

## Основи операційних систем

### УПРАВЛІННЯ ПАМ'ЯТТЮ

Пам'ять є найважливішим ресурсом, що вимагає ретельного управління мультизадачною операційною системою. Функціями ОС щодо керування пам'яттю є:



- 1) відстеження вільної і зайнятої пам'яті;
- 2) виділення пам'яті процесам і звільнення пам'яті при завершенні процесів;
- 3) витіснення процесів з оперативної пам'яті на диск, якщо розміри основної пам'яті недостатні для розміщення в ній усіх процесів, і повернення їх у оперативну пам'ять, коли в ній звільняється місце;
- 4) налагодження адрес програми на конкретну область фізичної пам'яті.

Усі ЕОМ використовують, у загальному випадку, два типи пам'яті: зовнішню — як правило, це дискові накопичувачі, і внутрішню — оперативну (див. стор. 5). Відповідно управління пам'яттю здійснюється на основі:



- алгоритмів без використання зовнішньої пам'яті;
- алгоритмів з використанням зовнішньої пам'яті, за якими відбувається переміщення процесів між оперативною пам'яттю і зовнішньою пам'яттю (дисковим накопичувачем).

Найпростішим способом розподілу пам'яті між кількома процесами є **статичне** керування пам'яттю. Усі процеси постійно розміщуються в оперативному запам'ятовуючому пристрої (ОЗП), при цьому кожному з них виділяється індивідуальний розділ, захищений від використання іншими процесами.

**Динамічне** керування пам'яттю забезпечує тимчасове вивантаження на диск припинених процесів, що дозволяє звільнити місце для поточного процесу. Основними технологіями динамічного керування є *підкачування (swap)* і *сторінкова* організація пам'яті, які будуть розглянуті пізніше.

Стосовно розташування процесів у основній пам'яті, виділяють два методи:

- метод суміжного розміщення;
- метод несуміжного розміщення.



Рис. 6. Суміжне розміщення (*contiguous allocation*) програми в основній пам'яті

Суміжне розміщення є найпростішим варіантом. В пам'яті, починаючи з деякої початкової адреси, виділяється одна неперервна ділянка адресного простору для розміщення програми (рис. 6).

При несуміжному розміщенні програма розбивається на кілька частин, що розташовуються в різних, необов'язково суміжних ділянках адресного простору (рис. 7).

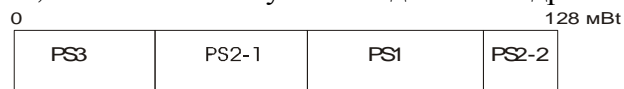


Рис. 7. Несуміжне розміщення (*non contiguous allocation*) програми в основній пам'яті

## Методи розподілу пам'яті без використання дискового простору

### Розподіл пам'яті фіксованими розділами

Найпростішим способом управління оперативною пам'яттю є поділ її на кілька розділів фіксованої величини. Для деяких ОС це могло бути виконано навіть вручну оператором під час старту системи чи під час її генерування. Чергова задача, що надійшла на виконання, розміщується або в загальну чергу (рис. 12), або в чергу для очікування звільнення деякого розділу.

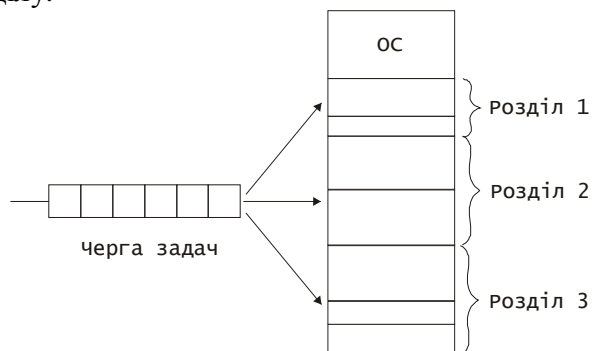


Рис. 12. Розподіл пам'яті фіксованими розділами

Підсистема управління пам'яттю в цьому випадку виконує задачі:

- 1) порівнюючи розмір програми, що надійшла на виконання і вільних розділів, вибирає придатний розділ,
- 2) здійснює завантаження програми і настроювання адрес.

При очевидній перевазі - простоті реалізації - даний метод має істотний недолік — неефективне використання наявної оперативної пам'яті: в кожному розділі може виконуватися тільки одна програма, рівень мультипрограмування заздалегідь обмежений числом розділів, не залежно від того, який розмір мають програми. Навіть якщо програма має невеликий обсяг, вона буде займати весь розділ, що приводить до неефективного використання пам'яті. З іншого боку, навіть якщо обсяг оперативної пам'яті машини дозволяє виконати деяку програму, розбивка пам'яті на розділи не дозволяє зробити цього.

### Розподіл пам'яті розділами змінного розміру

У цьому випадку пам'ять машини не поділяється заздалегідь на розділи. При завантаженні задачі виділяється необхідна їй пам'ять при умові, **що вистачає місця в пам'яті**, інакше задача не приймається на виконання і очікує у черзі вивільнення пам'яті. Після завершення задачі пам'ять звільняється, і на це місце може бути завантажена інша задача. Таким чином, у довільний момент часу оперативна пам'ять являє собою випадкову послідовність зайнятих і вільних ділянок (розділів) довільного розміру.

На рис. 13 показаний стан пам'яті в різні моменти часу при використанні динамічного розподілу. У момент  $t_0$  у пам'яті знаходиться тільки ОС, а до моменту  $t_1$  пам'ять розділена між 5 задачами, причому задача П4, завершуючись, залишає пам'ять. На вивільнене місце після задачі П4 завантажуються задача П6, що надійшла в момент  $t_3$ .

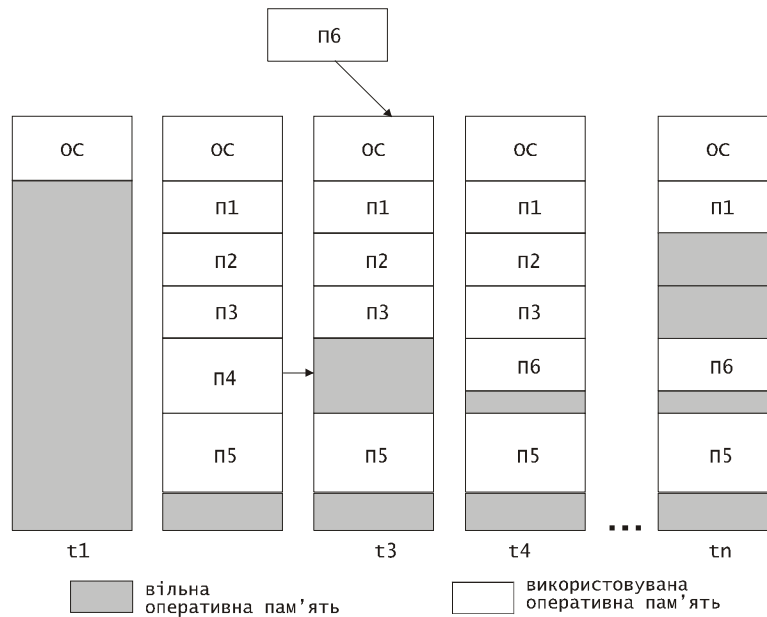


Рис. 13. Розподіл пам'яті з динамічними розділами

Вибір розділу для нової задачі може здійснюватися за різними правилами, наприклад, "перший, що зустрівся розділ достатнього розміру" чи "розділ, що має найменший достатній розмір" чи "розділ, що має найбільший достатній розмір". Усі ці правила мають свої переваги і недоліки.

При реалізації даного методу управління пам'яттю необхідно забезпечити у операційній системі:



- ведення таблиць вільних і зайнятих областей, у яких вказуються початкові адреси і розміри ділянок пам'яті,
- при надходженні нової задачі - аналіз запиту: перегляд таблиці вільних областей і вибір розділу, розмір якого достатній для розміщення задачі, що надійшла,
- завантаження задачі у виділений їй розділ і коректування таблиць вільних і зайнятих областей,
- після завершення задачі коректування таблиць вільних і зайнятих областей.

У порівнянні з методом розподілу пам'яті фіксованими розділами даний метод має набагато більшу гнучкість, але йому властивий недолік - *фрагментація пам'яті*.



*Фрагментація* - це наявність великого числа несуміжних ділянок вільної пам'яті малого розміру (фрагментів), що неможливо розмістити програму в жодній із них.

Якщо не вживати спеціальних заходів сумарний обсяг фрагментів може скласти значну величину, що набагато перевищує необхідний для виконання програми обсяг пам'яті.

### Переміщувані розділи

Одним з методів уникнення фрагментації є переміщення всіх зайнятих ділянок у бік старших або у бік молодших адрес, таким чином, щоб уся вільна пам'ять утворювала єдину вільну область (рис. 14).

На додаток до функцій, що виконує ОС при розподілі пам'яті зі змінними розділами, у даному випадку необхідно ще час від часу копіювати вміст розділів з одного місця пам'яті в інше, коректуючи таблиці вільних і зайнятих областей. Ця процедура називається "компресуванням". Компресування може виконуватися або при кожному завершенні задачі, або тоді, коли для нової задачі немає вільного розділу достатнього розміру. У

першому випадку потрібно менше обчислювальної роботи при коректуванні таблиць, а в другому – не так часто виконується процедура компресування.

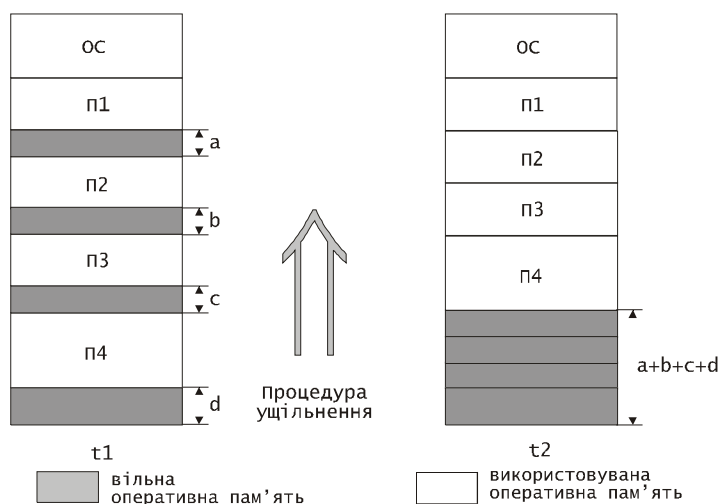


Рис. 14. Розподіл пам'яті переміщуваними розділами

Хоча процедура компресування і приводить до більш ефективного використання пам'яті, вона може вимагати значного часу, що часто переважає переваги даного методу.

### Методи розподілу пам'яті з використанням дискового простору

#### Поняття віртуальної пам'яті

Збільшення функціональності прикладного програмного забезпечення, при обмежених ресурсах оперативної пам'яті, поставило перед розробниками ОС проблему: розміщення в пам'яті програм, розмір яких перевищує вільну пам'ять. Одним із шляхів вирішення цієї проблеми був поділ програм на частини — *оверлей* (логічно завершенні блоки програми): 0-ий оверлей починав виконуватися першим, коли він закінчував своє виконання, завантажувалася інший оверлей. Всі оверлей зберігалися на диску і переміщалися між пам'яттю і диском засобами операційної системи, однак поділ програми на частини і планування черговості їх завантаження в оперативну пам'ять здійснювався програмістом. Такий підхід зменшує надійність виконання програми та операційної системи. Розвиток методів організації обчислювального процесу в цьому напрямку привів до появи методу, відомого як *віртуальна пам'ять*.



*Віртуальний ресурс* — ресурс, що надається користувачу чи програмі користувача таким, що має певні властивості, якими він у дійсності не володіє.

Наприклад:

- 1) користувачу може бути надана віртуальна оперативна пам'ять, розмір якої перевершує всю наявну в системі фізичну оперативну пам'ять;
- 2) користувач пише програми так, начебто в його розпорядженні є однорідна оперативна пам'ять великого обсягу, але в дійсності всі дані, які використовуються програмою, зберігаються на одному чи кількох різномірних запам'ятовуваних пристроях, як правило - дисках, і при необхідності частинами завантажуються у реальну пам'ять.

Таким чином, віртуальна пам'ять - це сукупність програмно-апаратних засобів, що дозволяють користувачам писати програми, розмір яких перевершує наявну оперативну пам'ять.

Операційна система при використанні віртуальної пам'яті повинна вирішувати задачі:

- розміщення даних на запам'ятовуючих пристроях різного типу, наприклад, частина програми в оперативній пам'яті, а частина на диску;
- переміщення, при необхідності, даних між запам'ятовуючими пристроями різного типу, наприклад, завантаження потрібної частини програми з диска в оперативну пам'ять;
- перетворення віртуальних адрес у фізичні.

Усі ці дії виконуються *автоматично*, тобто механізм віртуальної пам'яті є прозорим стосовно користувача та його програм. Найбільш розповсюдженими реалізаціями віртуальної пам'яті є: сторінковий, сегментний, сторінково-сегментний розподіл пам'яті, свопінг.

### Сторінковий розподіл

Віртуальний адресний простір кожного процесу поділяється на частини однакового, фіксованого для даної системи розміру, що називають віртуальними сторінками. Вся оперативна пам'ять обчислювальної машини також поділяється на частини такого ж розміру, що називають фізичними сторінками (чи блоками). Розмір сторінки вибирається рівним степеням двійки: 512, 1024 і т.д., з метою спрощення алгоритму перетворення адрес. При завантаженні процесу частина його віртуальних сторінок розміщується в оперативній пам'яті, а інші - у дисковій. Суміжні віртуальні сторінки не обов'язково розташовуються в суміжних фізичних сторінках.

*Сторінкова організація пам'яті* — апаратно підтримується процесорами, починаючи з i80386. В оперативній пам'яті знаходяться тільки активні сторінки, а інші розташовані на диску, віртуальній пам'яті. При доступу процесу до пам'яті віртуальна адреса перетворюється модулем управління пам'яттю операційної системи у фізичну адресу сторінки. Доступ здійснюється відразу, якщо сторінка знаходиться в оперативній пам'яті. Якщо дана сторінка знаходиться на диску, то вона спочатку завантажується в оперативну пам'ять. При цьому одна з активних сторінок витісняється на диск (рис. 15).

При завантаженні операційна система створює для кожного процесу інформаційну структуру — таблицю сторінок, у якій встановлюється відповідність між номерами віртуальних і фізичних сторінок, завантажених в оперативну пам'ять, або вказується, що віртуальна сторінка вивантажена на диск. Крім того, у таблиці сторінок міститься управляюча інформація: ознака модифікації сторінки; ознака заборони вивантаження (вивантаження певних сторінок може бути заборонене); ознака звертання до сторінки (використовується для підрахунку числа звертань за визначений період часу) та інші дані, що використовуються механізмом віртуальної пам'яті.

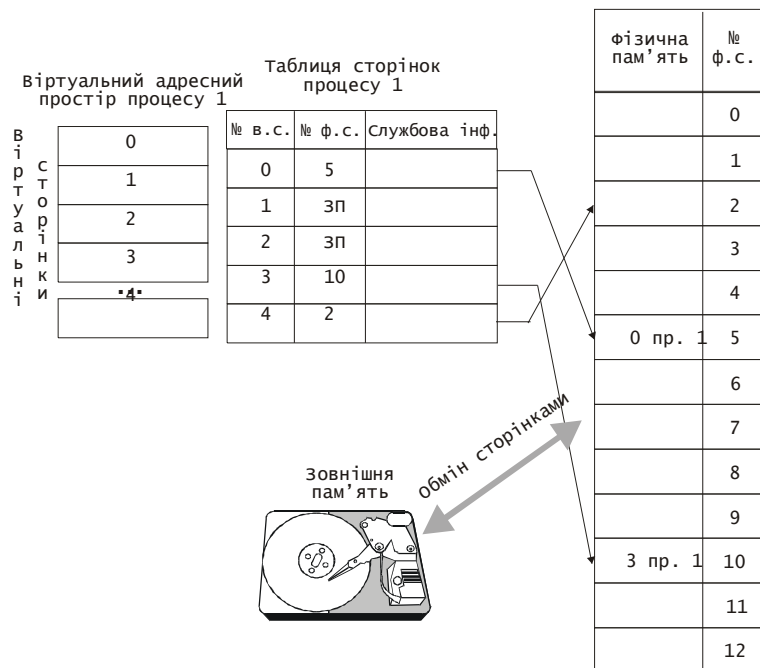


Рис. 15. Сторінковий розподіл пам'яті

При активізації чергового процесу в спеціальний регістр процесора завантажується адреса таблиці сторінок даного процесу. При кожному звертанні до пам'яті відбувається зчитування з таблиці сторінок інформації про віртуальну сторінку, до якої відбулося звертання, якщо дана віртуальна сторінка знаходиться в оперативній пам'яті, то виконується перетворення віртуальної адреси у фізичну, якщо ж потрібна віртуальна сторінка в даний момент вивантажена на диск, відпрацьовується алгоритм сторінкового переривання: процес, що виконується, переводиться в стан очікування, і активізується процес опрацювання сторінкового переривання для пошуку на диску необхідної віртуальної сторінки і завантаження її в оперативну пам'ять. Якщо в пам'яті є вільна фізична сторінка, то завантаження виконується негайно, якщо ж вільних сторінок немає, то вирішується питання, яку сторінку можна вивантажити з оперативної пам'яті.

У даній ситуації може бути використано кілька різних критеріїв вибору, найбільш використовувані з них:



- сторінка, яка найдовше не використовувалася;
- перша сторінка, що трапилася;
- сторінка, до якої останнім часом було найменше звертань.

У деяких системах використовується поняття *робочої множини* сторінок.



Робоча множина визначається для кожного процесу і являє собою перелік найбільш часто використовуваних сторінок, що повинні постійно знаходитися в оперативній пам'яті і тому не підлягають вивантаженню.

Рішення про вивантаження сторінки на диск приймається на основі аналізу атрибуту модифікації обраної сторінки. Якщо сторінка, що вивантажується, з моменту завантаження була модифікована, то її нова версія повинна бути переписана на диск. Якщо ні, то вона може бути просто знищена, тобто відповідна фізична сторінка стає вільною.

На продуктивність системи зі сторінковою організацією пам'яті впливають часові витрати, пов'язані з опрацюванням сторінкових переривань і перетворенням віртуальної адреси у фізичну. Якщо сторінкові переривання виникають часто, то система може витрачати велику кількість часу на свопінг сторінок. Щоб зменшити частоту сторінкових переривань, доцільно збільшувати розмір сторінки. Крім того, збільшення розміру

сторінки зменшує розмір таблиці сторінок, а значить зменшує витрати пам'яті. З іншого боку, якщо сторінка велика, значить велика і фіктивна область в останній віртуальній сторінці кожної програми, у середньому для кожної програми втрачається половина обсягу сторінки, тому сумарний обсяг, при великій сторінці, може скласти значну величину. Час перетворення віртуальної адреси у фізичну у значній мірі визначається часом доступу до таблиці сторінок. У зв'язку з цим таблицю сторінок розміщують в „швидких” запам'ятовуючих пристроях: це може бути, наприклад, набір спеціальних регістрів чи пам'ять з асоціативним пошуком та кешуванням.

Сторінковий розподіл пам'яті може бути реалізований у спрощеному варіанті, без вивантаження сторінок на диск. У цьому випадку всі віртуальні сторінки всіх процесів постійно знаходяться в оперативній пам'яті. Такий варіант сторінкової організації хоча і не надає користувачу віртуальної пам'яті, але майже виключає фрагментацію за рахунок того, що програма може завантажуватися в несуміжні області, а також того, що при завантаженні віртуальних сторінок ніколи не утворюються залишки.

### Сегментний розподіл

При сторінковій організації віртуальний адресний простір процесу поділяється механічно на рівні частини, що не дозволяє диференціювати способи доступу до різних частин програми (сегментів). Наприклад, можна заборонити звертатися з операціями запису і читання в кодовий сегмент програми, а для сегменту даних дозволити тільки читання. Крім того, розбивка програми на "осмислені" частини робить принципово можливим спільне використання одного сегмента кількома процесами.

Наприклад:

якщо два процеси використовують одну математичну підпрограму, то в оперативну пам'ять може бути завантажена тільки одна копія цієї підпрограми.

Віртуальний адресний простір процесу поділяється на сегменти (рис.16), розмір яких визначається програмістом.

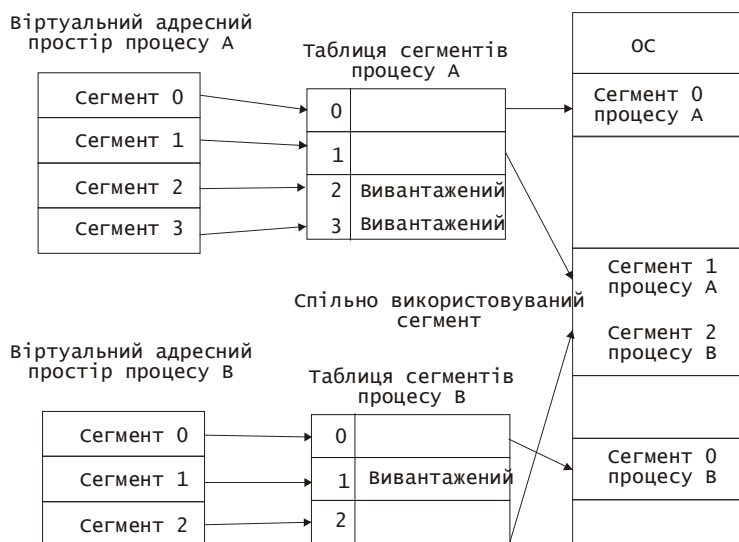


Рис. 16. Сегментний розподіл пам'яті

Окремий сегмент може являти собою підпрограму, масив даних і т.п. Іноді сегментація програми виконується за замовчуванням компілятором. При завантаженні процесу частина сегментів розміщується в оперативній пам'яті (при цьому для кожного з цих сегментів операційна система шукає придатну ділянку вільної пам'яті), а частина сегментів розміщується в дисковій пам'яті. Сегменти однієї програми можуть займати в оперативній пам'яті несуміжні ділянки. Під час завантаження система створює таблицю

сегментів процесу (аналогічну таблиці сторінок), у якій для кожного сегмента вказується початкова фізична адреса сегмента в оперативній пам'яті, розмір сегмента, правила доступу, ознака модифікації, ознака звертання до даного сегмента за останній проміжок часу та деяка інша інформація. Якщо віртуальні адресні простори кількох процесів включають той самий сегмент, то в таблицях сегментів цих процесів робляться посилання на ту саму ділянку оперативної пам'яті, у якій даний сегмент завантажується в єдиному екземплярі.

Система із сегментною організацією функціонує аналогічно системі зі сторінковою організацією: час від часу відбуваються переривання, пов'язані з відсутністю потрібних сегментів у пам'яті, при необхідності звільнення пам'яті деякі сегменти вивантажуються, при кожному звертанні до оперативної пам'яті виконується перетворення віртуальної адреси у фізичну. Крім того, при звертанні до пам'яті перевіряються права доступу до даного сегмента. Віртуальна адреса при сегментній організації пам'яті може бути представлена парою  $(g, s)$ , де  $g$  - номер сегмента, а  $s$  - зміщення у сегменті. Фізичну адресу отримують шляхом додавання початкової фізичної адреси сегмента, знайденого в таблиці сегментів за номером  $g$ , і зміщенням  $s$ .

Недоліком даного методу розподілу пам'яті є фрагментація на рівні сегментів і більш повільне в порівнянні зі сторінковою організацією визначення фізичної адреси.

### Сторінково-сегментний розподіл

Як видно з назви, даний метод являє собою комбінацію сторінкового і сегментного розподілу пам'яті і, внаслідок цього, об'єднує в собі переваги обох підходів. Віртуальний простір процесу поділяється на сегменти, а кожен сегмент у свою чергу поділяється на віртуальні сторінки, що нумеруються в межах сегмента. Оперативна пам'ять поділяється на фізичні сторінки. Завантаження процесу виконується операційною системою посторінково, при цьому частина сторінок розміщується в оперативній пам'яті, а частина в дисковій. Для кожного сегменту створюється своя таблиця сторінок, структура якої збігається зі структурою таблиці сторінок, використовуваної при сторінковому розподілі. Для кожного процесу створюється таблиця сегментів, у якій вказуються адреси таблиць сторінок для всіх сегментів даного процесу. Адреса таблиці сегментів завантажується в спеціальний реєстр процесора, коли активізується відповідний процес.

### Свопінг

Залежно від характеру виконуваних задач одна частина процесів може потребувати досить часто процесорного часу, а інша частина періодично, при настанні певних подій, наприклад виконання сервера для відправлення чи отримання електронної пошти. Одним із методів вивільнення оперативної пам'яті, для процесів, що потребують негайного виконання є свопінг.



**Свопінг** — тимчасове повне вивантаження процесу, що знаходяться в стані чекання, у зовнішню пам'ять.

Планувальник операційної системи не виключає перенесені процесу у зовнішню пам'ять із свого розгляду, і при настанні умов активізації процесу, що знаходиться в області свопінга на диску, цей процес переміщується в оперативну пам'ять. Якщо вільного місця в оперативній пам'яті не вистачає, то вивантажується інший процес.

При свопінгу, на відміну від розглянутих раніше методів реалізації віртуальної пам'яті, процес переміщується між пам'яттю і диском, і протягом деякого часу процес може бути відсутнім у оперативній пам'яті.

## Ієрархія запам'ятовуючих пристроїв. Кешування даних

Всю пам'ять обчислювальної машини можна представити у вигляді дерева запам'ятовуючих пристроїв рис. 17, що відрізняються середнім часом доступу і вартістю збереження даних у розрахунку на один біт. Для ефективного використання обчислювальних ресурсів необхідна наявність швидкої пам'яті, одночасно з доступною вартістю. Кеш-пам'ять представляє деяке компромісне вирішення цієї проблеми.

Пам'ять обчислювальної машини	
	внутрішні реєстри процесора,
	надоперативна пам'ять
	оперативна пам'ять
	пам'ять на дискових накопичувачах

Рис. 17. Ієрархія запам'ятовуючих пристроїв



*Кеш-пам'ять* — спосіб організації спільного функціонування двох типів запам'ятовуючих пристроїв, що відрізняються часом доступу, вартістю збереження даних, для зменшення середнього часу доступу до даних за рахунок динамічного копіювання в "швидкий" запам'ятовуючий пристрій найбільш часто використовуваної інформації з "повільного" запам'ятовуючого пристрою.

Кеш-пам'яттю часто називають не тільки спосіб організації роботи двох типів запам'ятовуючих пристроїв, але й пристрій швидкого запам'ятовування. Він коштує дорожче і, як правило, має порівняно невеликий обсяг.

Розглянемо окремий випадок використання кеш-пам'яті для зменшення середнього часу доступу до даних, що зберігається у зовнішній пам'яті. Для цього між процесором і зовнішньою пам'яттю розміщується швидкий запам'ятовуючий пристрій, кеш-пам'ять. Вміст кеш-пам'яті являє собою сукупність записів про всі завантажені в неї елементи даних. Кожен запис про елемент даних містить у собі адресу цього елемента даних в оперативній пам'яті, і управляючу інформацію: ознака модифікації й ознака звертання до даних за деякий останній проміжок часу.

У системах, обладнаних кеш-пам'яттю, кожен запит до оперативної пам'яті виконується у відповідності з наступним алгоритмом:



- 1) переглядається вміст кеш-пам'яті з метою визначення, чи не знаходяться потрібні дані в кеш-пам'яті; кеш-пам'ять не адресується, тому пошук потрібних даних здійснюється по вмісту - значенню поля "адреси в оперативній пам'яті", узятому з запиту;
- 2) якщо дані виявляються в кеш-пам'яті, то вони зчитуються з неї, результат передається процесору;
- 3) якщо потрібних даних немає, то вони копіюються з оперативної пам'яті в кеш-пам'ять, і результат виконання запиту передається процесору;

При копіюванні даних може виявитися, що в кеш-пам'яті відсутнє вільне місце, тоді, для витіснення з кеш-пам'яті обираються дані, до яких в останній період було найменше звертань. Якщо витіснені дані були модифіковані за час перебування в кеш-пам'яті, то вони переміщуються в оперативну пам'ять. Якщо ж ці дані не були модифіковані, то їх місце в кеш-пам'яті звільняється.

На практиці у кеш-пам'ять зчитується не один елемент даних, до якого відбулося звертання, а цілий блок даних, це збільшує ймовірність так званого "влучання у кеш", тобто перебування потрібних даних у кеш-пам'яті.

Покажемо, як середній час доступу до даних залежить від ймовірності влучання у кеш. Нехай маємо основний запам'ятовуючий пристрій із середнім часом доступу до даних  $t_1$  і кеш-пам'ять, що має час доступу  $t_2$ , відповідно  $t_2 < t_1$ . Позначимо через  $t$  середній час доступу до даних у системі з кеш-пам'яттю, а через  $p$  - ймовірність влучання у кеш. Відповідно до формули повної ймовірності:

$$t = t_1((1 - p) + t_2(1-p))$$

З неї видно, що середній час доступу до даних у системі з кеш-пам'яттю лінійно залежить від ймовірності влучання у кеш і змінюється від середнього часу доступу до основного запам'ятовуючого пристрою (при  $p=0$ ) до середнього часу доступу безпосередньо до кеш-пам'яті (при  $p=1$ ). У реальних системах ймовірність влучання у кеш складає приблизно 0,9. Високе значення ймовірності перебування даних у кеш-пам'яті пов'язано з наявністю в даних об'єктивних властивостей: просторової і часової локальності.

*Просторова локальність.* Якщо відбулося звертання до деякої адреси, то існує досить велика ймовірність, що найближчим часом відбудеться звертання до сусідніх адрес.

*Часова локальність.* Якщо відбулося звертання до деякої адреси, то існує досить велика ймовірність, що наступне звертання до цієї ж адреси відбудеться найближчим часом.

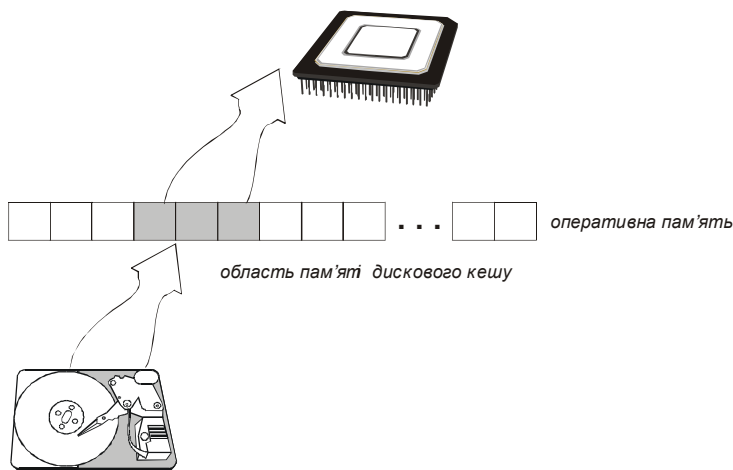


Рис. 18. Організація дискової кеш-пам'яті

Усі попередні викладки справедливі і для інших пар запам'ятовуючих пристроїв, наприклад, оперативна пам'ять-зовнішня пам'ять, у цьому випадку зменшується середній час доступу до даних, розташованих на диску, і функцію кеш-пам'яті виконує буфер в оперативній пам'яті (рис. 18).

### Зовнішня пам'ять




Зовнішня пам'ять - це сукупність запам'ятовуючих пристроїв і носіїв інформації, використовується для довгострокового зберігання даних і має властивості:

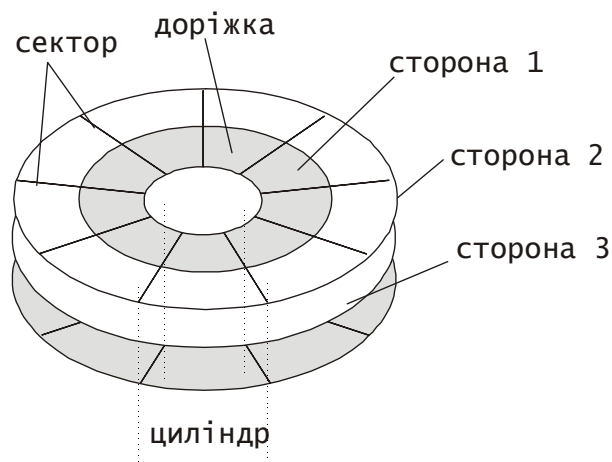
- довгострокового збереження інформації на носіїві при відключеному живленні комп'ютера.
- збереження досить великих обсягів даних.
- зберезувальну інформацію, перед використанням завантажують в оперативну пам'ять комп'ютера.

Інформація зберігається на фізичному носіїві довгострокової пам'яті у формі, обумовленій конкретним апаратним пристроєм. Для операційної системи при цьому не важливо, у якій саме фізичній формі зберігаються дані на носіїві. Незалежно від цієї форми ОС використовує стандартизований набір команд для управління апаратним пристроєм. На фізичному рівні робота з накопичувачами відбувається у всіх файлових системах однаково на рівні команд:

- 1) підвести зчитуючий/записуючий елемент до вказаного місця (сектора);
- 2) прочитати дані з вказаного місця (сектора);
- 3) записати дані у вказане місце (сектор).

Найбільш розповсюдженими видами зовнішньої пам'яті є дискові магнітні й оптичні накопичувачі. Поверхня диску розглядається як тривимірна матриця, вимірами якої є номери **поверхні, циліндра та сектору** (рис. 19).

-  **Сектор** — найменша фізична одиниця збереження даних. Кожний сектор має свою адресу, номер. Нумерація проводиться послідовно.
-  **Доріжка** — сектори, що належать одній поверхні і знаходяться на однаковій відстані від осі обертання.
-  **Циліндр** — сукупність усіх доріжок, що належать різним поверхням, і знаходяться на однаковій відстані від осі обертання.



*Рис.19. Структура дискового накопичувача*

Фізичне збереження і доступ до даних на зовнішньому носіїв забезпечується:

- 1) спеціальним електронним обладнанням — контролер дискових накопичувачів, з стандартним інтерфейсом обміну даними.
- 2) частиною операційної системи для логічної організації забезпечення управління даними (рис.20).

Логічна організація даних підтримується операційною системою за допомогою спеціальних програм і структур даних<sup>1</sup>, що дозволяє зберігати дані у вигляді єдиного логічного блоку, файлу. Несумісність різних операційних систем при роботі з однотипними носіями визначається саме різними принципами логічної організації та збереження даних.

<sup>1</sup> Структура даних — сукупність елементів даних різних типів, об'єднаних у єдиний логічний блок.

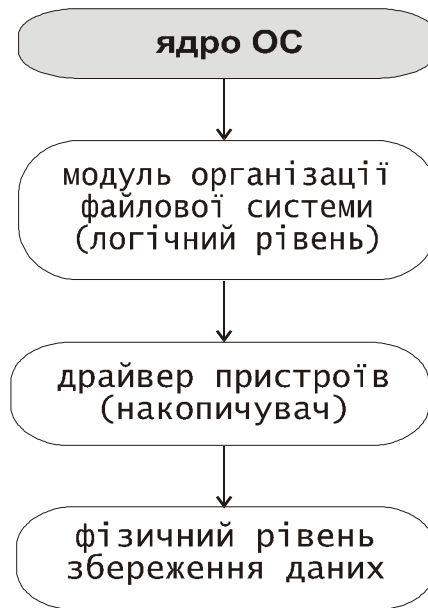



Рис.20. Організація файлової системи

 **Файл** — це поійменована область зовнішньої пам'яті, в яку можна записувати і з якої можна зчитувати дані, логічна одиниця збереження інформації на носієві.

Файли об'єднуються в групи — каталоги, що можуть містити підкаталоги утворюючи ієрархічну деревоподібну структуру з одним коренем (рис. 21.).

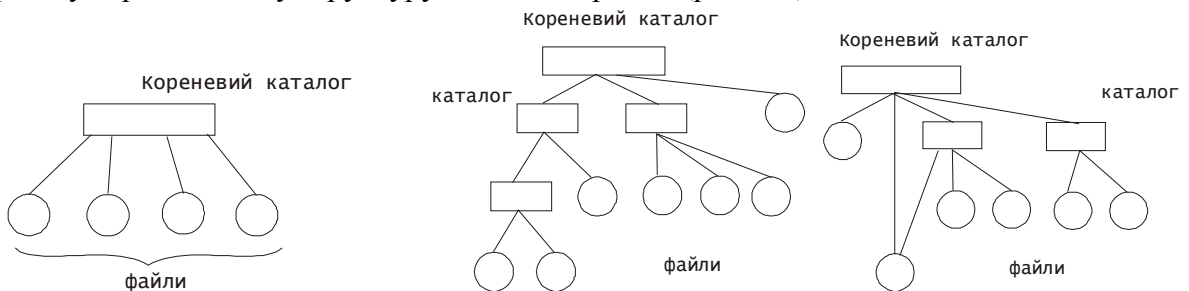




Рис. 21. Способи логічної організації збереження даних

 **Файлова система** — частина операційної системи, призначення якої полягає у забезпеченні програмно-апаратного інтерфейсу при роботі з даними, що зберігаються в ЗЗП.

 У широкому розумінні поняття "файлова система" включає:

- сукупність усіх файлів на диску;
- набори структур даних, використовуваних для керування файлами: каталоги файлів, дескриптори файлів, таблиці розподілу вільного і зайнятого простору на диску;
- комплекс системних програмних засобів, що реалізують функції управління файлами, зокрема: створення, вилучення, читання, запису, іменування, пошуку та інше.

Прикладне програмне забезпечення не звертається напряму до фізичного рівня, а робота з накопичувачем відбувається за допомогою викликів функцій файлової системи, що забезпечують виконання високорівневих операцій, таких як відкриття файлу, запис, зчитування даних та інше.

## Типи файлових систем

Слід зазначити, що не існує єдиного стандарту на файлову систему. Для кожної ОС, відповідно до виконуваних задач, існують “свої” реалізації логічного рівня для роботи з даними, що розрізняються способами організації структур даних.

*FAT (File Allocation Table) або FAT16* — файлова система операційної системи *Dos*. Розділ (*volume*) *FAT* займає цілу дискету або розділ жорсткого диску.

*VFat, Fat32* — модифіковані версії *FAT16*, для операційних систем сімейства *Windows 9x/ME*.

*NTFS* файлова система для *Windows NT*, розроблялася як надійна, стійка до апаратних помилок файлова системи.

*UFS (Unix File System)* — перша файлова система для операційної системи *UNIX*, всі сучасні версії походять від неї.

*Ext2* — достатньо функціонально розвинена файлова система з сімейства сумісних з *Linux*. На даний момент вважається найбільш популярною системою. Вона розроблена з врахуванням сумісності з наступними версіями, тому для установки нової версії коду системи не потрібно встановлювати її заново.

*Sysv* — файлові системи *System V/386, Coherent i Xenix*.

*Iso9660* — стандартна файлова система для *CD-ROM*. Досить популярне розширення стандарту *CD-ROM*, для автоматичної підтримки імен файлів нестандартної довжини.

*Nfs* — мережева файлова система, що забезпечує спільне використання однієї файлової системи кількома комп'ютерами.

*Hpfs* — файлова система, розроблена для *OS/2*.

*Minix* — одна з перших файлових систем, досить обмежена за своїми можливостями (відсутні деякі параметри, довжина імені файлу обмежена 30-ма символами) і доступних об'ємом (максимум 64 Мб на одну файлову систему).

## Виконання операцій з файлами

Для виконання операцій з файлами, копіювання, вилучення, перейменування, навігації тощо, усі операційні системи забезпечують необхідний мінімум команд .

Деякі команди для роботи з файловою системою  
DOS, ОС Windows XX (командний рядок), Linux

	<b>Windows 95</b>	<b>Linux</b>
запуск програми на виконання	file name	file name
копіювати	copy	cp
перенести	move	rn
вилучити	del	mv
пошук файлу (файлів)	find	find
створити каталог	md	mk
вилучити каталог	del	rm
ініціалізація файлової системи	format	mkfs

Для спрощення виконання операцій з файлами найчастіше використовують спеціальні програми — файл-менеджери, що забезпечують інтуїтивне виконання операцій з файлами у візуальному режимі (рис. 22,23).

У графічних середовищах використовуються графічні файл-менеджери, що представляють файли у вигляді піктограм, а каталоги у виді папок (рис.23).



Операції з файлами виконуються як операції над їх зображеннями, перенесення зображення файлу з однієї папки в іншу формує для операційної системи команду виконати копіювання вибраного файлу в новий каталог.

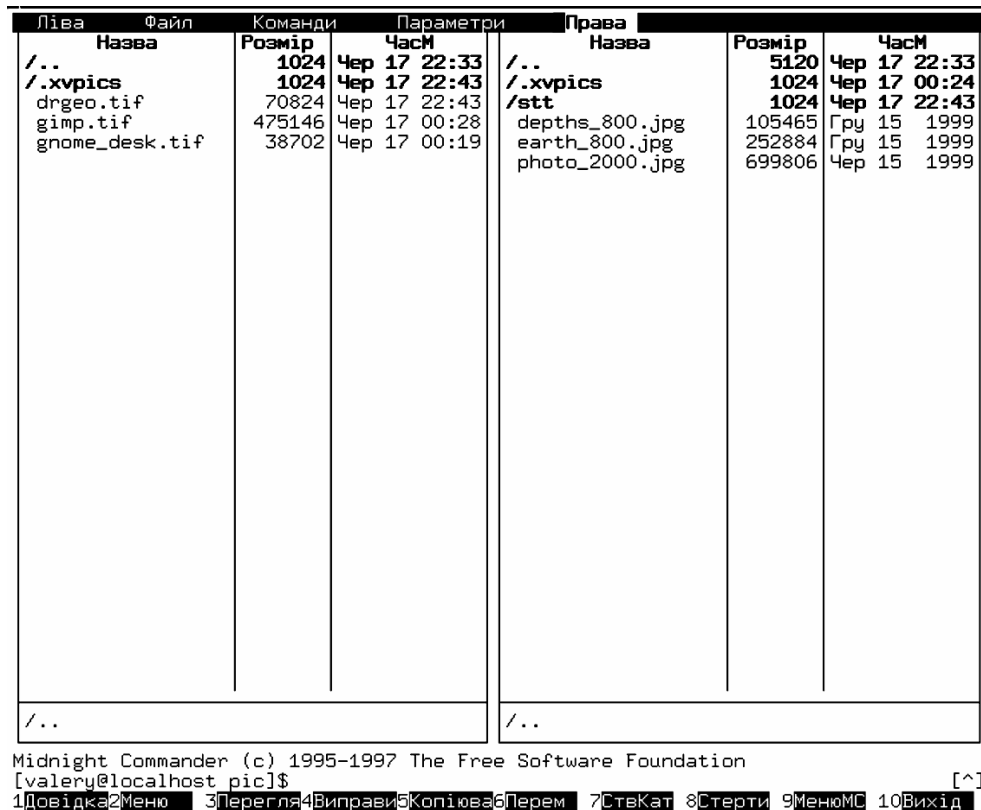


Рис. 22. Файл менеджер текстового режиму ОС Linux — mc

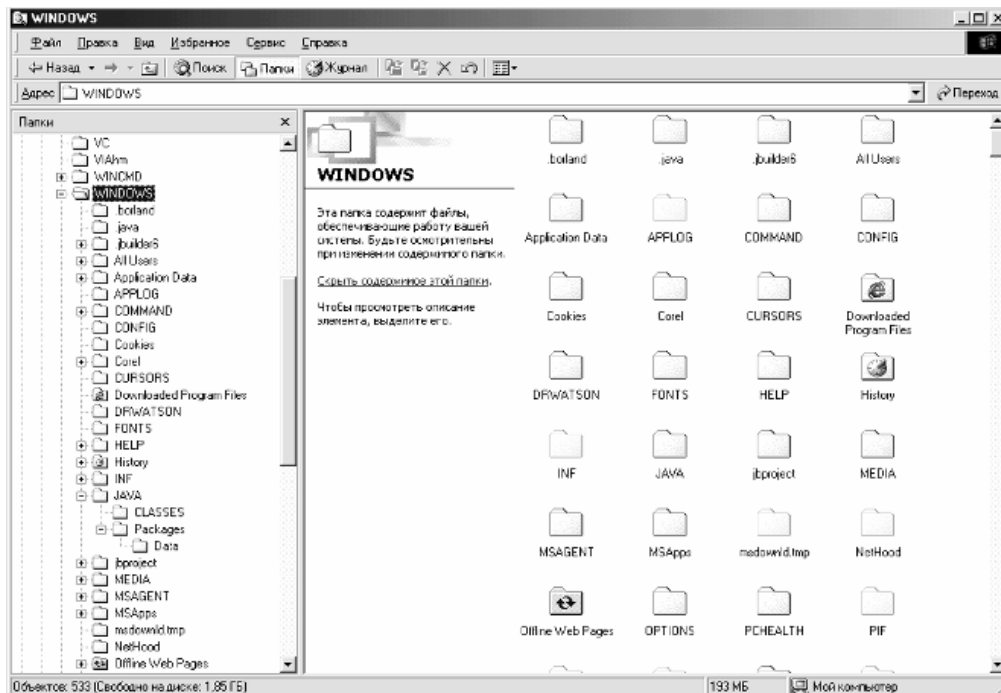


Рис. 23. Файл менеджер графічного середовища користувача ОС Windows — Провідник

## УПРАВЛІННЯ УВЕДЕННЯМ/ВИВЕДЕННЯМ

Однією з головних функцій ОС є управління пристроями введення-виведення комп'ютера. ОС повинна передавати пристроям команди, перехоплювати переривання й опрацьовувати помилки; вона також повинна забезпечувати незалежний інтерфейс між пристроями й іншою частиною системи.

### Фізична організація пристроїв введення-виведення

Пристрої введення-виведення поділяються на два типи: *блок-орієнтовані та байт-орієнтовані* пристрої.



*Блок-орієнтовані пристрої* передають та отримують інформацію блоками фіксованого розміру. Найпоширеніший блок-орієнтований пристрій – дисковий накопичувач.



*Байт-орієнтовані пристрої* передають або приймають інформацію як послідовність байт. Прикладом байт-орієнтованих пристроїв є термінали, принтери, мережеві адаптери.

Зовнішній пристрій як правило містить механічні та електронні компоненти:

— механічний компонент — власне пристрій.

— електронний компонент називається контролером пристрою або адаптером;

Деякі контролери можуть управляти кількома пристроями, якщо інтерфейс між контролером і пристроєм стандартизований, то незалежні виробники можуть випускати сумісні контролери і пристрої.

Операційна система звичайно має справу не з пристроєм, а з контролером. Контролер, як правило, виконує прості функції, наприклад, перетворює потік бітів у блоки, що складаються з байтів, і здійснює контроль і виправлення помилок. Кожний контролер має кілька регістрів, що використовуються для обміну даними та управління. У деяких комп'ютерах ці регістри є частиною фізичного адресного простору, тому для таких систем немає спеціальних операцій введення-виведення, в інших — адреси регістрів введення-виведення, їх називають портами, утворюють власний адресний простір, що використовується спеціальними операціями зчитування або запису даних в порт (команди *IN* і *OUT* для процесорів *i86*).

ОС виконує операції введення-виведення, записуючи команди в регістри контролера. Наприклад, контролер гнучкого диска *IBM PC* приймає 15 команд, таких як *READ*, *WRITE*, *SEEK*, *FORMAT* і т.д. Коли команда прийнята, процесор звільнює контролер і виконує інші потоки команд. При завершенні команди контролер генерує переривання для передачі управління операційній системі, яка повинна перевірити результати операції.

### Організація програмного забезпечення введення-виведення

Основна ідея організації програмного забезпечення введення-виведення полягає у виділенні кількох рівнів, причому нижні рівні забезпечують маскуванню особливостей апаратури від верхніх, а ті, у свою чергу, забезпечують інтерфейс для використання програмним забезпеченням. Робота програми не повинна залежати від того, чи дані отримуються з гнучкого чи жорсткого диску.

Іншим важливим питанням при організації введення-виведення є опрацювання помилок. Опрацювання помилок необхідно проводити якнайближче до обладнання: якщо контролер виявляє помилку читання, то він повинен спробувати її відкоригувати і у випадку невдачі виправленням помилок повинен зайнятися драйвер пристрою. І тільки якщо нижній рівень не може виправити помилку, він повідомляє верхній рівень про помилку. Багато помилок можуть усуватися при повторних спробах виконання операцій введення-виведення, наприклад, помилки, викликані фізичними пошкодженнями робочих поверхонь магнітних чи оптичних дисків тощо.

Ще одне ключове питання - це використання блокуючих (синхронних) і неблокуючих (асинхронних) способів обміну.

Більшість операцій фізичного уведення-виведення виконується асинхронно — процесор починає передавання даних і перемикається на іншу задачу, поки не настане переривання завершення операції.

Програми користувача набагато легше писати, якщо операції уведення-виведення блокуючі - після команди *READ* програма автоматично припиняється доти, поки дані не потраплять у буфер програми. ОС виконує операції уведення-виведення асинхронно, але представляє їх для програм користувача у синхронній формі.

Остання проблема полягає в тому, що одні пристрої можуть спільно використовуватися у режимі розподілу часу, а інші – виділяються на час виконання певного завдання. Диски - це спільно використовувані пристрої, тому що одночасний доступ кількох користувачів до диска не являє собою проблему. Принтери - це виділені пристрої, не можна змішувати рядки, що друкуються різними користувачами.

Наявність виділених пристроїв створює для операційної системи деякі проблеми, що вирішуються розподілом програмного забезпечення уведення-виведення на чотири рівні :

- 1) опрацювання переривань
- 2) драйвери пристроїв
- 3) незалежний від пристроїв рівень операційної системи
- 4) рівень користувача.

### Опрацювання переривань

Одним з основних механізмів забезпечення роботи зовнішніх пристроїв є використання переривань.



*Переривання* – спеціальний сигнал, або команда, що подається процесору для негайного його перемикавання на опрацювання події (опрацювання переривання), за певним алгоритмом, що сформувала переривання.

### Драйвери пристроїв

Для управління фізичними пристроями використовуються спеціальне програмне забезпечення, драйвери пристроїв. Кожний драйвер керує пристроями одного типу або одного класу. В операційній системі тільки драйверу пристрою відомо про конкретні особливості якого-небудь апаратного обладнання з яким він працює: наприклад, драйвер дискового накопичувача має справу з доріжками, секторами, циліндрами, позиціонуванням головки та інше, що забезпечує коректну роботу накопичувача.

Можна уявити, що драйвери пристроїв є нижчим шаром, над яким розміщено програми (ОС, програми користувача), а під яким — контролери пристроїв. При отриманні вказівки виконати операцію, драйвер пристрою аналізує зайнятість пристрою. Якщо пристрій вільний під час надходження запиту, то він починає виконувати вказівку негайно, якщо зайнятий виконанням іншого запиту, то запит ставиться у чергу і буде виконаний у порядку черговості.

Перший крок у реалізації запиту уведення-виведення, наприклад, для диска, полягає в перетворенні його з абстрактної форми в конкретну. Для дискового драйвера це означає перетворення номерів блоків у номери циліндрів, поверхонь, секторів, перевірку, чи працює двигун, чи знаходиться зчитуючий елемент над потрібним циліндром. Тобто він повинен вирішити, які операції контролера потрібно виконати і у якій послідовності.

Після передачі команди контролеру драйвер повинен вирішити, чи блокувати себе до завершення заданої операції чи ні: якщо операція забирає значний час, як при друкуванні, то драйвер блокується до завершення операції, і опрацювання переривання не розблокує

його; якщо команда уведення-виведення виконується швидко (наприклад, прокручування екрану), то драйвер очікує її завершення без блокування.

### Незалежний від пристроїв рівень операційної системи

Велика частина програмного забезпечення уведення-виведення є незалежною від пристроїв, точна границя між драйверами і незалежними від пристроїв програмами визначається системою, оскільки функції, що могли б бути реалізовані незалежним способом, у дійсності виконані у вигляді драйверів для підвищення ефективності або з інших причин.

Наприклад: при створенні файлу або заповненні його новими даними необхідно виділити йому нові блоки. Для цього ОС повинна вести список вільних блоків диска. На підставі інформації про наявність вільного місця на диску може бути розроблений алгоритм пошуку вільного блоку, незалежний від пристрою і реалізований програмним шаром, що знаходиться вище шару драйверів.

Верхнім рівням програмного забезпечення не зручно працювати з блоками різної величини, тому даний шар забезпечує єдиний розмір блоку за рахунок об'єднання кількох різних блоків у єдиний логічний блок. У зв'язку з цим верхні рівні мають справу з абстрактними пристроями, що використовують єдиний розмір логічного блоку незалежно від фізичного розміру даних. Типовими функціями для незалежного від пристроїв рівня є:

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

### Користувацький рівень програмного забезпечення

Хоча велика частина програмного забезпечення уведення-виведення знаходиться у середині ОС, деяка його частина розміщується в бібліотеках, що зв'язуються з програмами користувача. Набір подібних процедур є частиною системи уведення/виведення. Стандартна бібліотека уведення/виведення містить велику кількість процедур, що виконують уведення-виведення і працюють як частина програми користувача. Прикладом може слугувати функція *write* мови *Pascal*, що приймає рядок, формат виводу як вхідну інформацію, потім формує рядок з символів *ASCII* і робить виклик системної функції для виведення цього рядка.

Іншою категорією програмного забезпечення уведення-виведення є підсистема спулінга (*spooling*). Спулінг — це спосіб роботи з виділеними пристроями в мультипрограмній системі.

Як приклад розглянемо типовий пристрій, що вимагає спулінга - принтер. Хоча технічно легко дозволити кожному процесу користувача відкрити спеціальний файл, зв'язаний із принтером, такий спосіб небезпечний через те, що процес користувача може монополізувати принтер на довільний час, замість цього створюється:

- 1) процес — монітор, який отримує виняткові права на використання цього пристрою,
- 2) спеціальний каталог, названий каталогом спулінга.

При друкуванні, процес користувача розміщує виведену інформацію в файл у каталозі спулінга. Процес-монітор по черзі роздруковує усі файли, що містяться в каталозі спулінга.

## ОБОЛОНКА КОРИСТУВАЧА

Користувачі здійснюють управління комп'ютером, використовуючи спеціальний модуль операційної системи — командний процесор або оболонку системи (shell). Основною функцією інтерфейсного модуля є отримання команд уведених за допомогою клавіатури або допоміжних пристроїв (маніпулятор “мишка”, “трекболл”), та виведення результату виконання введеної команди.



**Інтерфейс користувача** – засоби та сукупність команд операційної системи для здійснення управління користувачем засобами обчислювальної системи та виведення інформації для користувача.

Алгоритм роботи інтерфейсного модуля досить простий:

- 1) більшу частину часу він очікує введення команди користувача;
- 2) інтерпретує команду, що надійшла;
- 3) формує відповідний системний виклик або запускає визначену програму;
- 4) при необхідності видає на екран повідомлення про процес виконання;
- 5) після закінчення виконання команди переходить у режим очікування нової команди.

У сучасних операційних найбільш поширеними є:

— інтерфейс командного рядка, який дозволяє користувачеві набирати команди на клавіатурі після системного запрошення розташованого у рядку введення. Монітор працює в текстовому режимі.

— графічний інтерфейс користувача, який забезпечує уведення команд операційної системи за допомогою виконання дій з візуальним поданням об'єктів ОС. Файли, каталоги, команди, програми користувача подаються у вигляді значків (піктограм), меню, кнопок, вікон. Для задання команд використовується маніпулятор "мишка", "трекбол".

### Інтерфейс командного рядка

Найпершим способом забезпечення управління обчислювальною системою був інтерфейс командного рядка. При цій технології, як єдиний спосіб введення команд людини до комп'ютера використовується клавіатура, а комп'ютер виводить інформацію за допомогою алфавітно-цифрового дисплея (монітора). Таку комбінацію (монітор + клавіатура) стали називати *терміналом* або *консоллю*. Команди набираються в командному рядкові, що являє собою символ запрошення і миготливий прямокутник - курсор (рис.10.)

При натисканні клавіші на місці курсору з'являються символи, а сам курсор зміщується вправо, неправильно набраний символ можна вилучити. Введення команди закінчується натисканням клавіші Enter (або Return.) Після цього починається процес виконання програми, що супроводжується виведенням на екран (в міру необхідності) результатів виконання. Після закінчення виконання програми знову на екрані монітора з'являється системне запрошення.

```
[student@avalon student$ps -a | grep[
  gimp
  868 pts/0 00:00:30  gimp
[student@avalon student$ps -a | grep[
  gimp > myproc
[student@avalon student$_
```

Рис.10. Інтерфейс командного рядка.

У багатозадачних операційних системах оболонки користувача дозволяють запустити програму на виконання у фоновому режимі, не змушуючи чекати завершення попередньої програми (табл. 2)

Таблиця 2

## Засоби управління процесами консольного режиму ОС Linux

Дія	Команда
запуск програми на виконання	назва_програми
призупинити виконання активної задачі	[Ctrl z]
перервати виконання активної задачі	[Ctrl c]
запуск програми на виконання у фоновому режимі	назва_програми&
отримати перелік виконуваних завдань оболонкою користувача	jobs
перенести задачу у основний режим виконання [номер завдання — виводиться командою jobs]	fg %[номер завдання ]
перенести задачу у основний режим виконання [номер завдання — виводиться командою jobs]	bg %[номер завдання ]
отримати перелік виконуваних системою завдань	ps -all [номер завдання ]
вилучити виконуване завдання [номер завдання — виводиться командою ps]	kill -9
Отримати перелік виконуваних завдань	top

В різних операційних системах передбачено різні можливості щодо надання сервісних послуг при використанні командного рядка (табл. 3)

Таблиця 3

Додаткові можливості,  
що надаються командним процесором

	MS DOS	Unix
Зміна системного запрошення	+	+
Редагування командного рядка	+	+
Швидке повторення введення останньої команди	+	+
Ведення журналу введених команд	—	+
Написання сценаріїв для автоматизації виконання завдань	±	+
Контроль та управління виконанням процесів	—	+

### Графічний інтерфейс користувача

Ідея використання графічного інтерфейсу користувача зародилася в середині 70-х років, коли в дослідницькому центрі Xerox Palo Alto Research Center (PARC) була розроблена концепція візуального інтерфейсу. Передумовою появи графічного інтерфейсу з'явилося зменшення часу реакції комп'ютера на введену команду, зумовлене зростанням потужності центрального процесора, та додаткового обладнання. Перша система з графічним інтерфейсом 8010 Star Information System групи PARC з'явилася за чотири місяці до виходу у світ першого персонального комп'ютера фірми IBM у 1981 році. На перших етапах візуальний інтерфейс використовувався тільки для прикладного програмного забезпечення: текстовий редактор, електронні таблиці. Зростання попиту на обчислювальну техніку змусило розробників ОС використовувати засоби візуального управління в операційних системах: спочатку на комп'ютерах Atari і Apple Macintosh, а потім і на IBM-сумісних комп'ютерах.

Паралельно з розробкою графічних інтерфейсів для ОС проходив процес уніфікації використання клавіатури і мишки прикладними програмами. Злиття цих двох тенденцій привело до створення користувацького інтерфейсу, за допомогою якого, при мінімальних затратах часу і засобів на перенавчання, можна працювати з будь-якими програмними продуктами.

Основною концепцією сучасних графічних інтерфейсів є подання компонентів операційної системи (файл, каталог, програма) у вигляді візуальних графічних об'єктів, що мають певні властивості, команди операційної системи відображаються як зміна властивостей об'єктів (табл. 4).

Таблиця 4

Типові операції з об'єктами графічного інтерфейсу

Файл	Папка	Вікно
Відкрити	Відкрити	Відкрити
Копіювати	Копіювати	Перенести
Перенести	Перенести	Змінити розміри
Переіменувати	Переіменувати	Закрити
Вилучити	Вилучити	
Виконати		

Концепція об'єктного підходу не нова. Дослідження в галузі психології довели, що мислення людини в процесі діяльності оперує поняттями на рівні об'єктів та зміни їх властивостей. Враховуючи особливості сприймання та діяльності і потреби максимально наблизити роботу користувача з ЕОМ до природної, зумовили виникнення об'єктно-орієнтованих мов програмування (Small Talk, C++, Object Pascal та інші). Використання об'єктно-орієнтованого підходу до розробки прикладного програмного забезпечення призвело до відповідних нововведень і в операційній системі. В графічному інтерфейсі користувача, програмному інтерфейсі операційної системи почали застосовувати об'єктно-орієнтовані парадигми.

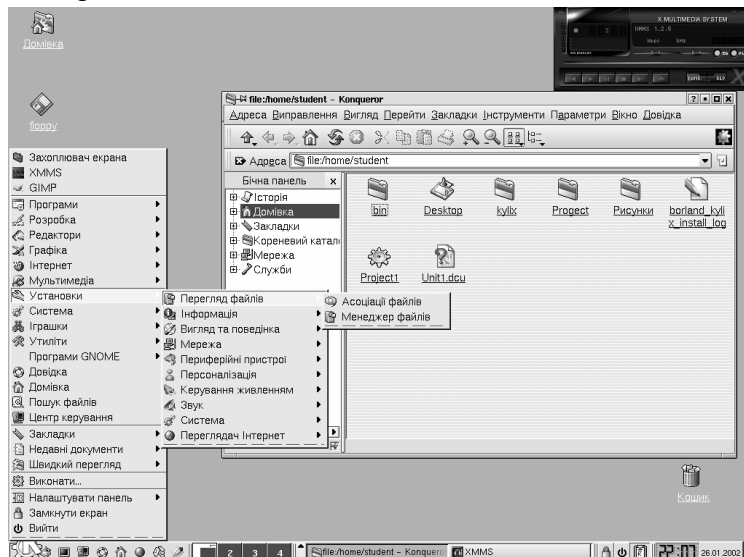


Рис 24. Робочий стіл користувача ОС Linux – графічний інтерфейс користувача

Першу реалізацію об'єктно-орієнтованих концепцій щодо інтерфейсу користувача було втілено в операційній системі фірми IBM OS/2 та у персональних комп'ютерах Next. Усі апаратні і програмні примітиви представлені як об'єкти з певними властивостями та засобами їх зміни: пам'ять, дисплей, принтер, папка, звукова карта, дисковод. Наступним розширенням ідеології об'єктно-орієнтованого інтерфейсу (OOI) стала концепція робочого столу (desktop), як аналогія робочого столу, на якому розміщуються (рис. 24).

- 1) документи — файли, папки з документами;
- 2) програми — інструменти для роботи з цими документами.

Протягом тривалого часу desktop-ідеологія була складовою частиною різних користувацьких інтерфейсів, починаючи з Macintosh і закінчуючи Workplace Shell операційної системи OS/2. У сучасних операційних системах Windows, MacOS, Linux "робочий стіл" користувача набув подальшого розвитку та вдосконалення.

### Основні складові графічного інтерфейсу користувача

Аналізуючи графічні інтерфейси користувачів, що надаються різними операційними системами, можна виділити деякі інваріантні елементи, об'єкти, способи їх використання відносно функціонального призначення (рис. 25, табл. 5).

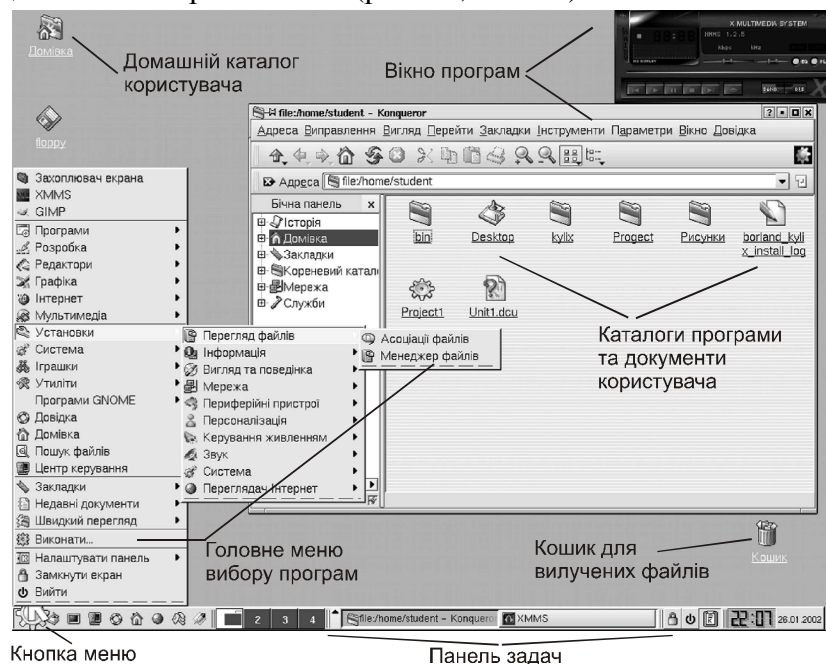


Рис. 25. Об'єктно-орієнтований інтерфейс користувача ОС Linux

Таблиця 5

### Елементи графічного інтерфейсу

Елемент інтерфейсу	Функціональне призначення
Головне меню (кнопка запуску програм)	Запуск прикладних програм на виконання
Панель задач	Перемикання між виконуваними завданнями
Кошик (Trash, Recycle bin)	Тимчасове зберігання вилучених файлів, каталогів
Вікно	Прикладна програма користувача, яка виконується

З розвитком графічних інтерфейсів поняття курсору і його зміст змінився. Оскільки дії відбуваються як зміна певних властивостей об'єктів, то об'єкти необхідно виділяти (*select*), переміщувати (*move*) та інше певним способом використовуючи графічний покажчик (*graphic pointer*). Цей універсальний покажчик і став використовуватися як графічний курсор (*graphic cursor*), який управляється маніпулятором "мишка". Використовуючи мишку можна:

"клацнути" (*click*) — швидко натиснути і відпустити клавішу мишки;

"двічі клацнути" (*double click*) — двічі швидко натиснути і відпустити клавішу мишки;

"натиснути" (*press*) — натиснути і утримувати клавішу мишки;

"відпустити" (*down*) — відпустити клавішу мишки;

"перевести курсор мишки" (*move the cursor*) — означає, що мишка без натискання клавіш просто переміщується в інше місце. При цьому змінюється положення курсора мишки, а сам він не повинен змінювати форми, якщо це не обумовлено окремо. Переміщення курсору не повинно впливати на розміщенні об'єкти на екрані.



З зображеними на екрані об'єктами за допомогою мишки можна виконати операції:

— "вибрати" (*choose*) - вказати мишкою на об'єкт і клацнути лівою клавішею мишки, або просто вказати на об'єкт;

— "відкрити" (*open*) - вказати на об'єкт і двічі клацнути лівою клавішею мишки, або вказати на об'єкт і клацнути лівою клавішею миши;

— "перемістити" (*move*) - переміщення курсору при натиснутій лівій клавіші. Це досить специфічна дія, тому як правило вона супроводжується візуальним ефектом на екрані: курсор як би "тягне" за собою об'єкт. Звільнення об'єкта відбувається шляхом відпускання лівої клавіші мишки; при цьому об'єкт фіксується в місці, на яке вказує курсор.

Основним елементом графічного інтерфейсу є вікно, з яким можна виконувати операції:

- відкрити вікно — запустити на виконання певну програму;
- закрити вікно — припинити виконання програми;
- розгорнути на весь екран чи згорнути до попереднього розміру;
- мінімізувати вікно, розгорнути мінімізоване вікно;
- змінити розміри;
- перенести вікно в інше місце робочого столу;
- активізувати вікно.

Кожне вікно, що відображається на екрані, асоціюється з виконуваною програмою. Відкриті вікна під час роботи (для зручності) можна розміщувати каскадом; зліва направо чи зверху вниз; довільно за бажанням користувача використовуючи мишку або клавіатуру, вибравши необхідний варіант в меню. Вікно містить типові управляючі та інформаційні елементи (рис. 26):

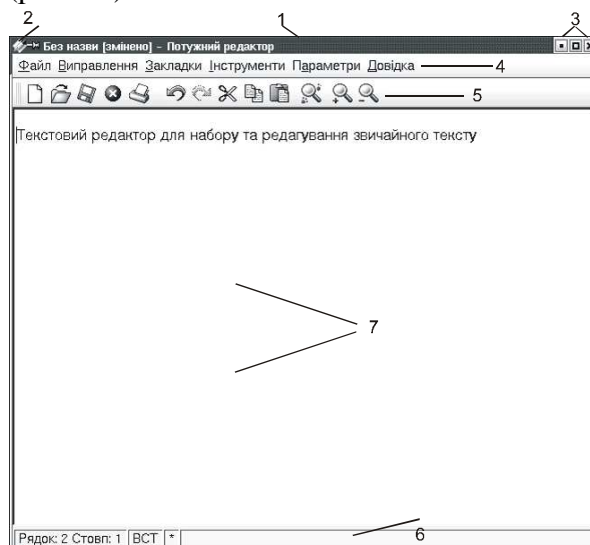


Рис. 26. Компоненти вікна

1) заголовок — рядок з назвою об'єкта і кнопками управління вікном.

2) *кнопка системного меню вікна* — дозволяє згорнути чи розгорнути вікно, змінити його розмір, перемістити його з допомогою клавіатури. При роботі з мишкою це меню не використовується.







3) *кнопки управління вікном* (табл.6).

4) *голове меню* — ієрархічний список команд відповідної програми. В ньому, як правило, наявні такі пункти:

- *Файл* – використовують для роботи з дисками, папками, файлами та ярликами. Залежно від типу обраного об'єкта може змінюватись перелік команд цього меню.
- *Виправлення* – використовується для вибору (відмічання) об'єктів, а також їх редагування.
- *Вид* – використовується для зміни параметрів перегляду інформації у вікні.
- *?* (*Допомога*) — отримання довідки.

Таблиця 6

Кнопки управління вікном  
різних графічних інтерфейсів користувачів

Елемент управління	Операційна система	
	Windows	Linux, графічний інтерфейс користувача KDE <sup>2</sup>
Розгорнути на весь екран		
Відновити (згорнути до неповного екрану) вікно		
Закрити вікно і відповідний додаток, з яким працював користувач		

5) *панель інструментів* — кнопки управління, які повторюють основні команди меню. Використовуються для швидкого доступу до послуг програми.

6) *рядок стану* — виводить інформацію про об'єкти, розміщені у вікні.

7) *робоча область* — виводиться основна інформація

Крім вікна графічний інтерфейс включає додаткові елементи (табл. 7) для введення певних значень параметрів шляхом набору з клавіатури, вибору з наперед заданого списку тощо.

*Перемикач* – для вибору зі списку параметрів.

*Лічильник* – віконце, в якому зображаються числові значення параметра.

*Список* – для вибору з переліку наперед вказаних значень.

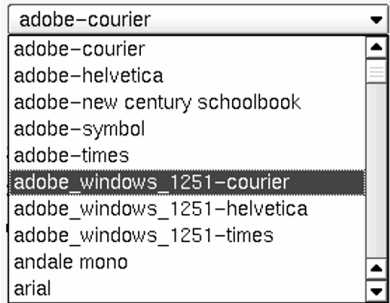

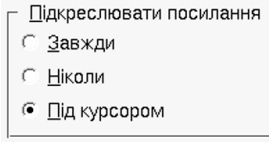

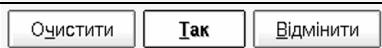
*Рядок уведення* – використовується для введення текстової інформації за допомогою клавіатури. Для появи курсору в рядку введення треба натиснути ліву кнопку мишки в полі рядка.

*Так* – закриває вікно зі збереженням всіх змін.

*Відмінити(скасувати)* – відмовитися від змін, закрити вікно.

<sup>2</sup> Робочий стіл користувача операційної системи Linux

## Елементи управління графічного інтерфейсу ОС Linux

Елемент управління	Зображення
Список	
Лічильник	
Перемикач	
Рядок введення	
Кнопки	

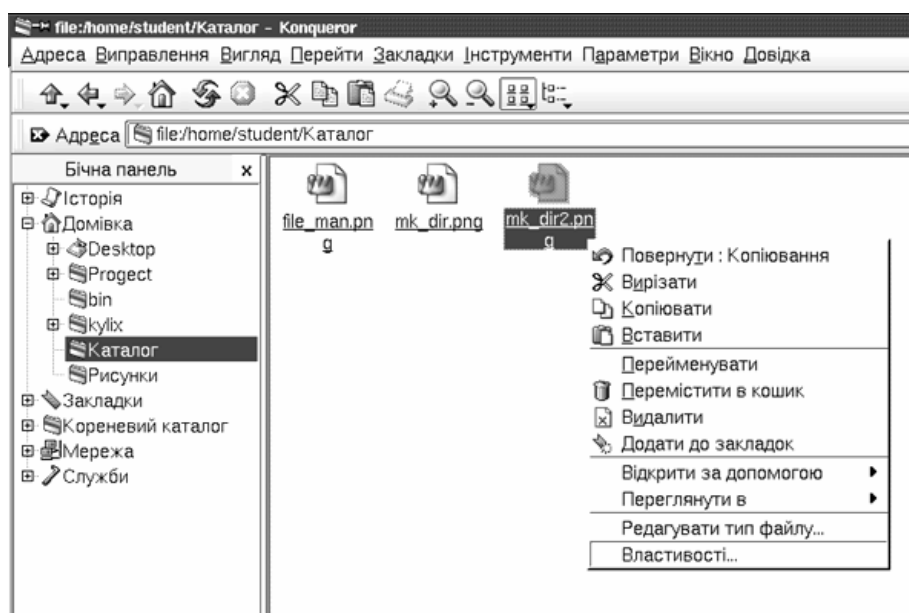


Рис. 27. Локальне меню для файла.

Для кожного об'єкту робочого столу визначений певний набір операцій, які можна проводити з вибраним об'єктом. Отримати перелік операцій можна, навівши курсор на об'єкт і натиснувши праву кнопку мишки, після чого відкриється *локальне меню* (рис. 27). Для виконання операції слід перемістити курсор на назву операції і натиснути ліву кнопку мишки.

## Налагоджування робочого середовища

Усі графічні інтерфейси користувача надають достатньо широкі можливості, щодо налагоджування параметрів робочого середовища:

- 1) вибір кольорової палітри,

- 2) встановлення кольору фону або фонове зображення,
- 3) вибір шрифту, видів курсору,
- 4) оформлення вікна,
- 5) звуків як реакції на певні події операційної системи та інше (рис.28).

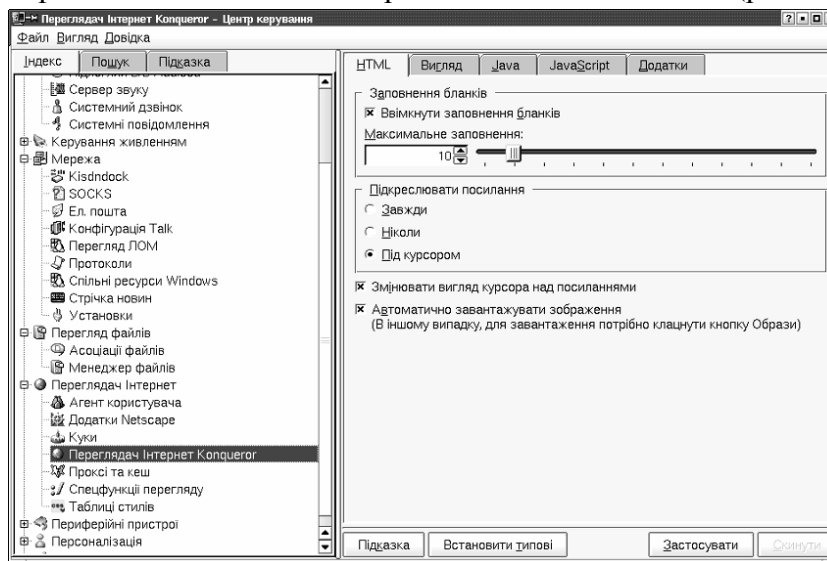


Рис. 28. Вікно налагоджування параметрів робочого столу.



Комплексний набір параметрів, які формують зовнішній вигляд робочого столу, називають *темою*.

Всі сучасні інтегровані середовища користувачів містять певну кількість готових тем.