## 15.2 ЗАДАЧІ ОС З УПРАВЛІННЯ ФАЙЛАМИ І ПРИСТРОЯМИ

Підсистема введення-виведення мультипрограмної ОС при обміні даними із зовнішніми облаштуваннями комп'ютера повинна розв'язувати ряд загальних задач, з яких найважливішими є такі:

– організація паралельної роботи пристроїв введення-виведення і процесора;

– узгодження швидкостей обміну і кешування даних;

– розподіл пристроїв і даних між процесами;

– забезпечення зручного логічного інтерфейсу між пристроями і іншою частиною системи;

– підтримка широкого спектрудрайверів з можливістю простого включення в систему нового драйвера;

– динамічне завантаження і вивантаження драйверів;

– підтримка декількох файлових систем;

– підтримка синхронних і асинхронних операцій уведення-виведення. У наступних підрозділах усі ці задачі розглянемо детальніше.

### 15.2.1 Організація паралельної роботи пристроїв введення-виведення

Кожний пристрій уведення-виведення обчислювальної системи – диск, принтер, термінал тощо – забезпечений спеціалізованим блоком управління, який називається контролером. Контролер взаємодіє з драйвером – системним програмним модулем, призначеним для управління цим пристроєм (рис. 15.1). Контролер періодично приймає від драйвера інформацію, яка виводиться на пристрій, а також команди управління, які говорять про те, що з цією інформацією потрібно зробити. Наприклад, вивести у вигляді тексту в певну область терміналу або записати в певний сектор диска.

Під управлінням контролера пристрій може деякий час виконувати свої операції автономно, не вимагаючи уваги з боку центрального процесора. Цей час залежить від багатьох чинників – об'єму інформації, яка виводиться, міри інтелектуальності контролера, швидкодії пристрою тощо. Навіть найпримітивніший контролер, який виконує прості функції, зазвичай витрачає досить багато час на самостійну реалізацію

подібної функції після отримання чергової команди від процесора. Це ж справедливо і для складних контролерів.

Процеси, які відбуваються в контролерах, протікають в періоди між видачами команд незалежно від ОС. Від підсистеми введення-виведення вимагається спланувати в реальному масштабі часу запуск і призупинення великої кількості різноманітних драйверів, забезпечивши прийнятний час реакції кожного драйвера на незалежні події контролера. З іншого боку, необхідно мінімізувати завантаження процесора задачами введення-виведення, залишивши якомога більше процесорного часу на виконання потоків користувача.

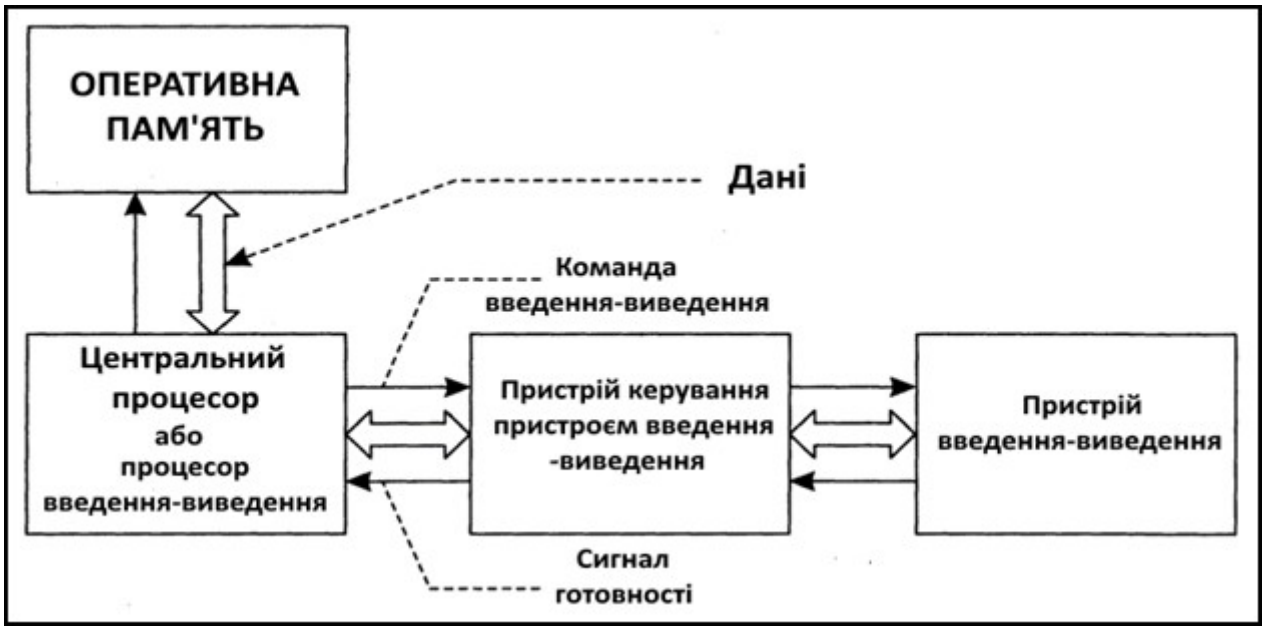


**Рисунок 15.1 –** Управління введенням-виведенням

Це задача є класичною задачею планування систем реального часу і розв'язується на основі багаторівневої пріоритетної схеми обслуговування за перериваннями. Для забезпечення прийнятного рівня реакції усі драйвери (чи частини драйверів) розподіляються за декількома пріоритетними рівнями відповідно до вимог часу реакції і часу використання процесора.

### 15.2.2 Узгодження швидкостей обміну і кешування даних

При обміні даними завжди виникає задача узгодження швидкості. Наприклад, якщо один процес користувача обчислює деякі дані і передає їх іншому процесу користувача через оперативну пам'ять, то в загальному випадку швидкості генерації даних і їх читання не співпадають. Узгодження швидкості досягається за рахунок ***буферизації даних*** в оперативній пам'яті і синхронізації доступу процесів до буфера. У тих спеціалізованих ОС, в яких забезпечення високої швидкості введення-виведення є першочерговою задачею (управління в реальному часі, послуги мережевої файлової служби тощо), велика частина оперативної пам'яті відводиться не під коди прикладних програм, а під буферизацію даних.

Проте буферизація тільки на основі оперативної пам'яті в підсистемі введення-виведення є недостатньою. Різниця між швидкістю обміну з оперативною пам'яттю, куди процеси поміщають дані для обробки, і швидкістю роботи зовнішнього пристрою часто стає занадто великою, щоб в якості тимчасового буфера можна було б використати оперативну пам'ять – її об'єму може просто не вистачити. Для таких випадків необхідно передбачити особливі заходи, і часто як буфер використовується дисковий файл, який називається також ***спул-файлом*** (від spool – шпулька, буфер). Типовий приклад застосування спулінгу дає організація виведення даних на принтер. Для друкарських документів об'єм в декілька десятків мегабайт – не рідкість, тому для їх тимчасового зберігання об'єму оперативної пам'яті явно недостатньо.

Іншим рішенням цієї проблеми є використання великої буферної пам'яті в контролерах зовнішніх пристроїв. Такий підхід особливо корисний в тих випадках, коли переміщення даних на диск занадто уповільнює обмін. Наприклад, в контролерах графічних дисплеїв застосовується буферна пам'ять, більша за об'ємом оперативної пам'яті, і це істотно прискорює виведення графіки на екран.

Буферизація даних дозволяє не лише погоджувати швидкості роботи процесора і зовнішнього пристрою, але і розв'язати іншу задачу – скоротити кількість реальних операцій уведення-виведення за рахунок кешування даних. Дисковий кеш є неодмінним атрибутом підсистем уведення-виведення практично в усіх операційних систем, значно скорочуючи час доступу до даних.

### 15.2.3 Розподіл пристроїв і даних між процесами

Пристрої уведення-виведення може надаватися процесам як в ***монопольне, так і в загальне (що розділяється)*** використання. При цьому ОС повинна забезпечувати контроль доступу тими ж способами, що і при доступі процесів до інших ресурсів обчислювальної системи – шляхом перевірки прав користувача або групи користувачів, від імені яких діє процес, на виконання тієї або іншої операції над пристроєм. Наприклад, певній групі користувачів послідовний порт дозволено забирати в монопольне володіння, а іншим користувачам це заборонено.

Операційна система може контролювати доступ не лише до пристрою в цілому, але і до окремих областей даних, які зберігаються або відображаються цим пристроєм. Диск є типовим прикладом пристрою, для якого важливо контролювати доступ не до пристрою в цілому, а до окремих каталогів і файлів.

При виведенні інформації на графічний дисплей окремі вікна екрану також є ресурсами, до яких необхідно забезпечити той або інший вид доступу для процесів, які протікають в системі. При цьому для кожної області даних або частини пристрою можуть бути задані свої права доступу, не пов'язані прямо з правами доступу до пристрою в цілому. Так, у файловій системі для кожного каталогу і файлу можна задати індивідуальні права доступу. Очевидно, що для організації загального доступу до частин пристрою або частин даних, які зберігаються на ньому, неодмінною умовою є задання режиму загального використання пристрою в цілому.

Один і той же пристрій в різні періоди часу може використовуватися як в режимах, які розділяються, так і в монопольних. Проте існують пристрої, для яких характерний один з цих режимів. Наприклад, послідовні порти іалфавітно- цифрові термінали частіше використовуються в монопольному режимі, а диски

– в режимі загального доступу. Операційна система повинна надавати ці пристрої в обох режимах, здійснюючи відстежування процедур захоплення і звільнення монопольно використовуваних пристроїв, а у разі загального використання, оптимізуючи послідовність операцій введення-виведення для різних процесів з метою підвищення загальної продуктивності, якщо це можливо.

Наприклад, при обміні даними декількох процесів з диском можна так упорядкувати послідовність операцій, що непродуктивні витрати часу на

переміщення головок істотно зменшуються. При цьому для окремих процесів можливе деяке уповільнення операції введення-виведення.

При розподілу пристрою між процесами може виникнути необхідність у розмежуванні області даних двох процесів один від одного. Така потреба виникає при загальному використанні так званих послідовних пристроїв, дані в яких на відміну від пристроїв прямого доступу не адресуються. Типовим представником такого роду пристроїв є принтер, який не виділяється в монопольне володіння процесам, і в той же час кожен документ має бути надрукований у вигляді послідовного набору сторінок. Для подібних пристроїв організовується черга завдань на виведення, при цьому кожне завдання є порцією даних, яку не можна розривати, наприклад, документ для друку. Для зберігання черги завдань використовується спул-файл, який одночасно погоджує швидкості роботи принтера і оперативної пам'яті і дозволяє організувати розбиття даних на логічні порції. Оскільки спул-файл знаходиться на пристроях прямого доступу, то процеси можуть одночасно виконувати виведення на принтер, поміщаючи дані у свій розділ спул-файлу.

### 15.2.4 Забезпечення логічного інтерфейсу між пристроями

Різноманітність пристроїв введення-виведення робить особливо актуальною функцію ОС зі створення екрануючого логічного інтерфейсу між периферійними пристроями і додатками. Практично всі сучасні операційні системи підтримують в якості основи такого інтерфейсу файлову модель периферійних пристроїв, коли будь-який пристрій виглядає для прикладного програміста послідовним набором байтів. З цим набором байтів можна працювати за допомогою уніфікованих системних викликів (наприклад, read і write), задаючи ім'я файлу-пристрою і зміщення від початку послідовності байтів. Для підтримки такого інтерфейсу підсистема введення-виведення повинна виконати досить велику роботу, враховуючи різницю в організації операцій обміну даними, наприклад, з жорстким диском і графічним терміналом. Привабливість моделі файлу-пристрою полягає в її простоті і уніфікованості для обладнання будь-якого типу, проте в багатьох випадках для програмування операцій уведення-виведення деякого пристрою вона є занадто бідною. Тому ця модель часто використовується тільки як базис, над яким підсистема введення-виведення будує

змістовнішу модель пристрою конкретного типу. Підсистем введення-виведення надає, як правило, специфічний інтерфейс для виведення графічної інформації на дисплей або принтер, для програмування операцій мережевого обміну тощо. При цьому розробник специфічного інтерфейсу завжди може спиратися на наявний базовий інтерфейс.

### 15.2.5 Підтримка широкого спектру драйверів

Перевагою підсистеми введення-виведення будь-якої універсальної ОС є наявність різноманітного набору драйверів для найпопулярніших периферійних пристроїв. Прекрасно спланована і реалізована операційна система може потерпіти невдачу на ринку тільки через те, що до її складу не включений достатній набір драйверів. У цьому випадку адміністратори і користувачі змушені шукати потрібний їм драйвер для наявного у них зовнішнього пристрою у виробників устаткування або, що ще гірше, займатися його розробкою. Саме в такій ситуації опинилися користувачі перших версій OS/2, і, можливо, ця обставина послужила свого часу не останньою причиною здачі позицій цієї непоганої операційної системи багатій на драйвери ОС Windows 3.x.

Щоб операційна система не відчувала нестачі в драйверах, потрібна наявність чіткого, зручного і відкритого інтерфейсу між драйверами і іншими компонентами ОС. Такий інтерфейс потрібний для того, щоб драйвери писали не лише безпосередні розробники цієї операційної системи, але і велика армія програмістів по всьому світу, в першу чергу – тих підприємств, які випускають зовнішні пристрої для комп'ютерів. Відкритість інтерфейсу драйверів, тобто доступність його опису для незалежних розробників програмного забезпечення, є необхідною умовою успішного розвитку операційної системи.

Драйвер взаємодіє, з одного боку, з модулями ядра ОС (модулями підсистеми введення-виведення, модулями системних викликів, модулями підсистем управління процесами і пам'яттю), а з іншого боку – з контролерами зовнішніх пристроїв (рис. 15.2). Тому існують два типи інтерфейсів: ***інтерфейс «драйвер-ядро»*** (Driver Kernel Interface, DKI) і інтерфейс ***«драйвер-пристрій»*** (Driver Device Interface, DDF).

Інтерфейс «драйвер-ядро» має бути стандартизованим у будь-якому випадку.

Іінтерфейс «драйвер-пристрій» має сенс стандартизувати тоді, коли підсистема введення-виведення не дозволяє драйверу безпосередньо взаємодіяти з апаратурою контролера, а виконує ці операції самостійно. Екранування драйвера від апаратури є дуже корисною функцією, оскільки драйвер в цьому випадку стає незалежним від апаратної платформи.


**Рисунок 15.2** – Структура системи введення-виведення

Підсистема введення-виведення може підтримувати декілька різних типів інтерфейсів DKI/DDI, надаючи специфічний інтерфейс для пристроїв певного класу.

Так, в ОС Windows NT для драйверів мережевих адаптерів існує інтерфейс стандарту NDIS (Network Driver Interface Specification), тоді як драйвери мережевих транспортних протоколів взаємодіють з верхніми шарами мережевого програмного забезпечення по інтерфейсу TDI (Transport Driver Interface).

Підсистема введення-виведення підтримує велику кількість системних функцій, які драйвер може викликати для виконання деяких типових дій. Прикладами можуть служити згадані операції обміну з регістрами контролера, ведення буферів для проміжного зберігання даних введення-виведення, синхронізація роботи декількох драйверів, копіювання даних з призначеного для користувача простору в простір системи тощо.

Для підтримки процесу розробки драйверів операційної системи випускається так званий пакет DDK (Driver Development Kit) – набір відповідних інструментальних засобів: бібліотек, компіляторів і відладчиків.

### 15.2.6 Динамічне завантаження і вивантаження драйверів

Окрім проблеми розробки нових драйверів існує також проблема включення драйвера до складу модулів працюючої ОС, тобто динамічного завантаження-вивантаження драйвера. Оскільки набір потенційно підтримуваних цією ОС периферійних пристроїв завжди істотно ширше набору пристроїв, якими ОС повинна управляти при установці на конкретній машині, то цінною властивістю ОС є можливість динамічно завантажувати в оперативну пам'ять необхідний драйвер (без зупинки ОС), і вивантажувати його після того, як потреба в підтримці пристрою минула, що може істотно заощадити системну область пам'яті.

Альтернативою динамічному завантаженню драйверів при зміні поточної конфігурації зовнішніх пристроїв комп'ютера є повторна компіляція коду ядра з необхідним набором драйверів, який створює між усіма компонентами ядра статичні зв'язки замість динамічних. Наприклад, таким чином вирішувалася ця проблема в ранніх версіях операційної системи UNIX. При статичних зв'язках між ядром і драйверами структура ОС спрощується, але цей підхід вимагає наявності початкових кодів модулів операційної системи, доступність яких швидше є виключенням, а не правилом. Крім того, в цьому варіанті працюючу попередню версію операційної системи необхідно зупинити і замінити новою, а перерви в роботі ОС в деяких додатках можуть і не допускатися.

Підтримка динамічного завантаження драйверів є практично обов'язковою вимогою для сучасних універсальних операційних систем.

### 15.2.7 Підтримка декількох файлових систем

Диски представляють особливий рід периферійних пристроїв, оскільки саме на них зберігається велика частина як призначених для користувача, так і системних даних. Дані на дисках організовуються у файлові системи, і властивості файлової системи багато в чому визначають властивості самої ОС – її відмовостійкість,

швидкодію, максимальний об'єм даних.

Популярність файлової системи часто призводить до її міграції з «рідної» ОС в інші операційні системи. Наприклад, файлова система FAT з'явилася спочатку у MS-DOS, але потім була реалізована в OS/2, сімействі MS Windows і багатьох реалізаціях UNIX. Зважаючи на це підтримка декількох популярних файлових систем для підсистеми введення-виведення також важлива, як і підтримка широкого спектру периферійних пристроїв. Важливо також, щоб архітектура підсистеми введення-виведення дозволяла досить просто включати до її складу нові типи файлових систем, без необхідності переписування коду.

### 15.2.8 Синхронні і асинхронні операції введення-виведення

Операція введення-виведення може виконуватися стосовно програмного модуля, який запросив операцію, в ***синхронному*** або ***асинхронному*** режимах. Сенс цих режимів той же, що і для розглянутих вище системних викликів. Синхронний режим означає, що програмний модуль призупиняє свою роботу до тих пір, поки операція введення-виведення не буде завершена (рис. 15.3, *а),* а при асинхронному режимі програмний модуль продовжує виконуватися одночасно з операцією введення-виведення (рис. 15.3, *б).*

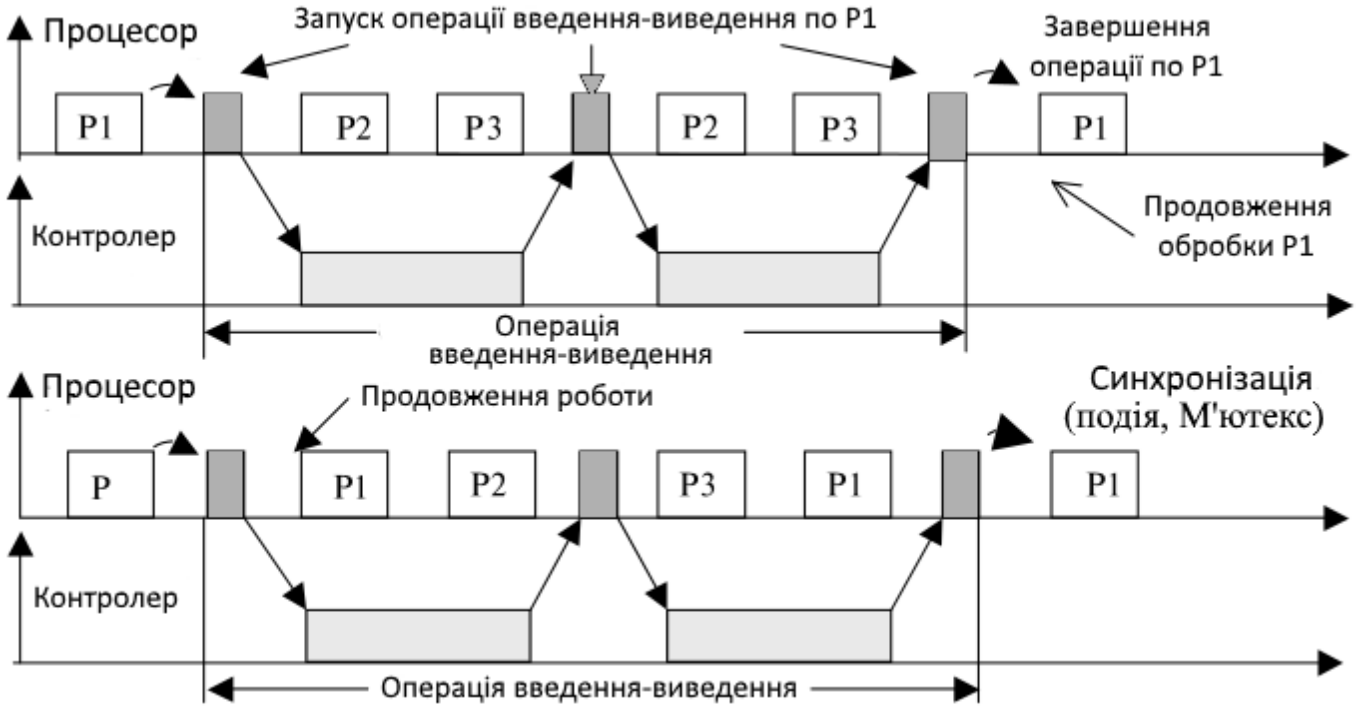


**Рисунок 15.3** – Два режими виконання операцій введення-виведення

Відмінність же полягає в тому, що операція введення-виведення може бути ініційована не лише призначеним для користувача процесом – в цьому випадку операція виконується в рамках системного виклику, але і кодом ядра, наприклад кодом підсистеми віртуальної пам'яті для прочитання відсутньої в пам'яті сторінки. Підсистема введення-виведення повинна надавати своїм клієнтам (процесам користувача і ядра) можливість виконувати як синхронні, так і асинхронні операції введення-виведення, залежно від потреб процесу. Системні виклики введення-виведення частіше оформляються як синхронні процедури в зв'язку з тим, що такі операції тривають довго і процесу користувача або потоку все одно доведеться чекати отримання результатів операції для того, щоб продовжити свою роботу.

Внутрішні ж виклики операцій уведення-виведення з модулів ядра зазвичай виконуються у вигляді асинхронних процедур, оскільки кодам ядра потрібна свобода у виборі подальшої поведінки після запиту операції введення- виведення.