

Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

УТИЛИТА ПОИСКОВ ОДИНАКОВЫХ ФАЙЛОВ
(аналог fdupes, но с фильтрацией по именам и типам)
БГУИР КР 1-40 02 01 306 ПЗ

Студент:

Григорик И. А.

Руководитель:

Глоба А. А.

Минск 2022

Оглавление

ВВЕДЕНИЕ.....	3
1 ОБЗОР ЛИТЕРАТУРЫ	5
2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ.....	7
2.1 Постановка задачи	7
2.2 Модуль примера	7
2.3 Модуль хранения данных о файлах	8
2.4 Модуль сбора информации о файлах.....	8
2.5 Модуль чтения информации о файле.....	8
2.6 Блок хеширования данных файла	8
2.7 Модуль преобразования хеша	8
2.8 Модуль обработки флагов.....	9
2.9 Блок проверки данных о файле.	9
2.10 Модуль удаления файлов	9
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ	10
3.1 Описание структур утилиты	10
ЛИТЕРАТУРА.....	11

ВВЕДЕНИЕ

В текущее время у каждого пользователя компьютера, вне зависимости от его опыта работы с компьютером, рано или поздно появляются одинаковые файлы. По различным причинам пользователь может не замечать их, или просто игнорировать, т.к. искать вручную все дублирующиеся файлы не так уж и просто. Именно поэтому используются специальные утилиты для автоматического поиска одинаковых файлов.

К примеру, `fdupes` – утилита, написанная Андрианом Лопесом, может искать одинаковые файлы из любого каталога в системе. Для этого используется получение хеша файла из его дескриптора. Следует пояснить, что такое хеш файла и файловый дескриптор:

- Хеш файла – это уникальный идентификатор файла, который высчитывается системой посредством определённых преобразований хранящейся в файле информации.
- Дескриптор файла – это целое неотрицательное число, с помощью которого процесс может обращаться к потоку ввода-вывода. Дескриптор может быть связан с файлом, сокетом или каталогом.

С помощью данных средств можно сверять файлы по содержанию, что и является целью поиска одинаковых файлов.

Данный курсовой проект представляет собой такую утилиту командной строки, подобную уже известной утилите `fdupes`, которая будет искать файлы с одинаковым содержимым, основываясь на хеше файла. Так же данная утилита будет сравнивать расширения файлов и их имена, помимо содержимого. Так же, с помощью некоторых управляющих флагов, утилита сможет производить действия над данными файлами, помимо простого поиска.

В рамках данной курсовой работы необходимо ознакомиться с библиотекой `openssl/md5.h`, которая используется для получения хеша файлов, или же с командой `MD5`, которые используются для получения MD5 хеша, с помощью которого файлы и будут проверяться на идентичность. В процессе разработки следует углубить знания по языку C / C++, а также осуществить взаимодействие пользовательского приложения с системой. В конце следует протестировать приложение и провести эксперименты на нескольких устройствах.

`OpenSSL` – полноценная криптографическая библиотека, с открытым исходным кодом, которая используется во многих проектах для хеширования MD5, MD2, SHA. Библиотека написана Эриком Янгом и Тимом Хадсоном, и получила популярность благодаря расширениям SSL/TLS, которые используются в веб-протоколе HTTPS.

Для качественного выполнения курсового проекта следует рассмотреть аналоги данной утилиты. Ключевой аналог – так же консольная утилита `fdupes`. Данная утилита использует хеш файлов, для их сравнения. Утилита

написана полностью на языке программирования C, и находится под лицензией MIT. Она же является примером подражания моего курсового проекта.

Следующая утилита – CloneSpy. Это так же бесплатный инструмент для очистки дискового пространства от дублирующихся файлов. Так же может находить файлы нулевой длины, у которых нету содержимого. Данный аналог представляет собой приложение с GUI, что не является целью проекта. Так же приложение разрабатывалось под операционную систему Windows, что так же не предпочтительно.

FSlint – так же утилита с графическим интерфейсом, но для Linux. Данная утилита представляет собой поиск одинаковых файлов, но с расширенным функционалом. Так же тут присутствуют флаги поиска для одинаковых имён, архивов и пустых директорий.

1 ОБЗОР ЛИТЕРАТУРЫ

Следует начать с определения утилиты.

Утилита – это некая вспомогательная компьютерная программа в составе программного обеспечения, которая используется для выполнения типовых задач, связанной с работой оборудования или ОС. В данном случае утилита используется для облегчения пользования компьютером.

Данная утилита основывается на получении хеша файла из его дескриптора, определения которых давались выше. Процедура получения хеша называется **хешированием**.

Хеширование может проводиться по разным алгоритмам. Основной смысл хеширования заключается в безопасности и надёжности, возможности сжимать любые куски информации в короткий стандарт сообщений, являющихся уникальными для каждой доли информации. Даже если один байт информации будет изменён – хеш так же поменяется.

Таким образом не надо хранить всё содержимое файла или каждый раз его открывать, чтобы сверить его информацию с другим файлом. Достаточно будет просто сгенерировать его хеш и запомнить его.

Недостатком хеширования является неизбежность коллизии.

Коллизия – это равенство значений хеш-функций на двух различных кусках информации. В данном случае это означает, что если функция сгенерирует одинаковый хеш для двух разных файлов, то в системе они будут считаться за одинаковые. Для решения вопроса коллизий создаются современные хеш-функции, в которых шанс появления коллизий стремится к минимуму.

Так же стоит пояснить, что длина строки зависит от конкретной хеш-функции, но одна функция не может сгенерировать две строки разной длины.

На данный момент популярны следующие хеш-функции:

- **SHA256** – одна из наиболее устойчивых к коллизиям функция. Недостаток: по сравнению с другими, имеет довольно большое время выполнения и большая длина хеш-слова (256 байт).
- **RIPEMD160** – так же устойчивая к коллизиям функция, которая, к тому же, имеет длину хеш-слова почти в два раза меньше (160 байт), чем SHA256. Время выполнения примерно такое же, как у SHA256.
- **MD5** – самая быстрая криптографическая хеш-функция из широко используемых, к тому же имеет наименьший размер хеша (128 байт). Недостаток: небезопасна. Легко подвергается коллизиям, поэтому не стала использоваться в проектах, по типу криптовалютных кошельков.

Так же хеширование используется для сокрытия данных, так как хеширование одностороннее (т.е. нельзя преобразовать хеш в первоначальные данные). Получить первоначальные данные можно только сгенерировав такую же строку, или создать коллизию, которая приведёт к конвертации другой строки

В данном случае сокрытие не требуется, и могут допускаться коллизии, поэтому может использоваться функция хеширования MD5.

Теперь же о файловых дескрипторах.

Файловый дескриптор в данном случае используется для отображения файла на память. Данный метод является эффективным, ибо помогает разгрузить систему, и вообще не использовать физическую память, чем помогает снизить нагрузку на диск для нескольких программ, обращающихся к одному и тому же файлу.

Так же следует заметить, что ОС Linux придерживается правила «всё есть файл», поэтому тип файла тут – понятие, которое отличается и часто путается с расширением файла в Windows. В Linux существуют всего три типа файлов: обыкновенные, специальные и директории. Следовательно, можно сделать вывод, что условие реализации фильтрации поиска по типам файла будет реализовываться относительно их расширения, что является подтипом обыкновенных файлов.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Сначала следует поставить конкретные функциональные требования разрабатываемой программе, разбить утилиту на модули и функциональные блоки. Данный подход в большинстве упростит понимание проектирования, сможет помочь устранить проблемы в архитектуре и обеспечить гибкость каждого из модулей. После того, как аналоги рассмотрены и проведен краткий экскурс в основные понятия, можно поставить конкретные цели разрабатываемому программному обеспечению.

2.1 Постановка задачи

Минимальное требование – просто поиск одинаковых файлов с определённой директории, с их выводом на экран. Так как в условии курсового проекта сказано, что должна быть фильтрация по типам, то это так же будет являться обязательным условием выполнения. В качестве поиска будет использоваться алгоритм хеширования MD5 из-за своей скорости и маленькой длине хешируемого слова, что является условием быстрого выполнения утилиты. Так же для оптимизации будут использоваться отображения в память, для упрощённого, для системы, получения хеша файла. Данный подход поможет не затрачивать лишнее дисковое пространство и в несколько раз ускорить работу утилиты. Так как присутствует условие фильтрации по типам, то утилита должна подстраиваться под передаваемые ей пользовательские флаги, следовательно, должен быть установлен обработчик входных флаговых значений.

Из вышеперечисленных методов можно составить определённые модули программы, которые будут обеспечивать полную функциональность выполнения. Так же следует выделить, что в данном случае расширение файла, и его имя является одним и тем же, следовательно, для этого не имеет смысла делать двух разных флагов. В качестве дополнительного задания решено находить файлы с различным содержанием, но одинаковым именем. Для этого следует применять флаг `-r` (reverse). Так же к флагам добавляется флаг статистики (`-s`), и флаг удаления файлов (`-d`). Первый будет использоваться для показа текущей статистики сбора файлов, второй для полного удаления дубликатов файлов после их поиска. Последний флаг – флаг примера (`-e`), который будет отображать корректное поведение программы. С него и стоит начать описание модулей программы.

2.2 Модуль примера

Данный модуль будет полностью симулировать применение программы. Он будет представлять собой открытие какого-то каталога, создания там нескольких файлов с одинаковым содержимым, выводом этого содержимого на экран, задержкой для проверки содержимого и удалением или поиском

одинаковых файлов. Модуль необходим для отображения корректного поведения программы, и будет использовать максимальное количество модулей, описанных ниже.

2.3 Модуль хранения данных о файлах

Данный модуль будет представлять собой некий массив самостоятельно написанной структуры, которая будет использоваться для хранения всех метаданных о каждом проверяемом файле. В данном случае метаданными будут служить хеш файла и его имя, для уникальных файлов, и имя файла и путь до него, для дублирующихся файлов. Данный модуль будет представлять из себя два вектора данных структур.

2.4 Модуль сбора информации о файлах

Данный модуль будет представлять собой рекурсивную функцию, используемую для прохождения по всем файлам с задающей директории. Модуль необходим для обновления информации модуля хранения данных о файлах. В нём же можно выделить ещё несколько блоков и модулей, которые будут описаны в заголовках **2.5**, **2.6** и **2.7**.

2.5 Модуль чтения информации о файле

Модуль будет реализовывать собой отображение данных файла в виртуальное адресное пространство, после чего будут задействован блок получения хеша файла и модуль преобразования хеша в шестнадцатеричную систему счисления.

2.6 Блок хеширования данных файла

Данный блок кода просто будет представлять собой получения хеша определённого файла основываясь не его отображении в виртуальном адресном пространстве. После выполнения этого блока необходимо освободить адресное пространство.

2.7 Модуль преобразования хеша

Данный модуль преобразовывает полученный хеш, который может состоять из нечитаемых символов или же из символов, неудобных для работы, в строку, определённой длины, состоящую из шестнадцатеричных чисел. Преобразование необходимо для корректной работы, ибо данные в неудобных форматах для чтения могут неправильно сравниваться.

2.8 Модуль обработки флагов.

Модуль представляет собой преобразование входных данных в определённые флаги, которые в дальнейшем будут использоваться в блоке проверки данных и выводе.

2.9 Блок проверки данных о файле.

Блок является проверкой данных конкретного файла со всеми файлами, которые хранятся в модуле хранения данных. Модуль может изменяться, в связи с изменением некоторых флагов. В случае выполнения определённых условий, данные файла будут заноситься в вектор дублирующихся файлов.

2.10 Модуль удаления файлов

Модуль является простым считыванием вектора дублирующихся файлов и их последующего удаления. Так же возможно не полное, а выборочное удаление файлов.

Все вышеперечисленные модули позволяют обеспечить полноту выполняемых действий, соответственно необходимы для выполнения курсовой работы.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Данная глава будет представлять собой ключевой раздел, дающий понимание работы моей утилиты, её структуру, с точки зрения описания отдельных функций и модулей, обработки данных, с приведением листинга и структурной диаграммой, вынесенной в приложение «А».

3.1 Описание структур утилиты

ЛИТЕРАТУРА

1. Лафоре, Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. – СПб. : Питер, 2004.
2. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон; пер. с англ. – СПб. : ДМК, 2004.
3. Лав Р. Системное программирование на Linux/ 2-е издание 2014.
4. Керниган Б. Язык программирования C/ 4-е издание М.:Питер, 2004. – 923 с.
5. Рочкинд М. Программирование для UNIX, 2-е изд. СПб, БХВ-Петербург, 2005.
6. Калле Р. Грокаем технологию биткойн / Р. Калле – СПб. : Питер, 2020.

[illegible]

[illegible]