## 16.2 ЛОГІЧНА ОРГАНІЗАЦІЯ ФАЙЛУ

У загальному випадку дані, що містяться у файлі, мають деяку логічну структуру. Ця структура є базою при розробці програми, призначеної для обробки цих даних. Наприклад, щоб текст міг бути правильно виведений на екран, програма повинна мати можливість виділити окремі слова, рядки, абзаци тощо. Підтримка структури даних може бути або цілком покладена на додаток, або в тому або іншому ступені цю роботу може взяти на себе файлова система.

У першому випадку, коли всі дії, пов'язані зі структуризацією і інтерпретацією утримуваного файлу, цілком належать до ведення додатка, файл представляється у ФС неструктурованою ***послідовністю даних.*** Додаток формулює запити до файлової системи на введення-виведення, використовуючи загальні для всіх додатків системні засоби, наприклад, вказуючи зміщення від початку файлу і кількість байт, які необхідно зчитати або записати. Потік байт, що поступив до додатку, інтерпретується відповідно до закладеної в програмі логіки.

Модель файлу, відповідно до якої вміст файлу представляється неструктурованою послідовністю (потоком) байт, стала популярною разом з ОС UNIX, а тепер вона широко використовується в більшості сучасних ОС, у тому числі в MS-DOS, Windows NT/2000, NetWare. Неструктурована модель файлу дозволяє легко організувати розподіл файлу між декількома додатками: різні додатки можуть по-своєму структурувати і інтерпретувати дані.

Інша модель файлу, яка застосовувалася в ОС OS/360, DEC RSX і VMS, нині використовується досить рідко – це ***структурований файл***. У цьому випадку підтримка структури файлу доручається файловій системі. Файлова система бачить файл як упорядковану послідовність логічних записів. Додаток може звертатися до ФС із запитами на введення-виведення на рівні записів.

Логічний запис є найменшим елементом даних, яким може оперувати програміст при організації обміну із зовнішнім пристроєм. Навіть якщо фізичний обмін з пристроєм здійснюється великими одиницями, операційна система повинна забезпечувати програмістові доступ до окремого логічного запису.

Файлова система може використовувати два способи доступу до логічних записів: читати або записувати логічні записи послідовно (***послідовний доступ***) або позиціонувати файл на запис з указаним номером (***прямий доступ***).

Очевидно, що ОС не може підтримувати всі можливі способи структуризації даних у файлі, тому в тих ОС, в яких існує підтримка логічної структуризації файлів, вона існує для невеликого числа широко поширених схем логічної організації файлу (рис. 16.2).

До таких способів структуризації належить представлення даних у вигляді записів, довжина яких фіксована в межах файлу (рис. 16.2, *а).* У такому разі доступ до *n*-го запису здійснюється або шляхом послідовного читання (*n-1*) попередніх записів, або прямо за адресою, обчисленою за її порядковим номером. Наприклад, якщо *L* – довжина запису, то початкова адреса *n*-го запису рівна *L*n*. Відмітимо, що при такій логічній організації розмір запису фіксований у межах файлу, а записи в різних файлах, що належать одній і тій же файловій системі, можуть мати різний розмір.

Інший спосіб структуризації полягає в представленні даних у вигляді послідовності записів, розмір яких змінюється в межах одного файлу. Якщо розташувати значення довжин записів так, як це показано на рис. 16.2, *б*, то для пошуку потрібного запису система повинна послідовно прочитати усі попередні записи. Обчислити адресу потрібного запису за її номером при такій логічній організації файлу неможливо, а, отже, не може бути застосований ефективніший метод прямого доступу.

Файли, доступ до записів яких здійснюється послідовно, за номерами позицій, називаються ***неіндексованими***, або ***послідовними***.

Іншим типом файлів є ***індексовані файли***, вони допускають швидший прямий доступ до окремого логічного запису. В індексованому файлі (рис. 16.2, *в)* записи мають одне або більше ключових (індексних) полів і можуть адресуватися шляхом вказівки значень цих полів.

Для швидкого пошуку даних в індексованому файлі передбачається спеціальна ***індексна таблиця***, в якій значенням ключових полів ставиться у відповідність адреса зовнішньої пам'яті. Ця адреса може вказувати або безпосередньо на шуканий запис,

або на деяку область зовнішньої пам'яті, займану декількома записами, до числа яких входить шуканий запис. В останньому випадку говорять, що файл має ***індексно-послідовну*** організацію, оскільки пошук включає два етапи: прямий доступ за індексом до вказаної області диска, а потім послідовний перегляд записів у вказаній області (рис. 16.3). Ведення індексних таблиць бере на себе файлова система.


**Рисунок 16.2** – Способи логічної організації файлів

Для пошуку потрібного запису за його ключем спочатку виконується пошук в індексному файлі. Після того як в ньому знайдено найбільше значення ключа, яке не перевищує шукане, триває пошук в головному файлі. Наприклад, нехай послідовний файл (головний) містить 1 млн записів. Для пошуку певного ключового значення потрібні в середньому 0,5 млн операцій доступу до записів. Якщо створити індексний файл, що містить 1000 елементів, то знадобиться в середньому 500 операцій доступу до індексного файлу, після чого ще потрібні в середньому 500 операцій доступу до головного файлу.

У результаті середня довжина пошуку зменшується з 0,5 млн до 1 тис. Ще кращого результату можна досягти використовуючи багаторівневу індексацію. При

цьому нижній рівень індексного файлу розглядається як послідовний файл, для якого створюється індексний файл верхнього рівня.

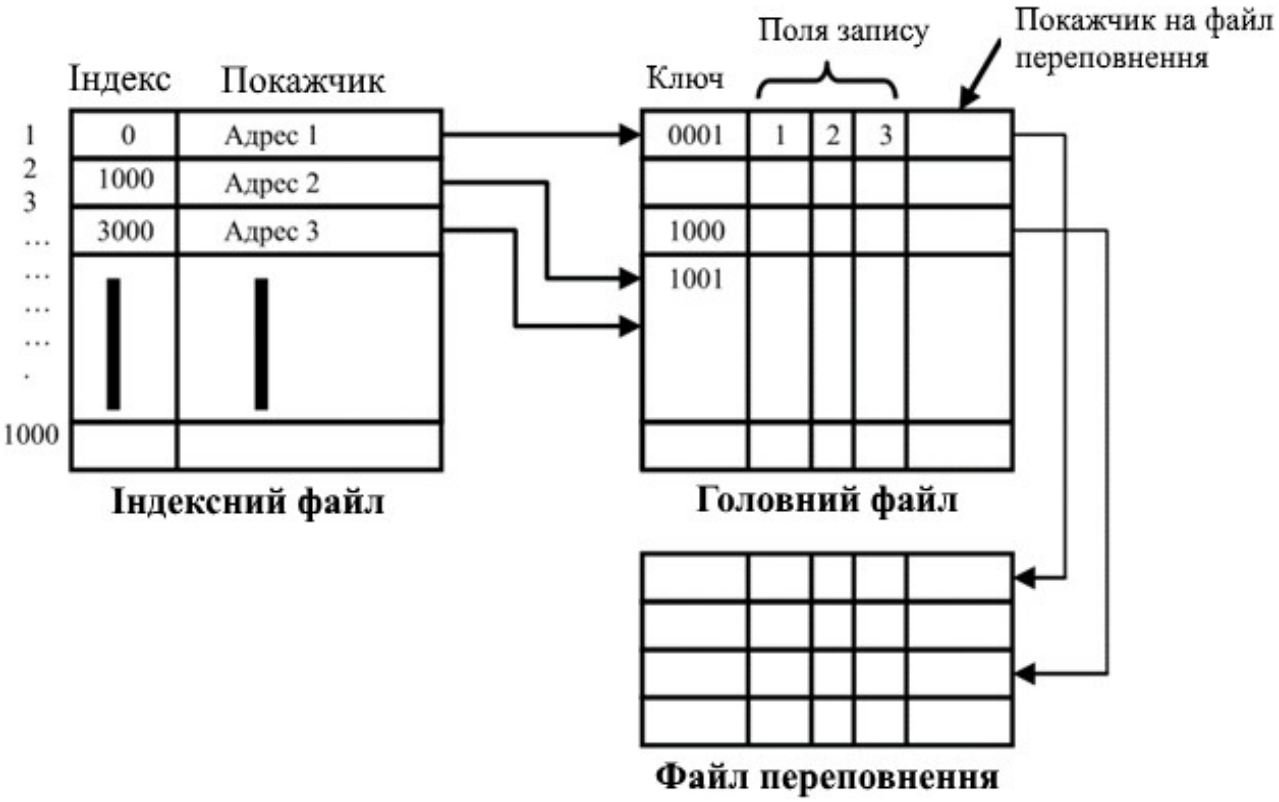


**Рисунок 16.3 –** Індексно-послідовний файл

Доповнення до файлу обробляються таким чином. У кожному записі головного файлу міститься додаткове поле, яке невидиме для додатка і є покажчиком на файл переповнення. Якщо у файлі робиться вставка нового запису, вона додається у файл переповнювання. Запис в головному файлі, безпосередньо передуючий новому запису в логічній послідовності, оновлюється і вказує на новий запис у файлі переповнювання.

Час від часу виконується злиття індексно-послідовного файлу з файлом переповнювання.