# Лекція № 1

# Основні функції операційних систем

Під операційною системою (ОС) звичайно розуміють комплекс керуючих та оброблюючих програм, що реалізують інтерфейс між апаратурою комп’ютера і користувачем, з одного боку, та оптимізують використання ресурсів обчислювальної системи при вирішенні конкретних задач, з іншого боку. ОС має виконувати наступні функції:

1. Забезпечувати завантаження користувальницьких програм в оперативну пам'ять й їхнє виконання.
2. Забезпечувати керування пам'яттю. У найпростішому випадку, це – вказівка єдиній завантажуваній програмі на кінцеву адресу пам'яті, доступної для використання, і на початкову адресу пам'яті, зайнятої системою.
3. Забезпечувати роботу із пристроями довгострокової пам'яті.
4. Надати стандартизований доступ до різних периферійних пристроїв.
5. Надати користувальницький інтерфейс.

У ряді ОС, функції цим вичерпуються (наприклад, у MS-DOS). Більш розвинені ОС надають додаткові можливості:

1. Паралельне (або псевдопаралельне) виконання декількох завдань.
2. Організація взаємодії завдань один з одним.
3. Організація максимальної взаємодії й поділу ресурсів.
4. Захист системних ресурсів, даних і програм користувача.
5. Аутентифікація користувача (перевірка користувача, чи він є тим, за кого себе видає, й має право виконати ту або іншу операцію).

## **1.1 Класифікація ОС**

В залежності від того, які з перерахованих вище функцій операційні системи виконують, їх можна розділити на:

#### ДОС (Дискові операційні системи)

Це системи, які беруть на себе виконання тільки перших чотирьох функцій. Як правило, це резидентний набір програм, і не більш того. ДОС завантажує користувальницьку програму в пам’ять і передає їй керування, після чого програма робить із системою, що їй заманеться. При завершенні програми вважається “гарним тоном” залишити машину в такому стані, щоб ДОС змогла продовжити роботу. Якщо програма приводить машину до якогось іншого стану, ДОС нічим цьому перешкодити не може.

Характерний приклад – різні завантажувальні монітори. Такі системи працюють одночасно тільки з однією програмою.

MS DOS для IBM PC – прямий спадкоємець такого монітора.

Існування подібних систем обумовлено їхньою простотою й малих ресурсів для їх реалізації.

Ще одна причина їх можливого використання навіть на досить потужних машинах – вимога програмної сумісності з ранніми моделями того ж самого сімейства комп’ютерів.

## **ОС загального призначення**

Це системи, що виконують всі перераховані функції, наприклад, IBM DOS й OC/360 і наші аналоги ОС ЄС.

Ці ОС розраховані на інтерактивну роботу одного або декількох користувачів у режимі поділу часу, при не дуже жорстких вимогах своєчасної реакції системи на зовнішні події. У таких системах велика увага приділяється захисту самої системи, програмного забезпечення й користувальницьких даних від помилкових і зловмисних дій програм і користувачів.

До цього класу ставиться відома ОС Windows 2000, а також системи сімейства UNIX.

## **Системи віртуальних машин**

Ці ОС допускають одночасну роботу декількох програм, але створююють при цьому для кожної програми ілюзію того, що машина перебуває в повному її розпорядженні, як і при роботі під керуванням ДОС.

Віртуальні машини – цінний засіб при розробці й тестуванні кросплатформних програм. Вони також використовуються для налагодження модулів ядра або самої операційної системи.

Для таких систем характерні високі накладні витрати й порівняно низька надійність. Тому вони рідко використовуються для промислового застосування. У системах віртуальних машин приділяється велика увага емуляції роботи апаратури.

Часто ці системи є підсистемами ОС загального призначення, наприклад підсистема W0W в Windows NT, Windows 2000, емулятор RT-II в VAX тощо.

## **Системи реального часу**

Ці системи призначені для полегшення розробки так званих програм **реального часу** – програм, які управляють некомп'ютерним устаткуванням, часто із жорсткими часовими обмеженнями.

Відмітною ознакою системи реального часу є здатність гарантувати певний час реакції. Важливо враховувати різницю між гарантованістю й просто високою продуктивністю й низькими накладними витратами. Далеко не всі алгоритми й технічні рішення, навіть й ті, які забезпечують відмінний середній час реакції, підходять для програм і ОС реального часу.

Типовими представниками цього класу систем є відомі системи OS-9 й OS-9000.

По суті, мультимедійні алгоритми (тобто потребуючі синхронізації зображення на екрані й звуку) також є системами реального часу. Розбіжність звуку й зображення фіксується людиною вже в межах 30 мсек.

На сьогодні існує кілька визначень систем реального часу (СРЧ) (real time operating systems (RTOS)), однак більшість з них суперечить одне одному.

Наведемо деякі з цих визначень для демонстрації різних поглядів на призначення і основні завдання СРЧ:

1. Системою реального часу називається система, в якій успішність роботи будь-якої програми залежить не тільки від її логічної правильності, а й від часу, за який вона отримала результат. Якщо часові обмеження не витримані, тоді фіксується збій в роботі систем.

Таким чином, часові обмеження повинні бути гарантовано витримані. Це вимагає від системи бути передбачуваною, тобто, незалежно від свого поточного стану і завантаженості, видавати потрібний результат за необхідний час. При цьому бажано, щоб система забезпечувала якомога більший відсоток використання наявних ресурсів. Прикладом завдання, де потрібна СРЧ, є управління роботом, що бере деталь зі стрічки конвеєра. Деталь рухається, і робот має лише невелике часове вікно, коли він може її взяти. Якщо він запізниться, то деталь вже не буде на потрібній ділянці конвеєра, і, отже, робота не буде зроблена, незважаючи на те, що робот знаходиться в правильному місці. Якщо він позиціонується раніше, то деталь ще не встигне під'їхати, і він заблокує їй шлях.

Іншим прикладом може бути космічний апарат, що знаходиться на автопілоті. Сенсорні серводатчики повинні постійно передавати в керуючий комп'ютер результати вимірювань. Якщо результат будь-якого вимірювання буде пропущено, то це може привести до неприпустимої невідповідності між реальним станом систем космічного апарату і інформацією про нього в керуючій програмі. Розрізняють жорсткі (hard) і слабкі (soft) вимоги реального часу. Якщо запізнення програми призводить до повного порушення роботи керованої системи, тоді говорять про жорсткі вимоги реального часу (жорсткі СРЧ).

Якщо ж запізнювання призводить тільки до втрати продуктивності, тоді говорять про слабкі вимоги реального часу (м'які СРЧ). Більшість програмного забезпечення орієнтоване на м'який реальний час, а завдання хорошої СРЧ – забезпечити рівень безпечного функціонування системи, навіть якщо керуюча програма ніколи не закінчить своєї роботи.

1. Стандарт POSIX 1003.1 визначає СРЧ наступним чином: реальний час в операційних системах – це здатність операційної системи забезпечити необхідний рівень сервісу в заданий проміжок часу.
2. Іноді системами реального часу називають системи постійної готовності (on-line системи), або інтерактивні системи з достатнім часом реакції. Звичайно це роблять фірми-виробники з маркетингових міркувань. Якщо інтерактивну програму називають працюючою в реальному часі, то це означає, що вона встигає обробляти запити від людини, для якої затримка в сотні мілісекунд навіть непомітна.
3. Часто поняття «система реального часу» ототожнюють з поняттям «швидка система». Це не завжди правильно. Час затримки реакції СРЧ на подію вже не так і важливий (він може досягати декількох секунд). Головне, щоб цього часу було достатньо для конкретного додатку і гарантовано. Часто алгоритм з гарантованим часом роботи менш ефективний, ніж алгоритм, що такою властивістю не володіє. Наприклад, алгоритм «швидкого» сортування (quicksort) в середньому працює значно швидше багатьох інших алгоритмів сортування, але його гарантована оцінка складності значно гірша.
4. У багатьох важливих сферах виконання додатків СРЧ вводяться свої поняття «реального часу». Так, про процес цифрової обробки сигналу кажуть, що він йде в «реальному часі», якщо аналіз (при введенні) і / або генерація (при виведенні) даних може бути проведено за той самий час, що і аналіз та / або генерація тих самих даних без цифрової обробки сигналу. Наприклад, якщо при обробці аудіо даних потрібно 2,01 секунди для аналізу 2,00 секунди звуку, тоді це не процес реального часу. Якщо ж потрібно 1,99 секунди, тоді це – процес реального часу. Виходячи з вищезазначеного, визначення системи реального часу можна подати в наступній інтерпретації.

**Визначення**. Система реального часу реагує в передбачуваний час на непередбачувану появу зовнішніх подій.

 Основні відмінності між ОС РЧ та звичайними ОС можна

продемонструвати за допомогою табл. 1.1.

Таблиця 1.1. Відмінності ОС РЧ

|  |  |  |
| --- | --- | --- |
|   | ОС РЧ  | Звичайна ОС  |
| Основна задача  | Можливість реагувати на події, що надходять зовні  | Оптимально розподілити ресурси комп’ютера між користувачами та задачами  |
| Як позиціонується  | Інструмент для створення конкретного програмно- апаратного комплексу  | Сприймається користувачем як сукупність додатків, готових до використання  |
|  На що орієнтована  | Обробка зовнішніх подій  | Обробка дій користувача  |
| Кому призначена  | Кваліфікованому розробнику  | Розробнику середньої кввліфікації  |

# Засоби крос-розробки

Це алгоритми, що призначені для розробки програм у двохмашинній конфігурації. При цьому редагування, компіляція й налагодження виконуються на інструментальній машині, а потім скомпільований код завантажується в цільову машину.

Прикладами таких ОС є системи програмування мікроконтролерів Intel. Такі системи, як правило, містять:

* набір компіляторів й асемблерів, що працюють на інструментальній машині з “нормальною” ОС;
* бібліотеки, які виконують велику частину функцій ОС при роботі програми (крім функції завантаження);
* засоби налагодження.

# Системи проміжних типів

Існують системи, які важко віднести до якогось одного з перерахованих типів. Найбільш відомими з таких систем є MS Windows 3.x й Windows 95. Вони, як ОС, використовують апаратні засоби процесора для захисту й віртуалізації пам'яті, і навіть забезпечують деяку подібність багатозадачності, але не захищають себе й програми від помилок інших програм (тобто поводяться, в цьому сенсі, як ДОС).

#### Сімейства операційних систем

Часто існує спадковість між різними ОС. Така спадковість обумовлена вимогами сумісності й переносимості прикладного ПЗ, крім того, запозиченню окремих вдалих рішень.

На основі такої спадковості можна побудувати “генеалогічні дерева” ОС.

Так, можна виділити кілька сімейств нині експлуатованих ОС, наприклад, сімейство UNIX, у тому числі ОС Linux, а також сімейств, які вже вимерли або вимирають, наприклад, системи для великих комп'ютерів IBM OS-360, IBM-Vm, OS/2.

# Архітектура ОС

Найбільш загальним підходом до структуризації операційної системи є поділ усіх її модулів на дві групи:

1. Ядро – модулі, що виконують основні функції ОС.
2. Модулі, що виконують допоміжні функції.

Модулі ядра виконують:

* + керування процесами,
	+ керування пам'яттю,
	+ керування пристроями введення-виведення.

Ядро виконує такі функції, як перемикання контекстів, завантаження / вивантаження сторінок пам'яті, обробку переривань. Ці функції недоступні для програм. Вони створюють для них **прикладне програмне середовище***.*

Додатки звертаються до ядра із запитами – **системними викликами** – для виконання тих чи інших дій (наприклад, відкриття й читання файлу, виводу графіки, одержання системного часу й т.п.).

Функції ядра, які можуть викликатися додатками, утворюють **інтерфейс**

**прикладного програмування.**

Функції, виконувані модулями ядра, є найчастіше використовуваними, тому швидкість їхнього виконання визначає продуктивність всієї системи в цілому. Саме тому більша частина модулів ядра є резидентною, тобто постійно перебуває в оперативній пам'яті.

**Допоміжні модулі** ОС оформлюються або у вигляді програм, або у вигляді бібліотек процедур. Оскільки частина модулів ОС виконуються як звичайні додатки (тобто в стандартному для даної ОС форматі), то часто буває складно провести чітку грань між ОС і додатками.

Рішення про те, чи є якась програма частиною ОС чи ні, приймає виробник ОС.

Допоміжні модулі ОС поділяють на наступні групи:

* утиліти – програми, що вирішують окремі завдання керування й супроводу комп'ютерної системи (наприклад, програми

стискування дисків, архівування даних);

* системні оброблюючі програми – текстові й графічні редактори, компілятори, компонувальники, відладчики і т. п.;
* програми надання користувачеві додаткових послуг – спеціальний користувальницький інтерфейс, калькулятор і т.п.
* бібліотеки процедур – спрощують розробку програм (наприклад, бібліотека математичних функцій, функцій введення-виведення й т.п.).

Допоміжні модулі ОС звертаються до функцій ядра за допомогою **системних викликів**. Поділ операційної системи на ядро й модулі-додатки забезпечує легку розширюваність ОС (рис.1.1).

Ядро

 ОС

Ут

и

л

і

т

и

Библ

і

отеки

Програми

користувача

Системн

і

оброблюючі

програми

  програм

##### Рис. 1.1. Загальна структура ОС

Допоміжні модулі ОС завантажуються в оперативну пам'ять тільки на час свого виконання, тобто є **транзитними**.

Важливою властивістю архітектури, заснованої на ядрі, є можливість захисту кодів і даних операційної системи за рахунок виконання функцій ядра в привілейованому режимі.

Забезпечити привілеї ОС неможливо без спеціальних засобів апаратної підтримки. Апаратура комп'ютера повинна підтримувати як мінімум два режими роботи: користувальницький режим і привілейований режим, його також називають режимом ядра або режимом супервізора*.*

Додатки ставляться в підлеглий стан за рахунок заборони виконання в користувальницькому режимі деяких критичних команд, пов'язаних з перемиканням процесора із завдання на завдання, керуванням пристроями введення-виведення, доступом до механізмів розподілу й захисту пам'яті. Умова дозволу виконання критичних інструкцій перебуває під контролем ядра й забезпечується за рахунок набору інструкцій, безумовно, заборонених для користувальницького режиму.

Аналогічно забезпечуються привілеї ядра при доступі до пам'яті. Наприклад, виконання інструкції доступу до пам'яті для додатка дозволяється, якщо інструкція звертається до області пам'яті, відведеної ОС даному додатку, і забороняється при звертанні до інших областей.

Між кількістю рівнів привілеїв, реалізованих апаратно, і кількістю рівнів привілеїв, підтримуваних ОС, немає прямої відповідності. Так, наприклад, на базі чотирьох рівнів, забезпечуваних процесорами Intel, ОС OS/2 будує трирівневую систему привілеїв, а Windows NT - дворівневу.

З іншого боку, якщо апаратура підтримує хоча б два рівні, програмним способом можна побудувати ОС із як завгодно розвиненою системою захисту. Розглянута архітектура ОС, заснована на привілейованому ядрі й додатках користувацького режиму, стала класичної. Її використовують багато відомих ОС: UNIX, VMS, OS/390, OS/2, Windows NT.

У деяких випадках, розроблювачі відступають від цієї класики, і привілейований режим використовується й для програм ОС.

У цьому випадку, звертання програм до ядра здійснюються швидше, але при цьому відсутній надійний апаратний захист пам'яті.

За своєю внутрішньою архітектурою ОС можна умовно поділити на монолітні ОС, ОС на основі мікроядра та об’єктно-орієнтовані ОС. ОС з монолітною архітектурою можна представити у вигляді:

* прикладний рівень – прикладні процеси;
* системний рівень – монолітне ядро ОС , що, у свою чергу,

складається:

* + з інтерфейсу між додатками та ядром;
	+ власне ядра ОС;
	+ інтерфейсу між драйверами та пристроями.

Основною перевагою монолітної архітектури є більша швидкодія, у порівнянні з іншими архітектурами. Однак, ця перевага досягається, в основному, за рахунок написання основної частини програми ОС на асемблері.

Недоліки монолітної архітектури:

* + системні виклики, що потребують переключення рівня привілеїв, повинні реалізовувати інтерфейси між додатками та ядром як переривання. Це значно збільшує час їх виконання;
	+ ядро не може бути перерване додатком. Це може призвести до того, що високопріоритетна задача може не отримати процесор, зайнятий виконанням низькопріоритетної задачі. Наприклад, низькопріоритетна задача запросила виділити пам’ять, зробила системний виклик, до закінчення якого сигнал на активізацію високопріоритетної задачі не може її активізувати;
	+ складність переходу на нову архітектуру процесора при невідповідності асемблерів;
	+ негнучкість та складнощі з розвитком, оскільки зміна частини ядра потребує його повної перекомпіляції.

**ОС на основі модульної організації** прибирає недоліки, пов’язані з інтерфейсом між додатками та ядром, полегшує модернізацію ОС та її встановлення на нові мікропроцесори.

При модульній архітектурі інтерфейс відіграє тільки одну роль – забезпечує зв’язки додатків із спеціальним модулем – менеджером процесів.

А ядро – подвійну роль:

* + керує взаємодією частин системи (менеджерів процесів та файлів);
	+ забезпечує неперервність виконання коду системи (тобто переключення задач відсутнє під час виконання функцій ядра).

В модульній архітектурі недоліки майже залишились, хоча вони

перейшли з рівня інтерфейсу на рівень ядра.