ТЕМА 4.1

ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ ТА КАТАЛОГАМИ

ПЛАН

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

3 ЗАХИСТ ФАЙЛІВ

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Історія систем управління даними в зовнішній пам'яті починається ще з магнітних стрічок, але сучасний вигляд вони придбали з появою магнітних дисків. До цього кожна прикладна програма сама вирішувала проблеми іменування даних і їх структуризації в зовнішній пам'яті. Це утрудняло підтримку на зовнішньому носієві декількох архівів довготривалого збереження інформації. Історичним кроком став перехід до використання централізованих систем управління файлами. **Система управління файлами** бере на себе розподіл зовнішньої пам'яті, відображення імен файлів в адреси зовнішньої пам'яті і забезпечення доступу до даних.

У цій лекції розглядаються питання структури, іменування, захисту файлів.

Файли є логічними інформаційними блоками, що створюються процесами.

Файлами управляє операційна система. Структура файлів, їх імена, доступ до них, їх використання, захист, реалізація і управління ними є основними питаннями розробки операційних систем.

З позиції користувача найбільш важливим аспектом файлової системи є її представлення, тобто що є файлом, як файли іменуються, який захист мають, які операції дозволено проводити з файлами і т. д.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Імена файлів. Файл є механізмом абстрагування. Він надає спосіб збереження інформації на диску і подальшого її зчитування, який повинен захистити користувача від подробиць про спосіб і місце зберігання інформації і деталей фактичної роботи дискових пристроїв.

Напевно, найбільш важливою характеристикою будь-якого механізму абстрагування є спосіб управління об'єктами і їх іменування, тому дослідження файлової системи розпочнеться з питання, що стосується імен файлів. **Коли процес створює файл, він привласнює йому ім'я. Коли процес завершується, файл продовжує існувати, і до нього по цьому імені можуть звертатися інші процеси.**

Конкретні правила складання імен файлів варіюються від системи до системи, але усі нині існуючі операційні системи в якості допустимих імен файлів дозволяють використовувати від однієї до восьми букв. Багато файлових систем підтримують імена завдовжки до 255 символів.

Деякі файлові системи розрізняють букви верхнього і нижнього регістрів, а деякі не роблять таких відмінностей. Система UNIX підпадає під першу категорію, а стара MS-DOS — під другу.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Багато операційних систем підтримують імена файлів, що складаються з двох частин, розділених точкою, як, наприклад, prog.c. Та частина імені, яка йде за точкою, називається **розширенням імені файлу** і, як правило, несе в собі деяку інформацію про файл. Приміром, в MS-DOS імена файлів складаються з 1-8 символів і мають (необов'язково) розширення, що складається з 1-3 символів. У UNIX кількість розширень вибирає сам користувач, так що ім'я файлу може мати два і більше розширень, наприклад homepage.html.zip, де .html вказує на наявність веб-сторінки в коді HTML, а .zip — на те, що цей файл (homepage.html) був стислий архіватором.

У деяких системах (наприклад, в усіх різновидах UNIX) розширення імен файлів використовуються відповідно до угод і не нав'язуються операційною системою.

Система Windows, навпаки, знає про розширення імен файлів і привласнює кожному розширенню цілком певне значення. Користувачі (чи процеси) можуть реєструвати розширення в операційній системі, вказуючи програму, яка стане їх «власником». При подвійному клацанні миші на імені файлу запускається програма, призначена цьому розширенню, з ім'ям файлу в якості параметра. Наприклад, подвійне клацання миші на імені file.docx запускає

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Структура файлу. Файли можуть бути структуровані декількома різними способами. Три найбільш вірогідні структури показані на рисунку 1. Файл на рисунку 1,а) є несистемною послідовністю байтів. По суті, операційній системі все одно, що міститься в цьому файлі, — вона бачить тільки байти. Яке-небудь значення цим байтам надають програми на рівні користувача. Такий підхід використов

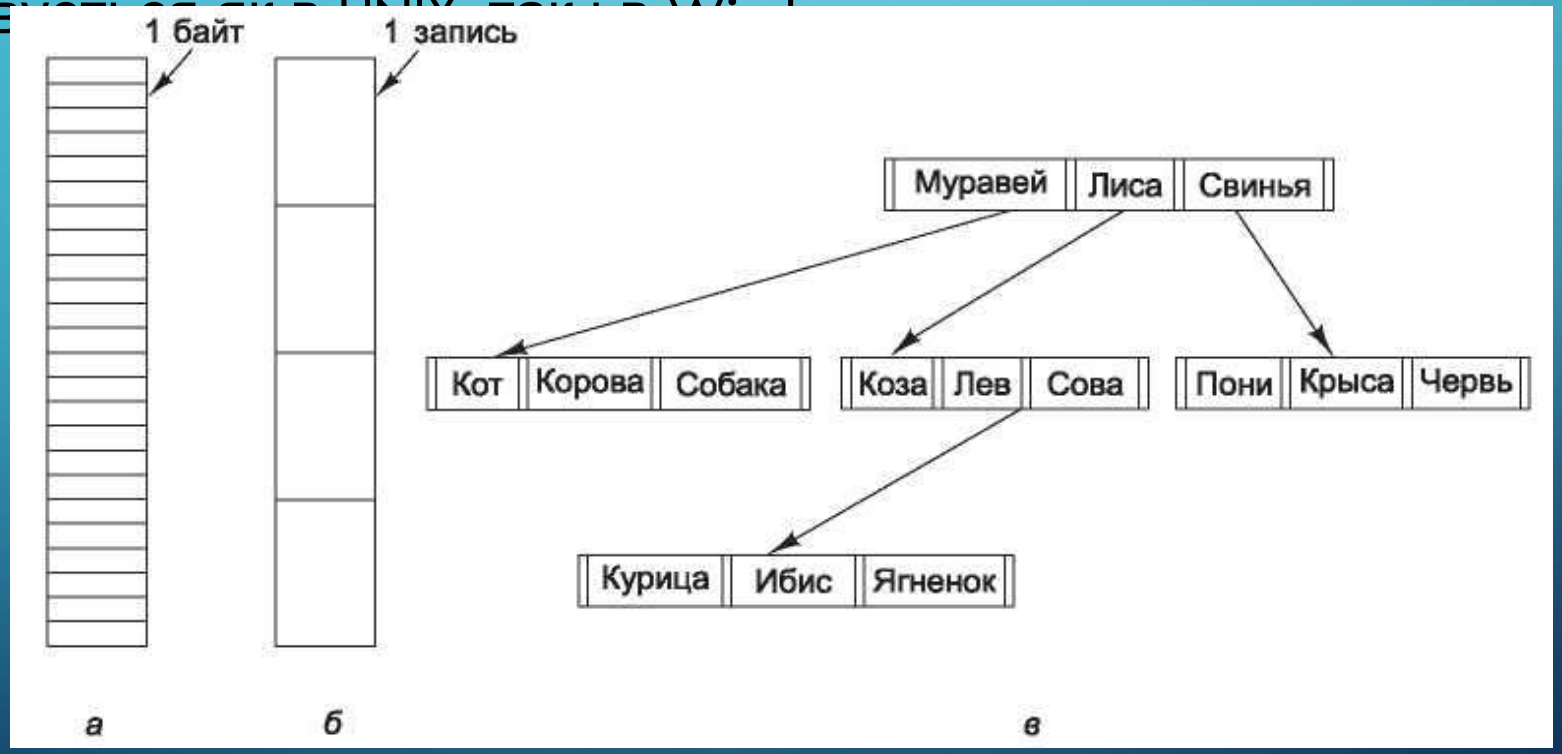


Рисунок 1 - Три типи файлів : а - послідовність байтів; б - послідовність записів;

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Коли операційна система вважає, що **файли — це не більше ніж послідовність байтів**, вона надає максимум гнучкості. Програми користувача можуть поміщати у свої файли усе, що їм заманеться, і називати їх, як їм зручно. Операційна система нічим при цьому не допомагає, але і нічим не заважає. Остання обставина може мати особливе значення для тих користувачів, які хочуть зробити що-небудь незвичайне. Ця файлова модель використовується усіма версіями UNIX (включаючи Linux і OS X) і Windows.

Перший крок назустріч деякій структурі показаний на рисунку 1,б. У цій моделі **файл є послідовністю записів фіксованої довжини**, кожна з яких має власну внутрішню структуру. Основна ідея файлу як послідовності записів полягає в тому, що операція читання повертає один із записів, а операція запису перезаписує або доповнює один із записів. Жодна сучасна універсальна система більше не використовує цю модель як свою первинну файлову систему.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Типи файлів. Багато операційних систем підтримують декілька типів файлів. Приміром, в системах UNIX (знову ж таки включаючи OS X) і Windows є **звичайні файли і каталоги**. У системі UNIX є також **символьні і блокові спеціальні файли**.

Звичайними вважаються файли, що містять інформацію користувача. Усі файли на рисунку 1 є звичайними.

Каталоги — це системні файли, призначені для підтримки структури файлової системи.

Символьні спеціальні файли мають відношення до вводу-виводу і використовуються для моделювання послідовних пристроїв вводу-виводу, до яких відносяться термінали, принтери і мережі.

Блокові спеціальні файли використовуються для моделювання дисків.

Нас в першу чергу цікавитимуть звичайні файли.

І ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Як правило, **до звичайних файлів відносяться або файли ASCII, або двійкові файли. ASCII-файли складаються з текстових рядків.** У деяких системах кожен рядок завершується символом повернення каретки. У інших системах використовується символ переводу рядка. Деякі системи (наприклад, Windows) використовують обидва символи. Рядки не обов'язково повинні мати однакову довжину.

Великою перевагою ASCII-файлів є можливість їх відображення і роздруку в початковому виді, також вони можуть бути відредаговані у будь-якому текстовому редакторові.

Усі інші файли відносяться до двійкових — це означає, що вони не являються ASCII-файлами. Їх роздрук буде незрозумілим набором символів. Зазвичай у них є деяка внутрішня структура, відома програмі, що використовує їх.

І ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Доступ до файлів. У найперших операційних системах надавався тільки один тип доступу до файлів — **послідовний**. У цих системах процес міг читати усі байти або записи файлу тільки по порядку, із самого початку, але не міг перестрибнути і читати їх поза порядком їх дотримання. Але послідовні файли можна було перемотати назад, щоб читати їх у міру потреби. Ці файли були зручні в ті часи, коли носієм в сховищах даних служила магнітна стрічка, а не диск.

Коли для зберігання файлів стали використовуватися диски, з'явилася можливість зчитувати байти або записи файлу поза порядком їх розміщення або діставати доступ до записів по ключу, а не по позиції. **Файли, в яких байти або записи могли бути зчитані у будь-якому порядку, стали називати файлами довільного доступу.** Вони затребувані багатьма додатками.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Файли довільного доступу є невід'ємною частиною багатьох додатків, наприклад систем управління базами даних.

Для визначення місця початку зчитування можуть бути застосовані два методи.

При першому методі позиція у файлі, з якою починається читання, задається при кожній операції читання read.

При другому методі для установки на поточну позицію надається спеціальна операція пошуку потрібного місця seek. Після цієї операції файл може бути зчитаний послідовно з тільки що встановленої позиції. Останній метод використовується в UNIX і Windows.

І ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Атрибути файлів. У кожного файлу є свої ім’я і дані. На додаток до цього усі операційні системи зв’язують з кожним файлом і іншу інформацію, приміром дату і час останньої модифікації файлу і його розмір. Ми називатимемо ці додаткові відомості **атрибутами файлу**. Також їх називають **метаданими**. Список атрибутів істотно варіюється від системи до системи. У таблиці 1 показані деякі з можливих атрибутів, але окрім них існують і інші атрибути. Жодна з існуючих систем не має усіх цих атрибутів, але кожен з них є присутнім в якій-небудь системі.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Таблиця 1 - Деякі з можливих атрибутів

Атрибут	Значення
Захист	Хто і яким чином може отримати доступ до файлу
Пароль	Пароль для діставання доступу до файлу
Творець	Ідентифікатор творця файлу
Власник	Поточний власник
Прапор «тільки для читання»	0 — для читання і запису; 1 — тільки для читання
Прапор «прихований»	0 — звичайний; 1 — не призначений для відображення в переліку файлів
Прапор «системний»	0 — звичайний; 1 — системний
Прапор «архівний»	0 — що пройшов резервне копіювання; 1 — що потребує резервного копіювання
Прапор «ASCII/двійковий»	0 — ASCII; 1 — двійковий
Прапор довільного доступу	0 — тільки послідовний доступ; 1 — довільний доступ
Прапор «тимчасовий»	0 — звичайний; 1 — що видаляється після закінчення роботи процесу
Прапори блокування	0 — незаблокований; ненульове значення — заблокований
Довжина запису	Кількість байтів в записі
Позиція ключа	Зміщення ключа усередині кожного запису
Довжина ключа	Кількість байтів в полі ключа
Час створення	Дата і час створення файлу
Час останнього доступу	Дата і час останнього доступу до файлу
Час внесення останніх змін	Дата і час внесення у файл останніх змін
Поточний розмір	Кількість байтів у файлі
Максимальний розмір	Кількість байтів, до якої файл може збільшуватися

ТЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Перші чотири атрибути відносяться до захисту файлу і повідомляють про те, хто може мати до нього доступ, а хто ні. Можливе застосування різноманітних схем. У деяких системах для доступу до файлу користувач повинен ввести пароль, в цьому випадку пароль може бути одним з атрибутів файлу.

Прапори є бітами або невеликими полями, за допомогою яких відбувається управління деякими конкретними властивостями або дозвіл їх застосування. Наприклад, приховані файли не з'являються в лістингу файлів. Прапор архівації є бітом, за допомогою якого відстежується, чи була нещодавно зроблена резервна копія файлу. Цей прапор скидається програмою архівації і встановлюється операційною системою при внесенні у файл змін. Таким чином програма архівації може визначити, які файли слід архівувати. Прапор «тимчасовий» дозволяє автоматично видаляти помічений ним файл після закінчення роботи процесу, що створив його.

Г ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Поля довжини запису, позиції ключа і довжини ключа є тільки у тих файлів, записи в яких можна шукати по ключу. Вони надають інформацію, необхідну для пошуку ключів.

Різні показники часу дозволяють відстежувати час створення файлу, останнього доступу до цього файлу, його останньої зміни. Ці відомості можуть виявитися корисними для досягнення різних цілей. Приміром, якщо початковий файл був змінений вже після створення відповідного об’єктного файлу, то він потребує перекомпіляції. Необхідну для цього інформацію надають поля часу.

Поточний розмір показує, наскільки великим є файл нині. Деякі старі операційні системи універсальних машин вимагають при створенні файлу вказувати його максимальний розмір, щоб дозволити операційній системі заздалегідь виділити максимальне місце для його зберігання. Операційні системи робочих станцій і персональних комп’ютерів досить розумні, щоб обійтися без цієї особливості.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Операції з файлами. Файли призначені для зберігання інформації з можливістю її подальшого витягання. Різні системи надають різні операції, що дозволяють зберігати і витягати інформацію. Далі розглядаються найбільш поширені системні виклики, що відносяться до роботи з файлами:

- ✔ **Create (Створити).** Створює файл без даних. Мета виклику полягає в оголошенні про появу нового файлу і установку ряду атрибутів.
- ✔ **Delete (Видалити).** Коли файл більше не потрібний, його треба видалити, щоб звільнити дисковий простір. Саме для цього і призначений цей системний виклик.
- ✔ **Open (Відкрити).** Перед використанням файлу процес повинен його відкрити. Мета системного виклику open — дати можливість системі витягнути і помістити в оперативну пам'ять атрибути і перелік адрес на диску, щоб прискорити доступ до них при подальших викликах.

І ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Операції з файлами.

- ✔ **Close (Закрити).** Після завершення усіх звернень до файлу потреба в його атрибутах і адресах на диску вже відпадає, тому файл має бути закритий, щоб звільнити місце у внутрішній таблиці. Багато систем встановлюють максимальну кількість відкритих процесами файлів, визначаючи сенс існування цього виклику. Інформація на диск пишеться блоками, і закриття файлу змушує до запису останнього блоку файлу, навіть якщо цей блок і не заповнений.
- ✔ **Read (Зробити читання).** Зчитування даних з файлу. Як правило, байти поступають з поточної позиції. Викликаючий процес повинен вказати об'єм необхідних даних і надати буфер для їх розміщення.
- ✔ **Write (Зробити запис).** Запис даних у файл, як правило, з поточної позиції. Якщо ця позиція знаходиться у кінці файлу, то його розмір збільшується. Якщо поточна позиція знаходиться десь в середині файлу, то нові дані пишуться поверх існуючих, які втрачаються назавжди.
- ✔ **Append (Додати).** Цей виклик є усіченою формою системного виклику write. Він може лише додати дані в кінець файлу. Як правило, у систем, що надають мінімальний набір системних викликів, виклик append відсутній, але багато систем надають безліч способів отримання того ж результату, і іноді в цих системах є присутнім виклик append.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФАЙЛИ. ОПЕРАЦІЇ НАД ФАЙЛАМИ

Операції з файлами.

- ✔ **Seek (Знайти).** При роботі з файлами довільного доступу потрібний спосіб вказівки місця, з якого беруться дані. Одним із загальноприйнятих підходів є застосування системного виклику seek, який переміщає покажчик файлу до певної позиції у файлі. Після завершення цього виклику дані можуть зчитуватися або записуватися з цієї позиції.
- ✔ **Get attributes (Отримати атрибути).** Процесу для роботи частенько необхідно зчитати атрибути файлу.
- ✔ **Set attributes (Встановити атрибути).** Значення деяких атрибутів можуть встановлюватися користувачем і змінюватися після того, як файл був створений. Таку можливість дає саме цей системний виклик. Характерним прикладом може послужити інформація про режим захисту. Під цю ж категорію підпадають більшість прапорів.
- ✔ **Rename (Перейменувати).** Нерідко користувачеві вимагається змінити ім'я існуючого файлу. Цей системний виклик допомагає вирішити це завдання. Необхідність в ньому виникає не завжди, оскільки файл може бути просто

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Зазвичай у файловій системі для впорядкування файлів є **каталоги або папки (або директорії)**, які самі по собі є файлами.

Системи з однорівневими каталогами. Найпростіша форма системи каталогів складається з одного каталогу, що містить усі файли. Іноді він називається **кореневим каталогом**, але оскільки він один-єдиний, то ім'я особливого значення не має. Ця система була широко поширена на перших персональних комп'ютерах частково через те, що у них був всього один користувач.

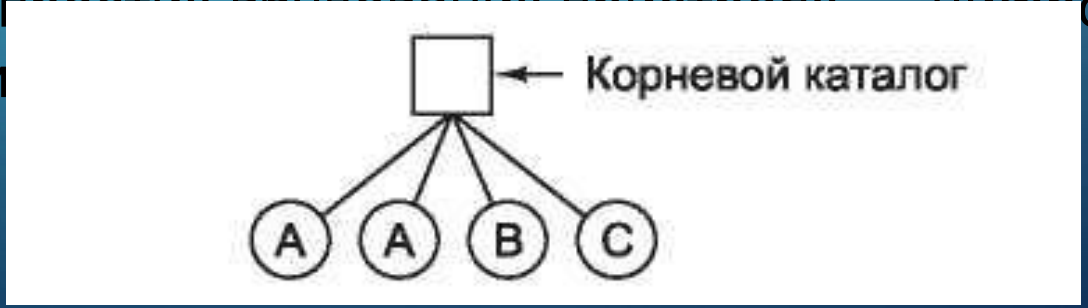
Приклад системи, що має всього один каталог, показаний на рисунку 2. На нім зображений каталог, в якому містяться чотири файли. **Переваги такої схеми полягають в її простоті і можливості швидкого знаходження файлів, оскільки пошук ведеться всього в одному місці.** Така система іноді все ще використовується в простих вбудованих пристроях — цифрових камерах і деяких переносних музичних плеєрах.

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Зазвичай у файловій системі для впорядкування файлів є **каталоги або папки (або директорії)**, які самі по собі є файлами.

Системи з одноріВневими каталогами. Найпростіша форма системи каталогів складається з одного каталогу, що містить усі файли. Іноді він називається **кореневим каталогом**, але оскільки він один-єдиний, то ім'я особливого значення не має. Ця система була широко поширена на перших персональних комп'ютерах частково через те, що у них був всього один користувач.

Приклад системи, що має всього один каталог, показаний на рисунку 2. На нім зображений каталог, в якому містяться чотири файли. **Переваги такої схеми полягають в її простоті і можливості швидкого знаходження файлів, оскільки пошук ведеться всього в одному місці.** Така система іноді все ще використовується в простих вбудованих пристроях, цифрових камерах і деяких переносних музичних



2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Ієрархічні системи каталогів. Однорівнева система більше підходить для дуже простих спеціалізованих додатків (і застосовувалася навіть на перших персональних комп'ютерах), але сучасним користувачам, працюючим з тисячами файлів, знайти що-небудь, якщо усі файли знаходяться в одному каталозі, буде практично неможливо.

Тому потрібний спосіб, що дозволяє згрупувати споріднені файли.

Для організації усього цього потрібна ієрархія (тобто **дерево каталогів**). Такий підхід дозволяє мати стільки каталогів, скільки необхідно для угрупування файлів природним чином. Крім того, якщо загальний файловий сервер спільно використовується декількома користувачами, як це буває у багатьох мережах організацій, кожен користувач може мати кореневий каталог для власної ієрархії. Саме цей підхід показаний на рисунку 3. На нім зображені каталоги A, B і C, які містяться в кореневому каталозі і належать різним користувачам, двоє з яких створили підкаталоги для проектів, над якими працюють.

Можливість створення довільної кількості підкаталогів надає користувачам потужний інструмент структуризації, що дозволяє організувати їх роботу. Тому таким чином влаштовані практично усі

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

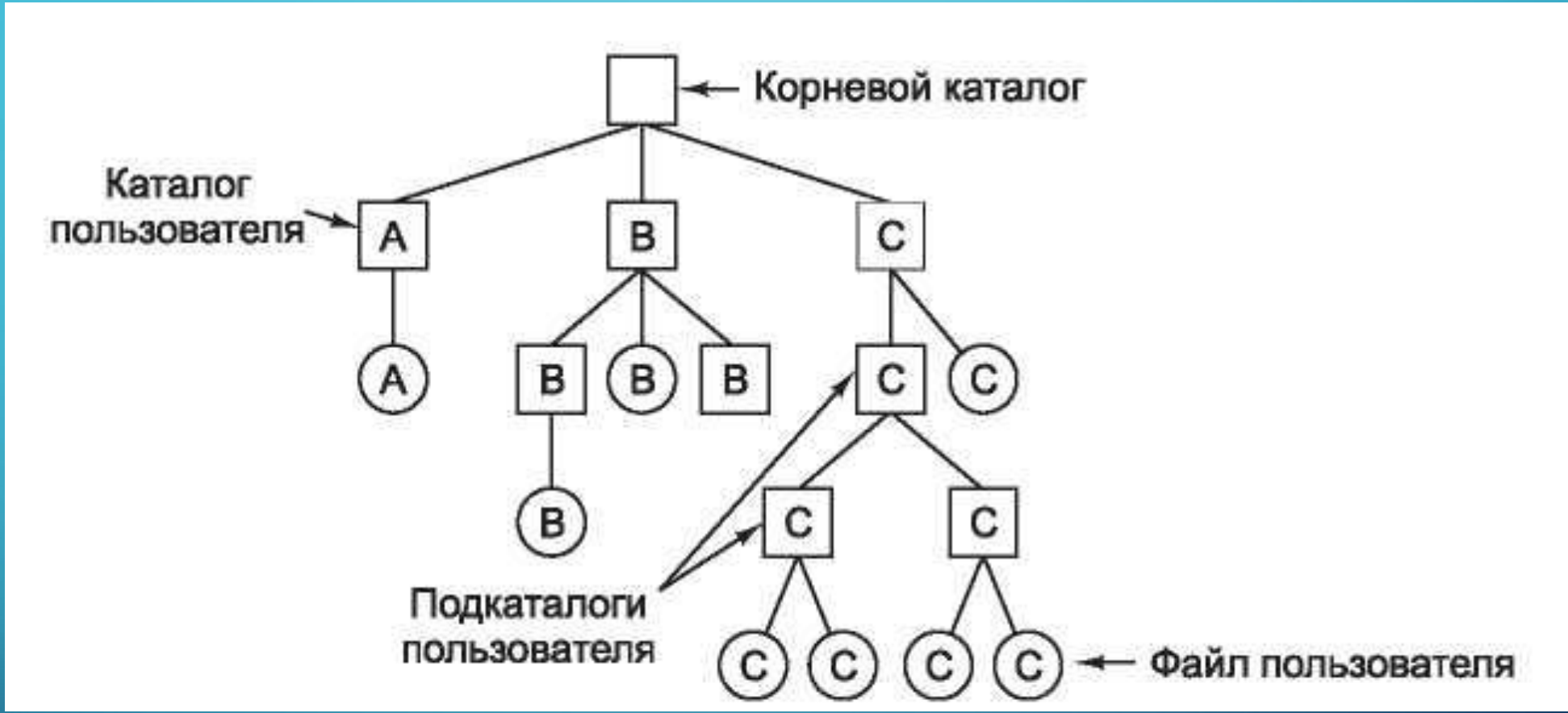


Рисунок 3 - Ієрархічна система каталогів

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Імена файлів. Коли файлова система організована у вигляді дерева каталогів, потрібний який-небудь спосіб вказівки імен файлів. Найчастіше для цього використовуються два методи. У першому методі кожному файлу дається **абсолютне ім'я (повне ім'я)**, що складається з шляху від кореневого каталогу до файлу.

Наприклад, ім'я /usr/ast/mailbox означає, що кореневий каталог містить підкаталог usr, який, у свою чергу, містить підкаталог ast, в якому міститься файл mailbox. Абсолютні імена файлів завжди розпочинаються з назви кореневого каталогу і є унікальними іменами. У системі UNIX компоненти шляху розділяються символом «слеш» — /. В системі Windows роздільником служить символ «зворотний слеш» — \. У цих двох системах одне і те ж ім'я виглядатиме таким чином:

Windows - \usr\ast\mailbox

UNIX - /usr/ast/mailbox

Якщо в якості першого символу в імені файлу використовується роздільник, то незалежно від символу, використовуваного в цій якості, шлях буде абсолютним.

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Імена файлів. Іншим різновидом імені є **відносне ім'я**. Воно використовується спільно з поняттям робочого каталогу (що називається також **поточним каталогом**). Користувач може визначити один каталог в якості поточного, і тоді усі імена файлів стануть розглядатися відносно робочого каталогу і не розпочинатимуться з кореневого каталогу. Приміром, якщо поточним робочим каталогом буде /usr/ast, то до файлу, що має абсолютне ім'я /usr/ast/mailbox, можна буде звертатися, просто вказуючи mailbox.

Більшість операційних систем, що підтримують ієрархічну систему каталогів, мають в кожному каталозі спеціальні елементи **«.» і «..»**, які зазвичай вимовляються як «крапка» і «крапка-крапка». **Крапка є посиланням на поточний каталог, а подвійна крапка — на батьківський каталог** (за винятком кореневого каталогу, де цей елемент є посиланням на сам кореневий каталог).

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Операції з каталогами. Допустимі системні виклики для управління каталогами мають більшу кількість варіантів від системи до системи, чим системні виклики, що управляють файлами. Розглянемо приклади, що дають уявлення про ці системні виклики і характер їх роботи (узяті з системи UNIX):

- ✔ **Create (Створити каталог).** Каталог створюється порожнім, за винятком крапки і подвійної крапки, які система поміщає в нього автоматично.
- ✔ **Delete (Видалити каталог).** Видалити можна тільки порожній каталог. Каталог, що містить тільки крапку і подвійну крапку, розглядається як порожній, оскільки вони не можуть бути видалені.
- ✔ **Opendir (Відкрити каталог).** Каталоги можуть бути зчитані. Приміром, для виведення імен усіх файлів, що містяться в каталозі, програма ls відкриває каталог для читання імен усіх файлів, що містяться в нім. Перш ніж каталог може бути зчитаний, він має бути відкритий по аналогії з відкриттям і читанням файлу.
- ✔ **Closedir (Закрити каталог).** Коли каталог зчитаний, він має бути закритий.

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Операції з каталогами.

- ✔ **Readdir (Прочитати каталог).** Цей виклик повертає наступний запис з відкритого каталогу. Раніше каталоги можна було читати за допомогою звичайного системного виклику read, але недолік такого підходу полягав в тому, що програміст вимушений був працювати з внутрішньою структурою каталогів, про яку він повинен був знати заздалегідь. На відміну від цього, readdir завжди повертає один запис в стандартному форматі незалежно від того, яка з можливих структур каталогів використовується.
- ✔ **Rename (Перейменувати каталог).** У багатьох відношеннях каталоги подібні до файлів і можуть бути перейменовані точно так, як і файли.
- ✔ **Link (Прив'язати).** Прив'язка є технологією, що дозволяє файлу з'являтися більш ніж в одному каталозі. У цьому системному виклику вказуються існуючий файл і нове ім'я файлу в деякому існуючому каталозі і створюється прив'язка існуючого файлу до вказаного каталогу з вказаним новим ім'ям. Таким чином, один і той же файл може з'явитися в декількох каталогах, можливо, під різними іменами. Подібна прив'язка іноді називається жорстким зв'язком, або жорстким посиланням (hard link).

2 КАТАЛОГИ. ОПЕРАЦІЇ НАД КАЛАТОГАМИ

Операції з каталогами.

- **Unlink (Відв'язати).** Видалити запис каталогу. Якщо відв'язуваний файл є присутнім тільки в одному каталозі (що найчастіше і буває), то цей виклик видалить його з файлової системи. Якщо він фігурує в декількох каталогах, то він буде видалений з каталогу, який вказаний в імені файлу. Усі інші записи залишаться. Фактично системним викликом для видалення файлів в UNIX (як раніше вже було розглянуто) являється unlink.

У приведеному списку перераховані найбільш важливі виклики, але існують і інші виклики, приміром для управління захистом інформації, пов'язаної з каталогами.

Ще одним варіантом ідеї прив'язки файлів є **символічне посилання (symbolic link)**. Замість двох імен, що вказують на одну і ту ж внутрішню структуру даних, що представляє файл, може бути створене ім'я, що вказує на дуже маленький файл, в якому міститься ім'я іншого файлу. Коли використовується перший файл, наприклад він відкривається, файлова система йде по вказаному шляху і у результаті знаходить ім'я. Потім вона починає процес пошуку усіх місць, де використовується це нове ім'я. Перевагою символічних посилань є те, що вони можуть перетинати межі дисків і навіть вказувати на імена файлів, що знаходяться на віддалених комп'

3 ЗАХИСТ ФАЙЛІВ

Наявність в системі багатьох користувачів припускає організацію контрольованого доступу до файлів. Виконання будь-якої операції над файлом має бути дозволене тільки у разі наявності у користувача відповідних привілеїв. Зазвичай контролюються наступні операції: читання, запис і виконання.

Списки прав доступу. Найбільш загальний підхід до захисту файлів від несанкціонованого використання - зробити доступ залежним від ідентифікатора користувача, тобто зв'язати з кожним файлом або директорією список прав доступу (access control list), де перераховані імена користувачів і типи дозволених для них способів доступу до файлу. Будь-який запит на виконання операції звіряється з таким списком.

У такої техніки є два небажані наслідки:

- конструювання подібного списку може виявитися складним завданням, особливо якщо ми не знаємо заздалегідь користувачів системи;
- запис в директорії повинен мати змінний розмір (включати список

3 ЗАХИСТ ФАЙЛІВ

Поділ користувачів на групи доступу. Для вирішення вищеперелічених проблем створюють класифікації користувачів, наприклад, в ОС Unix усі користувачі розділені на три групи:

- ✔ власник (Owner);
- ✔ група (Group). Набір користувачів, що розділяють файл і що потребують типового способу доступу до нього;
- ✔ інші (Univers).

У рамках такої обмеженої класифікації задаються тільки три поля (по одному для кожної групи) для кожної контрольованої операції. У результаті в Unix операції читання, запису і виконання контролюються за допомогою 9 біт (rwxrwxrwx).

УЗАГАЛЬНЕННЯ ВИВЧЕНОГО МАТЕРІАЛУ:
Усно дайте відповіді на питання.

1 Які типи файлів ви знаєте?
2 Що собою являють звичайні (чи регулярні) файли?
3 Що собою являють бінарні файли?
4 Що таке директорія або каталог?
5 Які атрибути файлів ви знаєте?
6 Що відбувається при операції створення файлу?
7 Що відбувається при операції відкриття файлу?
8 Що відбувається при операції позиціонування файлу?
9 Що відбувається при операції закриття файлу?
10 Що називають повним ім'ям файлу?
11 Яким чином у ОС реалізується захист файлів від несанкціонованого використання?

ДОМАШНЄ ЗАВДАННЯ

В Moodle пройти Тест до теми 4.1.