МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДВНЗ «УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

Факультет інформаційних технологій

Кафедра програмного забезпечення систем

«СИСТЕМНЕ ПРОГРАМУВАННЯ»

Конспект лекцій

УЖГОРОД – 2018

Системне програмування: конспект лекцій для студентів за напрямом підготовки 6.050103 «Програмна інженерія» факультету інформаційних технологій УжНУ / Розробник: к.т.н. Поліщук В.В. – Ужгород: 2018. – 56 с.

У конспекті лекцій з курсу «Системне програмування» розглянуто теоретичні основи, що входять до складу робочої програми; наведено теми лекційних та лабораторних занять, перелік запитань на підсумковий контроль та список рекомендованої літератури.

Укладач: к.т.н. Поліщук В.В., доцент кафедри програмного забезпечення систем факультету інформаційних технологій ДВНЗ «УжНУ».

Рецензенти:

- к.т.н., доц., декан факультету інформаційних технологій ДВНЗ «УжНУ» Повхан І.Ф.
- к.т.н., доцент кафедри інформатики та фізико-математичних дисциплін факультету інформаційних технологій ДВНЗ «УжНУ» Лях І. М.

Рекомендовано кафедрою програмного забезпечення систем від «24» січня 2018 р., протокол №5.

Рекомендовано Вченою радою факультету інформаційних технологій (протокол №8 від «23» лютого 2018 року).

ЗМІСТ

Мета та завдання навчальної дисципліни

Мета дисципліни полягає у формуванні знань, вмінь та навичок, необхідних для раціонального використання системних ресурсів ЕОМ; вивченні програмування для ОС Windows та методів розробки програм, що взаємодіють з операційною системою; надбанні навичок використання сучасних інформаційних технологій при розв'язанні задач, пов'язаних зі створенням програмного забезпечення прикладного та системного характеру; знайомстві студентів з перспективами розвитку технологій та методів системного програмування.

Основні **завдання** курсу:

- ознайомлення з основами побудови системного програмного забезпечення;
- вивчення технологій, засобів та методів системного програмування;
- ознайомлення з інструментальним програмним забезпеченням для створення системного програмного забезпечення;
- вивчення технологій створення системних програм.

У результаті вивчення навчальної дисципліни студент повинен

знати:

- призначення та функції інструментальних засобів для створення системного програмного забезпечення;
- основи програмування системних задач;
- методи сполучення прикладних програм з сучасним системним програмним забезпеченням.

вміти:

- застосовувати функції, що експортуються операційним середовищем;
- створювати нові прикладні та системні програми;
- вирішувати питання організації програмного інтерфейсу в системних програмах та модулях.

Програма навчальної дисципліни

Змістовий модуль 1. Принципи управління в ОС та архітектура мікропроцесорів

Тема 1. Основні поняття операційних систем: основні терміни і означення; функції, ресурси та взаємозв'язки ОС; покоління операційних систем; класифікації операційних систем; склад і загальна схема функціонування ОС; ієрархічна структура ОС; основні принципи побудови ОС.

Тема 2. Керування процесами: стани процесів та їх переходи; основні операції над процесами; обробка та типи переривань. Ядро ОС; класифікація процесів і види відношень між ними; взаємовиключення. Алгоритм Деккера; семафори. Моніторподібні засоби синхронізації.

Тема 3. Тема 3. Керування пам'яттю: організація та ієрархія пам'яті; стратегії керування пам'яттю; розподіли пам'яті.

Змістовий модуль 2. Процеси та потоки в ОС

Тема 4. Організація керування віртуальною пам’яттю:

механізм відображення віртуальних адрес в реальні; сторінкова, сегментна та комбінована організації віртуальної пам'яті.

Тема 5. Стратегії керування віртуальною пам'яттю: основні стратегії; локальність; робочі множини та підкачки сторінок.

Тема 6. Керування процесорами: рівні планування в ОС; дисципліни планування; мультипроцесорні системи; організація мультипроцесорної апаратури.

Теми лабораторних занять

№ з/п	Назва теми
1	Розробка комп'ютерної програми, що імітує алгоритм диспетчеризації задач
2	Розробка комп'ютерної програми, що імітує алгоритм розподілення пам'яті в мультизадачній
3	Розробка комп'ютерної програми, що імітує алгоритм управління введенням/виведенням в операційній системі
4	Розробка комп'ютерної програми, що імітує алгоритм синхронізації задач
5	Розробка комп'ютерної програми, що імітує алгоритм захисту адресного простору задач
6	Розробка комп'ютерної програми, що імітує алгоритм управління процесами
7	Розробка комп'ютерної програми, що імітує алгоритм управління виконання потоку
8	Розробка комп'ютерної програми створення бібліотеки, що динамічно підключаються

Тема 1. Основні поняття операційних систем

Операційна система (ОС) часто в більшій мірі визначає уявлення користувача про машину, ніж сама апаратура цієї машини. Очевидно, що необхідно мати чітке і адекватне означення операційної системи, знати які функції виконує операційна система і якими ресурсами вона керує.

1.1. Основні терміни і означення

Операційна система - це набір програм, як звичайних, так і мікропрограм, які забезпечують можливість використання апаратури комп'ютера. При цьому апаратура комп'ютера вважається "сирою" обчислювальною потужністю, а задача ОС - в тому, щоб зробити апаратуру доступною і по можливості зручною для користувача.

Користувач працює на так званій "розширеній" машині (рис.1.1).

Рисунок 1.1 - Структура "розширеної" машини

Під *програмним забезпеченням* (ПЗ) ЕОМ розуміють сукупність програм, процедур і правил разом з усією зв'язаною з цим документацією, яка дозволяє використовувати обчислювальну машину для розв'язання різних задач.

За задачами і функціями програмне забезпечення (ПЗ) можна поділити на дві групи: загальне та спеціальне.

Загальне ПЗ включає в себе засоби контролю, системи програмування, власне саму операційну систему.

Спеціальне ПЗ - пакети прикладних програм.

Операційна система - це система, яка керує виконанням інших програм. Частина ОС, що постійно знаходяться в основній пам'яті, називається *резидентною частиною* ОС або *ядром* ОС. Програми, що викликаються в основну пам'ять для виконання певних функцій, але не зберігаються там постійно,

називаються *транзитами.* Розділення ОС на ядро і транзити дозволяє зменшити область пам'яті, яку займає ОС і, відповідно, збільшити область програм користувачів.

Процес настроювання ОС на конкретну конфігурацію апаратних засобів і задач користувача, називається *генерацією системи.*

1.2. Функції, ресурси та взаємозв'язки ОС

ОС реалізує багато *функцій,* серед яких є такі:

1. визначає інтерфейс користувача;
2. забезпечує розподілення апаратних ресурсів між користувачами;
3. надає можливість працювати зі спільними даними в режимі колективного доступу;
4. планує доступ користувачів до спільних ресурсів;
5. забезпечує ефективне виконання операцій введення-виведення;
6. відновлює інформацію і обчислювальний процес у випадку помилки.

ОС керує такими основними *ресурсами:*

1. процесорами;
2. пам'яттю;
3. пристроями введення-виведення;
4. даними.

ОС *взаємодіє* з:

1. операторами ЕОМ;
2. прикладними програмістами;
3. системними програмістами (супроводжування ОС, налагоджування відповідно до вимог конкретної машини, доробка обслуговування нових типів пристроїв);
4. адміністраторами ОС (які встановлюють принципи і порядок роботи, взаємодіють з ОС для забезпечення певного порядку);
5. програмами;
6. апаратними засобами;

7. користувачами.

1.3. Покоління операційних систем

Нульове покоління (40-і роки).

ОС не було. Всі програми писалися на машинній мові і користувач мав доступ до всіх пристроїв.

Перше покоління (50-і роки).

Початок систем *пакетної обробки.* Декілька задач об'єднувались в групи (пакети). Запущена на виконання задача отримувала всі ресурси машини. Після завершення задачі (аварійного чи нормального) управління передавалося ОС, яка "прочищала" машину і забезпечувала введення та запуск нової задачі.

Друге покоління (початок 60-х років).

Характерною особливістю було те, що вони створювались як системи колективного користування з *мультипрограмним режимом роботи* і як перші *ОС мультипроцесорного типу.*

Третє покоління (середина 60-х - середина 70 років).

Початком цього покоління вважається поява в 1964 році сімейства комп'ютерів IBM-360. Це були громіздкі комп'ютери *(машини загального призначення),* призначені виконувати будь-які задачі з будь-якої області (але не кожному користувачу потрібні всі можливості) і тому дуже коштовні.

ОС для цих комп' ютерів були *багаторежимними системами.* Забезпечували пакетну обробку, режим реального часу, розподіл часу і мультипроцесорний режим. Недоліки: громіздкість і малоефективність, складність мов керування завданнями.

Четверте покоління (середина 80-х років і донині). Для цього покоління характерні:

1. Концепція розподіленої обробки даних за рахунок використання персональних комп'ютерів (ПК) і мереж.
2. Дружній інтерфейс, орієнтований на непідготовленого користувача.

Завдяки появі мікропроцесора були розроблені *персональні комп 'ютери.* З'являються системи з управлінням за допомогою меню. Важливу роль стали відігравати *системи баз даних.* Почала широко розповсюджуватись концепція *віртуальних машин.*

1.4. Класифікації операційних систем

Класифікація ОС може бути проведена за декількома критеріями.

1. За кількістю задач, що одночасно виконуються: однопрограмні та багатопрограмні ОС.
2. За кількістю користувачів: з одним користувачем та з декількома.
3. За типом інтерфейсу користувача: ОС безпосереднього доступу (діалогові), непрямого доступу (системи пакетної обробки).
4. За числом процесорів: однопроцесорні і мультипроцесорні.
5. За видом обробки даних: ОС із розподіленою обробкою даних (мережеві ОС) і з нерозподіленою обробкою.
6. За часом реакції системи: операційні системи реального часу та системи, що працюють зі швидкостями, визначеними внутрішніми особливостями самих систем.

Обчислювальна система працює в *однопрограмному режимі* (рис.1.2), якщо вона обслуговує тільки одну програму користувача. Користувач може працювати або в режимі безпосереднього доступу, або непрямого доступу (пакетний режим). Режим пакетної обробки дозволяє зменшити простої процесора, але, оскільки одна програма не може завантажити роботою всі пристрої, то це призводить до їх простою, що не є ефективним.

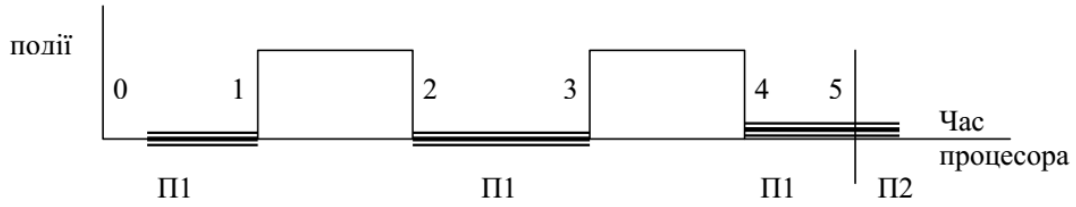


Рисунок 1.2 - Однопрограмний режим. 0-1, 2-3, 4-5 – виконання процесу П1, 5... - виконання П2, 1-2,3-4 - очікування зовнішньої події (процесор простоює).

Обчислювальна система працює в *мультипрограмному режимі,* якщо декілька програм одночасно знаходиться в пам'яті комп'ютера, причому виконання будь-якої з них може бути перервано для виконання іншої програми з наступним поверненням до виконання перерваної програми. Користувач може працювати як в режимі непрямого доступу (пакетна обробка), так і в режимі безпосереднього доступу. Розрізняють такі типи багатопрограмної роботи.

1. Класичне мультипрограмування характеризується тим, що правила переходу від однієї програми до іншої визначається з міркувань досягнення максимальної ефективності шляхом широкого використання суміщень.

Програма може бути перервана у двох випадках, а саме:

- якщо їй потрібне виконання якої-небудь події (природні переривання);
- якщо вона завершила виконання (рис. 1.3).

Недоліки: одна програма може монополізувати час процесора і заблокувати виконання інших. Мультипрограмування не сумісне з безпосереднім доступом і є різновидом пакетної обробки.

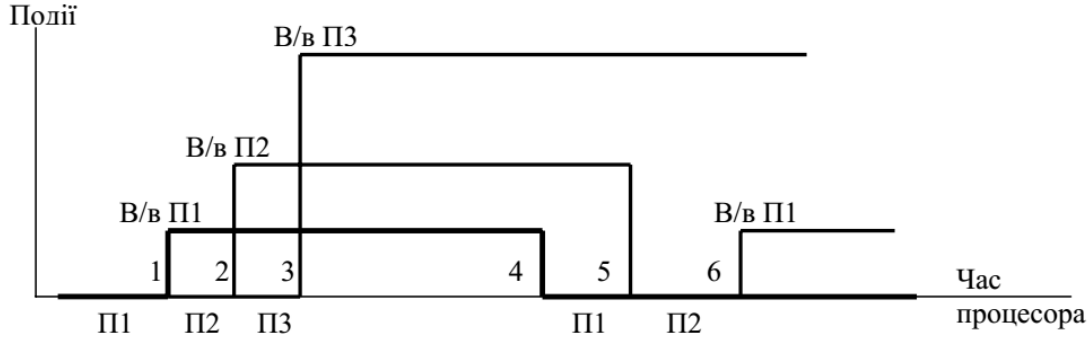


Рисунок 1.3 - Часова діаграма для класичного мультипрограмування (3-4 - простій процесора)

2. Під час *паралельної обробки* (рис.1.4) перехід від однієї програми до іншої виконується через досить короткий інтервал часу порівняно з швидкістю роботи машини, щоб створити у людини враження одночасного виконання декількох програм.

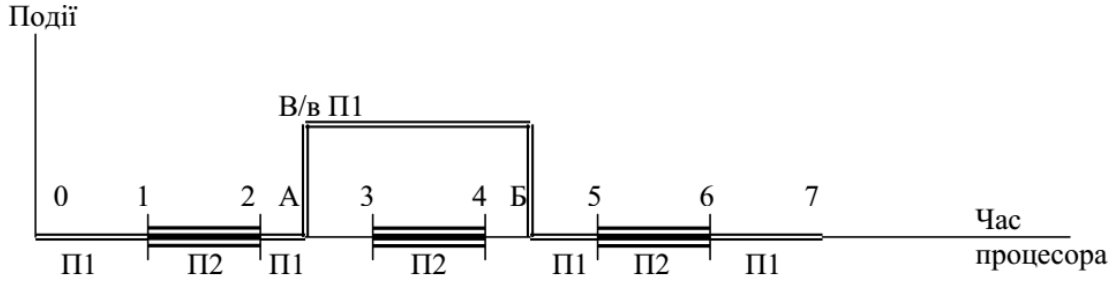


Рисунок 1.4 - Часова діаграма паралельної обробки
(А-3, 4-Б - простій процесора)

Недоліки: перехід від програми до програми можливий лише через строго визначені інтервали часу через примусові переривання. Примусові переривання генеруються системним таймером.

3. Системи з розподілом часу - це режим багатопрограмної роботи, коли перехід від однієї програми до іншої виконується за рахунок як "природних", так і "примусових" переривань (рис. 1.5).

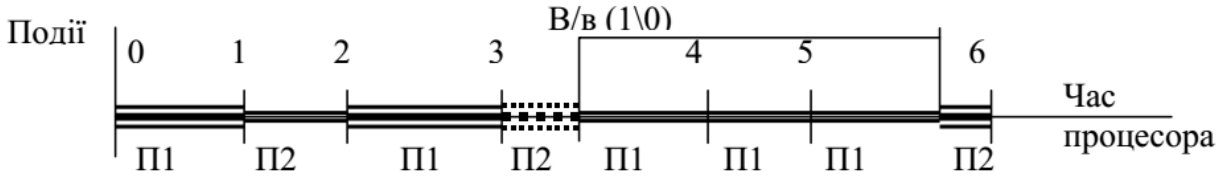


Рисунок 1.5 - Часова діаграма системи з розподілом часу

{Самостійна робота №1}

1.5. Склад і загальна схема функціонування ОС

З точки зору користувача ОС виконує такі основні функції:

1. Керування завданнями.
2. Керування задачами (процесами).
3. Керування даними.

4. Керування відновленням.

У відповідності з цим поділяється і керівна програма ОС. **Керування завданнями** визначає перехід від завдання користувача до задачі, що виконує машина. Підпрограми керування завданнями приймають і планують завдання по мірі їх надходження.

Керування задачами (процесами) слідкує за фактичним виконанням робіт; керування здійснюється відповідно з однією з дисциплін планування (LIFO, FIFO і т.д.).

Керування даними спрощує розміщення, пошук і підтримку всіх даних незалежно від способу їх організації або зберігання.

Керування відновленням дозволяє системі корегувати або виявляти помилку в роботі будь-якого елементу апаратури і замінювати або відключати непрацюючі пристрої.

З точки зору системних програмістів ОС має такі компоненти ОС:

1. Керування процесами,
2. Керування пам'яттю,
4. Керування процесорами,
5. Керування зовнішньою пам'яттю,
6. Керування пристроями введення-виведення.

1.6. Ієрархічна структура ОС

Ієрархічний підхід полягає в тому, що набори керівних програм рознесені по рівнях (шарах) спадаючих рангів. Цей підхід запропонував Дейкстра в 1968 році. Структурна схема ОС показана на рис.1.6.

Не існує твердого правила, які модулі слід розміщувати на якому рівні.

Кожен шар може використовувати тільки функції, які йому надають нижчі рівні, так, ніби вони є реальною машиною. ОС може бути створена і налагоджена по шарах, що зменшує складність розробки.

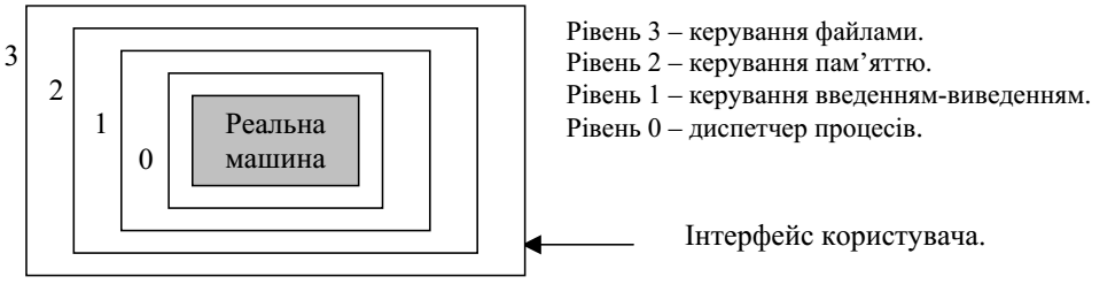


Рисунок 1.6 - Схема операційної системи з ієрархічною структурою

Розрізняють системи із строгою і прозорою ієрархією:

Строга ієрархія: 3 → 2 →1 → 0

Прозора ієрархія: 3 → 2, 1, 0

В системах з строгою ієрархією функції вищого рангу можуть звертатись до функцій тільки одного шару рангом нижче.

1.7. Основні принципи побудови ОС

1. *Частотний принцип* - для дій, які часто зустрічаються і забезпечуються умови їх швидкого виконання.
2. *Принцип модульності.*
3. *Принцип функціональної вибірковості* - виділяється частина важливих модулів, які завжди повинні бути в пам'яті (виділяється ядро).
4. *Принцип генерованості.*
5. *Принцип функціональної надлишковості* - забезпечується можливість виконання однієї і тієї ж роботи різними засобами.
6. *Принцип "за замовчуванням"* - оснований на зберіганні в системі деяких базових описів структур, процесів, конфігурації обладнання, що забезпечує початкову працездатність системи без її генерації.
7. *Принцип переміщуваності* - побудова модулів, виконання яких не залежить від їх місця в пам'яті.
8. *Принцип захисту* - необхідна розробка засобів, які виключають вплив одних програм на інші.
9. *Принцип незалежності* програм від зовнішніх пристроїв.
10. Принцип відкритої і нарощуваної ОС.

Тема 2. Керування процесами

1. Процес - це програма в стадії її виконання.
2. Процес - це об'єкт, якому виділяється процесор

Процес *- це система дій, яка реалізує певну функцію в обчислювальній системі і оформлена так, що керуюча програма обчислювальної системи може перерозподіляти* ***ресурси*** *цієї системи з метою забезпечення мультипрограмування.*

Ресурс *це засіб обчислювальної системи, який може бути виділений процесу на визначений інтервал часу*.

2.1. Стани процесів та їх переходи

За період свого існування процес проходить через різні дискретні стани, а саме:

1. *стан виконання* (якщо процесу в даний момент виділено центральний процесор);
2. *стан готовності* (якщо процес міг би відразу використати центральний процесор, наданий в його розпорядження);
3. *стан блокування* (якщо процес очікує появу деякої події, щоб продовжити виконання);
4. *стан призупинений блокований;*
5. *стан призупинений готовий*.

ОС створює список готових до виконання і заблокованих процесів. Список готових процесів впорядковується за пріоритетами. Наступним процесом, який отримує центральний процесор, буде перший процес з цього списку. Розблокування процесів виконується в тому порядку, в якому відбуваються очікувані иподії.

Процес записується до списку готових процесів відповідно до свого пріоритету. Якщо всі процеси мають однаковий пріоритет, то процес записується

в кінець списку. По мірі завершення виконання попередніх процесів процес переміщується до головної частини списку. Коли він стає першим в списку готових процесів, то по завершенні попереднього процесу йому виділяється центральний процесор (ЦП) і він переходить із стану готовності в стан виконання. По завершенні кванту часу, виділеного процесу, центральний процесор.

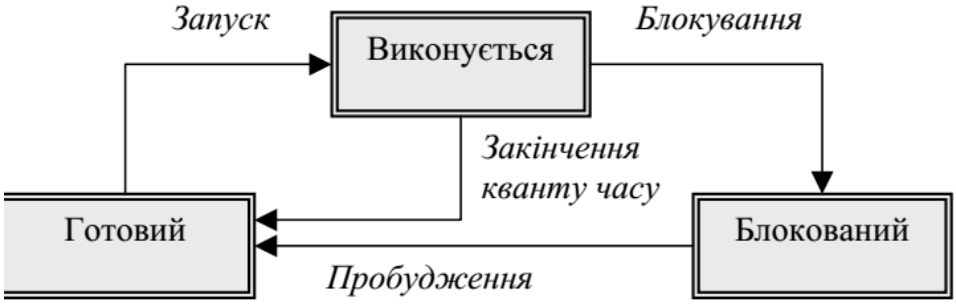


Рисунок 2.1 - Діаграма станів процесу

Представником процесу в ОС є **блок керування процесом PCB (programm control block).** Це структура даних, яка містить інформацію про даний процес, в тому числі поточний стан процесу, пріоритет, унікальний ідентифікатор процесу, покажчик виділених ресурсів, пам'яті, область зберігання регістрів. В PCB, наприклад, може бути записана інформація, необхідна для перезапуску процесу. В багатьох ЕОМ передбачена апаратна підтримка операцій з PCB. Вводиться спеціальний регістр, який вказує на PCB поточного процесу; є спеціальні команди процесора, які забезпечують швидкі операції з PCB.

2.2. Основні операції над процесами

Системи керування процесами повинні мати можливість виконувати деякі операції над процесами, в тому числі:

1. створення процесу;
2. знищення процесу;
3. призупинення процесу;
4. відновлення процесу;
5. зміна пріоритету процесу;

6. блокування процесу;
7. пробудження процесу;
8. запуск (вибір) процесу.

Створення процесу складається з операцій присвоєння імені процесу, включення цього імені в список імен процесів, відомих системі, визначення початкового пріоритету процесу, формування блоку керування процесом, виділення процесові початкових ресурсів. Процес може породити новий процес, який називається дочірнім, а той, що породжує - батьківським.

Знищення процесу. Процес видаляється з системи, ресурси віддаються системі, ім'я процесу стирається зі списків імен відомих ОС, а РСВ звільняється. В деяких системах дочірні процеси знищуються при знищенні батьківського, в інших продовжують існування незалежно від батьківських.

Зміна пріоритету означає зміну значення пріоритету в блоці керування даним процесом.

Призупинення процесу бувають короткі і довготривалі. У випадку тривалого призупинення ресурси процесу повинні бути звільнені. Причинами призупинень можуть бути такі явища:

a) система працює не цілком надійно, є ознаки можливої відмови. Процес відновлюється після виправлення помилки;
b) призупинення, ініціатором яких є користувач, який має сумнів в правильності проміжних результатів;
c) призупинення під час короткочасних піків навантаження системи.

Відновлення (активація) - операція підготовки процесу до повторного запуску з тієї ж точки, в якій він був призупинений.

Процес, що виконується в однопроцесорній системі, може призупинити лише сам себе. В усіх інших станах процеси можуть бути призупинені іншими процесами.

2.3. Обробка та типи переривань. Ядро ОС

Переривання *- це подія, під час якої змінюється нормальна послідовність команд, що виконується процесором.* Під час переривання виконується така послідовність дій:

a) керування передається операційній системі;

b) операційна система запам'ятовує стан перерваного процесу, інколи PCB перерваного процесу;

c) ОС аналізує тип переривання і передає керування відповідній програмі обробки переривань;

d) після обробки переривання ОС може відновити виконання перерваного процесу.

Типи переривань:

1. *SVC (Super Visor Call) - переривання.* Це запит, який генерується програмою користувача для отримання конкретної системної послуги (вимога на операцію введення-виведення, збільшення розміру виділеної пам'яті, зв'язок з оператором). Механізм SVC дозволяє захистити ОС від користувача. Ця команда повинна мати параметри для вибору функції ОС (аналог в MS-DOS - INT 21H).
2. *Переривання введення-виведення.* Ці переривання ініціюються апаратурою введення-виведення. Вони сигналізують центральному процесору про зміни стану каналу або пристроїв введення-виведення.
3. *Зовнішні переривання.* Причинами цих переривань можуть бути різні події - закінчення відрізку часу, натиснення кнопки переривання на пульті, сигнал переривання від іншого процесора в мультипроцесорній системі.
4. *Переривання по рестарту* (коли користувач натискає кнопку рестарту на пульті керування або коли інший процесор передає команду рестарту).
5. *Переривання по контролю (помилці) програми* (ділення на нуль, трасування та ін.).
6. *Переривання по контролю (помилці) машини* (апаратні помилки).

Для обробки кожного з цих переривань в складі операційної системи передбачені програми, які називаються обробниками переривань (IH -Interrupt Handler).

Всі операції, пов'язані з процесами, виконуються під керуванням ядра ОС. Це лише невелика частина ОС, але яка дуже часто використовується, і тому вона резидентно розміщується в основній пам'яті. Інші частини ОС переміщуються в зовнішню пам'ять і назад по мірі необхідності.

Основні функції ядра ОС такі:

1. обробка переривань;
2. створення і знищення процесів;
3. переключення процесу із стану в стан;
4. синхронізація процесів, організація взаємодії між процесами;
5. маніпулювання РСВ;
6. підтримка операцій введення-виведення;
7. підтримка розподілу і перерозподілу пам'яті;
8. підтримка роботи файлової системи;
9. підтримка механізмів виклику і повернення під час звертання до процедур;
10. підтримка функцій по веденню обліку роботи машини.

Реалізація ядра. Значна частина функцій ядра реалізується мікро-програмними засобами, що забезпечує більшу швидкість виконання функцій операційних систем .

2.4. Класифікація процесів і види відношень між ними

За часом розвитку процеси поділяються на:

1. *процеси реального часу* - гарантовано закінчуються до настання конкретного моменту часу;
2. *інтерактивні процеси* - існують не більше інтервалу часу допустимої реакції ЕОМ на запит користувача;

3. *пакетні процеси* - процеси, що не ввійшли в попередні класи;
4. *еквівалентні процеси* - мають однаковий результат обробки одних і тих же вхідних даних, по одній і тій же або різних програмах, на одному і тому ж або різних процесорах;
5. *тотожні процеси* - це еквівалентні процеси, в яких обробка даних відбувається за однією тією ж програмою, але траси не збігаються. (Траса - порядок і тривалість перебування процесу в допустимих станах на інтервалі існування).
6. *рівні процеси* - тотожні процеси, у яких траси збігаються;
7. *послідовні процеси* - інтервали існування не перетинаються в часі;
8. *паралельні процеси* - існують одночасно на певному відрізку часу;
9. *комбіновані процеси* - на деякому інтервалі часу є хоча б одна точка, в якій вони існують одночасно.

За місцем розвитку процеси бувають:

1. *внутрішні процеси* - виконуються на центральному процесорі (ЦП) (або процесорах);
2. *зовнішні* - виконуються на процесорах, відмінних від ЦП під керуванням ОС;
3. *процеси користувача* - процеси виконання програм користувача;
4. *системні процеси* - процеси виконання програм ОС.

За способами зв'язків між собою процеси можуть бути взаємозв'язаними, ізольованими, інформаційно-незалежними, взаємодіючими і конкурентними.

1. Взаємозв'язані процеси - це такі, між якими за допомогою системи керування процесами підтримуються зв'язки: функціональні, просторово-часові, зв'язки з керування, інформаційні.

2. Ізольовані процеси - це процеси, які не підтримують зв'язки один з одним. Повністю ізольованих процесів в ОС не існує, є процеси зі слабкими зв'язками.

3. Інформаційно-незалежні процеси - це взаємозв'язані процеси, які спільно використовують ресурси, але інформаційного зв'язку не мають.

4. Взаємодіючі процеси - це взаємозв'язані процеси, які мають інформаційні зв'язки між собою

5. Конкурентні процеси - це взаємозв'язані процеси, які мають зв'язок по ресурсах.

Взаємозв'язані процеси вступають у відношення між собою.

Види відношень між процесами:

1. *Відношення передування* - для двох процесів це означає, що перший процес повинен переходити в активний стан раніше другого.
2. *Відношення пріоритетності* - процес з пріоритетом Р може бути переведений в активний стан під час виконання двох умов:

- в стані готовності до даного процесора немає процесів з вищим пріоритетом;
- даний процесор або вільний в цей час, або використовується процесом з нижчим пріоритетом.

3. *Відношення взаємовиключення* - декілька процесів використовують спільний ресурс; сукупність дій над спільним ресурсом в складі одного процесу називається критичною секцією; критична область одного процесу не повинна виконуватись одночасно з критичною областю іншого.

Особливості кожної конкретної взаємодії між двома або більше процесами визначаються ***задачами синхронізації.***

Спеціальні засоби операційних систем (ОС) встановлюють деякі обмеження на послідовність виконання взаємозв'язаних паралельних процесів, забезпечуючи тим самим їх синхронізацію та спілкування. Невірна реалізація задач синхронізації може призвести до тупиків у системі. Для розв'язання задач синхронізації можуть застосовуватись різні механізми синхронізації. В багатьох ЕОМ є механізми апаратної підтримки алгоритмів синхронізації.

2.5. Взаємовиключення. Алгоритм Деккера

Розглянемо наступний приклад. Декілька паралельних процесів звертаються до спільної змінної P1 і додають до неї "1". Кожний процес має власну копію фрагменту програми:

LOAD P1

ADD 1

STORE P1.

Нехай P1=20. Припустимо, що один із процесів виконав команди завантаження LOAD і додавання ADD, після чого значення акумулятора стало рівним 21. Уявимо, що в цей момент закінчився виділений квант часу і перший процес віддає процесор другому процесу. Другий процес виконує всі три команди і встановлює значення змінної P1 рівним 21. Потім керування передається першому процесу, який також присвоює змінній P1 значення 21. Таким чином, через нескоординований доступ до спільної змінної система фактично губить одне звертання до змінної P1. Вірна загальна сума повинна скласти 22.

Цю задачу можна розв'язати, якщо кожному процесу дати монопольне, виключне право доступу до змінної P1. Коли один процес збільшує цю змінну, всі інші процеси, яким також необхідно її змінити в цей самий час, очікують. Це і називається ***взаємовиключенням***.

Коли процес звертається до поділюваних даних, то говорять, що він знаходиться на *критичній ділянці.* Якщо будь-який процес знаходиться на критичній ділянці, то інші процеси можуть виконуватись, але без входу на цю критичну ділянку.

Найпростішим виходом із цієї ситуації було б заборонити переривання на критичній ділянці, але в системах з розподілом часу і в системах реального часу час грає дуже важливу роль. Система може не відреагувати на будь-яку зовнішню подію, або час відповіді збільшиться.

Розглянемо декілька варіантів реалізації взаємовиключення з дотриманням таких обмежень:

1. задача повинна бути розв'язана чисто програмними засобами на машині, що не має спеціальних команд взаємовиключення;
2. не повинно бути ніяких обмежень щодо швидкості виконання асинхронних паралельних процесів;
3. процеси, що знаходяться поза межами своїх критичних ділянок, не заважають іншим процесам входити в їх власні критичні ділянки;
4. не повинно бути нескінченого відтермінування моменту входу процесів у критичні ділянки.

2.6. Семафори. Моніторороподібні засоби синхронізації

Семафор - захищена змінна, значення якої може зчитуватись і мінятись за допомогою спеціальних операцій P, V та операцій ініціалізації.

Розрізняють двійкові та семафори з лічильником. *Двійкові* семафори можуть приймати значення '0' або '1'. Семафори з лічильником можуть приймати невід'ємні цілі значення. Семафори з лічильником корисні, якщо деякий ресурс виділяється з пула ідентичних ресурсів. Під час ініціалізації такого семафора в його лічильнику вказується кількісний покажчик об'єму ресурсів пула.

Операції над семафорами:

- *операція P(Б):* якщо S>0 то S:=S-1
  інакше (очікувати на S);
- *операція V(S):* якщо (один або більше процесів очікують на S)
  то (дозволити одному з них продовжити роботу)
  інакше S:=S+1.

Семафори та операції над ними можуть бути реалізовані як програмно, так і апаратно. Як правило, вони реалізуються в ядрі ОС, де виконують керування зміною станів процесів. За допомогою семафорів можна реалізувати багато задач синхронізації: взаємовиключення, блокування-відновлення, та інші.

Реалізація взаємодії «блокування-відновлення"

Дано два процеси, один з яких може виявити подію, на яку очікує другий процес, але не може виявити її самостійно (рис. 2.6 а).

Реалізація взаємодії "виробник-споживач"

Виробник - це процес, що генерує інформацію. Споживач - це процес, що використовує цю інформацію. Будемо вважати, що вони взаємодіють один з одним за допомогою змінної БуферЧисла (БЧ), і швидкості цих процесів різні. Якщо не вжити спеціальних заходів синхронізації, то інформація буде губитись (споживач повільний), або дублюватись (споживач швидкий). Змінні: ЧЗ - число занесено, ВД -виключний доступ, БЧ - буфер числа (рис. 2.6 б).

```
Programm
блокування_відновлення;
        var     подія : семафор;
procedure Процес1;
   begin
      інші_оператори;
      P(подія);
      інші_оператори;
   end;
procedure Процес2;
   begin
      інші_оператори;
      V(подія);
      інші_оператори;
   end;
BEGIN
   ініціалізація_семафора(подія,
0);
   ParBegin
      Процес1;
      Процес2;
    ParEnd;
END.
```

а)

```
Programm виробник_споживач;
   var ВД : семафор;
               БЧ : ціла;
               ЧЗ : семафор;
procedure процес_виробник;
   var НР : ціла;
   begin
      обчислення      наступного
результату {НР};
      P(ВД);
      БЧ:=НР;
      V(ЧЗ);
   end;
procedure процес_споживач;
   var    НР : ціла;
   begin
      P(ЧЗ);
      НР:=БЧ;
      V(ВД);
      записати        наступний
результат {НР};
   end;
BEGIN
   ініціалізація_семафора(ВД, 1);
   ініціалізація_семафора(ЧЗ, 0);
   ParBegin
      процес_виробник;
      процес_споживач;
    ParEnd;
END.
```

б)

Рис. 2.6. Синхронізація за допомогою семафорів

Операції P і V з циклом активного очікування можуть бути реалізовані за допомогою алгоритму Деккера або команди testandset. Щоб виключити цикли активного очікування, команди операцій над семафорами реалізуються в ядрі ОС. Можлива реалізація неподільності операцій P і V простою забороною переривань (в однопроцесорних системах).

Алгоритм Деккера і семафори мають деякі недоліки. Наприклад, невірне використання цих примітивів може призвести до порушення працездатності системи паралельної обробки. Необхідні механізми, які б:

а) спрощували розв'язання складних задач паралельної обробки;

б) спрощували доведення коректності програм;

в) важко псувались або невірно використовувались.

В значній мірі ці проблеми вирішуються за допомогою ***моніторів***.

Монітор *- це засіб забезпечення паралелізму, який містить як дані, так* і *процедури, необхідні для реалізації конкретного розподілу ресурсу.* Щоб забезпечити виділення необхідного йому ресурсу, процес повинен звернутися до конкретної процедури монітора. В кожний момент часу лише один процес може увійти в монітор. Внутрішні дані монітора доступні лише з середини, що забезпечує розробку надійних програм користувача.

Вводяться примітиви: WAIT - очікування, SIGNAL - сигналізації. За командою WAIT створюється черга процесів, що очікують на деякий ресурс, якщо він зайнятий. SIGNAL виконується, коли деякий ресурс звільняється і повідомляє процеси, що очікували на цей ресурс, про можливість його зайняття. Процеси можуть знаходитись в режимі очікування за межами монітора з різних причин, тому введене поняття змінної умови, а саме: WAIT(умова), SIGNAL(умова).

Тема 3. Керування пам'яттю

3.1. Організація та ієрархія пам'яті

Пам'ять розділяється на байти, слова (декілька байт). Кожна комірка пам'яті має свою адресу. Множина адрес, доступних програмі називається *адресним простором.*

Апаратно-реалізовані функції.

1. *Розшарування пам'яті.* Збільшується швидкість доступу до оперативної пам'яті. Сусідні за адресами комірки розміщуються в різних блоках пам'яті, що дає можливість звертатися одночасно до декількох комірок.
2. *Регістр-переміщення.* Забезпечує можливість динамічного переміщення програм в пам'яті.
3. *Захист пам'яті.* Виконується за допомогою граничних регістрів.
4. *Віртуальна пам'ять.* Дає можливість вказувати в програмі адреси, які не обов'язково відповідають фізичним адресам основної пам'яті. Віртуальні адреси під час роботи програми динамічно перетворюються за допомогою апаратних засобів в реальні адреси основної пам'яті.

Розрізняють такі види пам'яті:

- *зовнішню (вторинну)* пам'ять (ЗП);
- *основну (первинну)* пам'ять (ОП);
- *кеш-пам'ять* (КП).

Зв'язок між різними типами пам'яті показано на рис.3.1.

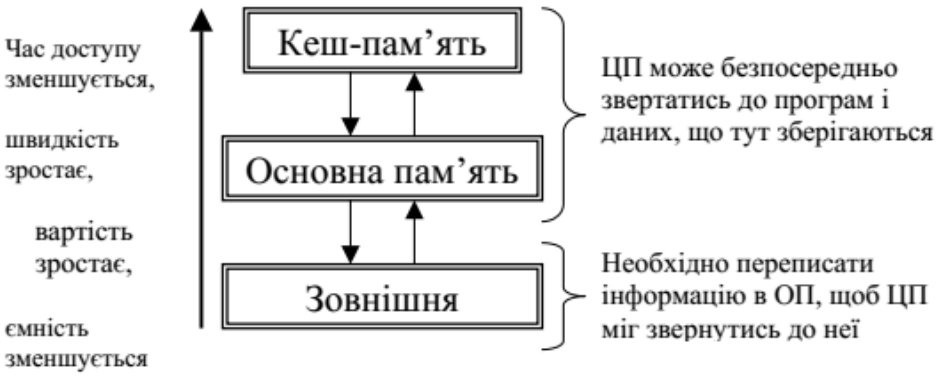


Рисунок 3.1 - Ієрархічна організація пам'яті

3.2. Стратегії керування пам'яттю

Стратегії керування пам'яттю спрямовані на те, щоб забезпечити найкращі можливості використання ресурсів основної пам'яті.

Стратегії керування пам' яттю поділяються:

1. ***Стратегії вибірки*** ставлять за мету визначити, коли необхідно записати черговий блок програми або даних в основну пам'ять. Ця категорія стратегій поділяється на:

a) стратегії вибірки за запитом;

b) стратегії випереджувальної вибірки.

2. ***Стратегії розміщення*** (рис. 3.2) визначають, в яке місце основної пам'яті необхідно помістити програму або дані. Вони поділяються на стратегії:

a) першого придатного,

b) найбільш придатного,

c) найменш придатного.

Стратегія першого придатного - нові дані розміщуються в першій зустрінутій ділянці, що підходить за розмірами.

Стратегія найбільш придатного - програма розміщується в ділянці пам'яті, що найточніше підходить за розміром.

Стратегія найменш придатного - програма займає максимально вільну ділянку пам'яті.

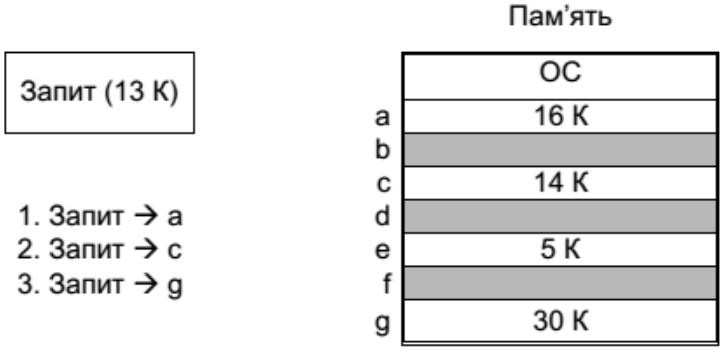


Рисунок 3.2 - Стратегії розміщення інформації в пам'яті (1 - першого придатного, 2 - найбільш придатного, 3 - найменш придатного)

3. ***Стратегії заміщення*** визначають блок програми або даних, який треба вивести з основної пам'яті, щоб звільнити місце для нових даних або програм.

3.3. Розподіли пам'яті

При зв'язному розподілі пам'яті кожна програма записується в один суцільний блок пам'яті. При незв'язному розподілі програма розбивається на ряд блоків (сегментів), які можуть розміщуватись в ділянках пам'яті, які не обов'язково сусідні.

Системи віртуальної пам'яті передбачають незв'язний розподіл пам'яті. Переваги незв'язного розподілу пам'яті в тому, що може виконуватись програма, яку неможливо розмістити в одному блоці. Зв'язний розподіл пам'яті для однопрограмної системи наведено на рис.3.3.

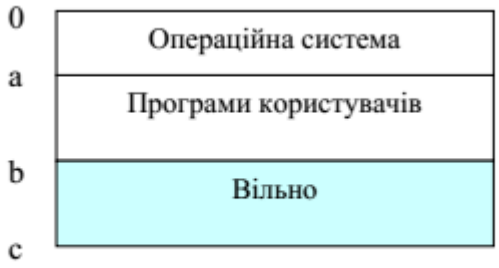


Рисунок 3.3 - Зв'язний розподіл пам'яті в однопрограмних ОС

Трансляція і завантаження в абсолютних адресах

Пам'ять розбивається на розділи фіксованого розміру. В кожному розділі розміщується одне завдання. ЦП швидко переключається з завдання на завдання, створюючи ілюзію одночасного виконання. Розділи мають фіксований розмір. До кожного розділу утворюється своя черга програм, які можуть виконуватись лише в цьому розділі (рис.3.4. а).

Недоліком такої організації пам'яті є неефективне використання ресурсів, оскільки трансляція відбувається в абсолютних адресах, і програма може виконуватись лише в одному розділі.

Трансляція і завантаження переміщуваних модулів

Існує одна черга завдань, які можуть надходити до будь-якого вільного розділу. Однак, існуючі транслятори та завантажувачі є значно складнішими (рис.3.4.б).

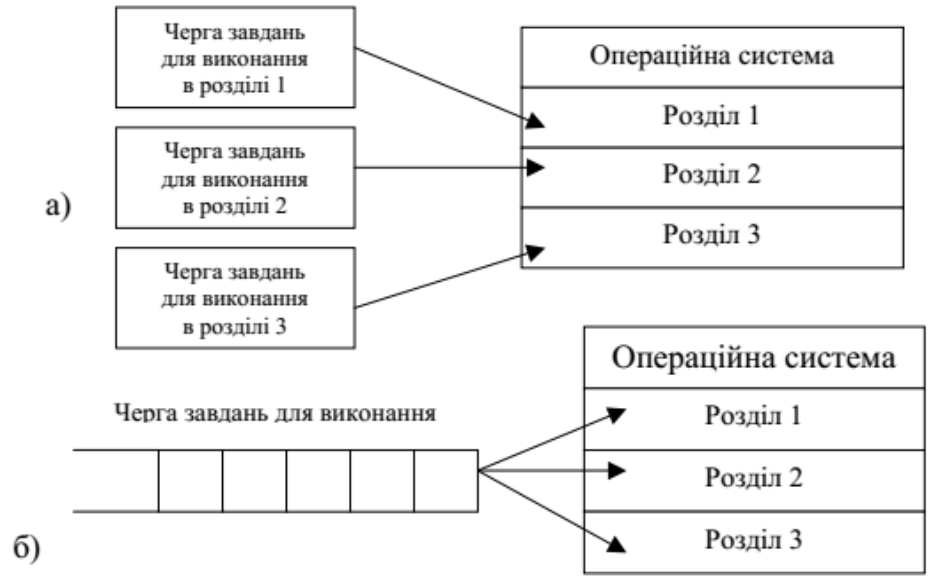


Рис. 3.4. Мультипрограмування з фіксованими розділами

Захист пам 'яті в мультипрограмних системах

В мультипрограмних системах зі зв'язним розподілом пам'яті захист пам'яті реалізується за допомогою декількох граничних регістрів. Програми, які хочуть звернутись до послуг ОС, використовують SVC-команди (перериванням).

Мультипрограмування з змінними розділами при зв'язному розподілі пам'яті не дотримуються фіксованих границь розділів, а навпаки, кожному завданню виділяється стільки пам'яті, скільки необхідно. Однак, коли завдання завершуються, утворюються дірки (фрагментація пам'яті). Долають це явище такими способами:

a) об'єднання вільних сусідніх ділянок пам'яті (злиття);
b) ущільнення пам'яті або "збір сміття" (рис. 3.5).

До недоліків ущільнення пам'яті слід віднести те, що воно забирає ресурси системи: під час проведення ущільнення збільшується час відповіді системи; необхідно зберігати дані про розміщення програм.

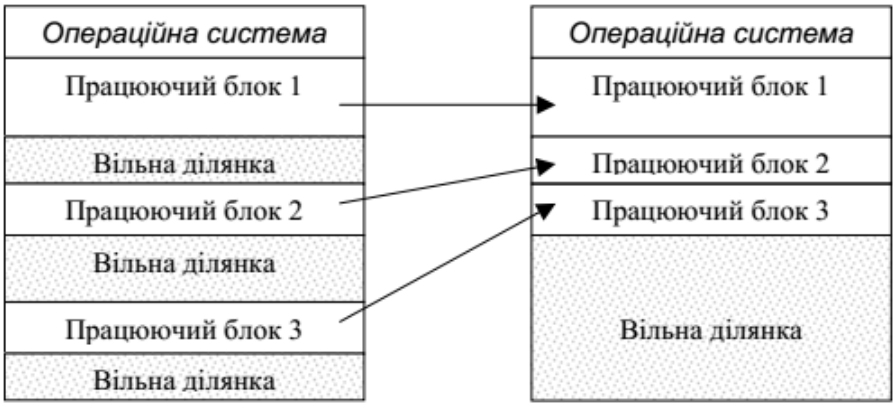


Рис. 3.5. Ущільнення памяті

Мультипрограмування зі свопінгом - це схема розподілу пам'яті, під час реалізації якої програми користувача не зберігаються в основній пам'яті до їх завершення. Пам'ять на короткий період виділяється одному завданню, потім це завдання виводиться і замість нього вводиться інше завдання. Перші системи з розподілом часу були системами зі свопінгом. На основі систем зі свопінгом були створені системи зі сторінковою організацією. Розрізняють системи з свопінгом, в яких в основній пам'яті розміщується:

- одна програма користувача;
- декілька програм користувача.

Тема 4. Організація керування віртуальною пам’яттю

Існує три способи реалізації віртуальної пам'яті: сторінкова; сегментна; комбінована.

В системах віртуальної пам'яті адреси, що формуються програмами, не обов'язково збігаються з адресами реальної основної пам'яті.

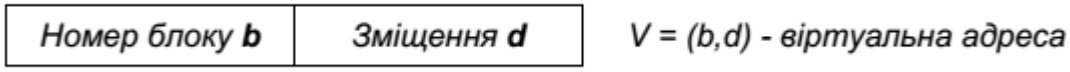
Термін *віртуальна пам 'ять* асоціюється з можливістю адресувати простір пам'яті набагато більший, ніж ємність первинної (реальної, фізичної) пам'яті конкретної обчислювальної машини.

4.1. Механізм відображення віртуальних адрес в реальні

Сутність концепції віртуальної пам'яті полягає в тому, що адреси, які формуються програмами, відокремлюються від адрес реальної пам'яті. *Віртуальні адреси* - це адреси, на які посилається процес, що виконується. Адреси, що існують в реальній основній пам'яті, називаються *реальними адресами.* Діапазон віртуальних адрес називається віртуальним адресним простором. Для встановлення відповідності між реальними і віртуальними адресами застосовуються механізми динамічного перетворення адрес. Суміжні адреси віртуального простору не обов'язково будуть суміжними в реальній пам'яті (штучна суміжність), але користувач звільнений від необхідності враховувати розміщення в реальній пам'яті. Віртуальний адресний простір завжди значно ширший за реальний адресний простір.

Механізм динамічного перетворення адрес повинен вести таблиці, які показували б які комірки віртуальної пам'яті знаходяться в реальній пам'яті і де саме. Якщо б відображення виконувалось побайтно, то ці таблиці займали б більше місця, ніж самі процеси. Тому елементи інформації групуються в блоки і система слідкує, в яких місцях реальної пам'яті знаходяться блоки віртуальної пам'яті.

Формат віртуальної адреси в системі поблочного відображення наступний:

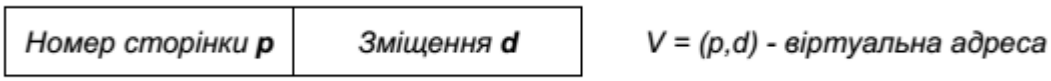


Якщо всі блоки мають однаковий розмір, то така організація віртуальної пам'яті називається *сторінковою,* а блоки сторінками. Якщо блоки можуть бути різних розмірів, то така організація пам'яті називаються *сегментною,* а блоки сегментами.

Кожен процес має свою таблицю відображення адрес, яку система веде в реальній пам'яті. Реальна адреса *а* цієї таблиці завантажується до спеціального регістра ЦП, який називається *регістром початкової адреси* (РПА) таблиці відображення блоків (ТВБ). ТВБ містить по одному рядку для кожного блоку процесу, причому блоки йдуть в послідовному порядку.

4.2. Сторінкова, сегментна та комбінована організації віртуальної пам'яті

Формат віртуальної адреси в системі з сторінковою організацією пам'яті має такий вигляд:



Процес виконується до тих пір, поки його поточна сторінка знаходиться в основній пам'яті. Сторінки переписуються з зовнішньої пам'яті в основну і розміщуються в блоках, які називаються *сторінковими кадрами* і мають розмір сторінки. Сторінка, яка надходить з зовнішньої пам'яті, може бути розміщена в будь-якому вільному сторінковому кадрі. Сторінкові кадри починаються з адрес первинної пам'яті, кратних, як правило, розміру сторінки.

Перетворення адрес сторінок методом прямого відображення

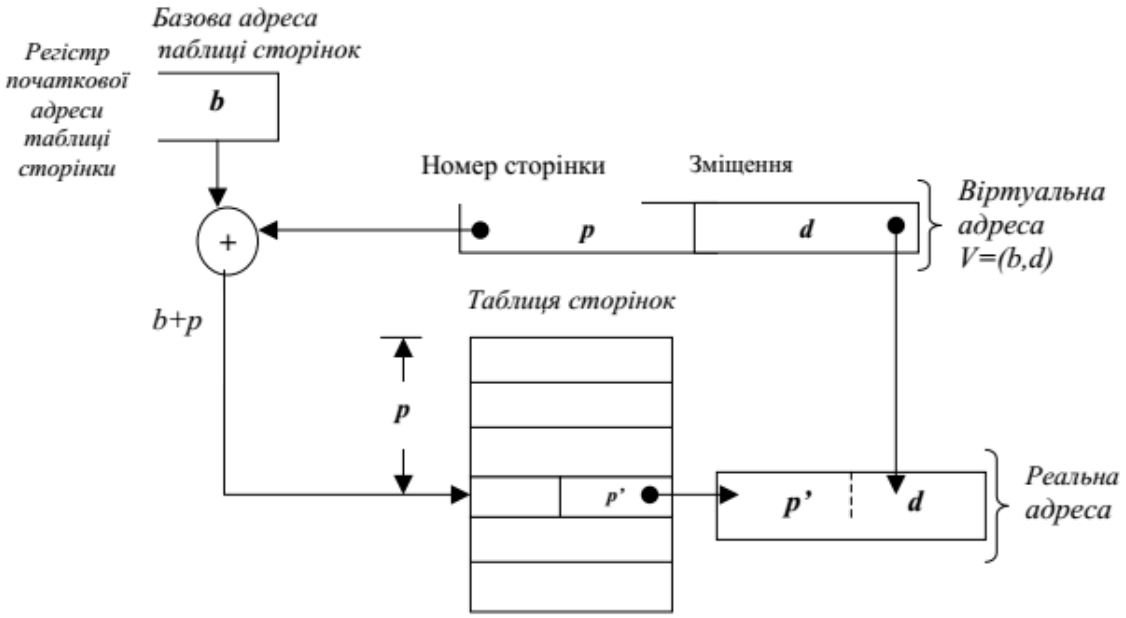


Рис. 4.1. Перетворення адрес сторінок методом прямого відображення

Формат рядка таблиці сторінок може мати такий вигляд:

Біт-ознака присутності сторінки	*Адреса зовнішньої пам'яті (якщо сторінки немає в реальній пам'яті)*	*Номер сторінкового кадру (якщо сторінка знаходиться в реальній пам'яті)*
I	S	p’

Якщо I=0, то сторінки немає в реальній пам'яті, а якщо до неї будуть звертання, то формується переривання за відсутністю сторінки і необхідна сторінка завантажується в реальну пам'ять. Якщо 1=1, то сторінка знаходиться в реальній пам'яті.

Недоліком такого перетворення є те, що швидкість виконання програми знижується майже вдвічі, оскільки необхідно виконувати два цикли зчитування пам'яті.

Перетворення адрес сторінок асоціативним відображенням. Одним з методів прискорення динамічного перетворення адрес є застосування асоціативної пам'яті для розміщення таблиці перетворення сторінок (рис.4.2), оскільки асоціативна пам'ять значно швидша ніж звичайна пам' ять з довільним доступом.

Недоліком асоціативного перетворення є дуже велика вартість.

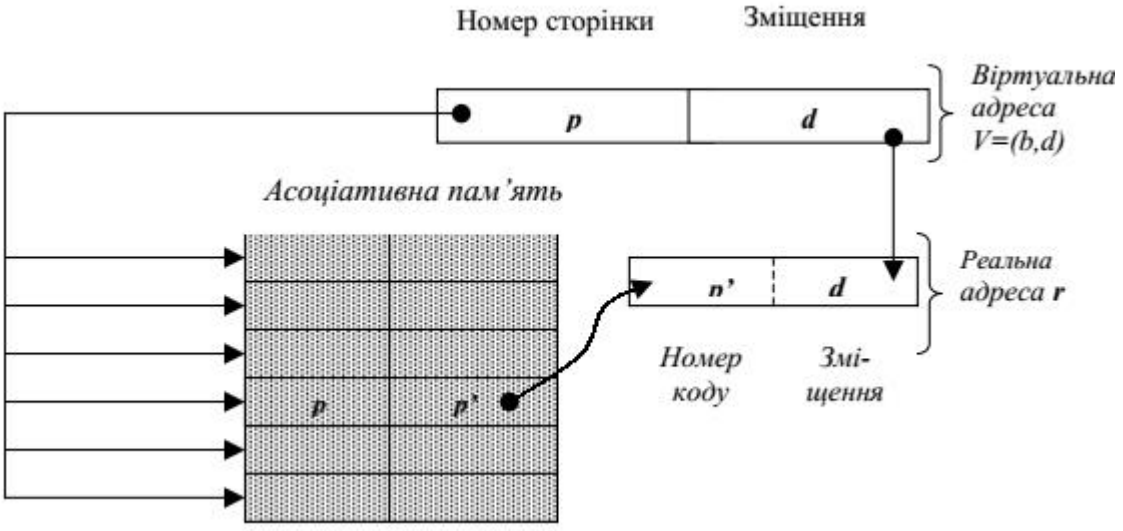


Рис. 4.2. Перетворення адрес сторінок асоціативним відображенням

Перетворення адрес сторінок комбінованим відображенням

Цей підхід передбачає використання асоціативної пам'яті для збереження тільки тих рядків повної таблиці відображення сторінок для процесу, до яких були звертання протягом останнього інтервалу часу. Операція (b+p) виконується лише тоді, коли сторінка не знайдена в асоціативній таблиці. Швидкодія складає 90% від повної асоціативної пам'яті.

В системах з сегментною організацією пам'яті програми можуть займати багато блоків основної пам'яті, причому не обов'язково однакового розміру. Блоки можуть розміщуватись один біля одного, перекриватись частково або повністю. Процес виконується тоді, коли сегмент знаходиться в основній пам'яті. Сегменти передаються до основної пам'яті повністю. В загальному випадку перетворення адрес сегментів може бути прямим, асоціативним, комбінованим (рис. 4.3.).

Формат рядка таблиці сегментів може мати такий вигляд:

Ознака наявності сегмента в пам'яті	*Адреса сегмента в зовнішній пам'яті*	*Довжина сегмента*	*R, W, E, A - біти ознак захисту: R - доступ для читання; W - доступ для запису; E - доступ для виконання; A - доступ для доповнення.*				*Базова адреса сегмента, якщо сегмент знаходиться в ОП*
I	A	L	R	W	E	A	S'

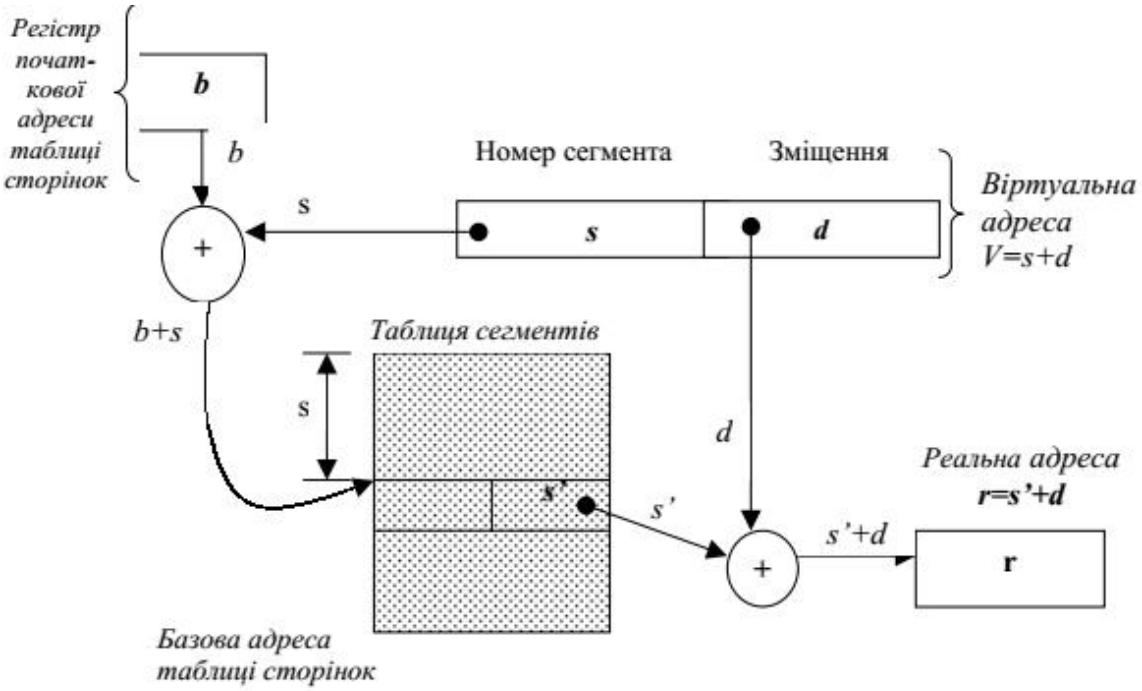
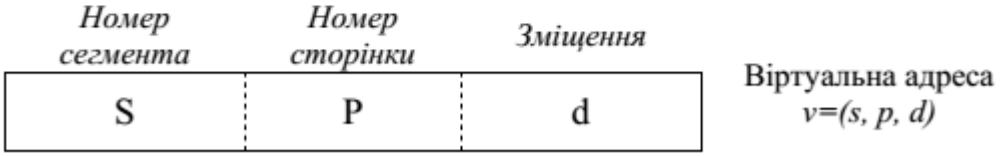


Рис. 4.3. Перетворення віртуальної адреси в сегментній системі

У системах із сторінково-сегментною організацією застосовується трикомпонентна (тривимірна) адресація, тобто віртуальна адреса визначається як функція трьох змінних *V=(s; p; d)*, де *s* - номер сегмента, p - номер сторінки, *d* - зміщення в межах сторінки. Формат віртуальної адреси такий:



Структура таблиць, які система веде при сторінково-сегментній організації. На самому верхньому рівні знаходиться *таблиця процесів* (одна на всю систему), яка містить по рядку для кожного процесу відомого системі, і цей рядок вказує на *таблицю сегментів* цього процесу (по одній на кожен процес). Кожний рядок таблиці сегментів процесу вказує на *таблицю сторінок* відповідного сегмента, а кожний рядок таблиці сторінок вказує або на сторінковий кадр, в якому розміщена дана сторінка, або на адресу зовнішньої пам'яті, де можна знайти цю сторінку. В сторінково-сегментній системі організації віртуальної пам'яті переваги від колективного використання програм і даних більші, ніж при чисто сегментній організації, оскільки в таблицях сегментів для різних процесів є рядки, які вказують на одну й ту саму таблицю сторінок.

Тема 5. Стратегії керування віртуальною пам'яттю

5.1. Основні стратегії

1. ***Стратегії вибірки*** (вштовхування). Їх мета - визначити, в який момент слід переписати сторінку або сегмент із вторинної пам'яті в первинну. *Вибірка за запитом* передбачає, що система очікує посилання на сторінку або сегмент, і тільки дочекавшись цього посилання від працюючого процесу, переписує дану сторінку або сегмент у первинну пам'ять. При *вибірці з випередженням* передбачається, що система заздалегідь намагається визначити, до яких сторінок або сегментів звернеться процес. Якщо вірогідність звернення висока і в первинній пам'яті є в наявності вільне місце, то відповідні сторінки або сегменти будуть переписані в первинну пам' ять ще до того, як до них відбудеться звернення.

2. ***Стратегії розміщення.*** В системах зі сторінковою організацією розміщення вирішуються просто. Будь-яка сторінка, що надходить із зовнішньої пам'яті, розміщується у вільний сторінковий кадр, бо всі сторінки одного розміру. В системах з сегментною організацією потрібні стратегії розміщення, розглянуті вище, в розділі, що стосувався мультипрограмування із змінними розділами.

3. ***Стратегії заміщення*** (виштовхування). Їх основна мета - вирішити, яку сторінку або сегмент необхідно видалити з основної пам'яті, щоб звільнити місце для сегмента або сторінки, що надходять, якщо первинна пам'ять повністю зайнята. Розглянемо такі стратегії заміщення сторінок:

1. Принцип оптимальності - необхідно заміняти ту сторінку, до якої в подальшому не буде звертань протягом довгого часу. Недоліком цього принципу є те, що його важко реалізувати, оскільки важко прогнозувати майбутнє.
2. Заміщення випадкової сторінки - застосовується рідко, бо можна замінити сторінку, до якої багато звертань.
3. Принцип LIFO (Last in First Out) - "виштовхування" першої сторінки, що надійшла, тобто тієї, яка знаходилась в пам'яті найдовше. Однак, цей

принцип може призводити до заміщення активних сторінок. Той факт, що сторінка постійно знаходиться в оперативній пам'яті, може свідчити про те, що вона постійно в роботі.

4. Принцип LRU (Least recently used) - виштовхування сторінки, яка довше всіх не використовується. Цей принцип використовується рідко. Може статися так, що сторінка, до якої не було звернень, може стати потрібною.
5. Принцип LFU (Least frequently used) - виштовхування сторінки, яка використовується рідше за інші. Враховується інтенсивність використання сторінок, але інтенсивність використання сторінки може змінитись в майбутньому.
6. Принцип NUR - виштовхування сторінки, яка не використовувалась останнім часом. Дуже близька стратегія до LRU. Вводяться два апаратних біти:

а) *біт-ознака звернення* = 0, якщо до сторінки не було звернень і
= 1, якщо звернення були;

б) *біт-ознака модифікації* = 0, якщо сторінка не змінювалась і
= 1, якщо сторінка змінювалась. Біти-ознака звернення періодично скидаються в "нуль" операційною системою. Алгоритм NUR передбачає існування чотирьох груп сторінок: група 1 - звертань і модифікацій не було; група 2 - звертань не було, модифікації були; група 3 - звертання були, модифікацій не було; група 4 - звертання і модифікації були. Сторінки груп з меншими номерами виштовхуються в першу чергу.

5.2. Локальність

Більшість стратегій керування пам'яттю базуються на концепції локальності. Її сутність в тому, що розподіл запитів процесів на звертання до пам'яті має, як правило, нерівномірний характер з високим ступенем локальної концентрації. Розрізняють часову і просторову локальність.

Часова локальність означає, що до комірок пам'яті, до яких недавно проводилось звертання, з великою ймовірністю будуть звертання в найближчому майбутньому з таких причин:

1. наявність програмних циклів;
2. наявність підпрограм;
3. наявність стеків;
4. наявність змінних, що використовуються як лічильники для накопичення сум.

Просторова локальність означає, що якщо були звертання до деякої комірки пам'яті, то з великою ймовірністю можна очікувати звертання до сусідніх комірок. Причини цього такі:

1. організація даних у вигляді масивів;
2. послідовне виконання коду програми;
3. тенденція програмістів розміщувати поруч описи зв'язаних змінних.

Наслідком локалізації звернень до пам'яті є те, що програма може ефективно виконуватись, коли в пам'яті знаходиться множина найбільш популярних сторінок. На основі вивчення властивостей локальності Денінг сформулював *теорію робочих множин:* допоки в основній пам'яті є множина найбільш інтенсивно використовуваних сторінок процесу, частота переривань за відсутності сторінки змінюється мало.

5.3. Робочі множини та підкачки сторінок

Робоча множина - це множина сторінок, до яких процес активно звертається протягом деякого відрізку часу свого виконання. Для ефективного виконання процесу необхідно, щоб його робоча множина знаходилась в первинній пам'яті. В іншому випадку виникає режим інтенсивної підкачки сторінок, причому це можуть бути одні і ті самі сторінки. Такий режим називається *пробуксовкою (трешингом)*.

Робоча множина сторінок W(t, w) в момент часу ї є набір сторінок до яких процес звертався протягом інтервалу [т-w; т]. Змінна називається розміром вікна робочої множини.

Реальна робоча множина процесу - це множина сторінок, які повинні знаходитись в первинній пам' яті, щоб процес міг ефективно виконуватись. Під час роботи процесу його робоча множина динамічно змінюється.

За використання саме **стратегії підкачки сторінок** говорять такі аргументи:

1. шлях, який вибере програма, передбачити неможливо;
2. в оперативній пам'яті будуть знаходитись лише необхідні сторінки;
3. накладні витрати на визначення сторінок, які потрібно передати до оперативної пам'яті, мінімальні.

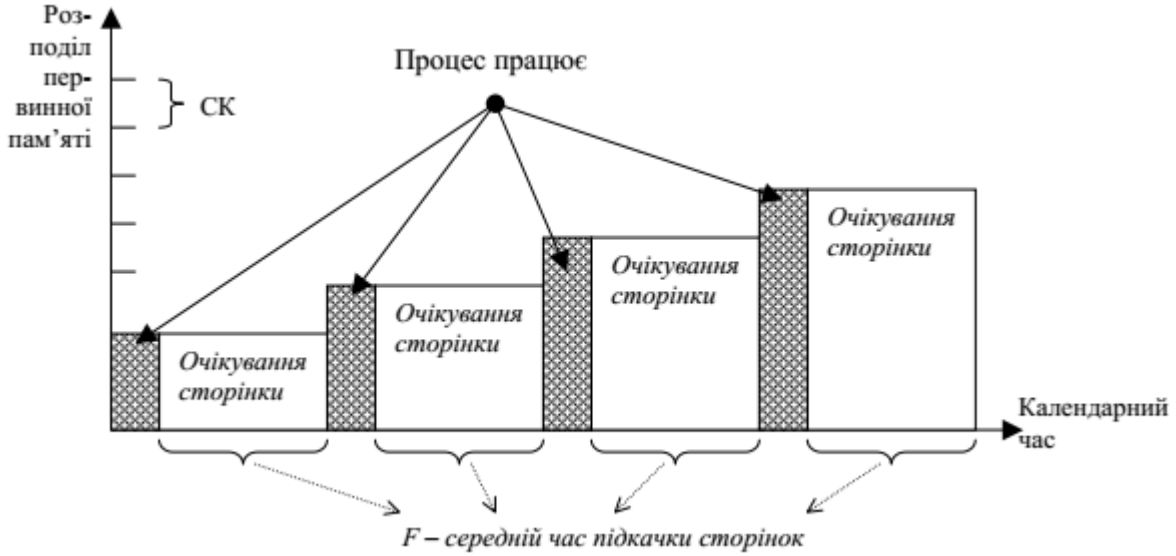


Рис. 5.1. Підкачка сторінок за запитом

Недоліки підкачки сторінок за запитом такі: великі періоди очікування на підкачку сторінок, тобто добуток "простірвчас" занадто великий.

Добуток "простірвчас" - це комплексний покажчик, що відображає і об' єм, і час використання пам' яті процесом. Зменшення цього добутку за рахунок періоду очікування процесом потрібних йому сторінок є головною метою всіх стратегій керування пам'яттю.

Система намагається заздалегідь передбачити, які сторінки потрібні процесу, а потім, коли в основній пам'яті з'являється вільне місце, завантажує туди підготовлені сторінки. Поки процес працює зі своїми поточними сторінками, система вже переглядає сторінки, готує їх до використання.

Переваги цього методу полягають в тому, що

I. якщо правильно вибрати рішення з вибірки сторінки, то час виконання процесу значно зменшиться;

II. оскільки апаратура дешевіє, то наслідки неоптимальних рішень стають менш важливими. Можна придбати додаткову пам'ять, щоб забезпечити накопичування додаткових сторінок, які при даному механізмі підкачки будуть передаватись в основну пам'ять.

Звільнення сторінок та їх розмір

Сторінки, які більше не потрібні, повинні якимось чином виводитися з робочих множин. Існує деякий період часу, під час якого непотрібні сторінки все ж таки ще залишаються в основній пам'яті. Ця проблема буде вирішена за допомогою компіляторів та операційних систем, які будуть автоматично виявляти ситуації, що потребують звільнення сторінок, причому робити це набагато ефективніше.

Щодо розміру сторінок, то для правильного підбору необхідно враховувати багато факторів, як то:

1. чим менший розмір сторінки, тим більша кількість сторінок та сторінкових кадрів буде в системі і тим більші будуть розміри таблиці сторінок. Якщо таблиці сторінок займають первинну пам' ять, то, з огляду на це, треба збільшувати розміри сторінок;

2. при великих розмірах сторінок до первинної пам'яті потрапляє велика кількість інформації, яка не завжди може знадобитися. З цих міркувань треба зменшити розміри сторінок;

3. оскільки обмін даними з дисковою пам'яттю займає відносно багато часу, і ми хочемо звести до мінімуму число обмінів, то слід збільшувати розміри сторінок.

Тема 6. Керування процесорами

Процеси мають можливість виконувати певну роботу, коли в їх розпорядженні є фізичні процесори. В даній главі розглядаються питання, коли, в яких випадках і яким процесам необхідно виділяти процесор, іншими словами, *питання планування завантаження процесора.*

6.1. Рівні планування в ОС

Процеси можуть виконувати конкретну роботу, якщо отримують в користування фізичні процесори. Розрізняють три основні рівні планування в ОС:

• *планування верхнього рівня,* яке деколи називають плануванням завдань або плануванням допуску. Засоби цього рівня визначають яким завданням буде дозволено конкурувати за захват ресурсів системи. Завдання, що входять у систему, стають процесами або групами процесів (рис.6.1);

• *планування проміжного рівня.* Засоби цього рівня визначають, яким процесам буде дозволено змагатися за захват центрального процесора (ЦП). Планувальник проміжного рівня оперативно реагує на коливання системного навантаження, короткочасно призупиняє і знову активізує процеси, що забезпечує рівномірну роботу системи і допомагає досягненню глобальних цільових швидкісних характеристик;

• *планування нижнього рівня.* Засоби цього рівня визначають, якому із готових до виконання процесів буде виділений вільний ЦП і фактично виділяють ЦП даному процесу (тобто, виконують диспетчерські функції). Планувальник нижнього рівня називають *диспетчером.* Він працює з великою частотою, багато раз у секунду, і тому завжди повинен бути в основній пам'яті.

Рис. 6.1. Рівні планування

Планування з переключенням і без переключення

Якщо після передачі ЦП деякому процесу, відібрати ЦП не можна до завершення процесу, то говорять про дисципліну *планування без переключення.* Якщо ЦП відібрати можна, то це дисципліна *планування з переключенням.* Планування з переключенням потрібно в системах реального часу або інтерактивних системах, де необхідно гарантувати прийнятний час відповіді.

Обидва підходи мають свої недоліки:

- в схемі з переключенням в основній пам'яті треба розміщувати декілька процесів, щоб швидко перемикались з одного на інший, перемикання збільшує час виконання процесів;
- в схемах без перемикання довго очікують короткі завдання.

Перемикання здійснюється за рахунок інтервального таймера і переривань. За перериванням від таймера керування передається ОС, яка вирішує, якому процесу віддати ЦП.

Пріоритети процесів

Пріоритети процесів можуть назначатися ззовні або присвоюватись автоматично, можуть бути зароблениими або купленими, статичними або динамічними. Статичні пріоритети не змінюються під час виконання процесу, механізми статичної пріоритетності легко реалізуються. Динамічні пріоритети змінюються в залежності від змін зовнішньої для даного процесу ситуації, а тому схеми динамічної пріоритетності набагато складніші в реалізації. Куплені пріоритети купуються за гроші користувачами.

6.2. Дисципліни планування

Ідеї мультипрограмування пов'язані з наявністю черг процесів. Черги мають місце при звертанні до каналів, модулів ОС, наборів даних, ЦП. Використання багатьма процесами того чи іншого ресурсу, який в кожен момент часу може обслуговувати тільки один процес, виконується за допомогою дисциплін розподілу ресурсів, основою яких є: дисципліни формування черг на ресурси та дисципліни обслуговування черг.

Дисципліни обслуговування черг - це сукупність правил, які визначають порядок вилучення процесу з черги з подальшим наданням ресурсу цьому процесу для використання.

Дисципліни формування черг - це сукупність правил, які визначають порядок розміщення процесів в черзі.

Для формування і обслуговування черг використовуються одні й ті ж або близькі дисципліни планування.

1. ***Планування за терміном завершення.*** Виконання певних завдань повинно закінчуватись раніше певного терміну, наприклад, за додаткову оплату.
2. ***Планування за принципом FIFO (first-in-first-out).*** Найбільш простою дисципліною планування є принцип FIFO: першим прийшов - першим обслуговується.

3. ***Планування за принципом LIFO (last-in-first-out).*** Останній прийшов обслуговується першим. Це дисципліна планування без переключення. Проста в реалізації. Недоліки - ті ж, що і в FIFO.
4. ***Циклічне планування (RR - round robin).*** При цьому плануванні диспетчирування виконується за принципом FIFO, однак кожного разу процесу виділяється обмежена кількість часу ЦП, яка називається часовим квантом. Якщо процес не закінчується за виділений квант часу, то ЦП у нього відбирається і віддається наступному процесу. Процес, у якого перехопили ЦП, переходить у кінець списку готових до виконання процесів.
5. ***Планування за принципом SJF (shortest-job-first).*** При плануванні SJF найбільш коротке завдання виконується першим. Це дисципліна планування без переключення. Основна проблема, яка виникає при реалізації цього принципу, полягає в необхідності визначення часу, протягом якого буде виконуватись завдання. Цей принцип забезпечує зменшення середнього часу очікування в порівнянні з FIFO, але час очікування коливається в широких межах. Користувачі можуть обманювати систему, задаючи мінімальний оціночний час виконання.
6. ***Планування за принципом SRT (shortest-remaining-time)*** - планування за найменшим часом, що залишився до завершення завдання. Це аналог SJF з переключенням. Короткі завдання виконуються майже відразу. Може використовуватись в системах з розподілом часу. Недоліки: велика дисперсія часу очікування.
7. ***Планування за принципом HRN (highest-response-next)*** - планування за найбільшим відносним часом реакції. Це дисципліна планування без переключення, відповідно до якої пріоритет кожного завдання є не тільки функцією часу обслуговування цього завдання, але також і часу, витраченого завданням на очікування обслуговування. Після того, як завдання отримує ЦП, воно виконується до завершення. Динамічно пріоритети обчислюються за формулою:

Пріоритет = (Час очікування + Час обслуговування) / Час обслуговування

8. ***Багаторівневі черги.*** Механізм планування повинен:
   - віддавати перевагу коротким завданням;
   - віддавати перевагу завданням, які лімітуються введенням-виведенням;
   - якомога швидше визначати характер завдання і відповідним чином планувати його виконання.

6.3. Мультипроцесорні системи

З появою дешевих мікропроцесорів та їх комплектів з'явилась можливість створювати недорогі мультипроцесорні системи (МПС). Перевагами МПС є їх надійність і висока продуктивність. Головна мета застосування мультипроцесорних систем збільшення продуктивності за рахунок використання паралелізму під час виконання програм. Однак є багато складностей:

1. люди думають послідовними категоріями;
2. мови паралельного програмування не отримали достатнього поширення;
3. налагоджувати паралельні програми набагато складніше, ніж послідовні.

Реальний прогрес можливий лише тоді, коли ОС і компілятори зможуть автоматично виявляти і виконувати розпаралелення.

Паралелізм в програмах може бути *явним або неявним. Явний паралелізм* програміст сам вказує у своїй програмі. Однак, це пов'язано з рядом негативних моментів. Так, програми з явним вказуванням паралелізму стають більш складними для їх модифікації. З другого боку, спроможність явного вказування паралелізму вимагає від програміста високої кваліфікації. Малоймовірно, що ним будуть указані всі можливі ситуації.

Більш реальним є виявлення *неявного паралелізму* за допомогою спеціальних механізмів розпаралелювання, які включаються в операційні системи,

компілятори та апаратні засоби комп'ютерів. Найбільш поширеними способами пошуку неявного паралелізму: розщеплення циклу; редукція висоти дерева; правило "ніколи не чекати".

Розщеплення циклу

У програмному циклі передбачається багатократне виконання деякої послідовності операторів, які називаються тілом циклу, до тих пір, поки не стане істинною відповідна умова завершення. В деяких випадках у тіло циклу входять оператори, які можуть виконуватись паралельно. Розглянемо такий цикл.

```
FOR i=1 TO 4 DO
A(i) = B(i) + C(i);
```

У даному програмному циклі послідовний процесор виконує по черзі такі операції:

```
A(1)=B(1)+C(1);
A(2)=B(2)+C(2);
A(3)=B(3)+C(3);
A(4)=B(4)+C(4).
```

У мультипроцесорній системі ці оператори можна виконувати паралельно за допомогою 4-х процесорів, завдяки чому значно зменшується час виконання всього циклу.

Компілятор, який виконує автоматичне розпаралелення, може перетворити приведений вище цикл у таку форму:

```
COBEGIN;
    A(2)=B(2)+C(2);
    A(3)=B(3)+C(3);
    A(4)=B(4)+C(4);
COEND;
```

де конструкція COBEGIN/COEND вказує обчислювальній системі на можливість паралельного виконання операторів.

Редукція висоти дерева

Застосовуючи асоціативні, комутативні та дистрибутивні властивості арифметики, компілятори можуть виявляти неявний паралелізм в алгебраїчних виразах і генерувати об'єктний код для мультипроцесорних систем з указанням операцій, які можуть виконуватись одночасно.

Звичайний компілятор діє за такою схемою:

1. Виконуються операції у вкладених дужках, тобто ті операції, які мають більш глибокий рівень вкладеності.
2. Потім виконуються операції в дужках.
3. Виконуються операції піднесення до степіні.
4. Потім виконуються операції множення і ділення.
5. Дальше - операції додавання і віднімання.
6. Якщо операції є рівноцінними, то вони виконуються зліва направо.

Правило "ніколи не чекати "

Правило "ніколи не чекати" говорить по те, що краще доручити деякому процесору роботу, яка або буде, або не буде використана в подальшому, ніж залишити цей процесор бездіяльним.

Стратегія "ніколи не чекати" полягає в тому, щоб завжди виконувався третій оператор паралельно з першим. Якщо значення D не зміниться, то результат виконання третього оператора буде вже готовий і все обчислення завершиться швидше.

6.4. Організація мультипроцесорної апаратури

Дійсно мультипроцесорні комплекси можна охарактеризувати таким чином:

1. Мультипроцесорний комплекс містить два і більше процесорів з приблизно однаковою продуктивністю.
2. Процесори мають доступ до спільної пам'яті.

3. Всі процесори мають колективний доступ до каналів, контролерів і пристроїв введення-виведення.
4. Весь комплекс працює під керуванням однієї ОС.

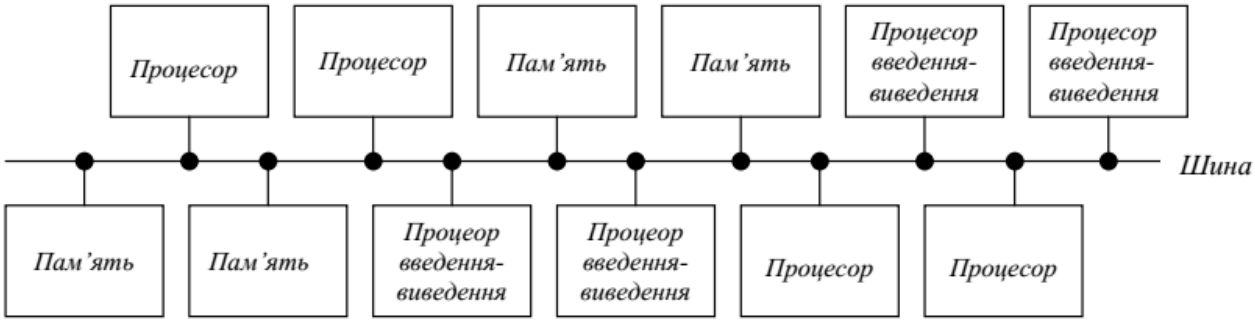
Розрізняють три найбільш розповсюджені види організації мультипроцесорних комплексів:

1. з спільною шиною;
2. з матрицею координатної комутації;
3. з багатопортовою пам'яттю.

Організація з спільною шиною використовує один канал зв'язку між всіма функціональними пристроями. Архітектура з спільною шиною дозволяє легко вводити нові пристрої, підключаючи їх безпосередньо до шини. Недоліки спільної шини такі:

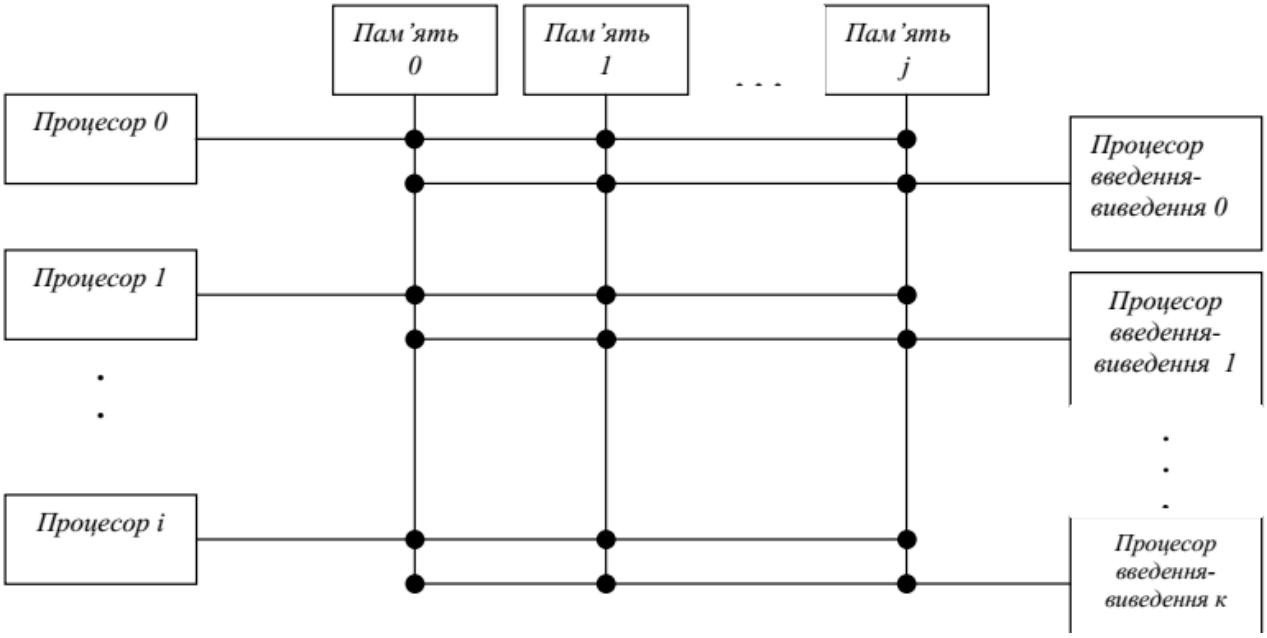
- несправності в шині виводять з ладу весь комплекс;
- швидкість передачі даних в системі обмежена пропускною здатністю шини.

Незважаючи на економічність і простоту через указані недоліки спільна шина застосовується лише в побудові невеликих мультипроцесорних комплексів.



Матриця координатної комутації.

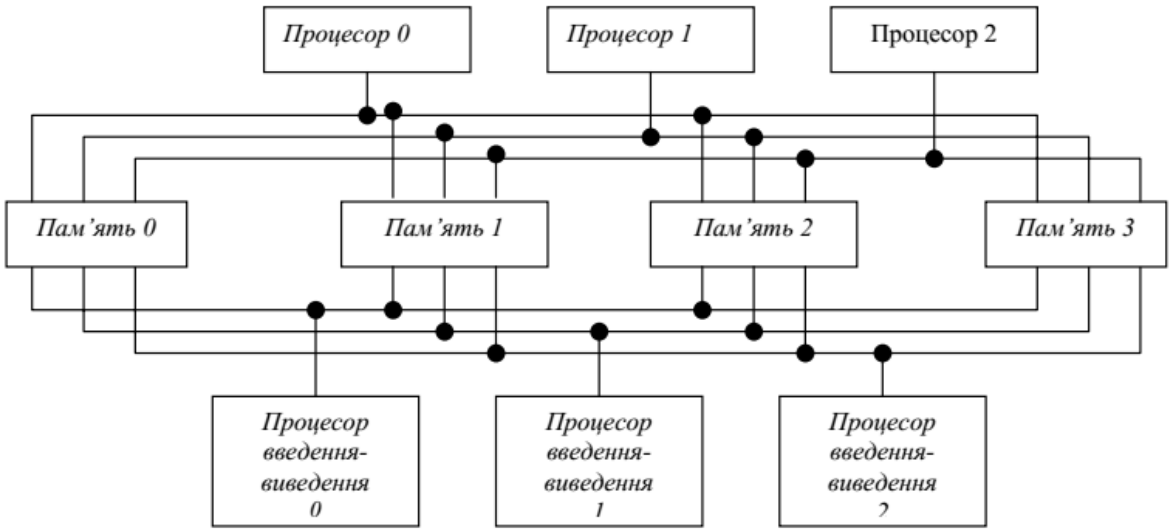
В цій схемі звертання до різних пристроїв пам' яті можуть виконуватись одночасно, що збільшує швидкодію всієї обчислювальної системи. Однак, апаратні засоби необхідні для побудови координатного комутатора можуть бути доволі складні, оскільки кожний вузол комутатора містить апаратуру керування, комутації і пріоритетного арбітражу.



Багатопортова пам'ять характеризується тим, що апаратура керування, комутації і пріоритетного арбітражу розміщується в інтерфейсі кожного пристрою пам'яті. За такої організації кожний функціональний пристрій може отримати доступ до кожного пристрою пам'яті, але тільки по конкретному порту, або каналу зв'язку. Портам пам' яті присвоюються постійні пріоритети для розв'язування конфліктних ситуацій між пристроями, що одночасно звертаються до однієї і тієї ж пам' яті.

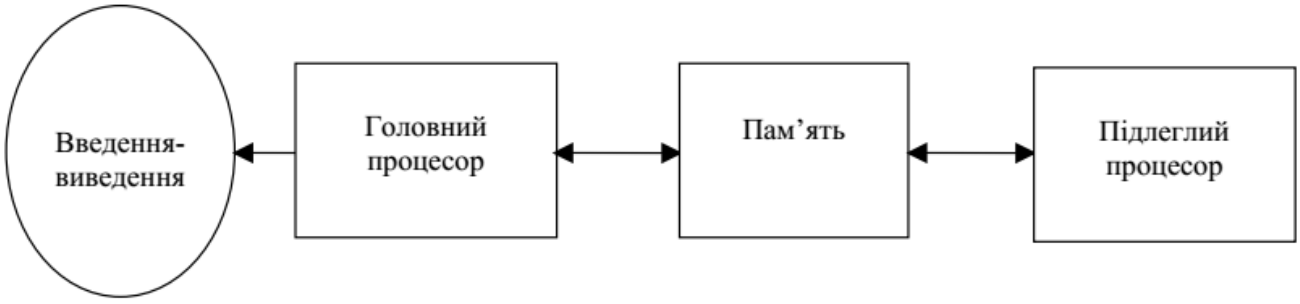
Крім того, багатопортова пам'ять може дозволяти доступ до різних пристроїв пам' яті різним підгрупам центральних процесорів або процесорів введення-виведення, що дуже важливо для систем високої секретності.

Недоліком систем з багатопортовою пам'яттю є великі витрати кабелю на виконання необхідних з'єднань.



Організація "головний-підлеглий "

При організації операційної системи "головний-підлеглий" операційна система виконується тільки на головному процесорі. На підлеглих процесорах можуть виконуватись тільки програми користувачів.



Коли процесу, який виконується на підлеглому процесорі потрібна послуга ОС, він генерує сигнал переривання. Якщо підлеглих процесорів багато, то можуть створюватись черги до головного процесора і таким чином обчислювальна потужність підлеглих процесорів використовується не в повній мірі.

Ще одним недоліком цього типу організації є недостатня надійність, оскільки вихід головного процесора з ладу може призвести до виходу з ладу всієї системи(в деяких системах підлеглий процесор може взяти на себе функції головного).

Організація з роздільними моніторами

При організації з роздільними моніторами кожний процесор має власну операційну систему, яка певним чином реагує на переривання від програм користувачів, що працюють на цьому процесорі. Кожний процесор керує своїми ресурсами, незалежно від інших процесорів, наприклад, файлами або пристроями вводу-виводу. Процес, запланований для виконання на якомусь певному процесорі, виконується на ньому до завершення. Доступ до спільних таблиць виконується з використанням механізму взаємо виключення.

Не передбачається ніякої взаємодії процесорів при виконанні індивідуального процесу. Деякі процесори можуть лишатися вільними, в той самий час як один процесор виконує довготривалий процес.

Симетрична організація

Вдалішою є система з симетричною організацією. Характеристики цієї організації такі:

1. Найбільш складна для реалізації. Всі процесори ідентичні і кожен з них може керувати будь-яким пристроєм введення-виведення або звертатися до пам'яті.
2. ОС переміщується від одного процесора комплексу до іншого. Процесор, який в даний момент відповідає за ведення системних таблиць і виконання системних функцій, називається моніторним процесором. В кожен конкретний момент часу моніторним може бути лише один процесор, що усуває конфлікти під час роботи з системною інформацією.
3. Процес може виконуватись на будь-якому процесорі в різні періоди часу. Процесори можуть кооперуватися для виконання одного завдання.
4. Оскільки програми ОС виконуються на багатьох процесорах, реєнтерабельний код і взаємовиключення обов'язкові для ОС.
5. Необхідні програмні і апаратні засоби для розв'язання конфліктних ситуацій.

6. Навантаження є точніше збалансованим і організація в цілому є найбільш надійною з розглянутих.

Продуктивність мультипроцесорних систем

Збільшення кількості процесорів в n разів не означає, що продуктивність збільшиться в n разів через:

- додаткові витрати на роботу ОС;
- збільшення конкуренції за системні ресурси;
- затримки в апаратурі комунікації і маршрутизації.

Відомі мультипроцесорні системи, в яких при двох процесорах продуктивність зростає в 1.8 рази, при 3-х - в 2,1 рази, при 4-х - в 3,6 рази.

Перелік питань на підсумковий контроль

1. Опишіть поняття операційної системи, програмне забезпечення та генерація системи.
2. Наведіть основні функції операційних систем.
3. Наведіть основні ресурси операційних систем та взаємозв’язки.
4. Вкажіть, які покоління операційних систем розрізняють.
5. Вкажіть за якими критеріями класифікують операційні системи.
6. Вкажіть, які розрізняють типи багатопрограмної роботи.
7. Вкажіть, які функції виконує ОС з точки зору користувача.
8. Розкрийте поняття ієрархічної структури ОС.
9. Вкажіть основні принципи побудови ОС.
10. Вкажіть поняття процесу та наведіть стани процесів.
11. Опишіть поняття переходу процесу зі стану в стан на основі діаграми.
12. Вкажіть, що собою представляє блок керування процесом РСВ.
13. Вкажіть основні операції над процесами.
14. Опишіть поняття переривань та їх послідовність дій.
15. Охарактеризуйте типи переривань.
16. Опишіть поняття ядро ОС та його основні функції.
17. Вкажіть, як поділяються процеси за часом розвитку.
18. Вкажіть, як поділяються процеси за місцем розвитку.
19. Вкажіть, як поділяються процеси за способами зв’язків.
20. Вкажіть види відношень між процесами.
21. Вкажіть, що собою представляє задача синхронізації.
22. Вкажіть, що собою представляє взаємовиключення і деякі варіанти їх реалізацій.
23. Вкажіть, що собою представляють семафори.
24. Вкажіть основні операції над семафорами.
25. Розкрийте поняття мониторoподібні засоби синхронізації.
26. Вкажіть, що собою представляють м’ютекси.

27. Вкажіть, що собою представляє критична секція.

28. Охарактеризуйте поняття організації пам’яті.

29. Опишіть ієрархічну організацію пам’яті.

30. Вкажіть на які види стратегій поділяється керування пам’яттю.

31. Розкрийте поняття зв’язного розподілу пам’яті.

32. Розкрийте поняття незв’язного розподілу пам’яті.

33. Розкрийте поняття трансляції та завантаження в абсолютних адресах.

34. Розкрийте поняття завантаження переміщувальних модулів.

35. Мультипрограмування з змінними розділами при зв’язному розподілі пам’яті.

36. Мультипрограмування зі стопінгом.

37. Розкрийте поняття віртуальної пам’яті.

38. Розкрийте поняття концепції віртуальної пам’яті.

39. Опишіть поняття механізму відображення віртуальних адрес в реальні.

40. Розкрийте поняття перетворення адрес сторінок методом прямого відображення.

41. Розкрийте поняття перетворення адрес сторінок методом асоціативного відображення.

42. Розкрийте поняття сегментної організації віртуальної пам’яті.

43. Розкрийте поняття комбінованої сторінково-сегментної організації пам’яті.

44. Опишіть стратегію вибірки керування віртуальною пам’яттю.

45. Опишіть стратегію розміщення керування віртуальною пам’яттю.

46. Опишіть стратегію заміщення керування віртуальною пам’яттю.

47. Опишіть концепцію локальності стратегій керування пам’яттю.

48. Вкажіть поняття керування пам’яттю на основі робочих множин.

49. Розкрийте поняття підкачки сторінок по запиту стратегій організацій пам’яті.

50. Розкрийте поняття підкачки сторінок з упередженням.

51. Вкажіть основні рівні планування в ОС.

52. Вкажіть, що собою представляє планування з переключенням і без переключення.

53. Розкрийте поняття пріоритетів процесів.

54. Вкажіть, що собою представляють дисципліни обслуговування та формування черг центрального процесору.

55. Опишіть дисципліни планування FIFO, LIFO та RR.

56. Опишіть дисципліни планування SJF, SRT та HRN.

57. Опишіть принципи роботи багаторівневих черг.

58. Опишіть поняття мультипроцесорних систем.

59. Опишіть спосіб пошуку неявного паралелізму – розщеплення циклу.

60. Опишіть спосіб пошуку неявного паралелізму – редукція висоти дерева.

61. Опишіть спосіб пошуку неявного паралелізму – правило «ніколи не чекати».

62. Вкажіть, яким чином можна охарактеризувати мультипроцесорні комплекси.

63. Опишіть вид організації мультипроцесорних комплексів з спільною шиною.

64. Опишіть вид організації мультипроцесорних комплексів з матрицею координатної комутації.

65. Опишіть вид організації мультипроцесорних комплексів з багато портовою пам’яттю.

66. Опишіть варіант організації ОС для мультипроцесорних комплексів – головний-підлеглий.

67. Опишіть варіант організації ОС для мультипроцесорних комплексів – з розподільними моніторами.

68. Опишіть симетричну організацію для мультипроцесорних комплексів.

69. Опишіть поняття продуктивності мультипроцесорних систем.