15.3 ОРГАНІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВВЕДЕННЯ-ВИВЕДЕННЯ

Основна ідея організації програмного забезпечення введення-виведення полягає в розбитті його на декілька рівнів (шарів), причому нижні рівні забезпечують екранування особливостей апаратури від верхніх, а ті, у свою чергу, забезпечують зручний інтерфейс для користувачів.

Ключовим принципом є незалежність від пристроїв. Вид програми не повинен залежати від того, читає вона дані з гнучкого чи з жорсткого диска. Дуже близькій до ідеї незалежності від пристроїв є ідея однакового іменування, тобто для іменування пристроїв мають бути прийняті єдині правила.

Іншим важливим питанням для програмного забезпечення введення- виведення є обробка помилок. Помилки слід обробляти як можна ближче до апаратури. Якщо контролер виявляє помилку читання, то він повинен спробувати її скоректувати. Якщо ж це йому не вдається, то виправленням помилок повинен зайнятися драйвер пристрою. Багато помилок можуть зникати при повторних спробах виконання операцій введення-виведення. наприклад, помилки, викликані наявністю порошинок на головках читання або на диску. І тільки якщо нижній рівень не може впоратися з помилкою, він повідомляє про помилку верхній рівень.

Ще одне ключове питання – це використання ***блокуючих (синхронних)*** і ***неблокуючих (асинхронних) передач***. Більшість операцій фізичного введення-виведення виконуються асинхронно – процесор починає передачу і переходить на іншу роботу, поки не настає переривання. Програми користувача набагато легше писати, якщо операції введення-виведення блокуючі . ОС виконує операції введення-виведення асинхронно, але представляє їх для програм користувача в синхронній формі. Остання проблема полягає в тому, що одні пристрої є такими, що ***розділяються***, а інші **–** ***виділеними***. Диски – це пристрої, які розділяються, оскільки одночасний доступ декількох користувачів до диска не є проблемою. Принтери – це виділені пристрої, тому що не можна змішувати рядки, які друкуються різними користувачами. Наявність виділених пристроїв створює для операційної системи деякі проблеми.

Для вирішення поставлених проблем доцільно розділити програмне забезпечення введення-виведення на чотири шари (рис. 15.4):

1) обробка переривань;

2) драйвери пристроїв;

3) незалежний від пристроїв шар операційної системи;

4) призначений для користувача шар програмного забезпечення.

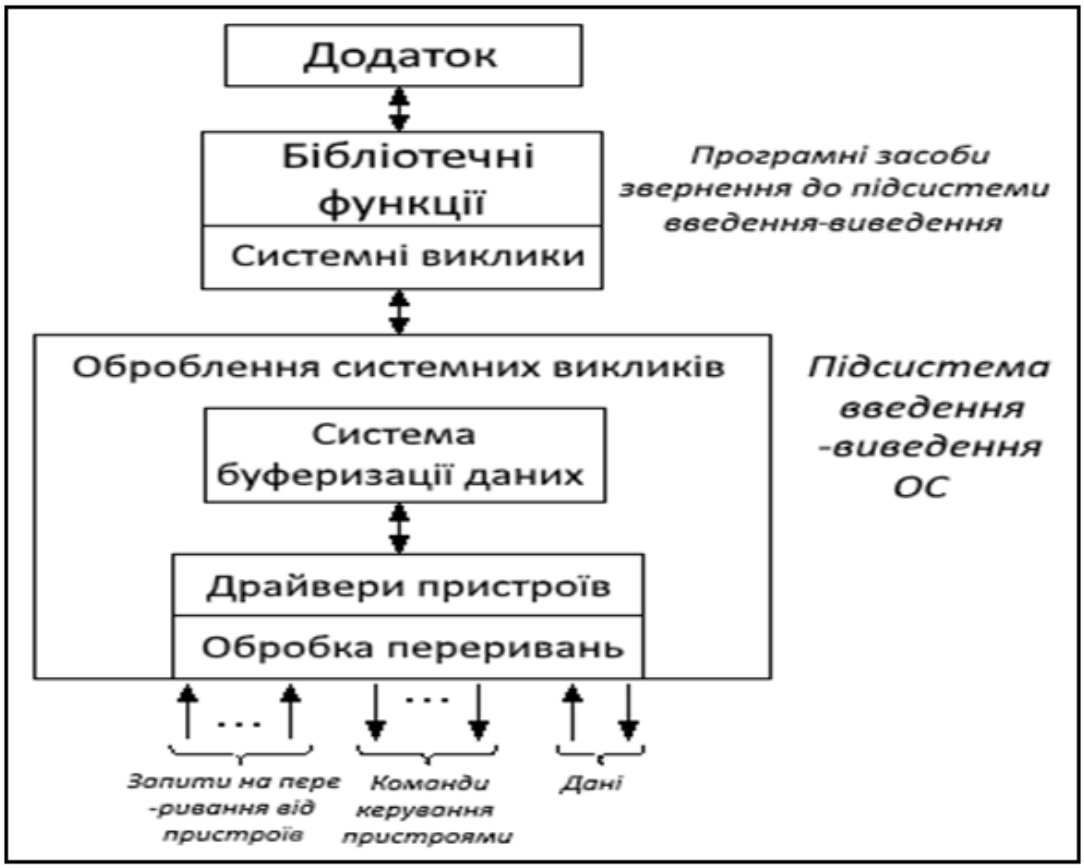

Рисунок 15.4 – Багаторівнева організація підсистеми введення-виведення

Багатошарова побудова програмного забезпечення, характерна для операційних систем, виявляється особливо природною і корисною при побудові підсистеми введення-виведення при великій різноманітності пристроїв введення-виведення, які мають істотно різні характеристики (принтер і диски, графічний монітор і мережевий адаптер тощо).

Ієрархічна структура програмного забезпечення дозволяє дотримуватися балансу між двома дуже суперечливими вимогами. З одного боку, необхідно врахувати всі особливості кожного пристрою, а з іншого – забезпечити єдине логічне

представлення і уніфікований інтерфейс для пристроїв усіх типів. При цьому нижні шари підсистеми введення-виведення повинні включати індивідуальні драйвери, написані для конкретних фізичних пристроїв, а верхні шари повинні узагальнювати процедури управління цими пристроями.

Також можна надавати спільний інтерфейс, якщо не для всіх пристроїв, то принаймні для груп пристроїв, які мають деякі загальні характеристики, наприклад для принтерів певного виробника або для всіх матричних принтерів.

Обробка переривань. Переривання мають бути приховані як можна глибше в надрах операційної системи, щоб як можна менша частина ОС мала з ними справу. Найкращий спосіб полягає в дозволі процесу, який ініціював операцію введення-виведення, блокувати себе до завершення операції і настання переривання. Процес може блокувати себе, використовуючи, наприклад, виклик DOWN для семафора, або виклик WAIT для змінної умови, або виклик RECEIVE для очікування повідомлення. При настанні переривання процедура обробки переривання виконує розблокування процесу, який ініціював операцію введення-виведення, використовуючи виклики UP, SIGNAL або посилаючи процесу повідомлення. У будь-якому випадку ефект від переривання полягатиме в тому, що раніше заблокований процес тепер продовжить своє виконання.

Драйвери пристроїв. Увесь залежний від пристрою код поміщається в драйвер пристрою. Кожен драйвер управляє пристроями одного типу або, можливо, одного класу.

У операційній системі тільки драйвер пристрою знає про конкретні особливості якого-небудь пристрою. Наприклад, тільки драйвер диска має справу з доріжками, секторами, циліндрами, часом встановлення головки і іншими чинниками, які забезпечують правильну роботу диска.

Драйвер пристрою приймає запит від пристрою програмного шару і вирішує, як його виконати. Типовим запитом є читання *n* блоків даних. Якщо драйвер був вільний під час надходження запиту, то він починає виконувати запит негайно. Якщо ж він був зайнятий обслуговуванням іншого запиту, то запит, який знову поступив, приєднується до черги інших запитів, і він буде виконаний, коли настане його черга.

Перший крок в реалізації запиту введення-виведення, наприклад, для диска,

полягає в перетворенні його з абстрактної форми в конкретну. Для дискового драйвера це означає перетворення номерів блоків на номери циліндрів, головок, секторів, перевірку того, чи працює мотор чи знаходиться головка над потрібним циліндром. Коротше кажучи, він повинен вирішити, які операції контролера потрібно виконати і в якій послідовності.

Після передачі команди контролеру драйвер повинен вирішити, чи блокувати себе до закінчення заданої операції чи ні. Якщо операція займає значний час, як при друці деякого блоку даних, то драйвер блокується до тих пір, поки операція не завершиться, і обробник переривання не розблокує його. Якщо команда введення-виведення виконується швидко (наприклад, прокрутка екрану), то драйвер чекає її завершення без блокування.

Незалежний від пристроїв шар операційної системи. Велика частина програмного забезпечення введення-виведення є незалежною від пристроїв. Точна межа між драйверами і незалежними від пристроїв програмами визначається системою, оскільки деякі функції, які могли б бути реалізовані незалежним способом, насправді виконані у вигляді драйверів для підвищення ефективності або з інших причин.

Типовими функціями для незалежного від пристроїв шару є:

- забезпечення загального інтерфейсу до драйверів пристроїв;
- іменування пристроїв;
- захист пристроїв;
- забезпечення незалежного розміру блоку;
- буферизація;
- розподіл пам'яті на блок-орієнтованих пристроях;
- розподіл і звільнення виділених пристроїв;
- повідомлення про помилки.

Зупинимося на деяких функціях цього переліку. Верхнім шарам програмного забезпечення незручно працювати з блоками різної величини, тому цей шар забезпечує єдиний розмір блоку, наприклад, за рахунок об'єднання декількох різних блоків в єдиний логічний блок. У зв'язку з цим верхні рівні мають справу з абстрактними пристроями, які використовують єдиний розмір логічного блоку

незалежно від розміру фізичного сектора. При створенні файлу або заповненні його новими даними необхідно виділити йому нові блоки. Для цього ОС повинна вести список або бітову карту вільних блоків диска. На підставі інформації про наявність вільного місця на диску може бути розроблений алгоритм пошуку вільного блоку, який незалежний від пристрою і реалізовується програмним шаром, що знаходиться вище за шар драйверів.

Призначений для користувача шар програмного забезпечення. Хоча велика частина програмного забезпечення введення-виведення знаходиться всередині ОС, деяка його частина міститься в бібліотеках, які зв'язуються з призначеними для користувача програмами. Системні виклики, які включають виклики введення-виведення, робляться бібліотечними процедурами. Якщо програма, написана мовою *C,* містить виклик *count = write(fd, buffer, nbytes),* то бібліотечна процедура *write* буде пов'язана з програмою.

Набір подібних процедур є частиною системи введення-виведення. Зокрема, форматування введення або виведення виконується бібліотечними процедурами. Прикладом може служити функція *printf* мови *C*, яка приймає рядок формату і, можливо, деякі змінні, як вхідну інформацію, потім будує рядок символів ASCII і робить виклик *write* для виведення цього рядка. Стандартна бібліотека введення-виведення містить велике число процедур, які виконують уведення-виведення і працюють як частина програми користувача.

Іншою категорією програмного забезпечення введення-виведення є підсистема ***спулінгу*** (spooling). Спулінг – це спосіб роботи з виділеними пристроями в мультипрограмній системі. Розглянемо типовий пристрій, який вимагає спулингу – принтер. Хоча технічно легко дозволити кожному процесу користувача відкрити спеціальний файл, пов'язаний з принтером, такий спосіб небезпечний через те, що процес користувача може монополізувати принтер на довільний час. Замість цього створюється спеціальний процес – монітор, який отримує виняткові права на використання цього пристрою. Також створюється спеціальний каталог, який називається каталогом спулінгу. Для того щоб надрукувати файл, процес користувача поміщає інформацію, яка виводиться, в цей файл, і поміщає його в каталог спулінгу. Процес-монітор по черзі роздруковує всі файли, які містяться в каталозі спулінгу.