Підсистеми ядра ОС

- **Підсистема управління введенням-виведенням**
- **Підсистема управління оперативною пам'яттю**
- **Підсистема управління задачами (процесами)**
- **Підсистема управління даними (файлові системи)**
- **Підсистема забезпечення безпеки**

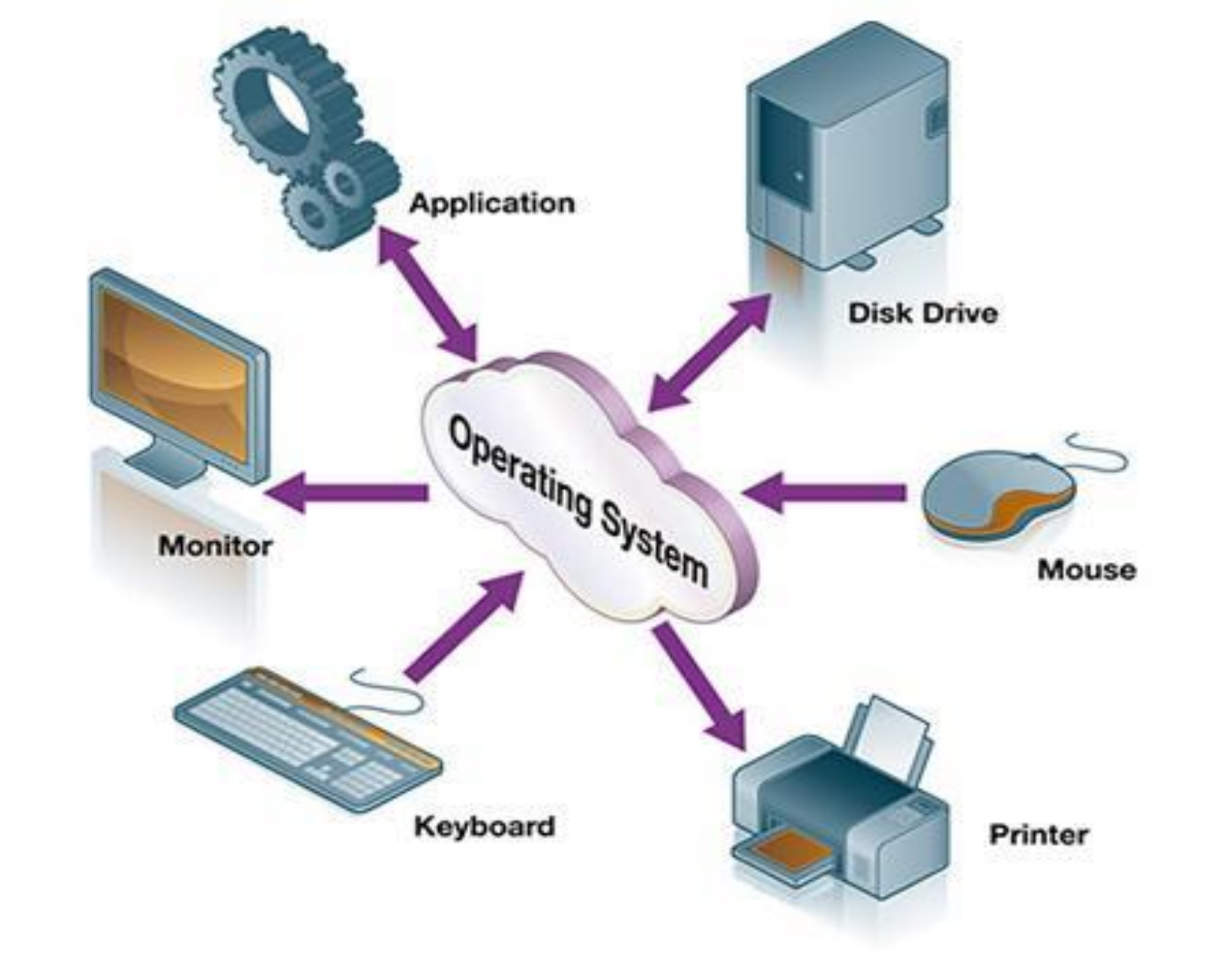
Управління пристроями

План лекції:

- Основні задачі управління пристроями
- Класифікації пристроїв
- Переривання
- Архітектура підсистеми введення/виведення
- Способи організації введення/виведення
- Буферизація
- Кешування дисків
- Драйвери пристроїв
- Керування пристроями в MS-DOS, Windows, Linux

Периферийні пристрої (ПП) –

всі основні апаратурні блоки комп'ютера, за виключенням процесора и основної пам'яті.



Характерні риси:

- ✔ велике різноманіття типів і моделей ПП
- ✔ швидкий прогрес технологій →
- ✔ збільшення продуктивності пристроїв →
- ✔ поява додаткових можливостей апаратури

ОСНОВНІ ЗАДАЧІ КЕРУВАННЯ ПРИСТРОЯМИ

- забезпечення надійної роботи ПП
  - дублювання даних,
  - використання завадозахищених кодів, контрольних сум даних і т.п.
  - виявлення апаратних помилок і збоїв,
  - компенсацію їх за рахунок надмірності даних і повторного виконання операцій

ефективне використання можливостей пристроїв

- скорочення часу на обмін даними
  - за рахунок підвищення швидкості обміну
  - за рахунок розпаралелювання роботи ПП і процесора
- підвищення продуктивності і скорочення кількості операцій В\В
  - за рахунок збереження даних в пам'яті для подальшого використання

- легке підключення нових ПП

  технологія «Plug & Play» - можливість оперативного під’єднання ПП без виключення комп’ютера

- максимальна стандартизація роботи з ПП

  зміна апаратури не повинна приводити до модифікації прикладного ПЗ

- Додаткові задачі:

  Збереження даних в ущільненому вигляді

  Шифрування даних

  . . .

КЛАСИФИКАЦІЯ ПП І ЇХ АРХІТЕКТУРА

Програмна архітектура (архітектура) пристрою - сукупність тих структурних особливостей, які впливають на роботу програм з пристроєм

Контролер (адаптер) пристрою - поставляється разом з пристроєм і містить електронні схеми управління пристроєм

Конструктивно контролер - плата, що вставляється в роз'єм шини комп'ютера, або розташована в корпусі пристрою.

!!! Програми працюють з ПП через контролери ПП

↓

Ідентичні поняття: “пристрій” ≡ “контролер пристрою”

Класифікація 1

- Пристрої послідовного доступу (sequential access)
  - наявність певного природного порядку
  - обробка даних складна

  приклади: магнітна стрічка, клавіатура, миша, модем, …

- Пристрої довільного доступу (random access).
  - можливе звернення до різних порцій даних в будь-якому порядку
  - ефективність роботи слабо залежить від порядку звернення
  - наявність адресації даних і операцій пошуку потрібної адреси

  приклади: магнітні диски, інші дискові пристрої, монітор ПК, …

Класифікація 2

- символьні – пристрої, які можуть передавати дані послідовно, байт за байтом:
  - спонтанно генерують вхідні дані (клавіатура, модем, миша, джойстик)
  - представляють дані у вигляді лінійного потоку (принтер, звукова карта)
  - можуть здійснювати 2 основні операції: *get* і *put*
- блочні – пристрої, які можуть передавати блок байтів, як єдине ціле: магнітні і оптичні диски і стрічки, і т. д;
- мережеві (мережеві карти)
- всі решта (таймери, графічні дисплеї, телевізійні пристрої, відеокамери і т. п.)

Класифікація 3

- Фізичні пристрої – реально існуючий пристрій, “залізо”
- Логічні пристрої – поняття, що характеризує спеціально призначений пристрій в даній ОС

приклади:

- “завантажувальний диск”
- “пристрій стандартного виведення” (може бути змінено)
- “пристрій стандартного введення” (може бути змінено)

- Віртуальні пристрої – програмно реалізований об’єкт, який поводить себе подібно деякому фізичному пристрою

приклади: віртуальні диски в ОП, віртуальна пам’ять на дисках, віртуальні CD-DVD, віртуальні екрани

ПЕРЕРИВАННЯ

Переривання - сигнали, при надходженні яких нормальна послідовність виконання програми може бути перервана

- при цьому система запам'ятовує інформацію, необхідну для відновлення роботи перерваної програми
- передає управління підпрограмі обробки переривання (ISR - Interrupt Service Routine)
- по завершенню обробки, як правило, керування повертається перерваній програмі.

Типи переривань

1. Апаратні переривання від ПП
   - ✔ виникають при
     - переході в стан готовности
     - виникненні помилки виконання операції
   - ✔ більшість процесорів підтримує векторні переривання.
2. Внутрішні апаратні переривання (exceptions)
3. Програмні переривання
   - ✔ використовуються для переходу з режиму режиму користувача у режим ядра на момент виклику системних функцій з прикладної програми
   - ✔ замість адреси підпрограми вказується номер переривання

!!! Не кожен пристрій генерує переривання (монітор ПК)

АРХІТЕКТУРА ПІДСИСТЕМИ В/В

З програмної точки зору, пристрій (або його контролер) зазвичай представлений регістрами (одним або декількома).

Регістр пристрою - це адресоване (що має адресу) машинне слово, використовуване для обміну даними між пристроєм і процесором.

Два основних типи регістрів пристроїв:

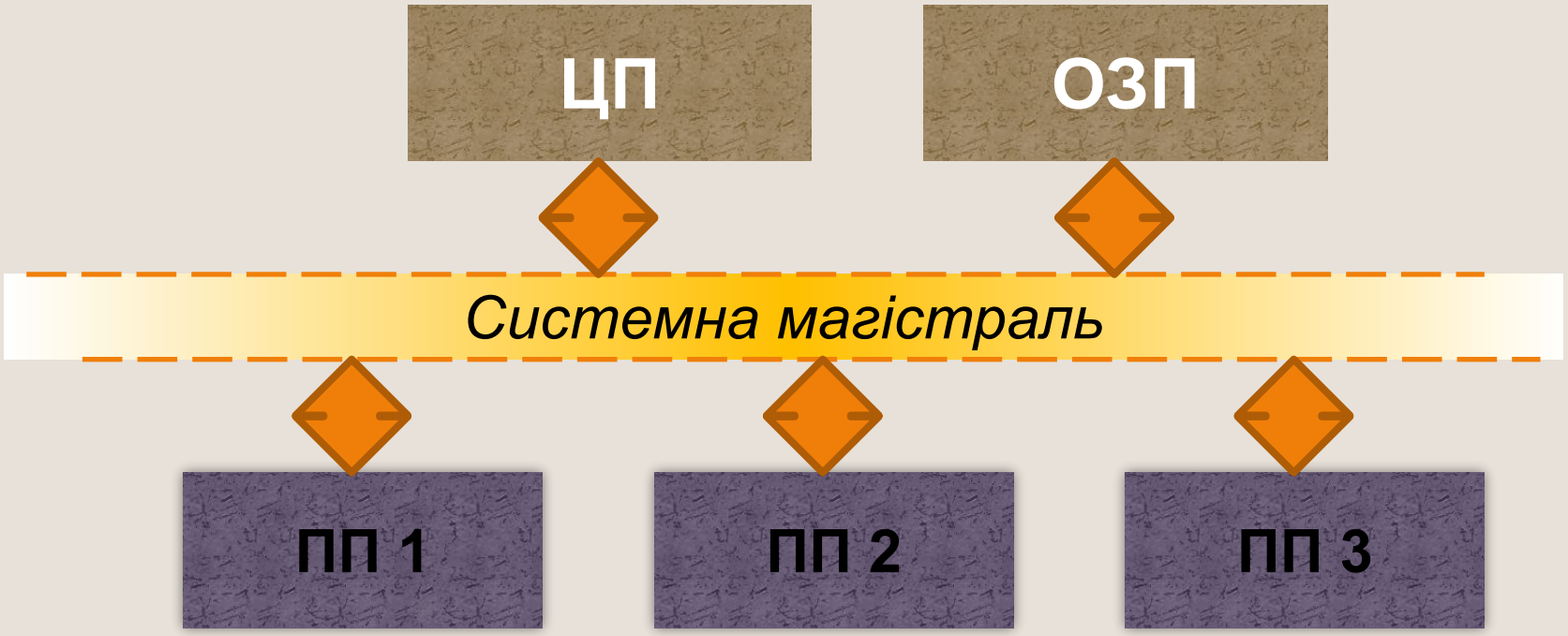
- **Регістр даних** (вхідних і вихідних) - для обміну даними.
  - запис у регістр – виведення даних на пристрій
  - читання з регістра - введення з пристрою
- **Регістр управління і стану** містять два типи бітів
  - **біти стану - для передачі процесору** інформації про поточний стан (біт готовності, біт помилки, біт занятості, …)
  - **біти керування - для передачі на пристрій** команд, що задають операцію, запускають її виконання, встановлюють режими роботи пристрою і т.п.

Типи конфігурацій :

- системи з **магістральною** архітектурою
- системи з **радіальною** архітектурою

Магістральна архітектура - підключення всіх наявних пристроїв (включаючи процесор та пам'ять) до єдиної системної магістралі (шини), яка об'єднує в собі лінії передачі даних, адрес і керуючих сигналів.

Спільне використання магістралі різними пристроями підпорядковується спеціальним правилам (протоколу), що забезпечує коректність роботи магістралі.

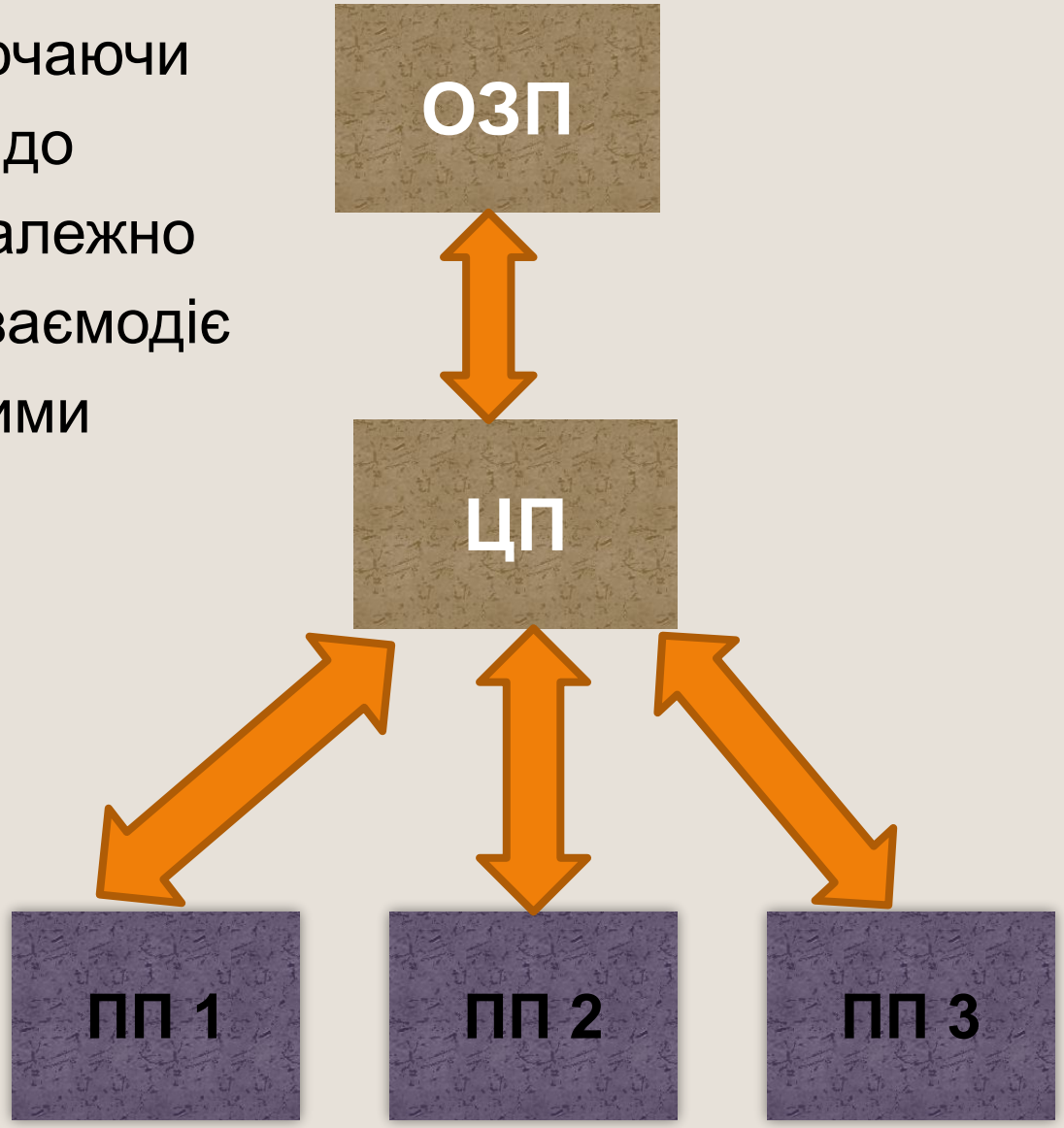


Особливості магістральної архітектури:

- ✔ однаковий спосіб підключення всіх пристроїв
- ✔ структура регістрів пристрою стандартизується (повинні відповідати стандарту даної магістралі)
- ✔ простота підключення нових типів пристроїв → зручна для відкритих обчислювальних систем (розрахованих на розширюваний набір ПП).

Радіальна архітектура –

кожен з пристроїв, включаючи пам'ять, підключається до процесора окремо, незалежно від інших пристроїв, і взаємодіє з процесором за власними правилами.



Особливості радіальної архітектури:

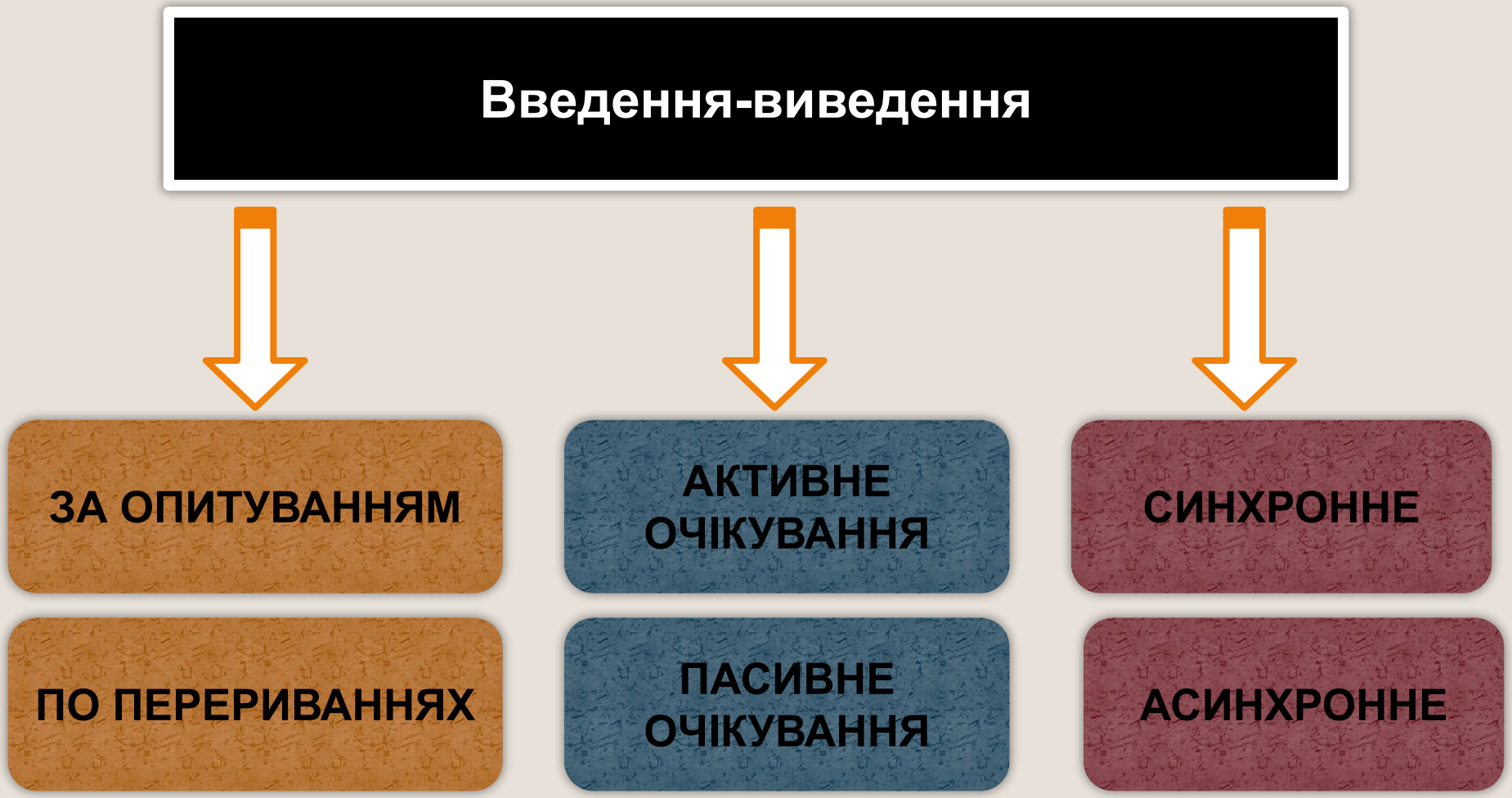
- ✔ індивідуальний вибір способу підключення, найбільш зручного для кожного типу пристроїв
- ✔ економія апаратних ресурсів і більш висока ефективність
- ✔ зручна у випадку, коли розрахована на постійний набір пристроїв. Розширення радіальної системи завжди викликає труднощі.

Контролер прямого доступу до пам'яті

(ПДП, англ. DMA - Direct Memory Access)

- **Без ПДП**: весь обмін даними йде через регістри процесора
- **З ПДП**: пряме перенесення даних з пристрою в пам'ять чи назад (процесор тільки ініціює операцію введення/виведення блоку даних, послявши відповідні команди контролеру ПДП) → часткове розвантаження процесора і магістралі.

Способи організації В/В



Логіка роботи драйвера ПП (приклад програми):

Нехай треба видати **N** байтів даних з масиву **A** на символьний пристрій **X.**

Архітектура пристрою представлена регистром данных **X.DATA** і прапором готовності **X.READY**.

Варіант а)

Введення-виведення без перевірки готовності

```
i:=1;
while i<=N do
begin
 X.DATA:=A[i];
 i:=i+1;
end;
```

!!!

1. Прапор **X.READ** завжди ***true*** → не потрібний
2. Якщо **X** – принтер, виведуться лише деякі літери (у моменти готовності)

Варіант б)

Введення-виведення за запитом готовності

```
i:=1;
while i<=N do
begin
 while not X.READY do
        ;
 X.DATA:=A[i];
 i:=i+1;
end;
```

!!! 1. Витрати часу на постійне опитування **X.READ.**

2. Якщо **X** не працює, система зависає.

Варіант в)

Введення-виведення по перериваннях

```
i:= 1;
while i<=N do
begin
 X_INT: if not X.READY
          return;
        X.DATA:=A[i];
        i:=i+1;
   end;
```

1. Пристрій не готовий – передача керування ОС. Працюють інші програми.
2. Якщо **X.READY** стає *true*, генерується апаратне переривання ПП.
3. Системний обробник повернеться до адреси **X_INT**.

Варіант **б** (В/В за запитом з циклом перевірки готовності)

\+ не витрачає часу на обробку переривань

\- можливий лише у випадку однозадачних ОС

Активне очікування (*busy waiting*) – спосіб очікування програмою деякої події, що оснований на постійній циклічній перевірці очікуваної умови

Варіант **в** (В/В з перериваннями)

\- витрачає деякий час на обробку переривань

\+ незамінний у випадку багатозадачних ОС

Пасивне очікування (*passive waiting*) – така реалізація очікування, при якій програма, що очікує, не витрачає процесорного часу

Операції В/В по відношенню до програмного додатку виконуються в синхронному чи асинхронному режимах.

Синхронний режим:

- додаток призупиняє свою роботу і чекає відгуку від пристрою
- додаток запускає операцію В/В і очікує її завершення
- так працюють операції В/В мов програмування → звичні для програмістів

Асинхронний режим

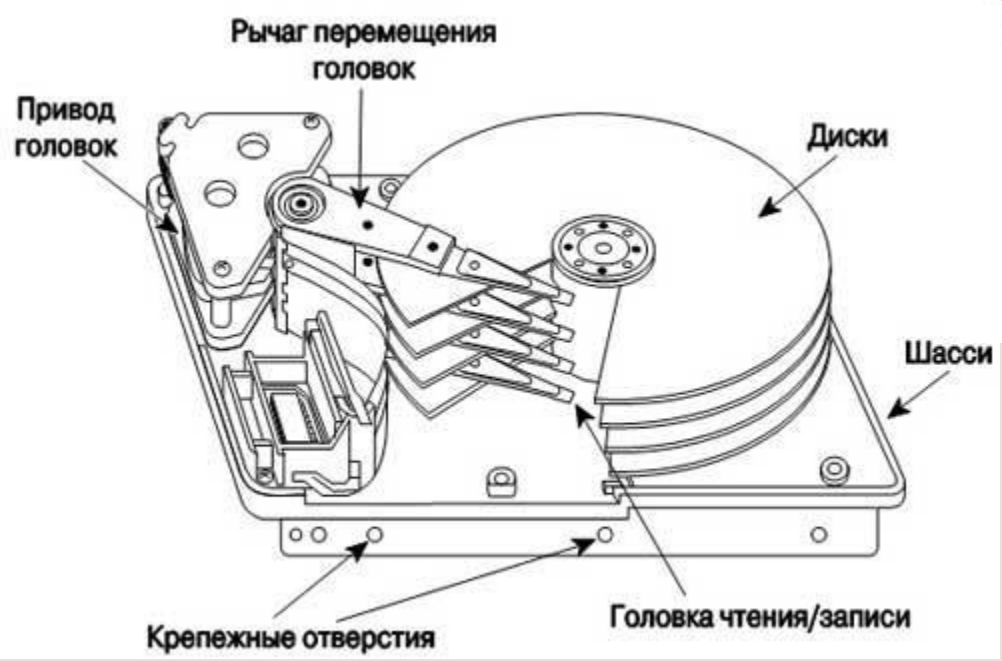
- додаток запускає функцію В/В, а функція одразу повертає керування додатку, не очікуючи її закінчення
- додаток продовжує роботу, паралельно з очікуванням відгуку від пристрою

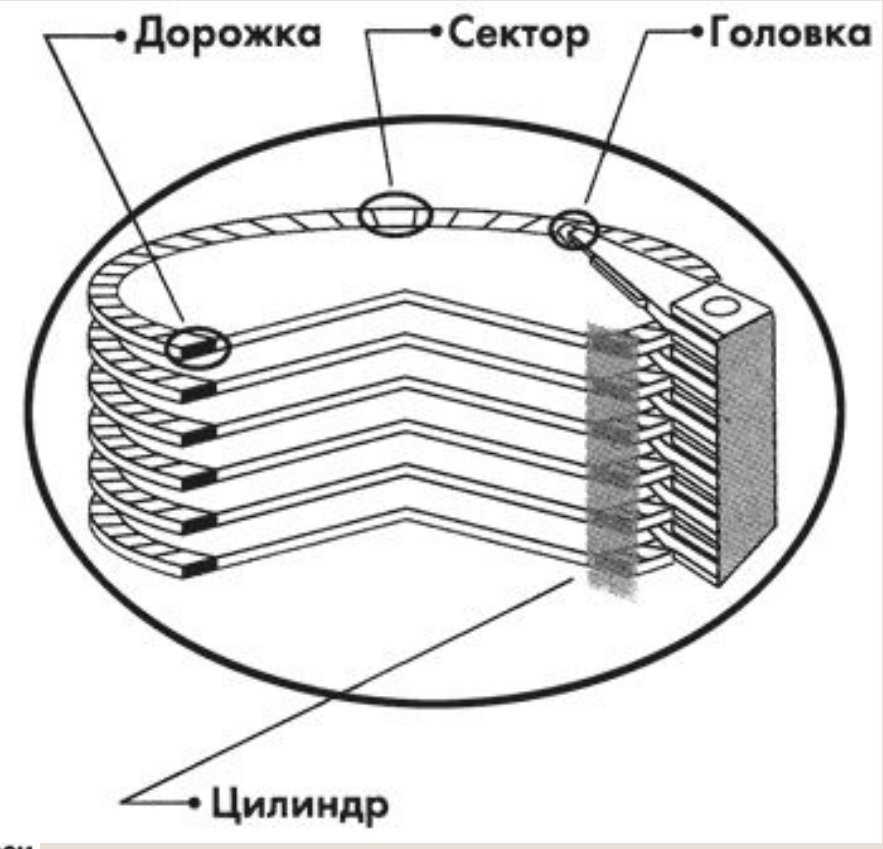
!!! ОС повинні для різних додатків забезпечити синхронну і асинхронну роботу з пристроями.

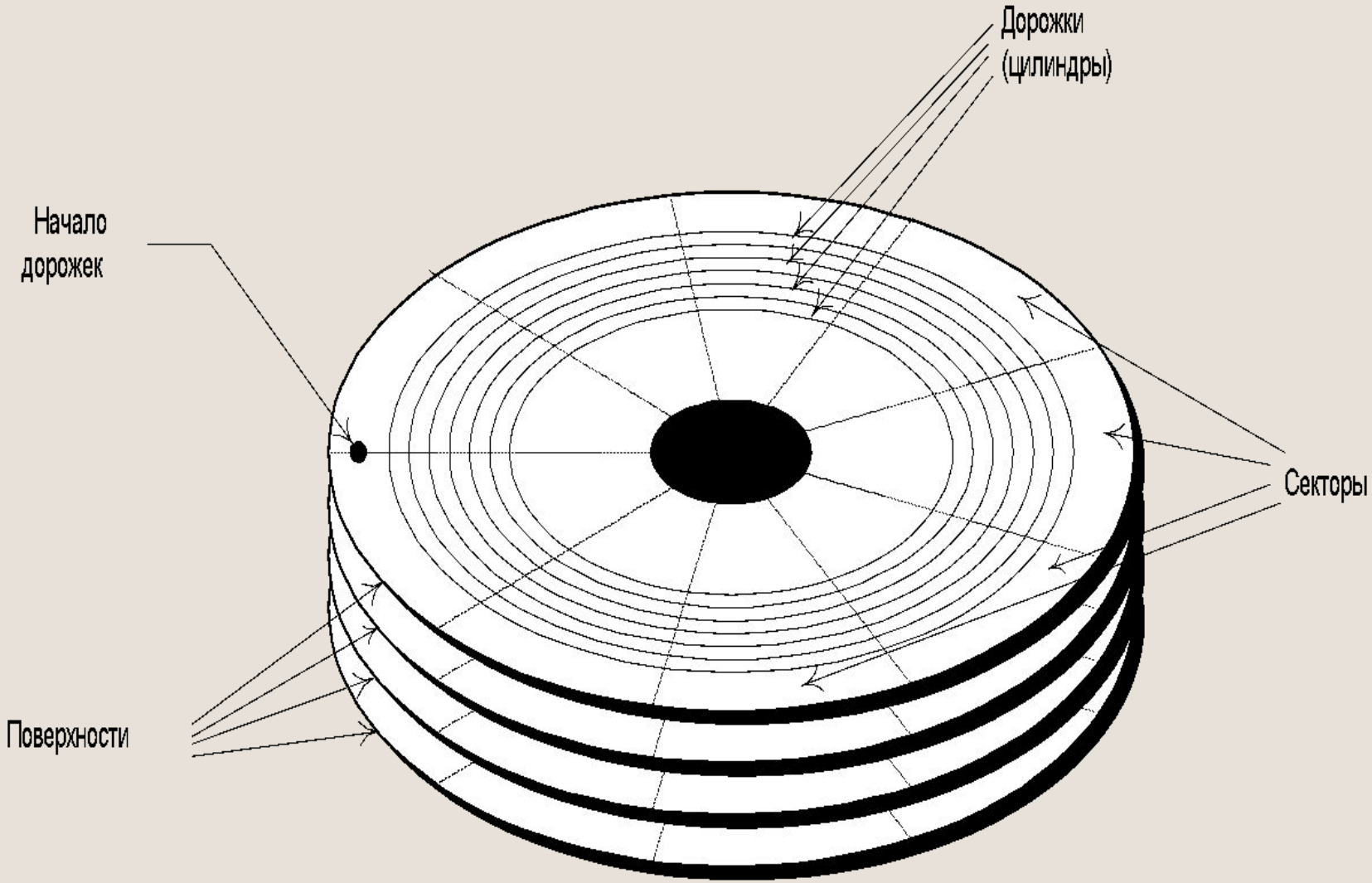
Повідомлення^

- Структура флеш-накопичувача USB

Структура магнітного диску


Рычаг перемещения головок
Привод головок
Диски
Шасси
Крепежные отверстия
Головка чтения/записи


Дорожка
Сектор
Головка
Цилиндр


Дорожки
(цилиндры)
Начало
дорожек
Сектора
Поверхности

Структура сектора диска

Структура доріжки

	Фізична адресна мітка (індексний отвір)	Початок кожної доріжки
GAP4A	Передіндексний синхронізуючий проміжок	
GAP5	Доіндексний проміжок	
IAM	Індексна адресна мітка	
GAP1	Після індексний проміжок	
CCCCC	Поле синхронізації	1-й сектор
IDAM CHRN CRC	Заголовок сектора (ідентифікатор), або секторний ID-маркер	
GAP2	Проміжок після ID-маркера	
DATA_IDA ...DATA ... CRC	Місце для розташування даних	
GAP3	Проміжок після даних	
...	... ...	...
CCCCC	Поле синхронізації	N-й (останній) сектор
IDAM CHRN CRC	Секторний ID-маркер	
GAP2	Проміжок після ID-маркера	
DATA_IDA ...DATA ... CRC	Місце для розташування даних	
GAP4	Завершуючий проміжок	

Розбиття доріжки на сектори

На диску:

H x C x N x S = . . .

Кількість байтів у секторі

Кількість секторів на доріжці

Кількість циліндрів на 1 поверхні (кількість циліндрів)

Кількість головок

Фізична нумерація секторів - 1÷N

Логічна нумерація секторів - 0÷L

БУФЕРИЗАЦІЯ

Буферизація – така організація В/В, при якій дані не передаються безпосередньо з пристрою в задану область пам'яті (або з області пам'яті на пристрій), а попередньо направляються у допоміжну область пам'яті, звану **буфером**.

Причини використання буферизації

1. Згладжування нерівномірності швидкостей процесів

!!! Чим більший буфер, тим менша ймовірність втрати даних через його переповнення.

2. Розпаралелювання введення та обробки

Після заповнення буфера його дані пересилаються у програму для обробки, а їх обробка виконується паралельно з накопиченням наступної порції даних в буфері.

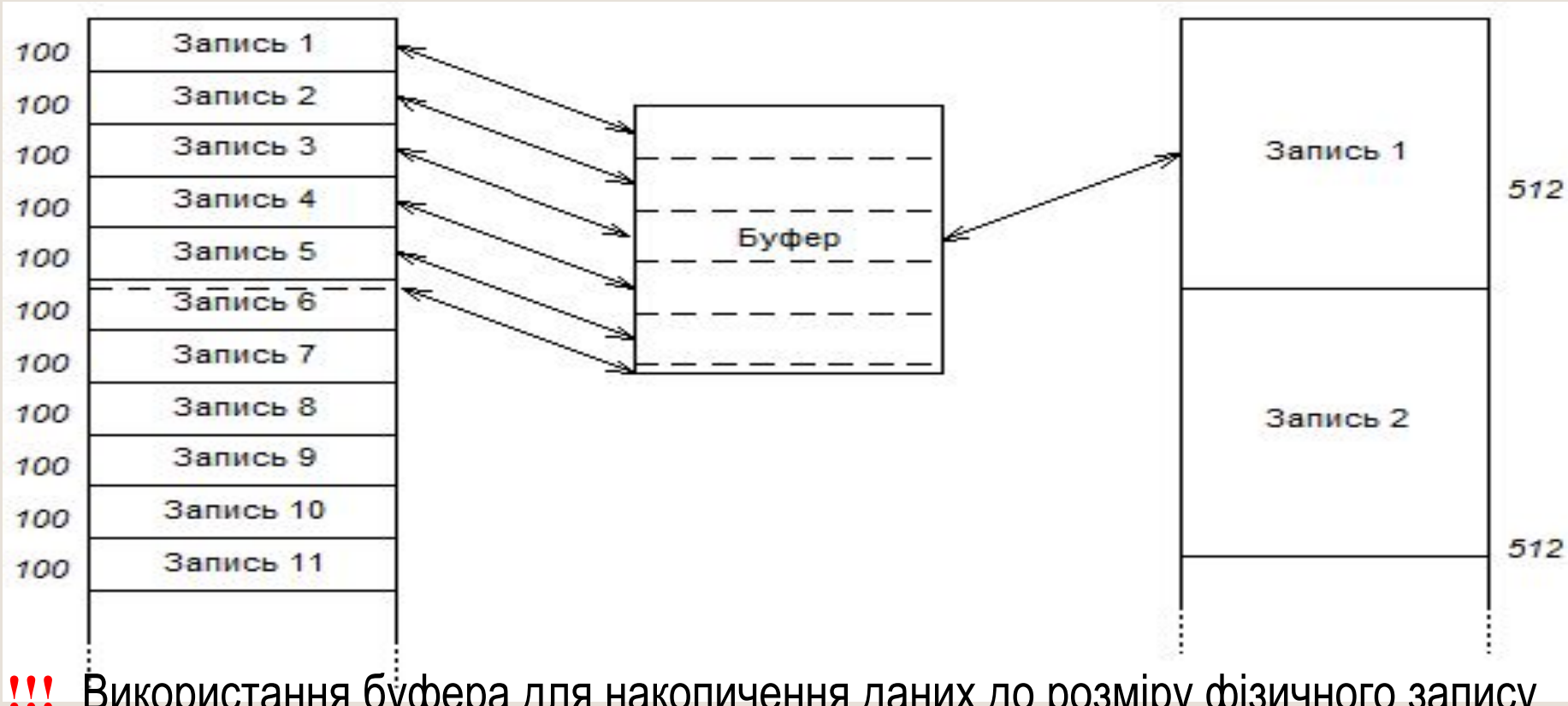
3. Редагування при інтерактивному введенні

Прикладна програма «не бачить» процесу редагування рядка, вона отримує весь рядок цілком після натискання, наприклад, клавіші Enter (можна легко відкоригувати помилки введення: «забити» невірний символ, повернутися в будь-яке місце рядка і внести зміни і т.п.)

4. Узгодження розмірів логічного та фізичного запису

Логічний запис – порція даних, зазначена в операторі В/В

Розмір **фізичного запису** визначається особливостями пристрою (для диска 512 байтів) і ніяк не пов'язаний з логікою програми.



!!! Використання буфера для накопичення даних до розміру фізичного запису дозволяє скоротити кількість операцій запису на диск і читання з диска.

5. Випереджуюче зчитування

- спеціальна форма буферизації, при якій система, виконавши зчитування потрібного блоку інформації, зчитує далі ще декілька блоків: наступні потрібні блоки вже будуть в пам'яті ⟶ пристрій вільний для інших операцій.

!!! При послідовному доступі кешування не допоможе.

6. Кешування

– особливий вид організації буферизації

Кешування дисків

Кешування - використання порівняно невеликої за обсягом, але швидкодіючої пам'яті для того, щоб зменшити кількість звертань до більш повільної пам'яті великого обсягу.

Гіпотеза про локальність посилань:

якщо в деякий момент часу відбулося звернення до певної ділянки даних, то найближчим часом можна з високою ймовірністю очікувати повторення звернень до тих самих або ж до сусідніх ділянок даних.

*** *«Cash» -«готівка» - ті дрібні гроші в гаманці, які дозволяють не звертатися щоразу в банк заради дрібних покупок.*

Сутність

- В якості кеша – масив буферів в системній ОП.
- Кожен буфер має:
  - Заголовок (адреса блоку диска, копію якого він містить)
  - Блок даних (відповідає розміру блоку даних (сектору) диску)
- Коли система отримує запит на читання, вона перевіряє (пошук лише по заголовках), чи немає цих даних у буфері. Якщо є – не треба читати з диска.
- У випадку запису змінених даних у заголовку помічається: буфер став “брудним” (не відповідає даним на диску) – не все потім треба записувати, а лише “брудні” буфери.

Проблема 1:

Блок в кеші не знайдений →

треба виділити буфер →

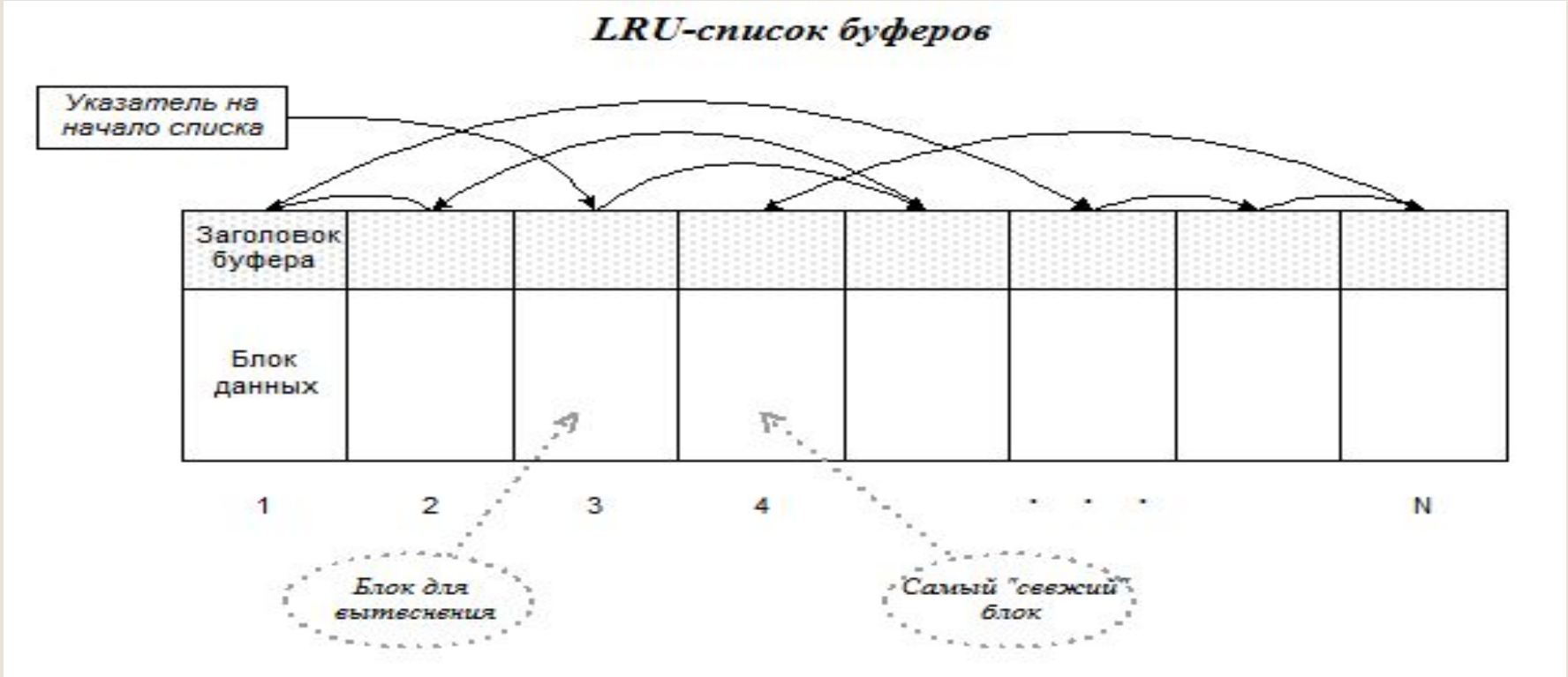
обсяг кеша обмежений →

треба “**витіснити**” з кеша один буферів →

Який буфер кеша “витісняти”?

Алгоритм ***LRU*** (Least Recently Used - «давно не використовуваний»)

1. Всі буфери пов’язують у зв’язаний список.
2. У заголовку буфера – посилання на наступний буфер.
3. При зверненні до блоку даних – переміщення буфера у кінець списку (поміняли покажчики).
4. В результаті: наприкінці списку – ті буфери, що довше не використовувались – кандидати на “витіснення”.



Проблема 2:

Закриття файлів, до якого відносяться “брудні” блоки →

примусове очищення всіх буферів (або буферів певного файлу)

Наприклад, в UNIX – через кожні 30 хв.

!!! Кешування операцій запису на диск створює певну небезпеку втрати даних.

Проблема 3:

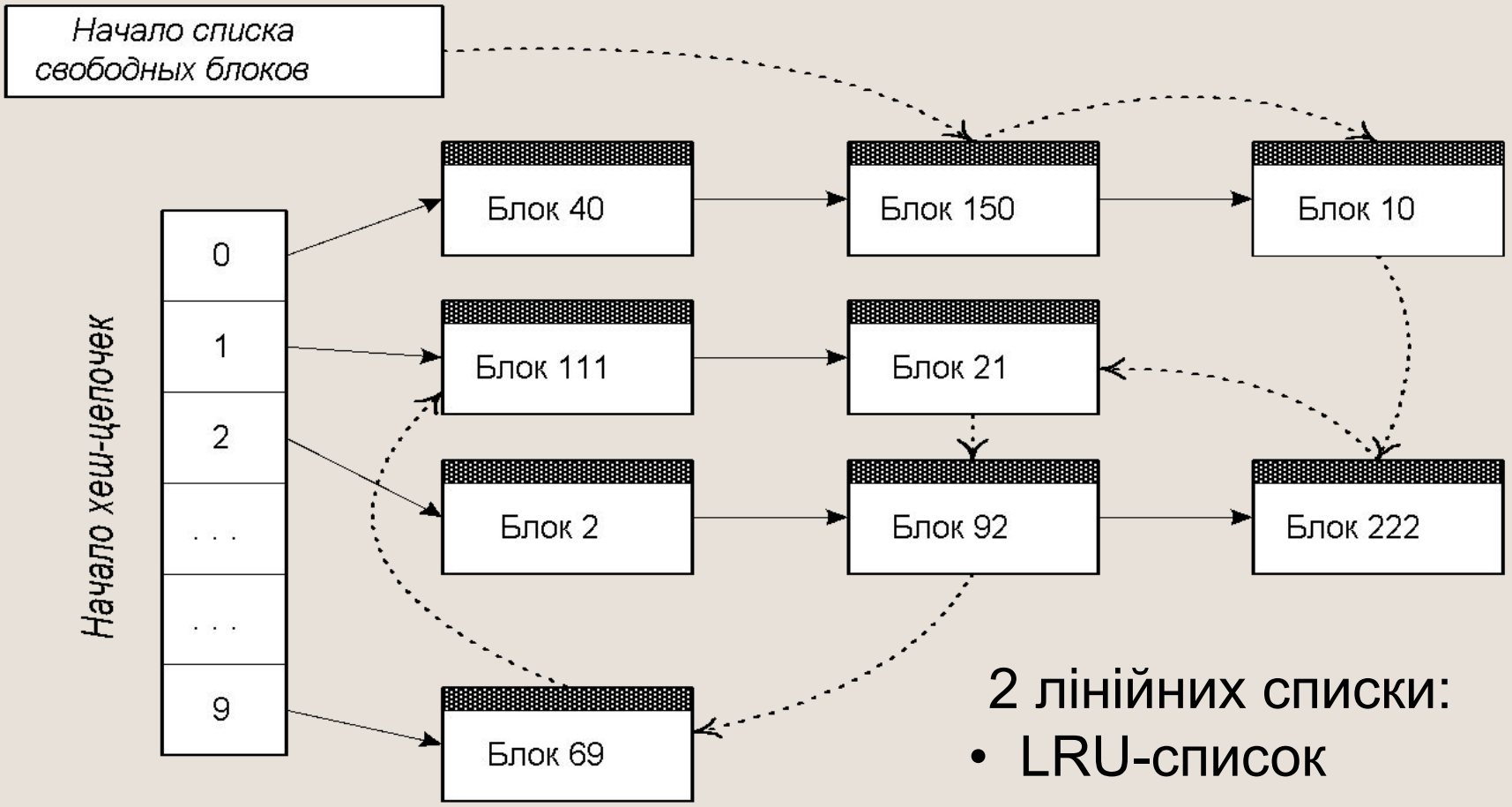
пошук необхідного блоку даних в (для цього система переглядає заголовки буферів) →

кеш складається з декількох сотень буферів →

час пошуку буде відчутний →

Вихід: оптимізація методів пошуку

Структура дискового кэша UNIX



2 лінійних списки:

- LRU-список
- «хеш-ланцюжок»

!!! Пошук скорочується ≈ в N разів

Драйвери пристроїв

Драйвер пристрою – системна програма, яка під управлінням ОС виконує всі операції з конкретним ПП

Драйвер – посередник між ОС і пристроєм.

Завдання драйвера:

- забезпечити можливість стандартного звернення до будь-якого пристрою, приховуючи від інших частин ОС специфічні особливості окремих пристроїв;
- досягти максимально ефективного використання всіх функціональних можливостей і особливостей конкретних пристроїв.

Всі драйвери стандартизувати не можна!

Два типи драйверів:

Тип	Загальні функції	Специфічні функції
Драйвери **для символьних пристроїв**	▪ читання даних ▪ запис даних ▪ ініціалізація пристрою (один раз, відразу після завантаження) ▪ відкриття і закриття пристрою ▪ . . .	функція «неруйнівного введення», тобто перевірки чергового символу . . .
Драйвери **для блочних пристроїв**		функції форматування, пошуку сектора . . .

Структура типового драйвера

1. Заголовок	інформація про драйвер і про керований пристрії: ім'я пристрою, тип пристрою, обсяг пам'яті на пристрої, адреси блоку стратегії і блоку переривань.
2. Блок стратегії	прийом заявок на виконання операції (заявка – стандартний запис, сформований ОС перед зверненням до драйвера) ведення черги заявок запуск операції та її завершення
3. Блок переривань	Виконує алгоритм В/В по перериваннях (система викликає цей блок, коли отримує сигнал переривання від пристрою)

Інші блоки (для різних пристроїв):

Блок ініціалізації

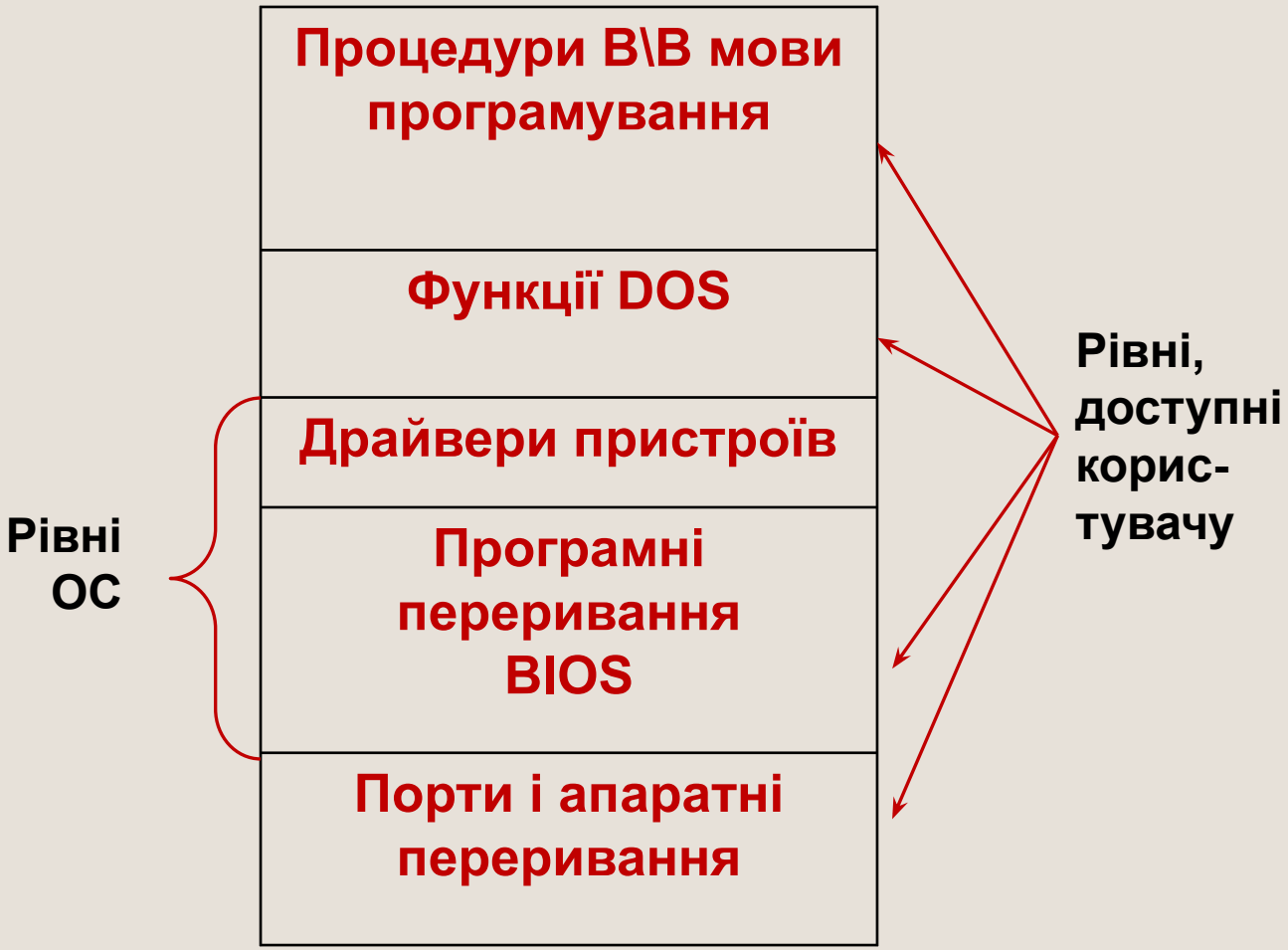
Блок зміни параметрів драйвера

. . .

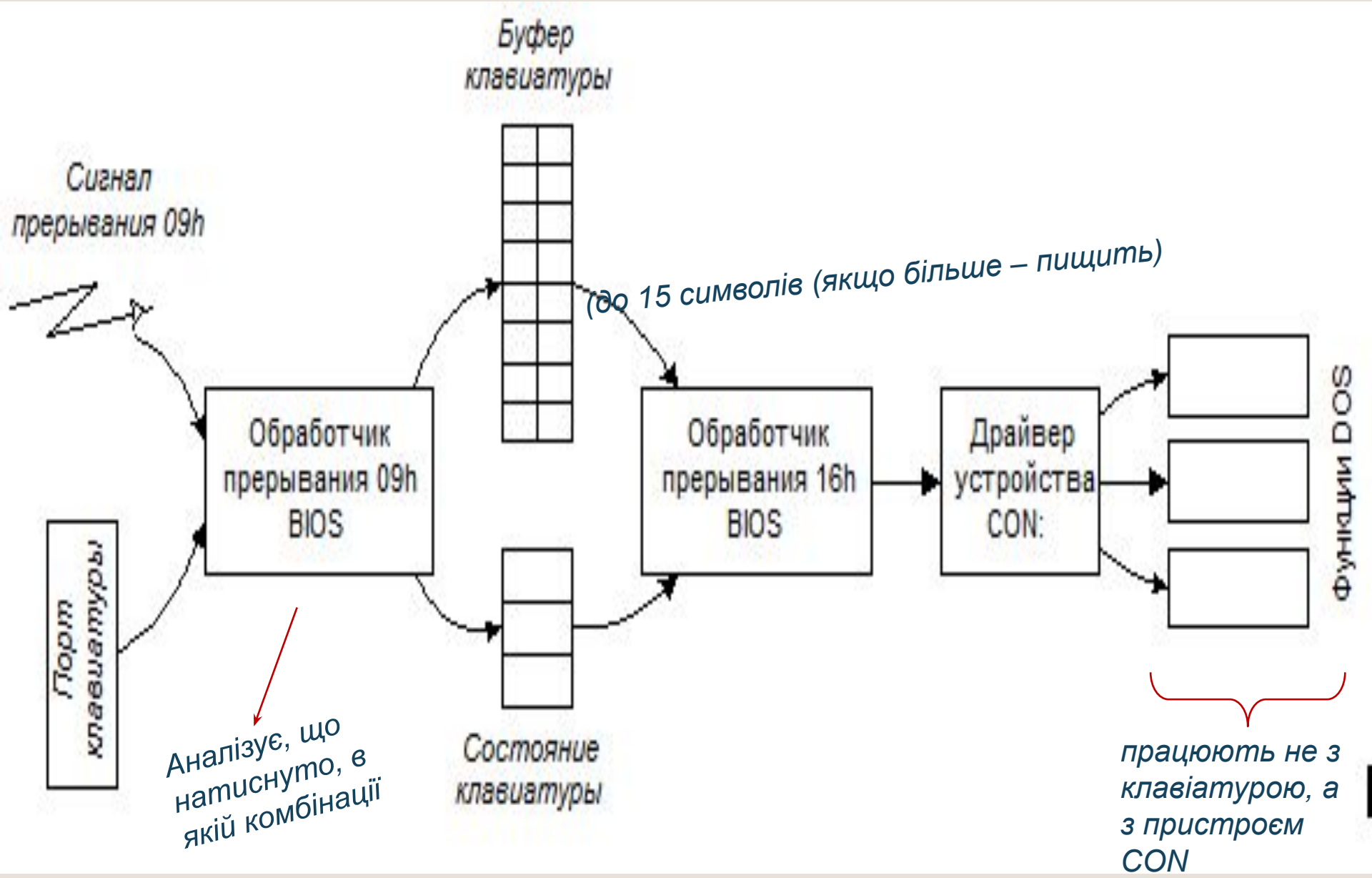
Керування пристроями у різних операційних системах

Керування пристроями в MS-DOS

Рівні доступу до пристроїв



Управління символьними пристроями (на прикладі клавіатури)



Самостійно!

- Керування пристроями в MS-DOS
- Керування пристроями в Windows
- Керування пристроями в Unix