СИСТЕМА ВВЕДЕННЯ/ВИВЕДЕННЯ. ФАЙЛОВА СИСТЕМА

4.1 Система введення/виведення

Основні принципи програмного забезпечення введення/виведення.

Незалежність від пристроїв. Цей принцип означає можливість написання програм, здатних отримувати доступ до будь-якого пристрою введення/виведення без попередньої вказівки конкретного пристрою.

Однакове іменування. Ім'я файлу або пристрою повинно являти собою просто текстовий рядок або ціле число і ніяк не повинно залежати від фізичного пристрою.

Способи здійснення введення/виведення.

Програмний ввід-вивід. Якщо, наприклад, потрібно роздрукувати на принтері рядок символів, то операційна система спочатку копіює ці дані в ядро. Потім ОС входить в цикл, на кожній ітерації якого друкує на принтері один символ. Після друку кожного символу процесор в циклі опитує готовність пристрою. Така поведінка процесора називається опитуванням, або активним очікуванням. Програмний ввід-вивід легко реалізується, але його недолік полягає в тому, що центральний процесор зайнятий на весь час операціями введення/виведення.

Ввід-вивід, керований перериваннями. Коли виконується системний виклик друку рядка, буфер копіюється в простір ядра, а перший символ рядка копіюється на принтер. Після цього центральний процесор викликає планувальник, який запускає на виконання якій-небудь інший процес. Після того як принтер надрукує символ і буде готовий прийняти наступний, принтер ініціює відповідне переривання. Воно перехоплюється обробником переривання і сигналізує ОС, що вивід символу завершений. Після цього ОС посилає на друк наступний символ, і т.д. Таким чином, під час друку символу на принтері процесор не зайнятий опитуванням готовності пристрою друку, а виконує іншу роботу.

Ввід-вивід з використанням прямого доступу до пам'яті (*DMA, Direct Memory Access*). Очевидний недолік введення/виведення керованого перериваннями, полягає в тому, що переривання відбуваються при друку

кожного символу. Обробка цих переривань займає певний час, тому така схема не є ефективною. Рішення проблеми полягає у використанні DMA. Контролер DMA поставляє принтеру символи по одному, не займаючи при цьому центральний процесор. По суті, цей метод майже не відрізняється від програмного введення/виведення, з тією лише різницею, що всю роботу замість центрального процесора виконує контролер DMA. Найбільший виграш від використання DMA полягає в зменшенні кількості переривань з одного на кожний друкований символ до одного на весь друкований буфер.

4.1.2 Програмні рівні введення/виведення

Програмне забезпечення введення/виведення зазвичай організовано у вигляді чотирьох рівнів, показаних на рисунку 4.1, де у кожного рівня є свої функції і строго визначений інтерфейс з сусідніми рівнями.

Рисунок 4.1 – Програмні рівні введення/виведення

Обробники переривань. Коли відбувається переривання, починає роботу обробник переривань. Після виконання необхідної роботи він може розблокувати драйвер, що запустив його.

Нижче наведені основні дії, що виконуються програмним забезпеченням після того, як відбулося апаратне переривання:

збереження всіх регістрів, що не збереглися апаратурою;

установка контексту для процедури обробки переривань (виконання цієї дії може включати встановлення TLB і таблиці сторінок);

установка покажчика стека для процедури обробки переривань;

4) видача підтвердження контролеру переривань (якщо ж централізованого контролера переривань немає, то дозвіл переривань);

копіювання вмісту регістрів в таблицю процесів;

запуск процедури обробки переривань, яка видобуває інформацію з регістрів контролера пристрою, що ініціював переривання;

вибір процесу, якому слід передати управління;

установка контексту блоку управління пам'яттю для наступного працюючого процесу;

завантаження регістрів для нового процесу;

запуск на виконання нового процесу.

Драйвери пристроїв. У кожного контролера є набір регістрів, які використовуються, для того щоб надавати команди певному пристрою і зчитувати стан цього пристрою. Команди, що видаються пристрою, залежать від конкретного типу цього пристрою, тому для управління кожним пристроєм введення/виведення потрібна спеціальна програма. Ця програма, називається драйвером пристрою і зазвичай пишеться виробником пристрою, причому для кожної операційної системи потрібні спеціальні драйвери. Кожен драйвер зазвичай підтримує тільки один тип пристроїв або ж клас близьких за типами пристроїв.

Щоб отримати доступ до регістрів контролера, драйвер пристрою має бути частиною ядра ОС. Драйвери зазвичай розташовуються під всією іншою частиною операційної системи.

ОС зазвичай класифікує драйвери по декількох категоріях у відповідності типами пристроїв, що ними обслуговуються. До найбільш загальних категорій відносяться *блокові пристрої* (наприклад, диски, що містять блоки даних), до яких можлива незалежна адресація, і *символьні пристрої*, такі як клавіатура і принтери, які формують або приймають потік символів.

ОС визначені стандартні інтерфейси, підтримувані всіма драйверами. Ці інтерфейси включають в себе набір процедур, які можуть викликатися із операційної системи для звернення до драйверу.

Драйвер пристрою повинен виконувати кілька функцій:

- обробку абстрактних запитів читання і запису, що надходять від незалежного від цього пристрою програмного забезпечення, розташованого над ним;
- перевірку вхідних параметрів;

перетворення абстрактних параметрів в конкретні (наприклад, дисковий драйвер може перетворювати лінійний номер блоку в номери головки, доріжки і сектора);

перевірку готовності і незайнятості пристрою;

власне управління пристроєм – видачу йому необхідної серії команд;

у багатьох випадках драйвер пристрою повинен чекати, поки контролер пристрою не виконає для нього певну роботу, тому він блокується до тих пір, поки переривання від пристрою його не розблокує; в інших випадках операція завершується без затримок, і драйверу не потрібно блокуватися;

по завершенні виконання операції драйвер повинен перевірити, чи виконана ця операція без помилок. Якщо все в порядку, то драйвер передає дані незалежного від пристроїв програмного забезпечення.

Незалежне від пристроїв програмне забезпечення введення/виведення.

Незалежне від пристроїв програмне забезпечення введення/виведення забезпечує наступні функції:

однаковий інтерфейс для драйверів пристроїв;

буферизацію;

повідомлення про помилки;

захоплення і звільнення виділених пристроїв;

розмір блоку, не залежний від пристрою.

Програмне забезпечення введення/виведення простору користувача.

Програмне забезпечення введення/виведення простору користувача складається бібліотек, приєднаних до програм користувача. Системні виклики вводу зазвичай складаються з бібліотечних процедур. Наприклад, якщо програма на мові С містить виклик *write*, то бібліотечна процедура *write* буде скомпонована з програмою і, відповідно, буде міститися в виконуваному файлі.

Хоча більшість таких процедур в основному здійснюють звернення до системного виклику з відповідними параметрами, але є й ряд процедур введення/виведення, що виконують певну роботу. Зокрема, бібліотечними процедурами виконуються операції форматного вводу і виводу (приклад - функція *printf* мови *C*).

4.2 Завдання файлової системи. Поняття файлу

4.2.1 Фізична організація файлової системи

Файл (file) – це поіменована область зовнішньої пам'яті, в яку можна записувати і з якої можна зчитувати дані (*суміжна область логічного адресного простору*). Основні цілі використання файлу:

довгострокове і надійне зберігання інформації;

спільне використання інформації.

Кожен файл має свій тип, що визначає, яка інформація зберігається в файлі. Основні типи файлів: програма (код) або дані. Дані підрозділяються на числові, символьні (текстові) і двійкові (довільна інформація).

Структура файлу

різних системах прийняті різні точки зору на структуру файлів. У ряді систем структура файлу прив'язувалася до типу пристрою, на якому він знаходиться. У деяких інших системах структура файлу була штучно ускладнена. Однак найбільш просту і уніфіковану точку зору з них запропонували автори системи UNIX: файл – це послідовність слів або байтів. Здавалося б, це очевидно, але перевага даного підходу до файлів в тому, що базове уявлення файлу і базові операції над ним (read, write) не залежать від типу пристрою. Можна сказати, що файли у своєму розвитку пройшли шлях, аналогічний розвитку архітектури комп'ютерів – спочатку в бік значних ускладнень, потім – спрощення та уніфікації.

Файли можна умовно підрозділяти на файли *простої та складної структури* (хоча точка зору на структуру файлу залежить від тієї програми, яка його обробляє).

Файли простої структури складаються з послідовності записів (*records*) – елементарних одиниць, в термінах яких виконуються операції обміну з файлом. Записи можуть бути: рядками, якщо це текстовий файл; двійковими даними фіксованої довжини; двійковими даними змінної довжини.

Файли складної структури можуть бути самого різного виду, наприклад: відформатованим документом Microsoft Office (такий файл, окрім власне тексту, містить керуючі символи перемикання шрифтів, кольорів і т.д.); завантажувальним модулем реального або віртуального двійкового коду, наприклад, portable executable (PE)-файлом для платформи NET; class-файлом для платформи Java; подібні файли складаються з декількох секцій, містять внутрішні посилання і таблиці і т.д.

Складна структура файлу може бути змодельована записами шляхом додавання відповідних керуючих символів.

Файли інтерпретуються операційною системою або програмами їх обробки.

Файлова система (ФС) – це частина операційної системи, що включає в себе:

сукупність всіх файлів на диску;

набори структур даних, використовуваних для управління файлами;

комплекс системних програмних засобів, що реалізують операції над файлами.

ФС грає роль «проміжного шару», що усуває для користувача/програміста всі складнощі фізичної організації довготривалого сховища даних і створює для програм більш просту логічну модель цього сховища, а також надає їм набір зручних у використанні команд для маніпулювання файлами.

Завдання, розв'язувані файловою системою, залежать від способу організації обчислювального процесу в цілому.

Основні функції ФС в однокористувацьких і однопрограмних ОС (типу MS-DOS):

іменування файлів;

програмний інтерфейс для додатків;

відображення логічної моделі файлової системи на фізичну організацію сховища даних;

стійкість файлової системи до збоїв харчування, помилок апаратних і програмних засобів.

однокористувацьких мультипрограмних ОС також мають бути передбачені засоби для блокування файлу і його частин, запобігання гонок, виключення тупиків, узгодження копій і т. п.

багатокористувацьких ОС з'являється ще одна задача – захист файлів одного користувача від несанкціонованого доступу іншого користувача.

Нарешті, ще більш складними стають функції ФС, яка працює у складі мережевої ОС.

Принципи розміщення файлів, каталогів і системної інформації на реальному пристрої описуються фізичною організацією файлової системи. Різні ФС мають різну фізичну організацію.

4.2.2 Будова диску

Основним типом пристрою для зберігання файлів є дискові накопичувачі. Ці пристрої призначені для зчитування і запису даних на жорсткі і гнучкі магнітні диски.

Жорсткий диск складається з однієї або декількох скляних або металевих пластин, кожна з яких покрита з одного або з двох сторін магнітним матеріалом. На кожній стороні кожної пластини шляхом відповідного намагнічування поверхні магнітного шару розмічені тонкі концентричні кільця – доріжки (tracks), на яких зберігаються дані (рисунок 4. 2).

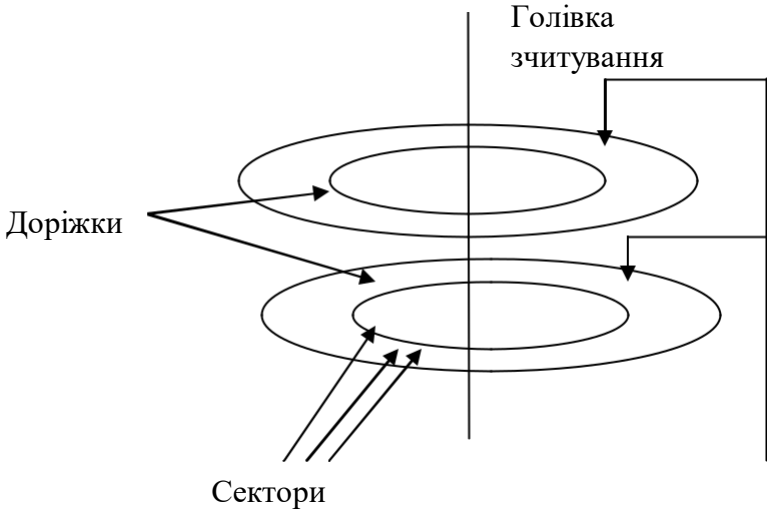

Рисунок 4.2 – Схема устрою накопичуваних жорстких магнітних дисків

Коли диск обертається, магнітна головка накопичувача зчитує двійкові дані з обраної магнітної доріжки або записує їх на магнітну доріжку. Для цього голівка може позиціонуватися над заданою доріжкою. Зазвичай всі головки закріплені на єдиному механізмі, що переміщується і рухаються синхронно. Тому, коли голівка фіксується на заданій доріжці однієї поверхні, всі інші голівки зупиняються над доріжками інших пластин з такими ж номерами. Сукупність доріжок одного і того ж радіусу на всіх поверхнях всіх пластин дискового пакета називається *циліндром*.

Кожна доріжка розбивається на фрагменти, звані секторами, або блоками, так що всі доріжки мають рівну кількість секторів, в які можна максимально записати одну і ту ж кількість байтів. Сектор має фіксований для конкретної ОС розмір, що виражається ступенем числа 2; найчастіше розмір сектора складає 512 байт. Враховуючи ж, що доріжки різного радіусу мають однакову

кількість секторів, щільність запису стає тим більшою, чим ближче розташовується доріжка до центру диска.

Сектор – це найменша адресуєма одиниця обміну даними дискового пристрою з оперативною пам'яттю. Щоб контролер міг знайти на диску потрібний сектор, необхідно задати йому всі складові адреси цього сектора: номер циліндра, номер поверхні, номер доріжки та власне номер сектора на цій доріжці.

ОС при роботі з диском використовує, як правило, власну одиницю дискового простору, звану *кластером* (в UNIX кластер називається блоком). При створенні файлу місце на диску йому виділяється кластерами. Доріжки і сектори створюються в результаті виконання процедури фізичного, або *низькорівневого форматування*, що передує використанню диска. Для визначення меж блоків на диск записується ідентифікаційна інформація. Низькорівневий формат диска не залежить від типу ОС, яка цей диск буде використовувати. Розмітку диска під конкретний тип файлової системи виконують процедури високорівневого, або *логічного форматування*. При високорівневому форматуванні визначається розмір кластера, а на диск записується інформація, необхідна для роботи файлової системи, у тому числі відомості про доступний і невикористаний простір, про межі областей, відведених під файли і каталоги, про пошкоджені області. Крім того, на диск записується *завантажувач операційної системи* – невелика програма, яка починає процес ініціалізації операційної системи після включення живлення або перезавантаження комп'ютера.

Перш ніж форматувати диск під певну файлову систему, його можна розбити на розділи. *Розділ* – це безперервна частина фізичного диска, яку операційна система являє користувачу як окремий логічний пристрій (часто використовуються також назви «логічний диск» і «логічний розділ»). Логічний пристрій функціонує так, ніби то це окремий фізичний диск. ОС різного типу зазвичай використовують єдине для всіх них уявлення про розділи, але створюють на його основі логічні пристрої, специфічні для кожного типу ОС. Так само як файлова система, з якою працює одна ОС, в загальному випадку не може інтерпретуватися ОС іншого типу, одні й ті ж логічні пристрої часто не можуть бути використані операційними системами різного типу. При цьому на кожному логічному пристрої може створюватися лише одна файлова система, на різних логічних пристроях одного і того ж фізичного диска можуть розташовуватися файлові системи різного типу.

Файлова система FAT.

Абревіатура FAT (file allocation table) розшифровується як «таблиця розміщення файлів». Цей термін відноситься до лінійної табличної структури

відомостями про файли — іменами файлів, їхніми атрибутами й іншими даними, що визначають місцезнаходження файлів (чи їхніх фрагментів) у середовищі FAT. Елемент FAT *визначає* фактичну область диска, у якій зберігається початок фізичного файлу.

файловій системі FAT логічний дисковий простір будь-якого логічного диска поділяється на дві області (рисунок 4.3): *системну область* і *область даних.*

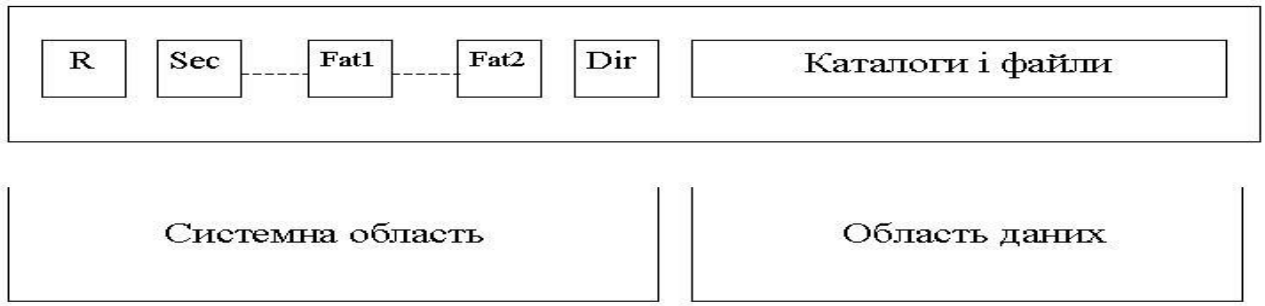

Рисунок 4.3 – Структура логічного диску

Системна область логічного диску створюється і ініціалізується при форматуванні, а потім оновлюється при маніпулюванні файловою структурою. Область даних логічного диску містить файли і каталоги, підпорядковані кореневому. Вона, на відміну від системної області, доступна через користувацький інтерфейс DOS. Системна область складається з наступних компонентів, розміщених у логічному адресному просторі підряд:

завантажуючого запису (*boot record, BR*);

зарезервованих секторів (*reserved sector, ResSecs*);

таблиці розміщення файлів (*file allocation table, FAT*);

кореневого каталогу (*root directory, Rdir*).

Також є файлові системи VFAT (віртуальна FAT) і FAT32, NTFS.

назву файлової системи ***NTFS*** входять слова «*New Technology*», тобто «нова технологія». Дійсно, NTFS містить ряд значних удосконалень і змін, що істотно відрізняють її від інших файлових систем. З погляду користувачів,

файли як і раніше зберігаються в каталогах (часто званих «папками» чи *фолдерами* в середовищі Windows). У NTFS на відміну від FAT робота на дисках великого обсягу відбувається набагато ефективніше; є засоби для обмеження в доступі до файлів і каталогів, введені механізми, що істотно підвищують надійність файлової системи, зняті багато обмежень на максимальну кількість дискових секторів і/чи кластерів.

Основні можливості файлової системи NTFS.

При проектуванні системи NTFS особлива увага була приділена наступним характеристикам:

Надійність. Високопродуктивні комп'ютери і системи спільного користування (сервери) повинні мати підвищену надійність, що є ключовим елементом структури і поводження NTFS. Одним зі способів збільшення надійності є введення механізму транзакцій, при якому здійснюється **журналювання** файлових операцій;

Розширена функціональність. NTFS проектувалася з врахуванням можливого розширення. У ній були втілені багато додаткових можливостей — вдосконалена відмовостійкість, емуляція інших файлових систем, могутня модель безпеки, рівнобіжна обробка потоків даних і створення файлових атрибутів, обумовлених користувачем;

Підтримка POSIX. Оскільки уряд США вимагав, щоб усі закуповувані ним системи хоча б у мінімальному ступені відповідали стандарту POSIX, така можливість була передбачена і в NTFS. До числа базових засобів файлової системи POSIX відноситься необов'язкове використання імен файлів з урахуванням регістра, збереження часу останнього звертання до файлу і механізм так званих «жорстких посилань» — альтернативних імен, що дозволяють посилатися на той самий файл по двох і більше іменах;

Гнучкість. Модель розподілу дискового простору в NTFS відрізняється надзвичайною гнучкістю. Розмір кластера може змінюватися від 512 байт до 64 Кбайт; він являє собою число, кратне внутрішньому кванту розподілу дискового простору. NTFS також підтримує довгі імена файлів, набір символів Unicode і альтернативні імена формату 8.3 для сумісності з
FAT.

4.2.3 Фізична організація і адресація файлу

Фізична організація файлу – це спосіб розміщення файлу на диску. Основними критеріями ефективності фізичної організації файлів є:

швидкість доступу до даних;

об'єм адресної інформації файлу;

ступінь фрагментації дискового простору;

максимально можливий розмір файлу.

Безперервне розміщення - це найпростіший варіант фізичної організації, при якому файлу надається послідовність кластерів диска, що утворюють безперервну ділянку дискової пам'яті. Цей метод має свої переваги: 1) висока швидкість доступу так як витрати на пошук і зчитування кластерів файлу мінімальні; 2) мінімальний об'єм адресної інформації; 3) необмежений максимально можливий розмір файлу.

Однак реалізувати цю схему складно через те, що розмір файлу, як правило, заздалегідь невідомий. Ще більш серйозною проблемою є фрагментація: через деякий час після створення такої файлової системи в результаті виконання численних операцій створення та видалення файлів простір диска неминуче перетворюється на «клаптикову ковдру», що включає в себе велику кількість вільних областей невеликого розміру.

Розміщення файлу в вигляді зв'язаного списку кластерів, коли на початку кожного кластера міститься покажчик на наступний кластер.

На відміну від попереднього способу, кожен кластер тут може бути приєднаний до ланцюжку кластерів якого-небудь файлу, а отже, фрагментація на рівні кластерів відсутня. Файл може змінювати свій розмір під час свого існування, нарощуючи число кластерів. Але маючи певні переваги цей метод також має свої недоліки: 1) складність реалізації доступу до довільного місця файлу; 2) кількість даних файлу, що містяться в одному кластері, не дорівнює ступеню числа 2 (так як одне машинне слово вже витрачено на номер наступного кластера), а багато програм читають дані кластерами, розмір яких дорівнює ступеню числа 2.

Використання зв'язаного списку індексів. Це – модифікація попереднього способу. Тут файлу також виділяється пам'ять у вигляді зв'язаного списку кластерів. При цьому номер першого кластера

запам'ятовується в записі каталогу, де зберігаються характеристики даного файлу, а інша адресна інформація відділена від кластерів файлу. З кожним кластером диска зв'язується особливий елемент – індекс, де всі індекси розташовуються в окремій області диска: в ОС MS-DOS це таблиця FAT, що займає один окремий кластер. Коли пам'ять вільна, всі індекси мають нульове значення. Якщо ж деякий кластер N призначений деякому файлу, то індекс цього кластера стає рівним або номеру M наступного кластера даного файлу, або приймає спеціальне значення, що є ознакою, що даний кластер є для файлу останнім. Індекс ж попереднього кластера файлу приймає значення N, вказуючи на новопризначений кластер.

При такій фізичній організації зберігаються всі переваги попереднього способу – мінімальність адресної інформації, відсутність фрагментації і відсутність проблем при зміні розміру файлу, але з'являються і додаткові переваги:

висока швидкість доступу (для доступу до довільного кластеру файлу не потрібно послідовно зчитувати його кластери – досить прочитати тільки сектори диска, які містять таблицю індексів, відрахувати потрібну кількість кластерів файлу по ланцюжку і визначити номер потрібного кластера);

дані файлу заповнюють кластер цілком, а значить, мають об'єм, рівний ступеню числа 2.

Недолік же цього методу тільки один: вся таблиця FAT повинна постійно знаходитися в ОП чи кожен раз цілком завантажуватися в ОП.

Перелік номерів кластерів, зайнятих файлом (метод і-вузлів). Такий перелік і служить адресою файлу. З кожним файлом зв'язана структура даних «*і-вузол*», що містить також атрибути файлу.

Недолік цього методу: довжина адреси залежить від розміру файлу, і для великого файлу може скласти значну величину. Але і переваги цього методу в тому, що: 1) висока швидкість доступу до довільного кластеру файлу, так як тут застосовується пряма адресація, яка виключає перегляд ланцюжка покажчиків при пошуку адреси довільного кластера файлу; 2) відсутня фрагментація на рівні кластерів; 3) кожен вузол повинен знаходитися в ОП тільки тоді, коли відповідний йому файл відкритий.

Модифікований підхід до переліку номерів кластерів. Даний метод використовується в ОС UNIX в традиційних файлових системах s5 і ufs, в ОС

Linux в найбільш поширених файлових системах ext2fs[1] та ext3fs (остання відрізняється від першої тільки наявністю журналювання). Для скорочення обсягу адресної інформації прямий спосіб адресації поєднується з непрямим.

стандартній для UNIX ФС ufs використовується наступна схема адресації кластерів файлу (рис. 4.4). Для зберігання адреси файлу виділено 15 полів, кожне з яких складається з 4 байтів. Якщо розмір файлу менше або дорівнює 12 кластерам, то номери цих кластерів безпосередньо перераховуються в перших дванадцяти полях адреси.

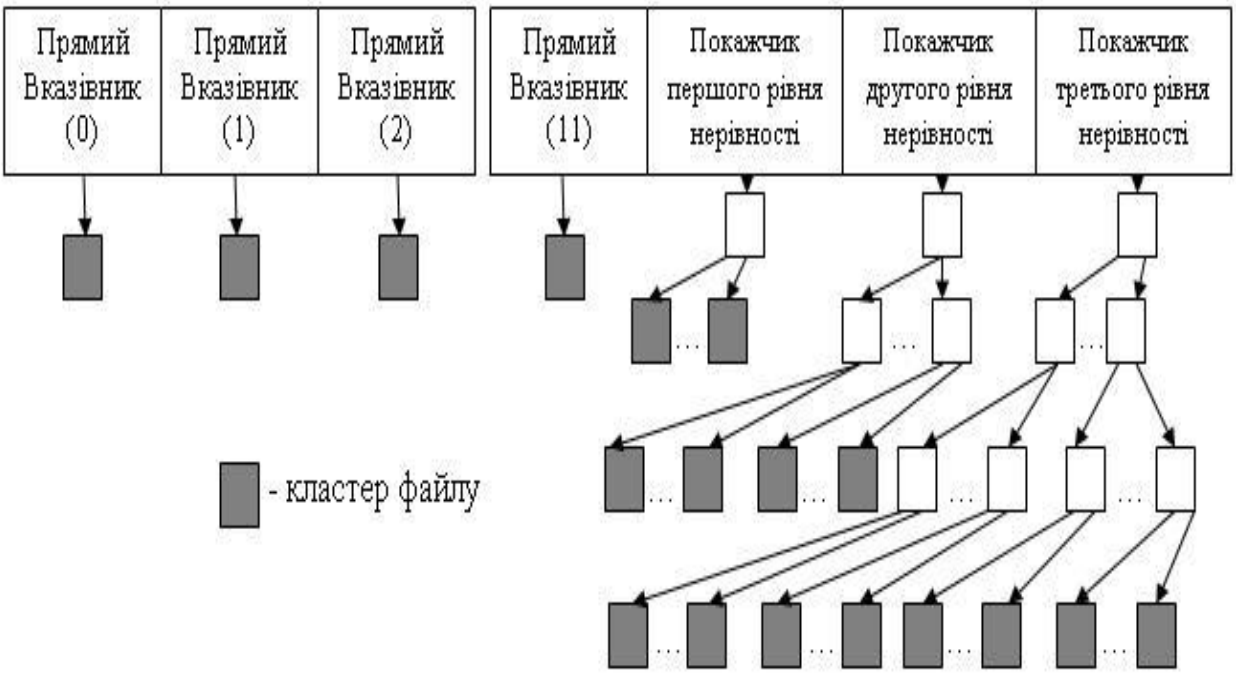

Рисунок 4.4 – Схема адресації кластерів файлу у файловій системі ufs

Якщо розмір файлу перевищує 12 кластерів, то наступне, 13-е поле містить адресу кластера, в якому можуть бути розташовані номери наступних кластерів файлу. Якщо розмір файлу перевищує 12 + 2048 = 2060 кластерів, то використовується 14-е поле, в якому знаходиться номер кластера, що містить 2048 номерів кластерів, кожен з яких зберігає 2048 номерів кластерів даних файлу, – тобто тут застосовується вже подвійна непряма адресація. І нарешті,

---

ext2fs – стандартна дискова файлова система для Linux, в якій дисковий простір ділять не на групи циліндрів, а просто на групи блоків (block groups), не прив'язані до геометрії диска. Такі групи за структурою схожі на групи циліндрів: кожна група блоків теж є зменшеною копією файлової системи із суперблоком, таблицею індексних дескрипторів тощо.

якщо файл включає більше 12 + 2048 +2048 2048 = 4196364 кластерів, то використовується останнє, 15-е поле для потрійний непрямої адресації.

Метод переліку адрес кластерів файлу задіяний і в файловій системі NTFS, використовуваної в ОС сімейства Microsoft Windows NT. Тут він доповнений досить природним прийомом, що скорочує об'єм адресної інформації: адресуються не кластери файлу, а безперервні області, що складаються з суміжних кластерів диска. Кожна така область, що називається групою (в *ext2fs*) або екстентом (*extent*), описується за допомогою двох чисел – початкового номера кластера та кількості кластерів у відрізку. Так як для скорочення часу операції обміну ОС намагається розмістити файл у послідовних кластерах диска, то в більшості випадків кількість послідовних областей файлу буде менше кількості кластерів файлу, так що обсяг службової адресної інформації в NTFS скорочується порівняно зі схемою адресації файлових систем ufs/s5.

4.3 Логічна організація файлової системи

ОС повинна надавати користувачеві зручності при роботі з даними на дисках. Для цього ОС підміняє фізичну структуру даних зручною для користувача логічною моделлю. При цьому логічна модель файлової системи матеріалізується у вигляді дерева каталогів, в символьних складових імен файлів і в командах роботи з файлами.

4.3.1 Типи файлів

Звичайні файли (або просто файли), містять інформацію довільного характеру, яку заносить в них користувач або яка утворюється в результаті роботи системних і користувальних програм.

Каталоги – це особливий тип файлів, які містять системну довідкову інформацію про набір файлів, згрупованих користувачами за якоюсь ознакою.

каталог можуть входити файли будь-яких типів, в тому числі інші каталоги, за рахунок чого утворюється деревоподібна структура, зручна для пошуку. Каталоги встановлюють відповідність між іменами файлів і їх характеристиками, використовуваними файловою системою для управління файлами.

Спеціальні файли – це файли, асоційовані з пристроями введення/виведення і використовувані для уніфікації механізму доступу до файлів і зовнішніх пристроїв. Спеціальні файли дозволяють користувачеві виконувати операції введення/виведення за допомогою звичайних команд запису у файл або читання з файлу. Сучасні ФС підтримують і інші типи файлів (символьні зв'язки, іменовані конвеєри, відображувані в пам'ять файли та ін.).

4.3.2 Ієрархічна структура файлової системи

Більшість файлових систем має ієрархічну структуру, в якій рівні створюються за рахунок того, що каталог більш низького рівня може входити в каталог більш високого рівня.

Граф, що описує ієрархію каталогів, може представляти собою дерево або мережу. Каталоги утворюють *дерево*, якщо файлу дозволено входити тільки в один каталог, або *мережу*, якщо файл може входити відразу в кілька каталогів. Наприклад, в MS-DOS і Windows каталог утворюють деревовидну структуру, а UNIX – мережеву. У деревоподібній структурі кожен файл є листом. Каталог самого верхнього рівня називається кореневим каталогом, або коренем (*root*).

При такій організації користувач звільнений від запам'ятовування імен всіх файлів; йому досить приблизно уявляти, до якої групи може бути віднесений той або інший файл, щоб шляхом послідовного перегляду каталогів знайти його. Ієрархічна структура зручна і для багатокористувацької роботи: кожен користувач зі своїми файлами локалізується в своєму каталозі або піддереві каталогів, але разом з тим, всі файли в системі логічно пов'язані.

Окремим випадком ієрархічної структури є однорівнева організація, коли всі файли входять в один каталог.

Імена файлів.

Всі типи файлів мають символьні імена. В ієрархічно організованих файлових системах зазвичай використовуються три типи імен файлів: прості і складні, де останні можуть бути абсолютними і відносними.

Просте, або коротке, символьне ім'я ідентифікує файл в межах одного каталогу. Прості імена привласнюють файлам користувачі і програмісти. В ієрархічних файлових системах різним файлам дозволено мати однакові прості символьні імена за умови, що вони належать різним каталогам, – тобто тут працює схема: «багато файлів – одне просте ім'я».

Для однозначної ідентифікації файлу в таких системах використовується так зване *повне ім'я*, яке являє собою ланцюжок простих символьних імен всіх каталогів, через які проходить шлях від кореня до даного файлу, плюс просте ім'я самого цього файлу.

деревоподібної файлової системи між файлом і його повним ім'ям мається взаємно однозначна відповідність: «один файл – одне повне ім'я».

файлових же системах, що мають мережеву структуру, файл може входити в кілька каталогів, а значить мати кілька повних імен; тут справедлива відповідність: «один файл – багато повних імен». Проте в обох цих випадках файл однозначно ідентифікується повним ім'ям.

Файл може бути також ідентифікований відносним ім'ям, яке визначається через поняття «поточний каталог». При цьому файл ідентифікується ланцюжком імен каталогів, через які проходить маршрут від поточного каталогу до даного файлу.

Хоча повне ім'я однозначно визначає файл, операційній системі простіше працювати з файлом, якщо між файлами і їх іменами мається взаємно однозначна відповідність. З цією метою ОС також привласнює файлу унікальне ім'я, так що справедливе співвідношення: «один файл – одне унікальне ім'я». Унікальне ім'я існує поряд з одним або декількома символьними іменами, присвоюється файлу користувачами або додатками, представляє собою числовий ідентифікатор і призначене тільки для ОС. Приклад – номер індексного дескриптора в системі UNIX.

Монтування

В обчислювальній системі може бути декілька дискових пристроїв. При цьому одна фізична пристрій за допомогою засобів ОС може бути представлене як кілька логічних пристроїв (зокрема, шляхом розбиття дискового простору на розділи).

Як організувати зберігання файлів в системі, що має кілька пристроїв зовнішньої пам'яті? Для цього можливо два рішення:

1. На кожному з пристроїв розміщується автономна ФС, тобто файли, що знаходяться на цьому пристрої, описуються деревом каталогів, ніяк не пов'язаним з деревами каталогів на інших пристроях. У такому випадку для однозначної ідентифікації файлу користувач, поряд з складовим символьним ім'ям файлу, повинен вказувати *ідентифікатор логічного пристрою*. Приклад – ОС MS-DOS.

Користувачеві надається можливість об'єднувати файлові системи, що знаходяться на різних пристроях, в єдину ФС, що описується єдиним деревом каталогів. Така операція називається *монтуванням*.

4.3.3 Атрибути файлів

Файл – це не тільки дані, але й атрибути-інформація, що описує властивості файлу. Приклади можливих атрибутів файлу:

тип файлу (звичайний файл, каталог, спеціальний файл і т. п.);
власник файлу;
творець файлу;
пароль для доступу до файлу;
інформація про дозволені операції доступу до файлу;
час створення, останнього доступу і останньої зміни;
поточний розмір файлу;
максимальний розмір файлу;
ознака «тільки для читання»;
ознака «прихований файл»;
ознака «системний файл»;
ознака «архівний файл»;
ознака «двійковий/символьний»;
ознака «тимчасовий» (видаляється після завершення процесу);
ознака блокування;
довжина запису у файлі;
покажчик на ключове поле в записі;
довжина ключа.

Набір атрибутів файлу визначається специфікою файлової системи. Користувач може одержувати доступ до атрибутів, використовуючи засоби, що надані для цих цілей файловою системою. Зазвичай дозволяється читати значення будь-яких атрибутів, а змінювати тільки деякі з них.