10.1 ОСНОВНІ ПОНЯТТЯ І ВИМОГИ ДО УПРАВЛІННЯ ПАМ'ЯТТЮ

Пам'ять є найважливішим ресурсом, який вимагає ретельного управління з боку мультипрограмної операційної системи. Розподілу підлягає вся оперативна пам'ять, що не зайнята операційною системою. Зазвичай ОС розташовується в наймолодших адресах, проте може займати і найстарші адреси. Функціями ОС з управління пам'яттю є:

– відстежування вільної і зайнятої пам'яті;

– виділення пам'яті процесам і звільнення пам'яті при завершенні процесів;

– витіснення процесів з оперативної пам'яті на диск, коли розміри основної пам'яті не достатні для розміщення в ній усіх процесів, і повернення їх в оперативну пам'ять, коли в ній звільняється місце;

– налаштування адрес програми на конкретну область фізичної пам'яті.

10.1.1 Переміщення

У багатозадачній системі доступна основна пам'ять розподіляється між багатьма процесами. Програміст не знає, які програми будуть резидентно знаходитися в основній пам'яті під час виконання його програми. Крім того, для максимального завантаження процесора бажано мати великий пул (буфер) готових до виконання процесів. Для цього необхідно завантажувати і вивантажувати активні процеси з основної пам'яті. Вимога, щоб вивантажена з пам'яті програма була знову завантажена в одне і те ж місце, де вона була раніше, було б великим обмеженням. Тому вкрай бажано, щоб програма могла бути переміщена в іншу ділянку пам'яті.

Таким чином, заздалегідь невідомо, де саме буде розміщена програма. Крім того, програма може бути переміщена з однієї області пам'яті в іншу в разі закінчення іншої програми. Очевидно, що ОС необхідно знати місце розташування управляючої інформації, а також точки входу для початку виконання процесу. Оскільки управлінням пам'яттю займається ОС, вона ж розміщує процес в основній пам'яті, то вона автоматично отримує відповідні адреси.

10.1.2 Захист

Кожен процес має бути захищений від випадкового або з якимсь наміром небажаного впливу інших процесів. Тому код інших процесів не повинен мати можливості без дозволу звертатися до пам'яті цього процесу. Під час роботи програми необхідно виконати перевірку всіх звернень до пам'яті, які генеруються процесом, щоб переконатися, що всі вони звертаються тільки до пам'яті, яка виділена цьому процесу. Механізми підтримки переміщень забезпечують і підтримку захисту. Вимоги щодо захисту пам'яті повинні виконуватися на рівні процесора (апаратно), а не на рівні ОС.

10.1.3 Спільне використання

Будь-який механізм захисту повинен мати достатню гнучкість, для того щоб забезпечити можливість декільком процесам звертатися до однієї і тієї ж області основної пам'яті. Наприклад, якщо декілька процесів виконують один і той же машинний код, то буде вигідно дозволити кожному процесу працювати з однією і тією ж копією цього коду, а не створювати свою власну копію. Процесам, які взаємодіють при виконанні деякого завдання, може знадобитися загальний доступ до одних і тих же структур даних. Система управління пам'яттю повинна забезпечувати керований доступ до різних ділянок пам'яті.

10.1.4 Типи адрес

Для ідентифікації змінних і команд використовуються символьні імена (мітки), віртуальні адреси і фізичні адреси (рис. 10.1).

Символьні імена привласнює користувач при написанні програми алгоритмічною мовою або на асемблері.

Віртуальні адреси виробляє транслятор, який перекладає програму на машинну мову. Оскільки під час трансляції, в загальному випадку, невідомо в яке місце оперативної пам'яті буде завантажена програма, то транслятор привласнює змінним і командам віртуальні (умовні) адреси, вважаючи за умовчанням, що програма буде розміщена, починаючи з нульової адреси. Є ще поняття відносної адреси, що є

окремим випадком віртуальної адреси (іноді – логічної адреси), коли адреса визначається положенням відносно деякої відомої точки (початку програми).

Сукупність віртуальних адрес процесу називається віртуальним адресним простором. Кожен процес має власний віртуальний адресний простір. Максимальний розмір віртуального адресного простору обмежується розрядністю адреси, властивій цій архітектурі комп'ютера, і, як правило, не співпадає з об'ємом фізичної пам'яті, наявним в комп'ютері. Наприклад, при використанні 32-розрядних віртуальних адрес цей діапазон задається межами 0000000016 і FFFFFFFF16.

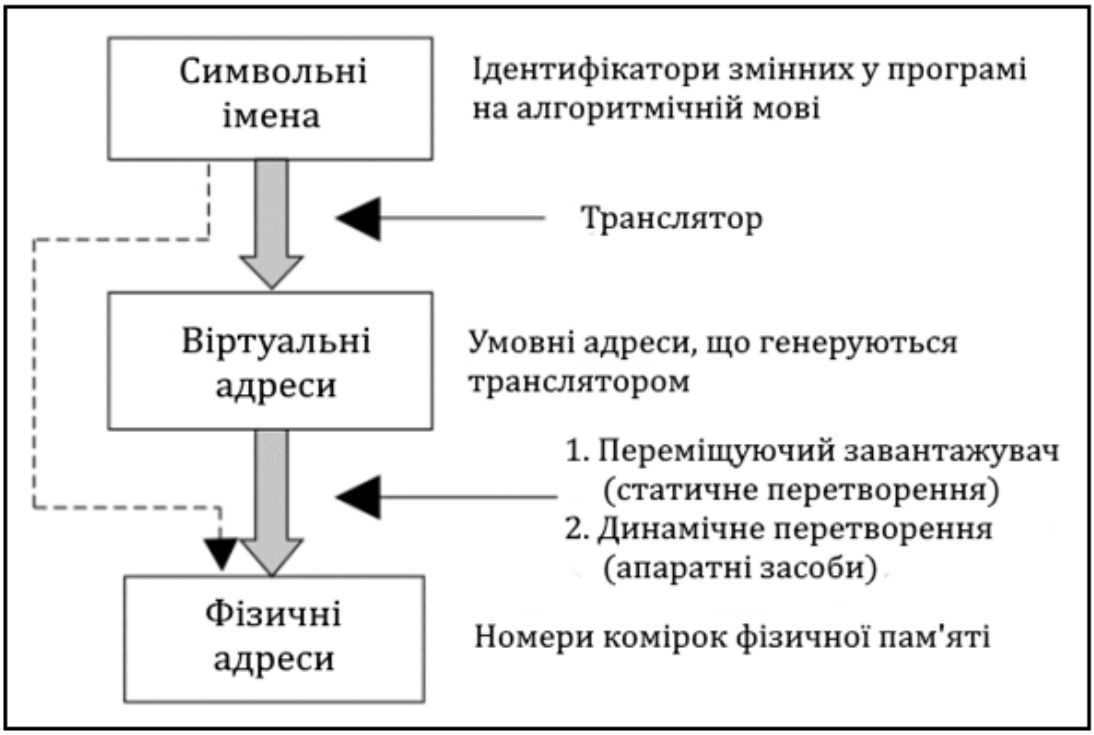

Рисунок 10.1 – Типи адрес

У різних операційних системах використовуються різні способи структуризації віртуального адресного простору. В одних ОС віртуальний адресний простір процесу подібно до фізичної пам'яті представлений у вигляді безперервної лінійної послідовності віртуальних адрес. Таку структуру адресного простору називають також плоскою (flat). При цьому віртуальною адресою є єдине число, яке представляє собою зміщення відносно початку віртуального адресного простору (звичайне це значення 000...000). Адресу такого типу називають лінійною віртуальною адресою. Недоліки такої системи: одна ділянка може повністю заповнитися, але при цьому

залишаться вільні ділянки. Можна звичайно переміщати ділянки, але це дуже складно.

Ці проблеми можна вирішити, якщо дати кожній ділянці незалежний адресний простір. Тому в багатьох ОС віртуальний адресний простір ділиться на частини, які називаються сегментами (або секціями, або областями, або іншими термінами). У цьому випадку окрім лінійної адреси може бути використана віртуальна адреса, що є парою чисел (n, m), де n визначає сегмент, а m – зміщення усередині сегменту.

Фізичні адреси (абсолютні, реальні) відповідають номерам елементів оперативної пам'яті, де насправді розташовані або будуть розташовані змінні і команди. Перехід від віртуальних адрес до фізичних може здійснюватися двома способами.

При застосуванні першого випадку заміну віртуальних адрес на фізичні робить спеціальна системна програма – переміщаючий завантажувач. Переміщаючий завантажувач на підставі наявних у нього даних про початкову адресу фізичної пам'яті, в яку належить завантажувати програму, і інформації, наданої транслятором про адресно-залежні константи програми, виконує завантаження програми, поєднуючи її із заміною віртуальних адрес на фізичні.

Другий спосіб полягає в тому, що програма завантажується в пам'ять у незміненому виді у віртуальних адресах, при цьому операційна система фіксує зміщення дійсного розташування програмного коду відносно віртуального адресного простору.

Під час виконання програми при кожному зверненні до оперативної пам'яті виконується перетворення віртуальної адреси у фізичну. Другий спосіб є гнучкішим, він допускає переміщення програми під час її виконання, тоді як переміщаючий завантажувач жорстко прив'язує програму до спочатку виділеної їй ділянки пам'яті. У той же час використання переміщаючого завантажувача зменшує накладні витрати, оскільки перетворення кожної віртуальної адреси відбувається тільки один раз під час завантаження, а в другому випадку – кожного разу при зверненні за цією адресою.

В деяких випадках (зазвичай в спеціалізованих системах), коли заздалегідь точно відомо, в якій області оперативної пам'яті виконуватиметься програма, транслятор видає виконуваний код відразу у фізичних адресах.

А тепер розглянемо алгоритми розподілу пам'яті, які застосовуються і застосовувалися в різних ОС.

Усі методи управління пам'яттю можуть бути розділені на два класи: методи, які використовують переміщення процесів між оперативною пам'яттю і диском, і методи, які не роблять цього (рис. 10.2).

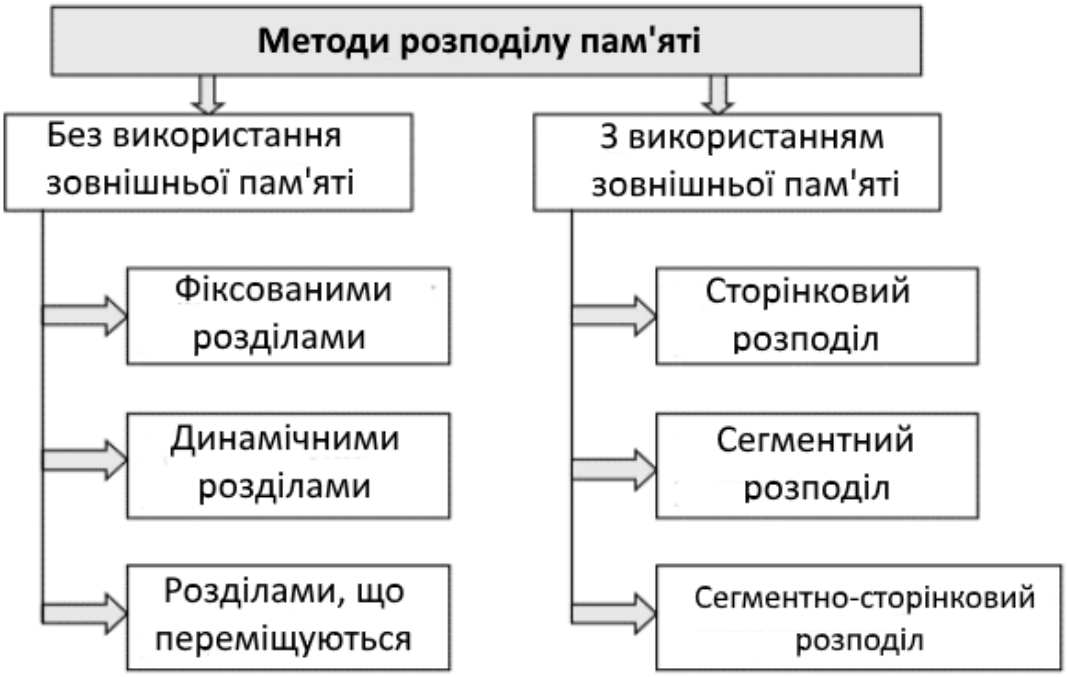

Рисунок 10.2 – Класифікація методів розподілу пам'яті

Чи слід призначати кожному процесу одну безперервну область фізичної пам'яті або можна виділяти пам'ять «ділянками»? Чи повинні сегменти програми, завантажені в пам'ять, знаходитися на одному місці впродовж усього періоду виконання процесу або можна їх час від часу переміщати? Що робити, якщо сегменти програми не поміщаються в наявну пам'ять?

Різні ОС по-різному відповідають на ці і інші базові питання управління пам'яттю. Далі будуть розглянуті найзагальніші підходи до розподілу пам'яті, які були характерні для різних періодів розвитку операційних систем. Деякі з них зберегли актуальність і широко використовуються в сучасних ОС, інші ж представляють в основному тільки пізнавальний інтерес, хоча їх і сьогодні можна зустріти в спеціалізованих системах.