Лабораторна робота №8

Тема. Управління файловою системою.

Мета. Ознайомитися з файловими системами операційних систем Windows та LINUX.

Теоретичні відомості

Файлова система – це система правил, згідно яких операційна система забезпечує зберігання даних на диску. Ці правила визначають формат зберігання даних, їх розміщення на диску, адресацію і ідентифікацію.

Кожен комп'ютерний файл зберігається на цифровому носії із певною місткістю. Накопичувач можна уявити як лінійний простір для читання і запису цифрової інформації. Кожен байт даних на ньому має своє зміщення від початку сховища, відомий як адреса, яка використовується для посилання на цей файл. Накопичувачі для посилання на байт інформації можуть використовувати адресацію у вигляді пари - сектор і зміщення.

Загалом, файлова система це структуроване представлення даних та набір метаданих, що описують ці дані. Вона записується до сховища під час операції форматування. Зазвичай ця структура оперує блоками, а не секторами. Блоки ФС – це групи секторів, за допомогою яких оптимізується адресація сховища. Файли, як правило, зберігаються на початку блоку та займають цілі блоки.

Крім даних користувача, файлова система також містить власні параметри (наприклад, розмір блоку), файлові дескриптори (такі як його розмір, розташування, фрагменти і т.д.), імена та ієрархію каталогів. Вона також може зберігати інформацію про безпеку, розширені атрибути та інші параметри.

Щоб відповідати різноманітним вимогам користувачів, таким як продуктивність сховища, стабільність і надійність, розробляється безліч типів (або форматів) файлових систем, здатних більш ефективно виконувати певні функції.

Файлові системи Windows

Microsoft Windows в основному використовує дві файлові системи: NTFS – основний формат за замовчуванням більшості сучасних версій цієї ОС, а також FAT – файлова система, успадкована від старої DOS, і exFAT, її подальше розширення. ReFS була також представлена Microsoft як файлова система нового покоління для серверних комп'ютерів, починаючи з Windows Server 2012. Файлову систему HPFS, розроблену Microsoft спільно з IBM, можна знайти тільки на дуже старих пристроях під управлінням Windows NT до версії 3.5.

FAT

FAT (File Allocation Table) – один із найпростіших типів ФС, який існує з 1980-х років. Вона складається з сектора дескрипторів ФС (завантажувального сектора або суперблоку), таблиці розподілу блоків (File Allocation Table) та простору для зберігання даних. Файли в FAT зберігаються у каталогах. Кожен каталог являє собою масив 32-байтових записів, кожен з яких визначає файл або його

розширені атрибути (наприклад, довге ім'я). Запис файлу описує його перший блок. Будь-який наступний блок можна знайти в таблиці розподілу блоків, використовуючи її як список із посиланнями. Таблиця розподілу блоків містить масив дескрипторів блоків. Нульове значення вказує на те, що блок не використовується, а ненульове відноситься до наступного блоку файлу або спеціального значення для кінця файлу.

NTFS

NTFS (New Technology File System) була представлена в 1993 році разом із Windows NT та нині є найбільш поширеною файловою системою для комп'ютерів на базі Windows. Більшість операційних систем лінійки Windows Server також використовують цей формат.

Цей тип ФС є досить надійним завдяки журналюванню та підтримує безліч функцій, включаючи контроль доступу, шифрування і т. д. Кожен файл в NTFS зберігається як файловий дескриптор у Головній файловій таблиці (Master File Table) та як вміст із даними. Головна файлова таблиця містить записи з усією інформацією про них: їх розміри, розподіл, імена тощо. Перші 16 записів таблиці резервуються для бітової карти (BitMap), яка веде облік всіх вільних та використаних кластерів, Журналу (Log), що застосовується реєстрації записів, і файлу BadClus, що містить інформацію про пошкоджені кластери. Перший та останній сектори файлової системи містять налаштування файлової системи (завантажувальний запис або суперблок). Цей формат використовує 48-бітові та 64-бітові значення для посилання на файли, що дозволяє їй підтримувати сховища з надзвичайно високою місткістю.

ReFS

ReFS (Resilient File System) – це остання розробка Microsoft, представлена разом із Windows 8, а тепер доступна і для Windows 10. Її архітектура абсолютно відрізняється від інших форматів Windows та в основному організована у вигляді B+-дерева. ReFS має високу стійкість до збоїв завдяки новим функціям. Найбільш визначною серед них є Копіювання при записі (Copy-on-Write): жодні метадані не змінюються без їх попереднього копіювання; дані не записуються поверх існуючих даних – вони поміщаються до іншої області на диску. При будь-яких змінах нова копія метаданих зберігається до вільної області у сховищі, а потім система створює посилання зі старих метаданих на нову копію. Таким чином, значна кількість старих резервних копій зберігається у різних місцях, забезпечуючи легкість відновлення даних, за умови що цю область не буде перезаписано.

Файлові системи macOS

Apple macOS застосовує два типи ФС: HFS+, розширення застарілої файлової системи HFS та APFS, формат, який використовується сучасними комп'ютерами Mac під управлінням macOS 10.14 і пізніших версій. Просунуті серверні продукти також використовують файлову систему Apple Xsan, кластерну файлову систему від StorNext і CentraVision.

HFS+

HFS+ використовує B-дерева для розміщення і пошуку файлів. Томи поділяються на сектори, зазвичай розміром 512 байтів, які потім групуються в блоки розподілу, кількість яких залежить від розміру всього тому. Інформація про вільні та використані блоки розподілу зберігається у Файлі розподілу (Allocation File). Всі блоки розподілу, присвоєні кожному файлу як екстенти зберігаються у файлі Extends Overflow. І, нарешті, всі атрибути файлу вказані у Файлі атрибутів. Надійність сховища підвищується внаслідок ведення журналу, який дозволяє відстежувати всі зміни в системі та швидко повертати її до робочого стану в разі непередбачених подій. Серед інших підтримуваних функцій – жорсткі посилання на каталоги, шифрування логічних томів, контроль доступу, компресія даних тощо.

APFS

Файлова система APFS має на меті розв'язання основних проблем своєї попередниці та розроблена для ефективної роботи з сучасними флеш- та твердотільними накопичувачами. Цей 64-розрядний формат використовує метод Копіювання при записі (Copy-on-Write) для підвищення продуктивності, що дозволяє копіювати кожен блок, перш ніж будуть застосовані зміни, і пропонує багато функцій для збереження цілісності даних та економії простору. Весь файловий вміст та його метадані, папки разом з іншими структурами APFS зберігаються в контейнері APFS. Суперблок контейнеру (Container Superblock) зберігає інформацію про кількість блоків у контейнері, розмір блоку тощо. Інформація про всі зайняті та вільні блоки контейнера організується за допомогою структур бітових карт (Bitmap Structures). Кожен том в Контейнері має свій власний Суперблок тому (Volume Superblock), який містить інформацію про цей том. Всі файли та папки тому записуються в B-дерево файлів і папок (File and Folder B-Tree), тоді як B-дерево екстентів (Extents B-Tree) відповідє за додаткові відомості – посилання на вміст файлу (початок файлу, його довжину в блоках).

Файлові системи Linux

Linux як операційна система з відкритим кодом спрямована на запровадження, тестування та використання різних типів файлових систем. Найпопулярніші формати для Linux охоплюють:

Ext

Ext2, Ext3, Ext4 – це просто різні версії "нативної" файлової системи Linux Ext. Цей тип активно оновлюється та вдосконалюється. Ext3 як розширення до Ext2 використовує транзакційні операції запису файлів з журналом. Ext4 як продовження Ext3 було доповнено підтримкою оптимізованої інформації про розподіл файлів (екстентів) та розширених атрибутів файлів. Ця ФС часто використовується як "коренева" файлова система для більшості дистрибутивів Linux.

ReiserFS

ReiserFS – альтернативна файлова система Linux, оптимізована для зберігання величезної кількості невеликих файлів. Вона оснащена потужною здатністю пошуку файлів та дозволяє розміщувати файли компактно, зберігаючи їх хвости або просто дуже малі файли разом із метаданими, щоб уникнути використання великих блоків ФС для цих цілей. Втім, розвиток цього формату системи було призупинено, і вона більше не отримує активної підтримки.

XFS

XFS – надійна журнальована файлова система, яка була створена компанією Silicon Graphics та спочатку використовувалась IRIX-серверами компанії. У 2001 році вона потрапила до ядра Linux і тепер підтримується більшістю дистрибутивів, деякі з яких, наприклад Red Hat Enterprise Linux, навіть використовують її за замовчуванням. Цей тип ФС оптимізований для зберігання дуже великих файлів та томів на одному хості.

JFS

JFS – файлова система, розроблена IBM для потужних комп'ютерних систем Компанії. JFS1 зазвичай використовують для JFS, в той час, як JFS2 – її другий реліз. Нині цей проєкт має відкритий вихідний код та застосовується в більшості сучасних версій Linux.

Btrfs

Btrfs – файлова система, заснована на принципі копіювання при записі (COW), що була розроблена компанією Oracle та підтримується основним ядром Linux з 2009 року. Btrfs виконує також функції менеджера логічних томів, здатна охоплювати кілька пристроїв і пропонує набагато вищу відмовостійкість, кращу масштабованість, простіше адміністрування та інші просунуті можливості.

F2FS

F2FS – це файлова система Linux, розроблена компанією Samsung Electronics та адаптована до специфіки запам'ятовувальних пристроїв на базі флеш-пам'яті NAND, які широко використовуються в сучасних смартфонах та інших обчислювальних системах. Цей тип працює на основі підходу лог-структурованої ФС (LFS) та враховує такі особливості флеш-пам'яті, як постійний час доступу та обмежену кількість циклів перезапису даних. Замість створення одного великого сегмента для запису, F2FS збирає блоки в окремі сегменти (до 6), які записуються одночасно.

Поняття "жорстких посилань", що використовуються в даному виді операційних систем, робить більшість типів ФС Linux схожими в тому, що ім'я файлу не розглядається як файловий атрибут і, швидше, визначається як псевдонім для файлу в певному каталозі. Файловий об'єкт може мати посилання з багатьох локацій, навіть багаторазово з одного каталогу під різними назвами. Це може призвести до серйозних і навіть непереборних труднощів при відновленні імен файлів після їх видалення або логічного пошкодження системи.

Файлові системи BSD, Solaris, Unix

Найпоширенішою файловою системою для цих операційних систем є UFS (Unix File System), що також має назву FFS (Fast File System). На цей час UFS (у різних виданнях) підтримується всіма операційними системами Unix-сімейства і є основною файловою системою ОС BSD та Sun Solaris. Сучасні комп'ютерні технології мають тенденцію застосовувати заміни для UFS у різних операційних системах (ZFS для Solaris, JFS і похідні формати для Unix тощо).

Операційна система виконує наступні функції щодо управління файловою системою:

- Створення файлів і присвоєння їм імен;
- Створення каталогів (папок) і присвоєння їм імен;
- Перейменовування файлів і каталогів;
- Копіювання і переміщення файлів;
- Видалення файлів і каталогів;
- Навігація по файловій структурі;
- Управління атрибутами файлів.

Створення файлів в ОС Windows

Функція CreateFileA

Створює або відкриває файл або пристрій вводу/виводу. Найбільш часто використовувані пристрої введення-виводу: файл, потік файлів, каталог, фізичний диск, том, консольний буфер, стрічковий накопичувач, комунікаційний ресурс, поштовий слот і канал. Функція повертає дескриптор, який можна використовувати для доступу до файлу або пристрою для різних типів вводу-виводу залежно від файлу або пристрою та вказаних прапорів і атрибутів.

```
HANDLE CreateFileA
(       LPCTSTR lpFileName,
        DWORD dwDesiredAccess,
        DWORD dwShareMode,
        LPSECURITY_ATTRIBUTES lpSecurityAttributes,
        DWORD dwCreationDistribution,
        DWORD dwFlagsAndAttributes,
        HANDLE hTemplateFile
       );
```

lpFileName - ім’я файлу або пристрою, який потрібно створити чи відкрити.

dwDesiredAccess - доступ до файлу: читання, запис, виконання або комбінація. GENERIC_READ, GENERIC_WRITE, GENERIC_EXECUTE, GENERIC_READ | GENERIC_WRITE, GENERIC_ALL.

dwShareMode - запит на спільний доступ до файлу або пристрою, який можна читати, писати, видаляти або комбінація. Якщо цей параметр дорівнює нулю, а CreateFile успішно виконано, файл або пристрій не можуть бути надані спільно, і

їх не можна буде знову відкрити, доки не буде закрито дескриптор файлу або пристрою. Можливі варіанти: 0, FILE_SHARE_DELETE (доступ до видалення дозволяє виконувати операції як видалення, так і перейменування), FILE_SHARE_READ (дозволяє наступні операції відкриття файлу або пристрою для запиту доступу для читання), FILE_SHARE_WRITE (дозволяє наступні операції відкриття файлу або пристрою для запиту доступу до запису.)

lpSecurityAttributes - вказівник на структуру SECURITY_ATTRIBUTES, яка містить необов'язковий десриптор безпеки та булеве значення, що визначає, чи може повернутий дескриптор успадковуватися дочірніми процесами.

dwCreationDistribution - дія, яку потрібно виконати з файлом або пристроєм, який існує або не існує. CREATE_ALWAYS - Завжди створює новий файл. Якщо вказаний файл існує і доступний для запису, функція перезаписує файл. CREATE_NEW - Створює новий файл, тільки якщо він ще не існує. Якщо вказаний файл існує, функція не працює. OPEN_ALWAYS - Завжди відкриває файл. Якщо вказаний файл існує, функція виконується успішно. Якщо вказаний файл не існує і є дійсним шляхом до місця для запису, функція створює файл. OPEN_EXISTING - Відкриває файл або пристрій, лише якщо вони існують. Якщо вказаний файл або пристрій не існує, функція не працює. TRUNCATE_EXISTING - Відкриває файл і обрізає його так, щоб його розмір дорівнював нулю байтів, тільки якщо він існує. Якщо вказаний файл не існує, функція не працює.

dwFlagsAndAttributes - Атрибути та прапорці файлу або пристрою, FILE_ATTRIBUTE_NORMAL є найпоширенішим значенням за замовчуванням для файлів. Цей параметр також може містити комбінації прапорів (FILE_FLAG_) для керування поведінкою кешування файлів або пристроїв, режимів доступу та інших прапорів спеціального призначення. Вони поєднуються з будь-якими значеннями FILE_ATTRIBUTE_. Цей параметр також може містити інформацію про якість обслуговування безпеки (SQOS), вказавши прапор SECURITY_SQOS_PRESENT. Деякі атрибути файлів: FILE_ATTRIBUTE_ARCHIVE - Файл має бути заархівований. Програми використовують цей атрибут для позначення файлів для резервного копіювання або видалення. FILE_ATTRIBUTE_READONLY - Файл доступний лише для читання. Програми можуть читати файл, але не можуть записувати або видаляти його. Деякі прапорці: FILE_FLAG_NO_BUFFERING - Файл або пристрій відкривається без системного кешування для читання та запису даних. Цей прапорець не впливає на кешування жорсткого диска або файли, відображені в пам'яті. FILE_FLAG_DELETE_ON_CLOSE - Файл слід видалити відразу після закриття всіх його дескрипторів, що включає вказаний дескриптор та будь-які інші відкриті або дубльовані дескриптори. FILE_FLAG_BACKUP_SEMANTICS - Файл відкривається або створюється для резервного копіювання або відновлення. Система гарантує, що процес, що викликає, перевизначає перевірки безпеки файлів, якщо процес має привілеї SE_BACKUP_NAME і

SE_RESTORE_NAME. FILE_FLAG_OVERLAPPED - Файл або пристрій відкривається або створюється для асинхронного введення-виводу. Коли наступні операції введення-виводу будуть завершені над цим дескриптором, подія, зазначена в структурі OVERLAPPED, буде встановлено в сигналізований стан. Якщо зазначено цей прапорець, файл можна використовувати для одночасних операцій читання та запису. FILE_FLAG_RANDOM_ACCESS - Доступ має бути випадковим. Система може використовувати це як підказку для оптимізації кешування файлів. Цей прапор не діє, якщо файлова система не підтримує кешований ввод-вивод і FILE_FLAG_NO_BUFFERING.
hTemplateFile Дійсний дескриптор файлу шаблону з правом доступу GENERIC_READ. Файл шаблону надає атрибути файлу та розширені атрибути для файлу, який створюється. Цей параметр може мати значення NULL.

Для зміни атрибутів файлу можна використати функцію:
BOOL **SetFileAttributes**(
  LPCTSTR lpFileName, // ім’я файлу
  DWORD dwFileAttributes // атрибути (див. ***dwFlagsAndAttributes***)
);
Змінити власника документа можна, змінивши структуру дескриптора безпеки.
Функція SetNamedSecurityInfo встановлює зазначену інформацію безпеки в дескрипторі безпеки зазначеного об'єкта.
DWORD SetNamedSecurityInfoA(
  [in] LPSTR pObjectName,
  [in] SE_OBJECT_TYPE ObjectType,
  [in] SECURITY_INFORMATION SecurityInfo,
  [in, optional] PSID psidOwner,
  [in, optional] PSID psidGroup,
  [in, optional] PACL pDacl,
  [in, optional] PACL pSacl
);
pObjectName вказівник на рядок, що закінчується нулем, що вказує ім'я об'єкта, для якого встановлюється інформація безпеки. Це може бути ім’я локального або віддаленого файлу чи каталогу у файловій системі NTFS, спільна мережа, ключ реєстру, семафор, подія, мьютекс, відображення файлів або таймер очікування.
ObjectType для файлів або директорій SE_FILE_OBJECT
SecurityInfo Набір бітових прапорів, які вказують тип інформації безпеки, яку потрібно встановити. Цей параметр може бути комбінацією бітових прапорів SECURITY_INFORMATION. Тип даних SECURITY_INFORMATION ідентифікує пов'язану з об'єктами інформацію безпеки, яка встановлюється або запитується. Ця інформація безпеки включає:

- Власник об'єкта
- Основна група об'єкта

- Список дискреційного контролю доступу (DACL) об’єкта
- Список системного контролю доступу (SACL) об'єкта

psidOwner вказівник на структуру SID, яка ідентифікує власника об'єкта. Якщо абонент не має константи SeRestorePrivilege, цей SID повинен міститися в маркері абонента і має увімкнути дозвіл SE_GROUP_OWNER. Параметр SecurityInfo має містити прапор OWNER_SECURITY_INFORMATION. Щоб встановити власника, абонент повинен мати доступ WRITE_OWNER до об’єкта або ввімкнути привілей SE_TAKE_OWNERSHIP_NAME. Якщо ви не встановлюєте ідентифікатор власника, цей параметр може мати значення NULL.
psidGroup вказівник на SID, який ідентифікує первинну групу об'єкта. Параметр SecurityInfo має включати прапор GROUP_SECURITY_INFORMATION. Якщо ви не встановлюєте SID основної групи, цей параметр може мати значення NULL.
pDacl вказівник на новий DACL для об'єкта. Параметр SecurityInfo повинен містити прапор DACL_SECURITY_INFORMATION. Абонент повинен мати доступ WRITE_DAC до об’єкта або бути власником об’єкта. Якщо ви не встановлюєте DACL, цей параметр може бути NULL.
pSacl вказівник на новий SACL для об'єкта. Параметр SecurityInfo має містити будь-який з таких прапорів: SACL_SECURITY_INFORMATION, LABEL_SECURITY_INFORMATION, ATTRIBUTE_SECURITY_INFORMATION, SCOPE_SECURITY_INFORMATION або BACKUP_SECURITY_INFORMATION.
Якщо встановлено SACL_SECURITY_INFORMATION або SCOPE_SECURITY_INFORMATION, абонент повинен мати привілей SE_SECURITY_NAME. Якщо ви не встановлюєте SACL, цей параметр може мати значення NULL.

Створення файлів Linux
#include <sys/stat.h>
#include <fcntl.h>
Створити файл можна з допомогою функції:
int creat(const char *path, mode_t mode);
Її аналогом є int open(const char *path, **int** oflag, **mode_t mode);**
Функція open() встановлює зв'язок між файлом і дескриптором файлу. Дескриптор файлу використовується іншими функціями вводу-виводу для посилання на цей файл. Аргумент path вказує на ім'я шляху, що називає файл.
Зміщення файлу, яке використовується для позначення поточної позиції у файлі, встановлюється на початок файлу.
Прапорці стану файлів і режими доступу до файлів опису відкритого файлу встановлюються відповідно до значення oflag:
O_RDONLY - Відкритий лише для читання.

O_WRONLY - Відкритий лише для написання.
O_RDWR - Відкритий для читання та письма. Результат є невизначеним, якщо цей прапор застосовано до FIFO.
Можна використовувати будь-яку комбінацію з наступного:
O_APPEND - Якщо встановлено, зміщення файлу має бути встановлено на кінець файлу перед кожним записом.
O_CREAT - Якщо файл існує, цей прапор не діє
O_DSYNC - Операції запису в дескриптор файлу повинні виконуватися, як визначено синхронізованим завершенням цілісності даних вводу-виводу.
O_EXCL - Якщо встановлено O_CREAT і O_EXCL, open() зазнає невдачі, якщо файл існує.
O_NOCTTY - Якщо набір і шлях ідентифікують термінальний пристрій, open() не спричиняє того, що термінальний пристрій стане керуючим терміналом для процесу.
O_NONBLOCK - Під час відкриття FIFO з встановленими O_RDONLY або O_WRONLY: Якщо встановлено O_NONBLOCK, open() лише для читання повернеться без затримки. Функція open() лише для запису повертає помилку, якщо жоден процес не має відкритий файл для читання. Якщо O_NONBLOCK очищений, open() лише для читання блокуватиме потік, що викликає, доки потік не відкриє файл для запису. Open() лише для запису блокуватиме потік, що викликає, доки потік не відкриє файл для читання.
O_RSYNC - Операції зчитування вводу-виводу над дескриптором файлу мають виконуватися на тому самому рівні цілісності, що визначено прапорами O_DSYNC та O_SYNC. Якщо обидва O_DSYNC і O_RSYNC встановлені в oflag, всі операції введення/виводу над дескриптором файлу мають завершитися, як визначено синхронізованим завершенням цілісності даних введення/виводу. Якщо і O_SYNC, і O_RSYNC встановлені у прапорах, усі операції введення-виводу над дескриптором файлу мають завершитися, як визначено синхронізованим завершенням цілісності файлу введення-виводу.
O_SYNC - Операції запису в дескриптор файлу повинні завершитися, як визначено синхронізованим завершенням цілісності файлу введення-виводу.
O_TRUNC - Якщо файл існує і є звичайним файлом, і файл успішно відкритий O_RDWR або O_WRONLY, його довжина має бути скорочена до 0, а режим і власник не змінюються. Це не має впливати на спеціальні файли FIFO або файли термінальних пристроїв. Його вплив на інші типи файлів визначається реалізацією. Результат використання O_TRUNC з O_RDONLY не визначено.
Якщо встановлено O_CREAT і файл раніше не існував, після успішного завершення open() позначає для оновлення поля st_atime, st_ctime і st_mtime файлу, а також поля st_ctime і st_mtime батьківського каталогу.

Аргумент mode визначає права доступу, які використовуються у разі створення нового файлу. Вони модифікуються звичайним способом, використовуючи umask процесу: права доступу створеного файлу рівні (mode & ~umask). Зауважте, що цей режим доступу стосується тільки до наступних звернень до нового файлу;. Нижче наведені символічні константи можна використовувати в mode:

S_IRWXU 00700 користувач (власник файлу) має права читання, запису та виконання.

S_IRUSR (S_IREAD) 00400 Користувач має право читання
S_IWUSR (S_IWRITE) 00200 Користувач має право запису
S_IXUSR (S_IEXEC) 00100 користувач має право на виконання
S_IRWXG 00070 група має права читання, запису та виконання
S_IRGRP 00040 група має право читання
S_IWGRP 00020 група має право запису
S_IXGRP 00010 група має право на виконання
S_IRWXO 00007 всі інші мають права читання, запису та виконання
S_IROTH 00004 всі інші мають право на читання
S_IWOTH 00002 всі інші мають право запису
S_IXOTH 00001 всі інші мають право на виконання

mode завжди повинен бути вказаний при використанні O_CREAT, у всіх інших випадках цей параметр ігнорується.

Змінити права доступу до файлу можна з допомогою функцій
int chmod(const char *_path_, mode_t _mode_);
int fchmod(int _fd_, mode_t _mode_);
Нові дозволи файлу вказуються в *mode*, який є бітовою маскою, створеною шляхом АБО об'єднання нуля або більше з наступного:
S_ISUID (04000) set-user-ID (встановити ефективний ідентифікатор користувача процесу)
S_ISGID (02000) set-group-ID (встановити ефективний ідентифікатор групи процесу)
S_IRUSR (00400) читає власник
S_IWUSR (00200) пише власник
та ін.

Завдання

1. Створити за допомогою API функцій файл для запису результатів виконання лабораторної роботи №3 (відповідно до свого варіанту).
2. Створити файл для запису результатів виконання лабораторної роботи №4 (відповідно до свого варіанту).
3. Реалізувати зміну прав доступу користувача до файлу.
4. Результати виконання роботи відобразити у звіті.