Режим мониторинга и защиты характеризуется включенной настройкой защиты. Представляет из себя интервальную проверку хеш-сумм файлов в охраняемой директории, проверку их наименований, на случай удаления, или добавления неизвестных. Критические изменения сопровождаются системными уведомлениями.

При переходе из режима ожидания, в режим мониторинга и защиты, а также в обратном порядке, происходят действия описанные в пункте 4.3 настоящей пояснительной записки.

5.3.3 Последовательность действий пользователя

Для использования программы для контроля изменений и целостности группы файлов, необходимо:

- 1 Проверить соответствие необходимым требованиям к программному обеспечению. В случае несоответствия произойдет установка необходимого программного обеспечения.

- 2 Установка программ front_sfc, back_sfc.

- 3 Запуск программы front_sfc.

- 4 Настройка параметров посредством интерфейса, описанного в подпункте 5.3.1 настоящей пояснительной записки.

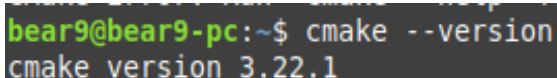
- 4 Запуск программы back_sfc.

- 5 При необходимости изменение настроек мониторинга, или режима работы программы посредством повтора пункта 3, 4.

Программа front_sfc завершается при входе соответствующего пункта в главное меню. Программа back_sfc продолжает работу в фоновом режиме, до перезагрузки машины. При последующей загрузке машины, будет достаточно выполнить только 4 шаг.

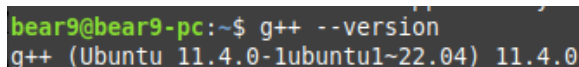
Возможный сценарий использования программы изложен ниже.

Проверка установленных библиотек, программ. Пример выполнения команд для проверки, приведены на рисунках 5.33–5.37.



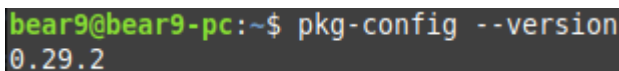
```
bear9@bear9-pc:~$ cmake --version
cmake version 3.22.1
```

Рисунок 5.33 – Проверка CMake



```
bear9@bear9-pc:~$ g++ --version
g++ (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
```

Рисунок 5.34 – Проверка компилятора



```
bear9@bear9-pc:~$ pkg-config --version
0.29.2
```

Рисунок 5.35 – Проверка pkg-config

```

bear9@bear9-pc:~$ ldconfig -p | grep boost log setup && ldconfig -p | grep boost log && ldconfig -p | grep boost
filesystem && ldconfig -p | grep boost regex && ldconfig -p | grep boost thread && ldconfig -p | grep boost dat
e_time && ldconfig -p | grep boost system && ldconfig -p | grep boost chrono && ldconfig -p | grep boost atomic
libboost_log_setup.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_log_setup.so.1.74.0
libboost_log_setup.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_log_setup.so
libboost_log_setup.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_log_setup.so.1.74.0
libboost_log_setup.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_log_setup.so
libboost_log.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_log.so.1.74.0
libboost_log.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_log.so
libboost_filesystem.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_filesystem.so.1.74.0
libboost_filesystem.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_filesystem.so
libboost_regex.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_regex.so.1.74.0
libboost_regex.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_regex.so
libboost_thread.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_thread.so.1.74.0
libboost_thread.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_thread.so
libboost_date_time.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_date_time.so.1.74.0
libboost_date_time.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_date_time.so
libboost_system.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_system.so.1.74.0
libboost_system.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_system.so
libboost_chrono.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_chrono.so.1.74.0
libboost_chrono.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_chrono.so
libboost_atomic.so.1.74.0 (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_atomic.so.1.74.0
libboost_atomic.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libboost_atomic.so

```

Рисунок 5.35 – Проверка Boost

```

bear9@bear9-pc:~$ sudo ldconfig -p | grep jsoncpp
libjsoncpp.so.25 (libc6,x86_64) => /lib/x86_64-linux-gnu/libjsoncpp.so.25
libjsoncpp.so (libc6,x86_64) => /lib/x86_64-linux-gnu/libjsoncpp.so

```

Рисунок 5.36 – Проверка jsoncpp

```

bear9@bear9-pc:~$ sudo ldconfig -p | grep openssl
libevent_openssl-2.1.so.7 (libc6,x86_64) => /lib/x86_64-li

```

Рисунок 3.37 – Проверка OpenSSL

После проверки библиотек, и до установки в случае необходимости, согласно алгоритму установки, приведенному выше, устанавливается программа front_src, back_sfc. Результат выполнения команд, приведен на рисунках 3.38, 3.39 соответственно.

```

bear9@bear9-pc:~/Рабочий стол/диск$ cd front_sfc/
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc$ mkdir build
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc$ cd build/
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ cmake ..
-- The C compiler identification is GNU 11.4.0
-- The CXX compiler identification is GNU 11.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Boost: /usr/lib/x86_64-linux-gnu/cmake/Boost-1.74.0/BoostConfig.cmake (
found version "1.74.0") found components: program options
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.2")
-- Checking for module 'jsoncpp'
-- Found jsoncpp, version 1.9.5
-- Configuring done
-- Generating done
-- Build files have been written to: /home/bear9/Рабочий стол/диск/front_sfc/bui
ld
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ make
[ 20%] Building CXX object CMakeFiles/front_sfc.dir/main.cpp.o
[ 40%] Building CXX object CMakeFiles/front_sfc.dir/menu.cpp.o
[ 60%] Building CXX object CMakeFiles/front_sfc.dir/config.cpp.o
[ 80%] Building CXX object CMakeFiles/front_sfc.dir/menu_func.cpp.o
[100%] Linking CXX executable front_sfc
[100%] Built target front_sfc

```

Рисунок 3.38 – Инсталляция front_sfc

```

bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ cd ../../
bear9@bear9-pc:~/Рабочий стол/диск$ cd back_sfc/
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc$ mkdir build
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc$ cd build/
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc/build$ cmake ..
-- The C compiler identification is GNU 11.4.0
-- The CXX compiler identification is GNU 11.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Boost: /usr/lib/x86_64-linux-gnu/cmake/Boost-1.74.0/BoostConfig.cmake (found vers
ion "1.74.0") found components: log_setup log filesystem regex thread date_time system chr
ono atomic filesystem
-- Found OpenSSL: /usr/lib/x86_64-linux-gnu/libcrypto.so (found version "3.0.2")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/bear9/Рабочий стол/диск/back_sfc/build
bear9@bear9-pc:~/Рабочий стол/диск/back_sfc/build$ make
[ 9%] Building CXX object CMakeFiles/back_sfc.dir/src/main.cpp.o
[ 18%] Building CXX object CMakeFiles/back_sfc.dir/src/configuration/config.cpp.o
[ 27%] Building CXX object CMakeFiles/back_sfc.dir/src/configuration/chek_conf_file.cpp.o
[ 36%] Building CXX object CMakeFiles/back_sfc.dir/src/daemon/daemon_works.cpp.o
[ 45%] Building CXX object CMakeFiles/back_sfc.dir/src/other/logger.cpp.o
[ 54%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/main_proc.cpp.o
[ 63%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/backup.cpp.o
[ 72%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/hash.cpp.o
[ 81%] Building CXX object CMakeFiles/back_sfc.dir/src/other/notification.cpp.o
[ 90%] Building CXX object CMakeFiles/back_sfc.dir/src/proc/encrypted.cpp.o
[100%] Linking CXX executable back_sfc
[100%] Built target back_sfc

```

Рисунок 3.38 – Инсталляция back_sfc

Производим запуск программы back_sfc, рисунок 5.39.

```

bear9@bear9-pc:~/Рабочий стол/диск/back_sfc/build$ ./back_sfc

```

Рисунок 5.39 – Запуск back_sfc

Возвращаемся в директорию с программой front_sfc, и запускаем ее, рисунок 5.40. Включаем защиту – рисунок 5.41, переходим к настройке.

```

bear9@bear9-pc:~/Рабочий стол/диск/back_sfc/build$ cd ../../
bear9@bear9-pc:~/Рабочий стол/диск$ cd front_sfc/build/
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc

```

Рисунок 5.40 – Запуск front_sfc

```

=====
|| 1. Enable protection      ||
|| 2. Disable protection    ||
|| 3. Get information about ||
|| 4. Force file restoration||
|| 5. Configure protection  ||
|| 6. Show help             ||
|| 7. Exit the program      ||
=====
Enter your choice: 1
Enable protection

```

Рисунок 5.41 – Включение защиты

Для определение пути к необходимой директории, воспользуемся следующей командой:

pwd

Скопируем ответ, рисунок 5.42, и внесем его в соответствующий пункт утилиты, рисунок 5.43. Настроим интервал проверки файлов, рисунок 5.44.

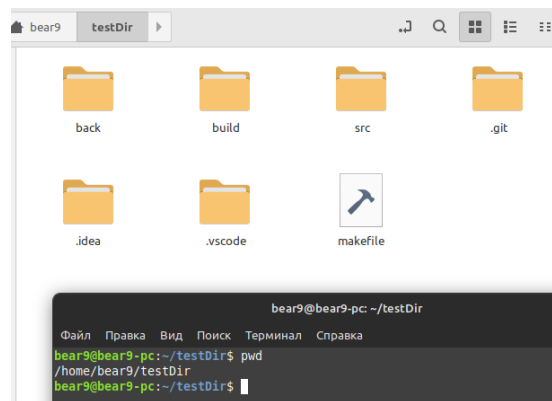


Рисунок 5.42 – Определение пути к директории

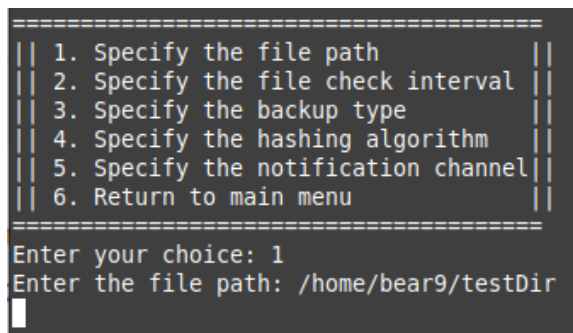


Рисунок 5.43 – Установка пути

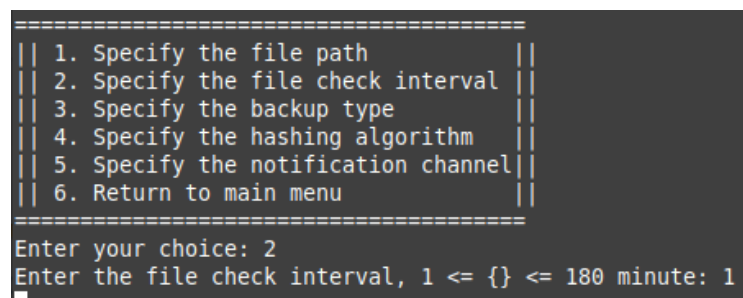


Рисунок 5.44 – Установка временного интервала

Выйдя из утилиты будет получено новое уведомление – рисунок 5.45. Проверим настройки мониторинга, рисунок 5.46.

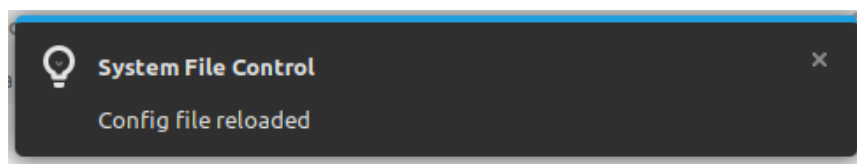


Рисунок 5.45 – Уведомление об обновлении настроек

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --file
Get information about protected files

*****
* File information:                               *
* Path: /home/bear9/testDir                       *
* Protection: Enabled                             *
* Interval: 1                                     *
* Backup Type: full                               *
* Hash Algorithm: MD5                             *
* Notification Channel: system                    *
*****
```

Рисунок 5.46 – Проверка настроек

После этих действий, в домашнем каталоге будет создана директория «control», содержащая в себе конфигурационный файл и сжатую, зашифрованную, резервную копию директории, рисунок 5.47.

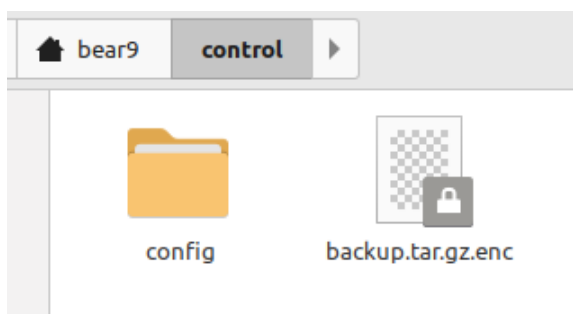


Рисунок 5.47 – Директория необходимая для работы программы

После этих действий, утилита работает в определенном для нее режиме. При удалении файла из охраняемой директории, рисунок 5.48, во время ближайшей проверки, интервал которой был выставлен ранее, будет выведено критическое системное уведомление – рисунок 5.49.

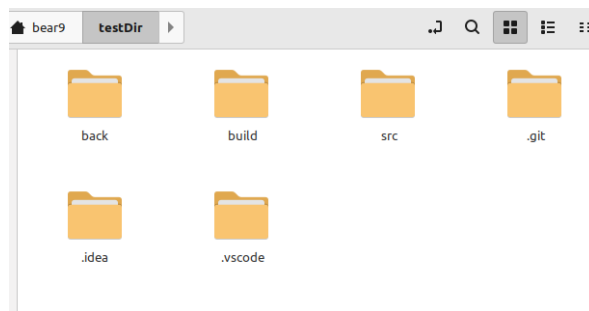


Рисунок 5.48 – Директория после удаления файла makefile

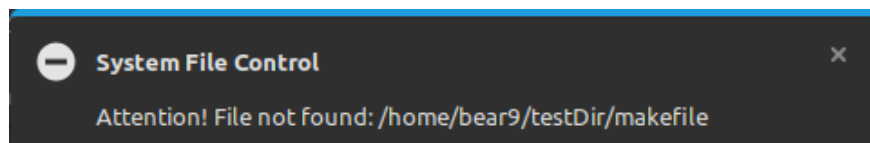


Рисунок 5.49 – Критическое системное уведомление

Вернем директорию в первоначально состояние, используя утилиту с соответствующим ключом, для принудительного восстановления, рисунок 5.50.

```
bear9@bear9-pc:~/Рабочий стол/диск/front_sfc/build$ ./front_sfc --force  
Force file restoration.
```

Рисунок 5.50 – Выполнение принудительного восстановления файлов

Просмотрим директорию, после выполнения восстановления, рисунок 5.51.

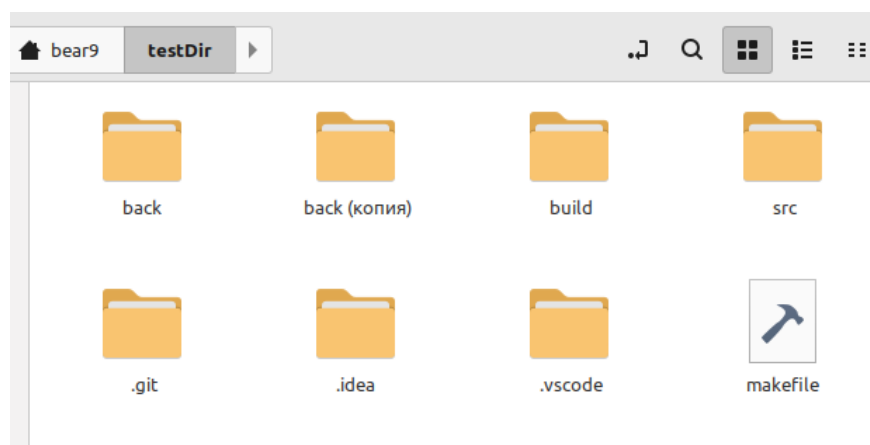


Рисунок 5.51 – Восстановленная директория

Изменив содержимое файла `makefile`, при проверке будет выведено уведомление о несоответствии хеш-суммы файла, изображенное на рисунке 5.52. Добавив новые файлы, для каждого нового, будет выведено уведомление, снимок экрана которого приведен на рисунке 5.53.

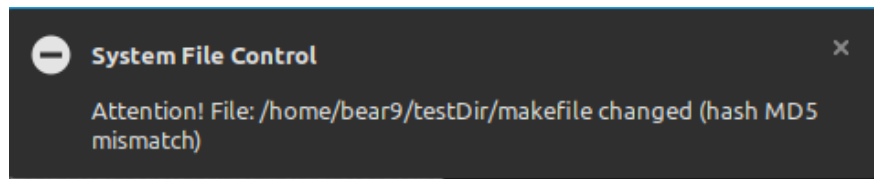


Рисунок 5.52 – Уведомление о несоответствии хеш-суммы

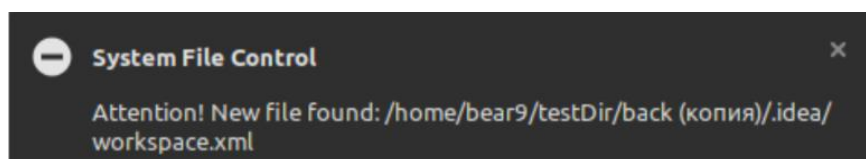


Рисунок 5.53 – Уведомление о новом файле

ЗАКЛЮЧЕНИЕ

Разработанная в ходе данного курсового проекта программа для контроля изменений и целостности группы файлов представляет собой ценный инструмент в современном информационном обществе. Кроме того, разработанная программа имеет потенциал для широкого применения в различных сферах и отраслях, например:

- в корпоративной безопасности;
- в научных исследованиях;
- в сфере финансовых услуг;
- в здравоохранении;
- в юридической сфере.

Курсовой проект был разработан в соответствии с поставленными задачами, весь функционал был реализован в полном объеме.

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 ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Керриск, М. Linux API. Исчерпывающее руководство / М. Керриск. – СПб : Питер, 2019. – 1248 с.
- 2 Никлаус, В. Алгоритмы и структуры данных. / Пер. с англ. Ткачев Ф. В. – М.: ДМК Пресс, 2010. – 272 с.: ил.
- 3 Прохорова, О. Информационная безопасность и защита информации. /СПб.: Питер, 2015. — 280 с.: ил.
- 4 Рожнова, Н. Г. Вычислительные машины, системы и сети. Дипломное проектирование : учебно–метод.пособие / Н. Г. Рожнова, Н. А. Искра, И. И. Глецевич. – Минск : БГУИР, 2014. – 96 с. : ил.
- 5 Дейтел, Х. Как программировать на C++ / Х. Дейтел, П. Дейтел. – М. : БИНОМ, 2001. – 1152 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Схема работы системы

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема работы подпрограммы `update_config_file`

ПРИЛОЖЕНИЕ В

(обязательное)

Схема работы подпрограммы full_backup

ПРИЛОЖЕНИЕ Г
(обязательное)
Текст программы

ПРИЛОЖЕНИЕ Д
(обязательное)
Ведомость документов