НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___”__червня___2017 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050102 «Комп'ютерна інженерія»

на тему: “Тег-орієнтоване файлове сховище”

Виконав: студент IV курсу, групи КВ-32
(шифр групи)

Мельник Олександр Олегович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н., доцент Романкевич В.О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___»____червня____2017 р.

ЗАВДАННЯ
на дипломний проект студента
Мельника Олександра Олеговича
(прізвище, ім’я, по батькові)

1. Тема проекту “Тег-орієнтоване файлове сховище”
_____
_____,
керівник проекту к.т.н., доцент Романкевич В.О.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___» травня 2017 р. №_____

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту див. ТЗ
_____
_____
_____

4. Зміст пояснювальної записки Аналіз існуючих рішень та обґрунтування теми дипломного проекту
Методи рішення та їх обґрунтування
Опис алгоритмів розробленого програмного забезпечення
Опис розробленого інтерфейсу

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслень, плакатів, презентацій тощо)

Тег-орієнтоване файлове сховище. Схема структурна

Видалення файлу зі сховища. Схема алгоритму

Виділення значення хеш-функції за частковим рядковим записом. Схема алгоритму

Вибір файлів за простим запитом. Схема алгоритму

6. Консультанти розділів проекту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення літератури за тематикою проекту	15.11.2016	
2	Розроблення та узгодження технічного завдання	30.11.2016	
3	Аналіз існуючих рішень	15.12.2016	
4	Підготовка матеріалів першого розділу дипломного проекту	30.12.2016	
5	Розроблення програмного забезпечення	01.03.2017	
6	Відлагодження програмного продукту	15.03.2017	
7	Підготовка матеріалів другого розділу дипломного проекту	01.04.2017	
8	Підготовка матеріалів третього розділу дипломного проекту	15.04.2017	
9	Підготовка матеріалів четвертого розділу дипломного проекту	01.05.2017	
10	Підготовка графічної частини дипломного проекту	15.05.2017	
11	Оформлення документації дипломного проекту	25.05.2017	

Студент _____ Мельник О.О.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Романкевич В.О.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

2 МЕТОДИ РІШЕННЯ ТА ЇХ ОБҐРУНТУВАННЯ

2.1 Операційна система, використана для розробки

Для вирішення завдання використано операційну систему Arch Linux (на базі GNU/Linux). GNU/Linux є оптимальним вибором, адже надає відмінну підтримку Python 3, є прогресивним сімейством ОС (Arch Linux — одна з найпрогресивніших, найсучасніших варіацій) та належить до вільного програмного забезпечення. Таким чином, вибираючи GNU/Linux, я роблю свій вклад до популяризації та розповсюдження вільного ПЗ.

Також найчастіше саме ОС на базі GNU/Linux використовуються ентузіастами файлових систем, зацікавленими в використанні нових ФС та файлових сховищ.

Дуже важливо, що GNU/Linux підтримує стандарт POSIX (хоч офіційної сертифікації не проведено, фактично підтримка наявна). Таким чином, вибираючи GNU/Linux, я роблю свій вклад до популяризації та розповсюдження POSIX.

Проте, загалом, ОС, використана для розробки, не має великого значення, адже тег-орієнтоване файлове сховище спроектоване як переносиме ПЗ, що не залежить від ОС і підтримує будь-яку ОС, що підтримує Python 3.

2.2 Мова програмування

Для вирішення завдання використано інтерпретовану мову програмування Python 3. Python є широко розповсюдженим та потужним інструментом розробки, дозволяє інтеграцію з об’єктними модулями для підвищення продуктивності та надає зручний доступ до багатьох ресурсів операційної системи та обчислювальної платформи, для якої створюється програма.

Також пакети Python 3 включено до стандартного набору більшості дистрибутивів GNU/Linux, OS X та інших операційних систем, подібних до Unix.

Python — інтерпретована мова програмування, що не потребує компіляції. Це дозволяє значне пришвидшення процесу розробки, спрощує тестування окремих підпрограм та модулів.

Значна перевага Python 3 — переносимість. Програми, написані на чистій мові Python без використання компільованих об’єктних модулів можуть бути виконані без модифікацій на будь-якій архітектурі та ОС, що підтримує Python.

2.3 Архітектура тег-орієнтованого файлового сховища

Тег-орієнтоване файлове сховище вирішено розробити як набір пов’язаних між собою модулів, що можуть бути вільно замінені для оптимального використання доступних ресурсів в конкретному випадку та/або для введення не передбачених на етапі планування функцій.

Наприклад, текстовий інтерфейс користувача зі стандартного пакету можна замінити іншим текстовим, або навіть графічним чи мережевим інтерфейсом, в тому числі FUSE-інтерфейсом для використання в якості користувацької файлової системи.

2.4 Ідентифікація збережених файлів в рамках материнської файлової системи

Для ідентифікації файлів у сховищі вирішено використовувати хеш-функцію SHA3 (алгоритм Keccak). Keccak є найсучаснішою стандартизованою розробкою в галузі хеш-функцій і обрана через це, адже кожен проект повинен, за можливості, використовувати найсучасніші технології, щоб спонукати їх розвиток в подальшому.

Для забезпечення якнайменшої ймовірності співпадіння значень хеш-функцій для файлів з різним вмістом, використовується SHA3-функція найбільшої розрядності: 512 біт. Хоч такі великорозрядні значення потребують більше дискової та оперативної пам’яті для роботи з ними, вони забезпечують найбільшу кількість можливих значень без співпадінь.

Відсутність співпадінь в основному форматі ідентифікатора, що використовується в програмі для визначення конкретного файлу, є дуже важливою, адже різні файли з однаковими ідентифікаторами будуть сприйматися як один і вміст одного з них буде втрачено. На жаль, боротися з цим ефектом дуже

складно, а взагалі позбутися — практично неможливо (побайтове порівняння файлів для визначення їх ідентичності кожного разу потребує забагато ресурсів, а використання кількох хеш-функцій лише знижує — хоч і дуже значно — ймовірність колізії).

512-бітні значення, проте, є дуже незручними у використанні кінцевим користувачем. Для більшої компактності значення кодуються у рядкове представлення відповідно до кодування base64. base64 — широко розповсюджене та відоме кодування, що дозволяє досить значне стиснення інформації (чотири символи відповідають трьом байтам, тоді як за шістнадцяткового представлення два символи відповідають одному байту — скорочення запису в півтора рази, або ж на третину) та використовує лише 64 символи ASCII, що можуть бути надруковані. Наприклад, важливі ідентифікатори в base64 користувач може надрукувати та зберігати, щоб безпомилково використати пізніше, коли буде потрібно.

Будь-яка програма-месенджер та поштовий клієнт/протокол дозволяють використання цього набору символів, що спрощує обмін інформацією. Додатковою перевагою base64 є також наявність реалізації кодування в стандартній бібліотеці Python 3.

Проте навіть закодовані в base64 ідентифікатори мають значну, незручну довжину (86 символів). Тому вирішено надати користувачу можливість використовувати неповні, часткові ідентифікатори у вигляді найменшого префіксу — таку кількість N перших символів ідентифікатора, яка на даний момент дозволяє однозначно знайти повний в таблиці наявних.

Для пошуку повного ідентифікатора за скороченим використано алгоритм на основі бітових операцій. Хоч такий підхід не очевидний для початківців та не-фахівців, він є більш ефективним (побітова операція виконується в одну машинну інструкцію).

2.5 Текстовий інтерфейс

Текстовий інтерфейс обрано за стандартний через величезну

розповсюдженість текстового інтерфейсу ОС та майже повну відсутність обчислювальних систем, що не дозволяють використання текстового інтерфейсу в тому чи іншому вигляді.

Також текстовий інтерфейс користувача дозволяє просте і зручне використання файлового сховища в скриптах та інших програмах. З графічним інтерфейсом це взагалі неможливо, тоді як програмний (API) чи мережевий інтерфейс незручні для користувача. Текстовий інтерфейс — оптимальний компроміс в даному випадку.

Формат інтерфейсу обрано подібним до того, що використовується утилітою git, адже git широко розповсюджена та популярна серед розробників, адміністраторів та інших фахівців, що найчастіше працюють з великими обсягами файлів. Отже, git-подібний інтерфейс буде інтуїтивно зрозумілим для цих користувачів.

2.6 Складні запити

Тег-орієнтоване файлове сховище дозволяє використання складних запитів з кількома (кількість необмежена) тегами, поєднаними логічними операціями ТА, АБО, НЕ.

Складні запити потребують окремої обробки (використовується алгоритм сортувальної станції для перетворення інфіксного (звичайного) запису виразів до постфіксного (оберненого польського), та стековий алгоритм для подальшої обробки отриманого виразу), проте є дуже корисними і зручними для користувача, позбавляючи необхідності додавати “складені” теги.

2.7 Конфігураційні файли та файли керуючих таблиць

Конфігураційні файли та файли керуючих таблиць, що визначають відповідності збережених файлів до тегів та значень хеш-функції зберігається в форматі Pickle, що надається стандартною бібліотекою функцій Python. Формат Pickle є оптимальним стосовно відношення затраченого дискового простору до часу, необхідного для приведення збережених даних до виду, придатного до

використання.

Збереження конфігураційних файлів та керуючих таблиць у файлах на диску дозволяє використовувати сховище в рамках декількох секцій, не втрачаючи дані, а тому є дуже важливою функцією.

2.8 Збереження імені та розширення файлу в якості тегів

Використання значення хеш-функції від вмісту в якості ідентифікатора файлу має багато переваг, проте один суттєвий недолік: для користувача значення хеш-функції не має ніякого сенсу, і зорієнтуватися, який з файлів отримав який ідентифікатор, завантаживши одночасно декілька файлів, буде досить складно. Крім того, файли з однаковими наборами тегів так теж взагалі неможливо розрізнити, не вивантажуючи.

Крім того, для файлу в абсолютній більшості сучасних файлових та операційних систем є дуже важливим розширення — частина імені після крапки. За розширенням файлу в більшості випадків визначається його тип та за допомогою якої прикладної програми можна працювати з даним файлом.

Тому розроблене тег-орієнтоване файлове сховище за замовчуванням при завантаженні до сховища файлу надає йому два особливих теги: “name/<ім’я файлу>” та “ext/<розширення файлу>”.

Тег “name/” ніяк не використовується сховищем і надається файлу лише заради можливості розрізняти файли відразу після завантаження.

Розширення з (найостаннішого в списку тегів) тегу “ext/” виводиться стандартним інтерфейсом при отриманні списку файлів, а також надається файлу при вивантаженні, щоб з ним можна було відразу працювати.

Якщо розширення було змінене при оновленні вмісту файлу, за замовчуванням тег старого розширення не знімається, а новий тег просто додається до кінця списку, так що виводитись буде саме нове розширення (за принципом “найостанніший”).

Також розширення можна змінити простою операцією вилучення/додавання відповідного тегу, а ім’я файлу, загалом, змінювати немає сенсу, адже воно ніяк не

використовується системою тег-орієнтованого файлового сховища.

Користувачу дозволено і рекомендується створювати власні “категорії” тегів шляхом додавання префіксу через косу риску або будь-яким іншим способом. При подальшому розвитку тег-орієнтованого файлового сховища планується можливість використання у виразах цілих категорій, а не окремих тегів.